REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Informatique Décisionnelle et Optimisation

Par :

BRAHIMI Souha
MIMOUNE Mohamed El Amine

Les Problèmes d'Ordonnancement Multi-Objectifs dans un Système de Production

Soutenu publiquement devant le jury composé de :

Pr. GASMI Abdelkader	**Université de M'sila**	**Président**
Dr. MOUHOUB Nasser Eddine	**Université de M'sila**	**Rapporteur**
Mr. BOUDAA Abdelghani	**Université de M'sila**	**Examinateur**

Promotion : 2021/2022

Remerciements

Nous remercions et nous louons Dieu, le tout puissant de sa grâce, son aide et sa miséricorde d'avoir accepté nos supplications, afin de nous mobiliser tous ces gens, qui nous ont donné de leur savoir, leur patience et leur temps précieux, où ils nous ont porté leur aide sans aucune pénurie. Sans cela, ce modeste travail ne serait pas ici devant vos mains.

Nous tenons à exprimer nos sincères remercîments et notre gratitude à notre encadreur, l'honorable professeur, Mr. N. Mouhoub pour ses précieuses orientations et conseils au cours de la préparation de ce mémoire, nous n'oublions pas son aide en tant qu'enseignant, où il n'a jamais cessé de nous donner de son bon enseignement, particulièrement ce qui est lié à ce mémoire.

Nous tenons à remercier vivement les membres de jury d'avoir accepté de juger notre modeste travail ainsi que tous les enseignants du département d'informatique de l'université Mohamed Boudiaf de Msila d'avoir contribuer à notre formation.

Table des matières

Chapitre 03 : Concepts de base des Algorithmes Génétiques

Chapitre 04 : Etude pratique

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction générale

De nombreux problèmes rencontres dans différents secteurs économiques, scientifiques et techniques, sont de nature combinatoire, selon le nombre de critères pris en compte, ces problèmes peuvent être mono ou multicritères. En effet, l'optimisation combinatoire regroupe une large classe de problèmes ayant des applications dans de nombreux domaines de l'industrie, parmi lesquels nous citons :

- Design des systèmes dans les sciences d'ingénieurs (mécanique, aéronautique, chimie, etc.)
- Transport : design de réseaux de transport, trace autoroutier, etc.
- Finances : investissements financiers, etc.
- Planification de trajectoires des robots mobiles, etc.
- Agronomie : programme de production agricole, etc.
- Environnement : gestion de la qualité de l'air, distribution de l'eau, etc.
- Télécommunications : design d'antennes, affectation de fréquences, etc.
- Ordonnancement et affectation : ordonnancement en production, localisation d'usines.

Ces problèmes issus du monde réel sont complexes et difficiles à résoudre, de plus ils sont rarement mono-objectif, et requièrent souvent la prise en compte de plusieurs critères conflictuels, ce qui amplifie encore le degré de difficulté. En effet, de tels problèmes sont réputes pour être particulièrement des problèmes NP-Difficiles.

Contrairement au cas mono-objectif, dans le cas multi-objectif, la notion de solution optimale n'a plus de sens et est remplacée par la notion de solution efficiente ou solution non dominée, dites Pareto optimales. Ainsi, l'optimisation multi-objectif s'intéresse aux particularités liées a l'existence de ces solutions optimales, et aux méthodes de résolution dédiées a ce type de problème.

De nos jours, les entreprises doivent être compétitives et ont besoin pour cela d'outils performants pour gérer la complexité de leur organisation. La productivité peut être affectée directement par la qualité de l'ordonnancement des opérations sur les machines car un atelier de production peut réaliser une grande variété de produit avec des coûts réduits, grâce à une meilleure utilisation des ressources.

Le problème étudie porte plus particulièrement sur un problème d'ordonnancement multi objectif d'un "Atelier d'extrusion Plastique», le but de ce problème d'optimisation est de minimiser les coûts de transitions entre commandes liées sur une même ligne de production tout en minimisant le temps d'ordonnancement total "le makespan", à la fois dont Les paramètres "coût" et "makespan" seront introduits dans la fonction objectif F, qui reste à déterminer. Nous proposons d'utiliser l'outil d'optimisation de l'algorithme génétique pour résoudre ce problème.

Cette problématique est détaillée dans le chapitre 4, qui traite de l'optimisation des fonctions d'ordonnancement, au niveau d'une usine d'extrusion plastique qui fait actuellement face à des problèmes majeurs liés aux matières premières polyéthylène, très chères après les chocs pétroliers. La nature de la production de matières plastiques a un impact très négatif sur la production. D'une part, passer d'une référence de commande à une autre crée une quantité temporaire de plastique à recycler, d'autre part, passé d'une commande à l'autre introduit des temps de non-production (perte de temps machine), qui peuvent être très coûteux du fait de pour du personnel supplémentaire.

Dans cette étude, nous proposons un puissant outil basé sur des principes d'intelligence artificielle pour répartir tout le travail (commandes) sur la machine, respectez au maximum un ensemble de contraintes (de précédences, de ressources,….), et cible (minimiser les coûts) modifications des commandes et heure associée à la fin de l'exécution de la tâche fabrication "Makespan").

L'étude porte principalement sur un problème d'ordonnancement qui implique plusieurs fonctions objectives. L'entreprise souhaite minimiser les objectifs temps (Makespan) et coût, il existe des algorithmes partiellement aléatoires qui permettent d'améliorer la recherche d'optima dans des problèmes complexes, 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. Pour la résolution de notre problème d'ordonnancement nous avons utilisé les Algorithmes Génétiques Ces derniers sont basés sur les concepts de la sélection naturelle et génétique. L'algorithme commence avec un ensemble de solutions possibles du problème (individus), constituant une population. Les individus sont formés par des variables, qui sont les paramètres à ajuster dans un dispositif.

Cette population est conçue aléatoirement à l'intérieur de limites prédéfinies. Certaines solutions de la première population sont utilisées pour former une nouvelle population, à partir d'opérateurs génétiques (croisement, mutation, etc.). Ceci est motivé par l'espoir que la nouvelle population soit meilleure que la précédente.

Au premier chapitre, nous verrons tout d'abord la définition du problème d'optimisation et les termes relatifs, notamment les problèmes multi-objectifs et on définit plusieurs types d'optimisation et les domaines d'applications d'Optimisation Combinatoire.

Dans le deuxième chapitre, nous avons introduit des définitions sur les problèmes d'ordonnancement (PO) afin d'aborder ses éléments et la typologie des (PO), à savoir les méthodes exactes et les méthodes approchées.

Le troisième chapitre sera consacré à présenter les Algorithmes Génétiques. Ces techniques d'optimisation seront appliquées à notre problème.

Dans le quatrième chapitre, nous présentons l'environnement permettant la mise en œuvre des algorithmes étudiées « MATLAB», nous verrons la représentation de notre problème par l'algorithme génétique et les différentes étapes d'implémentation qui respectent le problème choisi ainsi que les descriptions des chapitres précédents, nous avons également décrit l'atelier et la démarche de résolution, (codage du problème et la création de population ...etc.) ensuite nous avons expliqué la méthode utilisée pour résoudre et améliorer le problème d'ordonnancement à l'aide d'algorithme génétique, nous décrivons donc les étapes de la solution "la fonction objectif et fonction fitness" et la méthode utilisée "sélection par roulette", et nous verrons également comment fonctionne le programme, ce chapitre se termine par la présentation des expérimentations effectuées dans le logiciel.

Finalement, nous clôturons ce mémoire par une conclusion générale.

Chapitre 01

L’optimisation combinatoire multi objectifs

Chapitre 01 L'optimisation combinatoire multi objectifs

1. Introduction

Dans le monde réel, la plupart des problèmes nécessitent l'optimisation simultanée d'objectifs multiples, souvent interdépendants. Alors que les solutions optimales sont généralement bien définies dans l'optimisation à objectif unique, ce n'est pas le cas pour les problèmes à objectifs multiples, où il existe un ensemble de solutions qui sont des "compromis".

Il existe deux méthodes d'optimisation, la première est l'optimisation à objectif unique, qui est basée sur la minimisation (ou maximisation) d'une fonction objectif unique, dont le but est de trouver la solution optimale, appelée la solution optimale qu'est facilement définie comme une performance unique par contre, l'optimisation multi-objectifs optimise simultanément plusieurs fonctions objectifs souvent contradictoires, et on cherche la meilleure solution en fonction d'un ensemble de propriétés du problème (plus le temps de réponse est fort, plus la robustesse est forte, plus la temps de réponse, plus le temps de montée est long plus il est long, plus il est robuste, etc.) Le résultat d'un problème d'optimisation multicritères est généralement une variété de solutions, différenciées par différents arbitrages entre objectifs.

Ces solutions sont optimales lorsque tous les objectifs sont considérés simultanément car aucune autre solution dans l'espace de recherche n'est meilleure qu'elles.

Ce chapitre présent d'abord un ensemble de définitions liées à l'optimisation combinatoire, puis définit formellement les concepts de base de l'optimisation multi-objectifs, puis le domaine de résolution de problèmes de l'optimisation combinatoire.

2. Problèmes d'optimisation

Un problème d'optimisation, noté P (X, f), est caractérisé par un ensemble réalisable ou admissible X non-vide et une fonction objectif f qui associe un scalaire dans R à chaque élément x de l'ensemble X. Les éléments de X sont dits solutions réalisables.

Résoudre le problème P (X, f) revient à trouver parmi les solutions réalisables, une qui minimise ou maximise f, c'est-à-dire dans le cas d'un problème de minimisation,

trouver une solution $x^* \in X$ telle que $f(x) \geq f(x*)$ pour tout élément x dans X. Une telle solution est dite optimale et sera notée $x(X, f)$.

On peut dire aussi qu'un problème d'optimisation se définit comme la recherche du minimum ou du maximum (de l'optimum donc) d'une fonction donnée. On peut aussi trouver des problèmes d'optimisation pour lesquels les variables de la fonction à optimiser sont des contraintes à évoluer dans une certaine partie de l'espace de recherche.

2.1. Terminologie

Dans le but de bien éclaircir le domaine de notre thème, nous en présentons ici quelques termes que nous jugeons utiles pour la compréhension du travail qui sera abordé plus loin : [08]

Variables de décision : Le vecteur $x \in X$ est appelé vecteur de décision. Les éléments formants le vecteur de décision sont appelés variables de décision. Ce sont les variables recherchées dans le problème.

Dimension de problème : La dimension du problème est représentée par le nombre entier $n \in N^*$. Elle représente le nombre de variables de décision. Une solution à un problème de 3 dimensions est un vecteur de 3 éléments (3-uplet).

Espace de recherche et Solution admissible : L'ensemble X de dimension n contient les valeurs que le vecteur x peut prendre. Ces dernières sont appelées solutions admissibles ou faisables (ou encore solutions candidates). Une solution admissible est une affectation de toutes les variables de décision, qui répond aux contraintes du problème. L'ensemble X est connu sous différentes appellations : ensemble admissible, espace de solution, espace de recherche, domaine de recherche etc.

Contraintes : L'espace de recherche X est limité par quelques règles. Une contrainte est une condition que doivent respecter les vecteurs de décision du problème.

Solution optimale : C'est la meilleure solution $\bar{x} \in X$ du problème. . Elle est appelée solution globale ou optimale.

Fonction objectif : la fonction f, à valeur scalaire, sert de critère pour déterminer la meilleure solution du problème. Elle est appelée fonction d'objectif (fonction fitness, critère, etc.). Elle attribue à chaque solution $x \in X$ de l'espace de recherche, un nombre réel indiquant sa valeur (son score).

Minimisation : Une minimisation revient à trouver un élément $\bar{x}$de X tel que : $\forall\, x \epsilon X : f(\bar{x}) \geq f(\mathbf{x})$On dit qu'on cherche à minimiser la fonction f sur l'ensemble X. Dans ce cas la fonction d'objectif f est connue comme une fonction de coût. Le vecteur x est une borne inférieure de l'ensemble X.

Maximisation : Une maximisation revient à trouver un élément x de X tel que $: \forall x \in X : f(x) > f(x)$. On dit qu'on cherche à maximiser la fonction f sur l'ensemble X. Dans ce cas la fonction objectif f est connue comme une fonction d'utilité ou de profit. Le vecteur x est une borne supérieure de l'ensemble X.

Minimum et maximum stricte : On parle d'un minimum stricte (respectivement un maximum stricte) $si : \forall x \in X : f(x) < f(x)$ *(respectivement* $f(x) > f(x)$*)*.

Voisinage : Une solution voisine d'une solution x est définit comme une légère modification de la solution x. Cette modification est appelée aussi mouvement. Le voisinage de la solution x est l'ensemble des solutions voisines de x.

Optimum global et local : Un optimum global est la valeur de la fonction objectif de la solution globale. On peut avoir plusieurs solutions globales mais un seul optimum global. Un optimum local reflète la meilleure solution dans un entourage voisin. Dans la figure 1.1, on remarque qu'il existe deux solutions différentes $x1$ et $x2$ qui reflètent un seul optimum global $f1 = f2$. La solution x3 représente une solution locale reflétant un optimum local. [30]

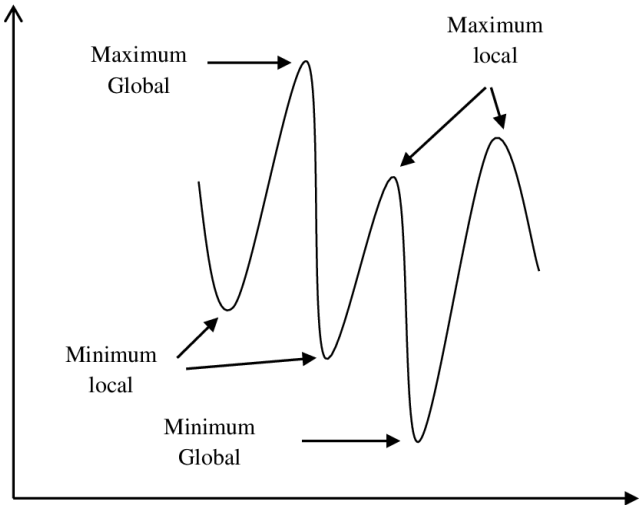

Figure 1 : Optimum local et global

3. La classification des problèmes d'optimisation

Il existe plusieurs familles d'optimisation. Il est important de bien identifier à quelle catégorie appartient le problème pour pouvoir choisir la conduite de la résolution. Les problèmes d'optimisation diffèrent selon l'espace de recherche, les contraintes, les objectifs, etc.

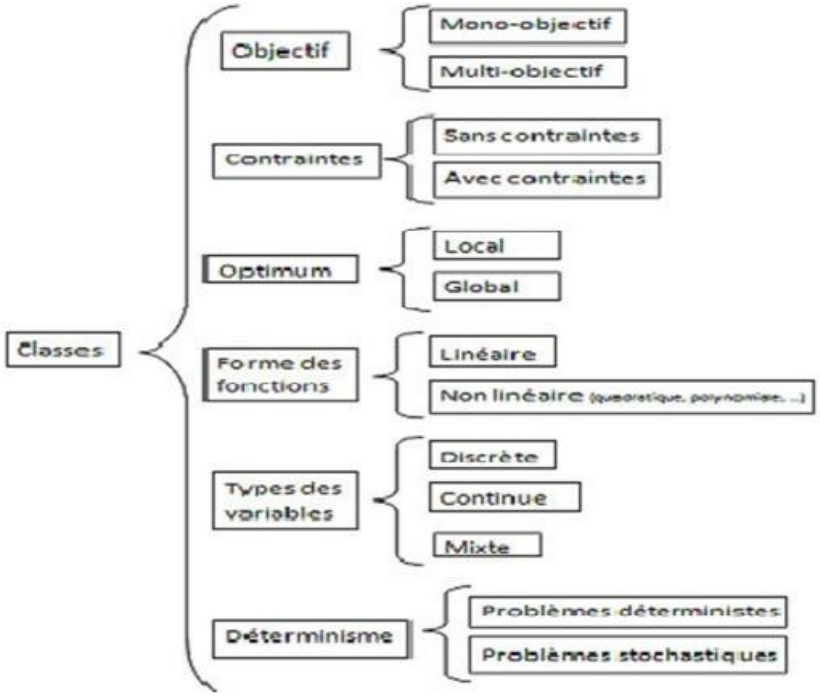

Figure 2 : Classification des problèmes d'optimisation

3.1. Les problèmes d'optimisation mono-objective ou multi-objective

Les problèmes mono-objectifs sont définis par une unique fonction objectif L'optimisation multi objectifs permet de chercher les valeurs des variables d'un problème qui maximisent ou minimisent un ou plusieurs fonctions objectif. Il peut s'agir par exemple de minimiser un coût de production, de rationaliser l'utilisation de ressources, d'améliorer les performances énergétiques d'un procédé industriel, etc. Elle procède donc par la définition au préalable des critères de qualité de la solution du problème, puis l'algorithme d'optimisation va résoudre le problème en cherchant les meilleures solutions en fonction de ces critères. Ainsi, la formulation du problème d'optimisation comporte les étapes suivantes :

- Exprimer les critères (ou fonctions) objectif d'optimalité
- Choisir les paramètres (ou variables) d'optimisation

- Définir un espace admissible pour les variables d'optimisation
- Définir les contraintes associées (impératives ou indicatives)

Donc l'optimisation multi objectifs permet de modéliser des problèmes réels faisant intervenir de nombreux critères (souvent conflictuels) et contraintes. Dans ce contexte, la solution optimale recherchée n'est plus un simple point, mais un ensemble de bons compromis satisfaisant toutes les contraintes [39]. Un problème d'optimisation multi objectif sous contraintes peut être défini comme suit :

$$C_l(\underset{X}{\rightarrow})l= \begin{cases} Min|Max\ fi\ \left(\underset{X}{\rightarrow}\right) i = 1,\dots,k \\ \\ 1,\dots,m \end{cases}$$
$$X \epsilon IR$$

Où « n » représente le nombre de variables, « x » un vecteur de décision, « k » le nombre d'objectifs (critères) et « m » le nombre de contraintes du problème. Pour les problèmes d'optimisation multi objectifs sous contraintes, il faut satisfaire un ensemble de contraintes tout en optimisant plusieurs fonctions objectives. En conséquence, même avec un petit nombre de variables et d'objectifs, le problème ne peut pas être traité simplement avec un algorithme classique de découpe. En effet, l'optimisation d'un problème multi objectifs est souvent plus difficile que l'optimisation des problèmes mono objectifs car la solution optimale ne se réduit plus à un seul point, mais à un ensemble de points non dominés [39].

3.2. Problème multi objectifs

Définition : Un problème multi objectifs ou multicritère peut être défini comme un problème dont on recherche l'action qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectives.

Les problèmes d'optimisation ont en général plusieurs solutions car la définition d'un optimum ne peut pas être établie dans les problèmes multi objectifs [1].

Une action (ou un vecteur de décisions) sera notée

$x = (x_1, x_2, \dots, x_n)$ Avec $x_i$les variables du problème et n le nombre de variables.

Les contraintes seront notées : $g_i(x)$avec $i = 1, \dots, m$ avec m le nombre de contraintes.

Le vecteur de fonction objectif sera noté $f(x) = (f_1(x), f_2(x), \dots, f_k(x))$avec f_i les objectifs ou critères de décision et « k » le nombre d'objectifs.

Un problème d'optimisation recherche l'action $x*$telle que les contraintes $g_i(x*)$ soient satisfaites pour $i = 1, \dots, m$ et qui optimise la fonction f:

$$f(x*) = \big(f_1(x*), f_2(x*), \dots, f_k(x*)\big).$$

L'union des domaines de définition de chaque variable et les contraintes forment un ensemble « E » que nous appelons l'ensemble des actions réalisables. Nous appellerons « F » l'ensemble des objectifs réalisables comme illustre la figure 3.

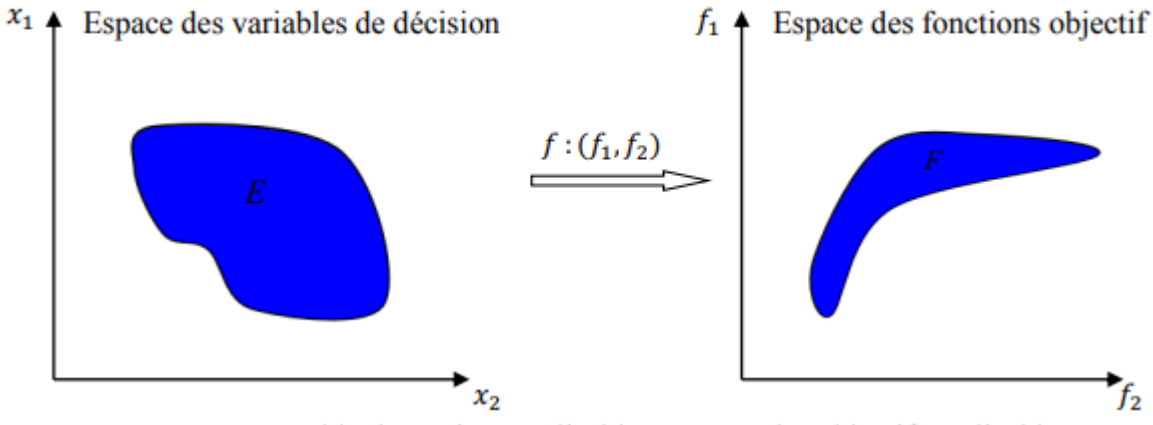

Figure 3 : L'ensemble des actions réalisables « E » et des objectifs réalisables « F »

Après avoir trouvé les solutions des problèmes multi objectifs que d'autres difficultés surviennent : il faut sélectionner une solution dans cet ensemble. La solution qui sera choisie par l'utilisateur va refléter les compromis opérés par le décideur vis-à-vis des différentes fonctions objectif.

3.2.1. La multiplicité des solutions

Lorsque l'on cherche la meilleure solution pour un problème d'optimisation multi-objectifs donné, on veut généralement trouver une solution, et une seule solution. En fait, cette situation est rarement rencontrée. La plupart du temps, il existe plusieurs solutions car certains objectifs sont contradictoires.

Ainsi, lorsque nous résolvons un problème d'optimisation multi-objectifs, nous obtenons de nombreuses solutions. Comme on peut s'y attendre, ces solutions ne seront pas optimales car elles ne minimiseront pas tous les objectifs du problème.

Une notion intéressante est celle de compromis, qui permet de définir la solution obtenue. En effet, lorsque nous résolvons un problème, la solution que nous obtenons est une

solution de compromis. Ils minimisent de nombreux objectifs tout en dégradant les performances des autres.

3.2.2. La difficulté principale d'un problème multi objectifs

La principale difficulté des problèmes multi-objectifs est qu'il n'y a pas de définition de la solution optimale. Les décideurs peuvent simplement exprimer le fait qu'une solution est meilleure qu'une autre, mais qu'aucune solution n'est meilleure que toutes les autres.

Par conséquent, la résolution de problèmes multi-objectifs ne réside pas dans la recherche de la solution optimale, mais dans un ensemble de solutions satisfaisantes pour lesquelles nous ne pouvons pas classer les opérations. Par conséquent, les méthodes de résolution de problèmes multi-objectifs sont des méthodes d'aide à la décision car le choix final sera laissé au décideur.

3.2.3. Résolution d'un problème multi objectifs

Deux types de comportement existent :

Le premier comportement : est de ramener un problème multi objectifs à un problème simple objectif.

Le second comportement : est de tenter d'apporter des réponses au problème en prenant en compte l'ensemble des critères.

La différence entre ces deux communautés s'exprime dans la figure 1.3.

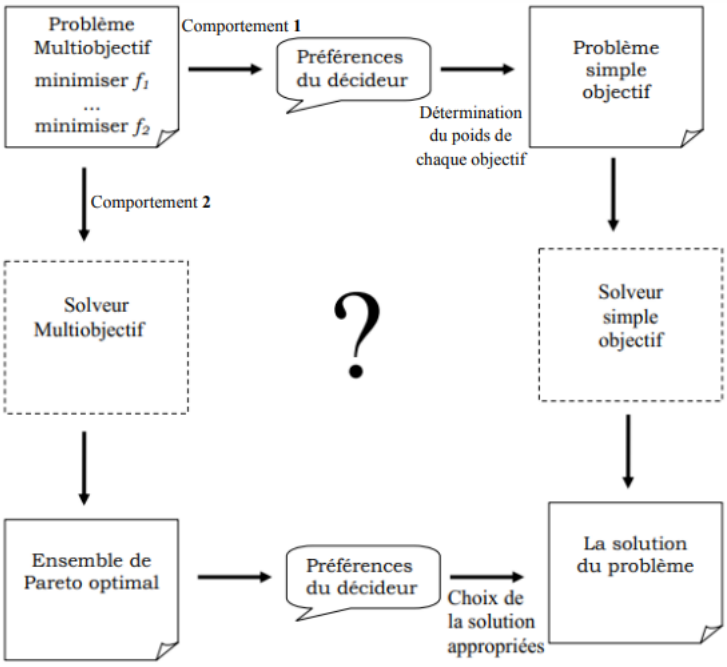

Figure 4 : Quel mode de résolution choisir ?

Soit le décideur intervient dès le début de la définition du problème, en exprimant ses préférences, afin de transformer un problème multi objectifs en un problème simple objectif (Comportement 1).

Soit le décideur effectue son choix dans l'ensemble des solutions proposées parle solveur multi objectif (Comportement 2). La principale qualité d'un solveur multi objectif est donc de rendre les décisions plus faciles en proposant un sous-ensemble représentatif de F (l'ensemble des objectifs réalisables). [34]

4. Domaines d'applications d'Optimisation Combinatoire

L'optimisation combinatoire intervient sur toute situation nécessitant une planification ou une organisation efficace. Elle est liée à la minimisation de risque, du temps ou du coût ainsi qu'à la maximisation de la qualité, du profit ou de l'efficacité.

L'industrie et l'ingénierie sont les domaines qui profitent le plus des études de l'optimisation combinatoire. On cite quelques exemples :

- La conception de réseaux (électriques, téléphoniques ..)
- La planifications des tâches des employés d'une entreprise.
- Le calcul d'itinéraire pour les applications de géolocalisation.
- La planification du trafic aérien ou routier.
- La gestion de portefeuilles.
- L'allocation de ressources matérielles et humaines.
- L'ordonnancement et la planification de la production.
- Maximisation du profit et minimisation du coût.[30]

5. Conclusion

Comme nous avons pu le voir tout au long de ce chapitre, l'optimisation combinatoire multiobjectifs n'est pas une tâche facile. L'optimisation multiobjectifs offre à l'utilisateur davantage de degrés de liberté pour modéliser son problème, Comme nous l'avons défini le concept d'un probleme multi objectif .

Nous avons également mentionné certains domaines de l'optimisation intégrative multi-objectifs, dont nous parllons avec détail dans le chapitre suivant.

Chapitre 02

Description des problèmes d’ordonnancement

Chapitre 02 Description des problèmes d'ordonnancement

1. Introduction

Les méthodes de résolution des problèmes d'ordonnancement puisent dans toutes les techniques de l'optimisation combinatoire (programmation mathématique, programmation dynamique, procédures par séparation et évaluation, théorie des graphes...).

Ces méthodes garantissent en général l'optimalité de la solution fournie. Mais les algorithmes dont la complexité n'est pas polynomiale ne peuvent pas être utilisés pour des problèmes de grande taille, d'où la nécessité de construire des méthodes de résolution approchée, efficaces pour ces problèmes souvent NP-difficiles [14].

L'objectif de ce deuxième chapitre est de présenter une typologie générale de construction de méthodes de résolution qui regroupe à la fois les méthodes exactes et les méthodes approchées.

L'analyse d'un problème particulier permet d'obtenir des propriétés sur la structure des solutions optimales, propriétés à partir desquelles on peut ou bien caractériser les solutions optimales ou bien réduire l'espace des solutions à explorer (notion de sous-ensemble dominant qui contient au moins une solution optimale).

Ces résultats permettent soit de prouver l'optimalité des solutions obtenues par un algorithme soit de rendre plus efficace les méthodes présentées ci-dessous.

2. Les problèmes d'ordonnancements

2.1. Définition

L'ordonnancement est une branche de la recherche opérationnelle et de la gestion de la production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison. Les problèmes d'ordonnancement sont présents dans tous les secteurs d'activités de l'économie, depuis l'industrie manufacturière jusqu'à l'informatique [29].

Un problème d'ordonnancement peut être défini comme étant le "processus d'organisation, choix et programmation dans le temps de l'usage des ressources pour traiter toutes les activités nécessaires afin d'aboutir à un objectif donné tout en satisfaisant un ensemble de contraintes sur les activités et les ressources" (Morton and Pentico, 1993).

Alors ce problème consiste à affecter des tâches à des ressources à des instants donnés pour répondre au mieux aux besoins exprimés par un client, au meilleur coût et dans les meilleurs délais, tout en tenant compte des contraintes.

3. Formulation d'un problème d'ordonnancement

Dans la définition du problème d'ordonnancement, quatre éléments fondamentaux interviennent : les tâches, les ressources, les contraintes et les objectifs. Alors, la formulation et la description de ce problème se fait par la détermination de ces quatre éléments dits "de base" [19].

3.1. Les tâches

Une tâche est un travail mobilisant des ressources et réalisant un progrès significatif dans l'état d'avancement du projet compte tenu du niveau de détail retenu dans l'analyse du problème [38].

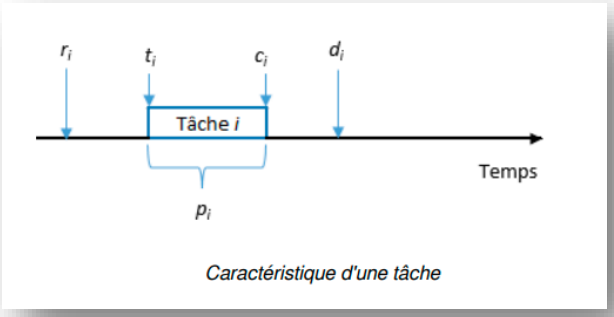

Figure 5 : Une représentation de la tâche en désignant ses principales caractéristiques.

On distingue deux types de tâches :

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

On note en général $N = \{J1, J2, \dots, Jn\}$ l'ensemble des tâches, chaque tâche est caractérisée par :

- $(\boldsymbol{P_{ij}})$ **:** La durée opératoire de la tâche i sur la machine j.
- $(\boldsymbol{r_i})$ **:** La date de disponibilité de la tâche i.
- $(\boldsymbol{S_i})$ **:** La date de début d'exécution de la tâche i.
- $(\boldsymbol{C_i})$ **:** La date de fin d'exécution de la tâche.
- $(\boldsymbol{d_i})$ **:** La date d'achèvement souhaitée de la tâche i.

- $(\boldsymbol{W_i})$ **:** Le facteur de priorité ou poids de la tâche i.
- $\boldsymbol{L_i = C_i - d_i}$ **:** Le retard algébrique de la tâche i.
- $\boldsymbol{T_i = \max(C_i - d_i, 0)}$ **:** Le retard vrai de la tâche i.
- $\boldsymbol{U_i = 1, si\ T_i > 0, U_i = 0, sinon}$ **:** L'indicateur de retard de la tâche i.

3.2. Les Ressources

1. Selon leurs disponibilités au cours du temps, on trouve :

- **Les ressources renouvelables**, comme c'est le cas pour les machines, personnels, équipements, etc. La ressource est dite renouvelable si après avoir été utilisé par une ou plusieurs opérations, elle est à nouveau disponible en même quantité.
- **Les ressources non-renouvelables**, souvent appelées ressources consommables ou bien ressources financières. On dit que la ressource est non-renouvelables si ça disponibilité décroît après avoir été allouée à une opération. C'est le cas pour la matière première, budget.
- **Les ressources doublement contraintes**, ces ressources combinent les contraintes liées aux deux catégories précédentes. Leur utilisation instantanées et leur consommation globale sont toutes les deux limitées. C'est le cas des ressources d'énergie (pétrole, électricité, etc.).

2. Selon leurs capacités, on trouve :

- **Les ressources disjonctives**, il s'agit des ressources qui ne peuvent exécuter qu'une seule opération à la fois c'est le cas par exemple de machine-outil ou robot manipulateur.
- **Les ressources cumulatives,** il s'agit des ressources qui peuvent être utilisé par plusieurs opérations simultanément (équipes d'ouvriers, poste de travail, etc.).

3.3. Les contraintes

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre conjointement les variables de décision. En d'autres termes, les contraintes représentent les conditions à respecter, lors de la construction de l'ordonnancement pour qu'il soit réalisable. Plus les contraintes sont nombreuses, plus le problème d'ordonnancement devient plus difficile.

Dans les problèmes d'ordonnancement, deux types de contraintes sont distingués : les contraintes temporelles et les contraintes de ressources. [13], [8] :

➢ **Les contraintes temporelles** décrivent les interdépendances temporelles entre les opérations, elles intègrent :

- Les contraintes de dates limites : c'est des contraintes imposées individuellement à chaque opération *i*. Par exemple, l'opération i ne peut débuter avant une certaine date (livraison de matière première, conditions climatiques, etc.) ou encore l'opération i ne peut commencer avant sa date de disponibilité r_i et doit être terminée avant une date d'échéanced_j.
- Les contraintes d'antériorité : c'est des contraintes qui relient la date de début ou la date de fin de deux opérations par une relation linéaire. De manière générale, c'est des contraintes qui décrivent le positionnement relatif de certaines opérations par rapport à d'autres, c'est le cas par exemple des gammes opératoires dans les d'ateliers de production.

➢ **Les contraintes de ressources** traduisent l'utilisation et la disponibilité des ressources utilisées par les opérations. Deux types de contraintes liées à la nature cumulative ou disjonctive des ressources peuvent alors être distingués.

- Les contraintes disjonctives : ces contraintes imposent la non-réalisation simultanée de deux opérations sur la même ressource.
- Les contraintes cumulatives : ces contraintes expriment le fait qu'à tout instant, le total des ressources utilisées ne dépasse pas une certaine limite fixée.

3.4. Les objectifs

Les objectifs dits aussi les critères d'évaluation sont les indicateurs de performance sur lesquels se base le choix d'un ordonnancement satisfaisant. En ordonnancement, les critères à optimiser consistent à minimiser ou maximiser une fonction objectif. Cette fonction objectif est généralement liée aux temps, aux ressources ou bien aux coûts.

➢ **Les objectifs liés au temps :** c'est la catégorie des objectifs les plus étudiés en optimisation, parmi les plus classiques, nous pouvons citer :

- Le temps total d'exécution : connue sous le nom de makespan et défini par $\boldsymbol{C_{max}} = \boldsymbol{\max_{i \epsilon E} C_i}$(*E* désigne l'ensemble d'opération à ordonnancer) qui représente la date de fin du job le plus tardif. La minimisation de ce critère est souvent rencontrée puisque ça conduit à une meilleure utilisation de ressources (productivité).
- La somme des dates d'achèvement des jobs : on lui réfère aussi comme total flow time ou total complétion time définie par$\sum_{i \epsilon E} \boldsymbol{C_i}$.

- Le retard algébrique maximal : connue sous le nom de lateness et défini par $\boldsymbol{L_{max} = \max_{i \epsilon E} L_i}$ tel que $\boldsymbol{L_i = C_i - d_i}$ est le retard algébrique pour chaque opération. L'objectif de la minimisation du retard algébrique maximal consiste donc à minimiser la quantité $\boldsymbol{L_{max}}$

➢ **les objectifs liés aux ressources :** les objectifs de ce type correspondent par exemple à :

- Maximisation de la charge d'une ressource e.g. maximisé l'utilisation de la machine ayant un bon rendement ou la machine la moins gourmand en énergie, etc.
- Minimisation de nombre de ressources nécessaires pour réaliser un ensemble d'opérations.

➢ **Les objectifs liés au coût :** ces objectifs consistent généralement à minimiser les coûts de lancement, de production, de stockage, ou de transport.

4. Les typologies des problèmes d'ordonnancement

Dans un problème d'atelier, une pièce doit être usinée par différentes opérations qui peuvent être liées exclusivement par des contraintes de précédence. Chaque opération utilise un certain nombre de ressources afin d'être traitée. Les opérations d'un atelier sont regroupées en des entités appelées gammes opératoires permettant d'apporter des modifications sur un produit.

Plus précisément, un atelier est caractérisé par les opérations qu'il contient et par son type. Une classification des problèmes d'ordonnancement est donnée dans (Figure 6**)**.

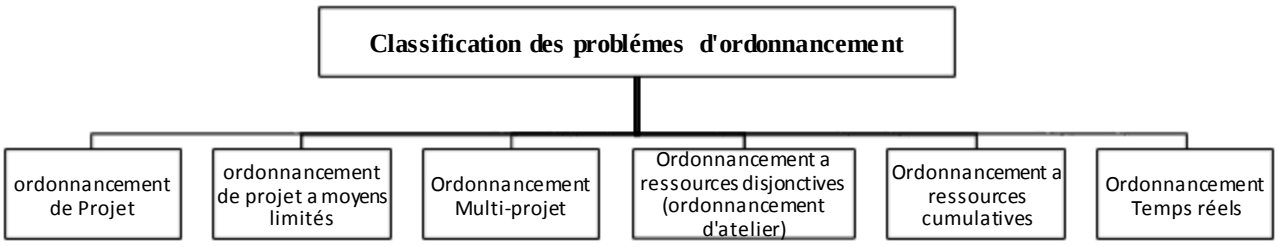

Figure 6 : Typologie des problèmes d'ordonnancement

Deux grandes familles de problèmes d'ordonnancement se présentent. La première famille regroupe les problèmes pour lesquels chaque job nécessite une seule opération. La deuxième regroupe ceux dont les jobs nécessitent plusieurs opérations.

5. Les différentes entités d'un problème d'ordonnancement

La notation ***α/ß/γ*** Elle permet en effet de caractériser un problème d'ordonnancement de manière précise.

- **Le champ *α* :** la structure de problème et se décompose en deux sous champ *α1 et α2, et α1* indique la nature de problème (job shop, flow-shop, etc.), *α2* précisant le nombre de machines ou de pools.
- **b) Le champ *ß* :** les types de contraintes prises en compte.
- **c) le champ *γ* :** fonction objectif ou la description des critères.

6. 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 [25]. Nous exposerons dans la partie qui suit, les grands principes de la théorie de la complexité des problèmes d'ordonnancement.

6.1. La classe P et NP

- Un problème est dit polynomial s'il existe un algorithme de complexité polynomiale permettant de répondre à la question posée dans ce problème, quelle que soit la donnée de celui-ci .La classe P est l'ensemble de tous les problèmes de reconnaissance polynomiaux **[2].** Pour le reste de la classe NP, on n'est pas sûr qu'il n'existe pas un algorithme polynomial pour résoudre chacun de ses problèmes. Ainsi, on sait que P est inclue dans NP mais on n'a pas pu prouver que P n'est pas NP.
- La classe NP (Non déterministe Polynomial) est celle des problèmes d'existence dont une proposition de solution est Oui et qui est vérifiable polynomialement. Parmi les problèmes décidables, les plus simples à résoudre sont regroupés dans la classe NP.

6.2. La classe NP-Complet et NP-Difficile

- La classe NP-Complet regroupe les problèmes les plus difficiles de la classe NP. Elle contient les problèmes de la classe NP tels que n'importe quel problème de la classe NP leur est polynomialement réductible. Entre eux, les problèmes de la classe NP-Complet sont aussi difficiles.

- La classe NP-Difficile regroupe les problèmes (pas forcément dans la classe NP) tels que n'importe quel problème de la classe NP leur est polynomialement réductible.

Pour les problèmes d'ordonnancement à une machine, certains peuvent être résolus par un algorithme polynomial, certains autres sont démontrés NP-difficiles. Certains ne sont ni démontrés NP-difficiles, ni polynomialement réservables. Ils restent donc ouverts. Et pour les problèmes d'ateliers, la plupart de ces problèmes ont été démontrés NP-difficiles.

7. Les méthodes de résolution

Nous avons vu précédemment qu'il existe des problèmes d'ordonnancement de complexités différentes. Les problèmes appartenant à la classe P ont des algorithmes polynomiaux permettant de les résoudre .Pour les problèmes appartenant à la classe NP, l'existence d'algorithmes polynomiaux semble peu réaliste. Ainsi, différentes méthodes de résolution sont largement utilisées pour appréhender les problèmes NP-difficiles.

Dans cette partie, nous exposons en générale les méthodes de résolution des problèmes d'optimisation qui sont classées en deux grandes catégories : les méthodes exactes et les méthodes approchées. On explique brièvement les méthodes les plus connues dans chaque catégorie. Elles sont schématisées comme suit :

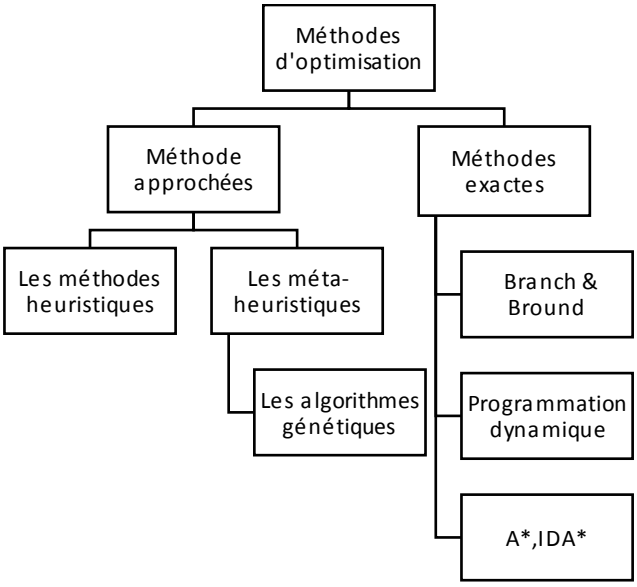

Figure 7 : un schéma montrant les différentes méthodes de résolution en optimisation combinatoire.

7.1. Les méthodes exactes

Les méthodes exactes utilisent surtout deux approches de résolution très connues :

La programmation dynamique et les procédures par séparation et évaluation. Ces méthodes sont souvent utilisées pour résoudre les problèmes combinatoires de manière exacte, en ordonnancement tout particulièrement. Ce sont des méthodes d'énumération implicite :

L'énumération explicite construit toutes les solutions réalisables et retient une parmi les meilleures. L'énumération implicite consiste à explorer l'ensemble de toutes les solutions réalisables en éliminant des sous-ensembles de solutions moins intéressants sans avoir à les construire.

Nous pouvons citer trois approches particulièrement célèbres : La programmation dynamique introduite par Bellman dans les années 50 [19], la méthode par séparation et évaluation (Branch and Bound en anglais : notée B&B) et l'algorithme de retour arrière (Backtracking).

7.2. Les méthodes approchées

Une méthode approchée est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objectif en un temps raisonnable, mais sans garantie d'optimalité. L'avantage principal de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles, D'un autre côté les algorithmes d'optimisation tels que les algorithmes de recuit simulé, les algorithmes tabous et lesalgorithmes génétiques ont démontré leurs robustesses et efficacités face à plusieurs problèmes d'optimisation combinatoires.

Les méthodes approchées englobent deux classes : **les heuristiques** et **les méta-heuristiques**. La particularité qui différencie les méthodes méta-heuristiques des méthodes heuristiques c'est que les méta-heuristiques sont applicables sur de nombreux problèmes. Tandis que, les heuristiques sont spécifiques à un problème donné.

7.2.1. Les heuristiques

Les méthodes heuristiques sont des méthodes spécifiques à un problème particulier. Elles nécessitent des connaissances du domaine du problème traité. En fait, ce sont des règles empiriques qui se basent sur l'expérience et les résultats acquis afin d'améliorer les

recherches futures. Plusieurs définitions des heuristiques ont été proposées par plusieurs chercheurs dans la littérature, parmi lesquelles :

- Définition 1 : « Une heuristique (règle heuristique, méthode heuristique) est une règle d'estimation, une stratégie, une astuce, une simplification, ou tout autre type de dispositif qui limite considérablement la recherche de solutions dans des espaces problématiques importants. Les heuristiques ne garantissent pas des solutions optimales. En fait, elles ne garantissent pas une solution du tout. Tout ce qui peut être dit d'une heuristique utile, c'est qu'elle propose des solutions qui sont assez bonnes la plupart du temps. » [12].
- Définition 2 : « Une méthode heuristique (ou simplement une heuristique) est une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillible de ce qui est la meilleure chose à faire. » [11].
- Définition 3 : « Les heuristiques sont des ensembles de règles empiriques ou des stratégies qui fonctionnent, en effet, comme des règles d'estimation. » [31].

➢ **Exemples d'heuristiques**

- **FIFO** (First In First Out) : la première tâche qui vient est la première tâcheordonnancée.
- **SPT** (Shortest Processing Time) : la tâche ayant le temps opératoire le plus court esttraitée en premier lieu.
- **LPT** (Longest Processing Time) : la tâche ayant le temps opératoire le plus importantest ordonnancée en premier lieu.
- **EDD** (Earliest Due Date) : cet algorithme choisit parmi les tâches exécutables celledont le délai est échu le plus tôt. Si aucune tâche n'est disponible, alors un temps libreest généré.
- **SRPT** (Shortest Remaining Processing Time) : cette règle, servant à lancer la tâcheayant la plus courte durée de travail restant à exécuter, est très utilisée pourminimiserles encours et dans le cas des problèmes d'ordonnancement préemptifs.
- **ST** (Slack Time) : à chaque point de décision, l'opération ayant la plus petite margetemporelle est prioritaire. Faute de disponibilité des ressources de production, cettemarge peut devenir négative.

7.2.2. Les Méta-heuristiques

Le mot méta-heuristique est dérivé de la composition de deux mots grecs :

- **heuristique** qui vient du verbe heuriskein et qui signifie 'trouver'.
- **méta** qui est un suffixe signifiant 'au –delà', 'dans un niveau supérieur'.

Les méta-heuristiques d'optimisation sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Elles sont apparues à partir des années 80, dans le but de résoudre au mieux des problèmes d'optimisation. Les méta-heuristiques s'efforcent de résoudre tout type de problème d'optimisation. Elles sont caractérisées par leur caractère stochastique, ainsi que par leur origine discrète.

Elles sont inspirées par des analogies avec la physique (recuit simulé, recuit micro canonique), avec la biologie (algorithmes évolutionnaires) ou encore l'éthologie (colonies de fourmis, essaims particulaires). Cependant, elles ont l'inconvénient d'avoir plusieurs paramètres à régler. Il est à souligner que les méta-heuristiques se prêtent à toutes sortes d'extensions, notamment en optimisation mono-objectif et multi-objectif.

Les méta-heuristiques utilisent des recherches stratégiques afin d'explorer plus efficacement l'espace de recherche, et souvent se focalisent sur les régions prometteuses. Ces méthodes commencent par un ensemble de solutions initiales ou une population initiale, et après, elles examinent étape par étape une séquence de solutions pour atteindre, oude s'approcher de la solution optimale du problème. Les méta-heuristiques ont plusieurs avantages par rapport aux algorithmes traditionnels. Les deux avantages les plus importants sont la simplicité et la flexibilité. Les méta-heuristiques sont souvent simples à implémenter, pourtant, elles sont capables de résoudre des problèmes complexes avec la capacité de s'adapter à plusieurs problèmes d'optimisation du monde réel, à partir du domaine de la recherche opérationnelle, d'ingénierie vers l'intelligence artificielle.

De nos jours les gestionnaires et les décideurs sont confrontés quotidiennement à des problèmes de complexité grandissante, qui surgissent dans des secteurs très divers. Le problème à résoudre peut souvent s'exprimer sous la forme générale d'un problème d'optimisation, dans lequel on définit une ou plusieurs fonctions objectif que l'on cherche à minimiser ou à maximiser par rapport à tous les paramètres concernés [9].

Les méta-heuristiques sont souvent inspirées de processus naturels. Il existe un grand nombre de méta-heuristiques différentes citons :

7.2.2.1. Le recuit simulé (SA : Simulated Annealing)

La méthode du recuit simulé a été introduite en 1983 par Kirkpatrick et al. Cette méthode originale est basée sur les travaux bien antérieurs de Metropolis et al. Cette méthode que l'on pourrait considérer comme la première méta-heuristique "grand public" a reçu l'attention de nombreux travaux et principalement de nombreuses applications.

Le principe de fonctionnement s'inspire d'un processus d'amélioration de la qualité d'un métal solide par recherche d'un état d'énergie minimum correspondant à une structure stable de ce métal. L'état optimal correspondrait à une structure moléculaire régulière parfaite. En partant d'une température élevée où le métal serait liquide, on refroidit le métal progressivement en tentant de trouver le meilleur équilibre thermodynamique. Chaque niveau de température est maintenu jusqu'à obtention d'un équilibre. Dans ces phases de température constante, on peut passer par des états intermédiaires du métal non satisfaisants, mais conduisant à la longue à des états meilleurs.

Dans la méthode SA, les mécanismes d'intensification et de diversification sont contrôlés par la température. La température t ne fait que décroitre pendant le processus de recherche, de sorte que la recherche tend à s'intensifier vers la fin de l'algorithme. L'idée est de diminuer petit à petit la chance d'accepter des solutions qui dégradent la fonction objectif,

L'algorithme (Algorithme 1) présente les principales caractéristiques d'un recuit simulé.

1: *initialise* : find an initial solution x, fix an annealing schedule $\mathcal{T}$, set initial temperature t
2: **repeat**
3: *neighbourhood search* : find a solution $x' \in \mathcal{N}(x)$
4: determine $\Delta C = f(x') - f(x)$
5: draw $p \sim \mathcal{U}(0,1)$
6: **if** $\Delta C < 0$ or $e^{-\Delta C/t} > p$ **then**
7: $x' \leftarrow x$
8: **end if**
9: update temperature t according to $\mathcal{T}$
10: **until** stopping criterion satisfied

Algorithme 1 : L'algorithme du recuit simulé

7.2.2.2. La recherche tabou (TS : Tabu Search)

La recherche tabou est une méta-heuristique originalement développée par Glover, 1986 et indépendamment par Hansen, 1986 [13]. Elle est basée sur des idées simples, mais elle est néanmoins très efficace. Cette méthode combine une procédure de recherche locale avec un certain nombre de règles et de mécanismes permettant à celle-ci de surmonter l'obstacle des optima locaux, tout en évitant de cycler. Elle a été appliquée avec succès pour résoudre de nombreux problèmes difficiles d'optimisation combinatoire : problèmes de routage de véhicule, problèmes d'ordonnancement, problèmes de coloration de graphes, etc.

Dans une première phase, la méthode de recherche tabou peut être vue comme une généralisation des méthodes d'amélioration locales. En effet, en partant d'une solution quelconque x appartenant à l'ensemble de solutions S, on se déplace vers une solution x' située dans le voisinage (x). Donc l'algorithme explore itérativement l'espace de solutions S.

1: *initialise* : find an initial solution x
2: **repeat**
3: *neighbourhood search* : find a solution $x' \in N^*(x)$
4: *update memory* : tabu list, frequency-based memory, aspiration level, ...
5: *move* $x \leftarrow x'$
6: **until** stopping criterion satisfied

Algorithme 2 : L'algorithme de recherche tabou classique

Afin de choisir le meilleur voisin x' dans N (x), l'algorithme évalue la fonction objectif f en chaque point x', et retient le voisin qui améliore la valeur dela fonction objectif, ou au pire celui qui la dégrade le moins.

7.2.2.3. Les algorithmes génétiques (GA : Genetic Algorithm)

L'algorithme génétique est choisis pour résoudre notre problème qu'il est adaptable à plusieurs types des problèmes, Robustes, Faciles à implémenter, Faciles à hybrider et Faciles à paralléliser.

Proposé dans les années 1975 par Holland [21], les algorithmes génétiques doivent leur popularité à Goldberg [10]. Avant la parution de son livre qui est une des références les plus citées dans le domaine de l'informatique, on a pu voir un certain nombre d'autres présentations, citons Goldberg [15], Holland[20], Schwefel [18]. Le sujet connaît une très grande popularité. Il existe aujourd'hui plusieurs milliers de références sur le sujet et le

nombre de conférences dédiées au domaine (que ce soit sur les techniques elles-mêmes ou sur les applications) ne fait qu'augmenter.

De manière générale, les algorithmes génétiques utilisent un même principe. Une population d'individus (correspondants à des solutions) évolue en même temps comme dans l'évolution naturelle en biologie. Pour chacun des individus, on mesure sa faculté d'adaptation au milieu extérieur par le fitness.

Les algorithmes génétiques s'appuient alors sur trois fonctionnalités :

- **La Sélection :**

Pour déterminer quels individus sont plus enclins à se reproduire, une sélection est opérée. Il existe plusieurs techniques de sélection, les principales utilisées sont la sélection par tirage à la roulette (roulette-wheel selection), la sélection par tournoi (tournament selection), la sélection par rang (ranking selection), etc [9].

- **Le Croisement :**

L'opérateur de croisement combine les caractéristiques d'un ensemble d'individus parents (généralement deux) préalablement sélectionnés, et génère de nouveaux individus enfants.

- **La Mutation et le remplacement :**

Les descendants sont mutés, c'est-à-dire que l'on modifie aléatoirement une partie de leur génotype, selon l'opérateur de mutation .Le remplacement (ou sélection des survivants), comme son nom l'indique, remplace certains des parents par certains des descendants. Le plus simple est de prendre les meilleurs individus de la population, en fonction de leurs performances respectives, afin de former une nouvelle population (typiquement de la même taille qu'au début de l'itération).

La représentation des solutions (le codage) est un point critique de la réussite d'un algorithme génétique. Il faut bien sûr qu'il s'adapte le mieux possible au problème et à l'évaluation d'une solution. Le codage phénotypique ou codage direct correspond en général à une représentation de la solution très proche de la réalité. L'évaluation d'une solution représentée ainsi est en général immédiate.

8. Conclusion

Nous avons présenté, dans ce chapitre, les méthodes de résolution utilisée dans la littérature.

Ces méthodes sont regroupées en deux classes : les méthodes exactes et les méthodes approchées et nous avons vu aussi 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[22], plusieurs études ont sélectionné les AG comme l'outil le plus performant pour résoudre des problèmes d'ordonnancement de différents types[16],[3],[40],[6].Aux chapitres suivants nous allons justifier quant à nous ce choix tout en commentant les résultats obtenus lors de la simulation .

Chapitre 03

Concepts de base des Algorithmes Génétiques

Chapitre 3 Concept de base des Algorithmes Génétiques

1. Introduction

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnistes. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable. Les algorithmes génétiques utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles au problème donné. La solution est approchée par « bonds » successifs, comme dans une procédure de séparation et évaluation (Branch & Bound), à ceci près que ce sont des formules qui sont recherchées et non plus directement des valeurs.

Dans ce chapitre, nous examinerons les concepts de base des algorithmes génétiques et l'application à notre problème.

2. Les algorithmes génétiques

Les algorithmes génétiques (AGs) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (chromosomes) initiales arbitrairement choisies. On évalue leur performance (fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante. Les AGs ont été initialement développés par John Holland (1975). C'est au livre de Goldberg (1989) que nous devons leur popularisation. Leurs champs d'application sont très vastes. Outre l'économie, ils sont utilisés pour l'optimisation de fonctions (De Jong (1980)), en finance (Pereira (2000)), en théorie du contrôle optimal (Krishnakumar et Goldberg (1992), Michalewicz, Janikow et Krawczyk (1992) et Marco et al. (1996)), ou encore en théorie des jeux répétés (Axelrod (1987)) et différentiels (Özyildirim (1996, 1997) et Özyildirim et Alemdar (1998)). La raison de ce grand nombre d'application est claire : simplicité et efficacité. Bien sûr d'autres techniques d'exploration stochastique existent, la plus connue étant le recuit simulé (simulated annealing – voir Davis (1987) pour une association des deux méthodes). Pour résumer, Lerman et Ngouenet (1995) distinguent 4 principaux points qui font la différence fondamentale entre ces algorithmes et les autres méthodes : [11], [12]

1. Les algorithmes génétiques utilisent un codage des paramètres, et non les paramètres eux-mêmes.
2. Les algorithmes génétiques travaillent sur une population de points, au lieu d'un point unique.
3. Les algorithmes génétiques n'utilisent que les valeurs de la fonction étudiée, pas sa dérivée, ou une autre connaissance auxiliaire.
4. Les algorithmes génétiques utilisent des règles de transition probabilistes, et non déterministes.

La simplicité de leurs mécanismes, la facilité de leur mise en application et leur efficacité même pour des problèmes complexes ont conduit à un nombre croissants de travaux. [31], [8].

3. Présentation des algorithmes génétiques (AGs)

Selon Lerman et Ngouenet [24] un algorithme génétique est défini par :

- **Individu/chromosome/séquence :** une solution potentielle du problème ;
- **Population :** un ensemble de chromosomes ou de points de l'espace de recherche
- **Environnement :** l'espace de recherche ;
- **Fonction de fitness :** la fonction - positive - que nous cherchons à maximiser.

Avant d'aller plus loin il nous faut définir quelques termes importants généralement définis sous l'hypothèse de codage binaire

•Définition 1(Séquence/Chromosome/Individu (Codage binaire)).

Nous appelons une séquence (chromosome, individu) A de longueur l(A) une suite **A = {a1, a2,..., al}** avec **∀i ∈ [1, l], ai ∈ V = {0,1}**.

Un chromosome est donc une suite de bits en codage binaire, appelé aussi chaîne binaire. Dans le cas d'un codage non binaire, tel que le codage réel, la suite **A** ne contient qu'un point, nous avons **A = {a}** avec **a ∈ℜ**. [37]

•Définition 2 (Fitness d'une séquence).

Nous appelons fitness d'une séquence toute valeur positive que nous noterons **f(A),** où f est typiquement appelée fonction de fitness. La fitness (efficacité) est donc donnée par une fonction à valeurs positives réelles.

Dans le cas d'un codage binaire, nous utiliserons souvent une fonction de décodage d qui permettra de passer d'une chaîne binaire à un chiffre à valeur réelle : **d : {0,1} l → ℜ** (où

l'est la longueur de la chaîne). La fonction de fitness est alors choisie telle qu'elle transforme cette valeur en valeur positive, soit f : **d ({0,1} l) →ℜ∗ +**. Le but d'un algorithme génétique est alors simplement de trouver la chaîne qui maximise cette fonction f. Bien évidemment, chaque problème particulier nécessitera ses propres fonctions d et f. [37]

Les AGs sont alors basés sur les phases suivantes :

1. **Initialisation** Une population initiale de N chromosomes est tirée aléatoirement.
2. **Évaluation** Chaque chromosome est décodé, puis évalué.
3. **Sélection** Création d'une nouvelle population de N chromosomes par l'utilisation d'une méthode de sélection appropriée.
4. **Reproduction**Possibilité de croisement et mutation au sein de la nouvelle population.
5. **Retour** à la phase d'évaluation jusqu'à l'arrêt de l'algorithme.

Ou alors sous une forme algorithmique :

Générer aléatoirement une population Pop de \Pop\ individus

TANT QUE Nombre de générations < Nombre de générations Maximum **FAIRE** **Évaluation :** Calculer la fitness de chaque individu x

Sélection : Établir de façon probabiliste les paires d'individus qui vont se reproduire en accordant une plus grande chance aux meilleurs individus

Croisement : Recombiner les paires sélectionnées en fonction de pc.

Mutation : Effectuer une mutation en fonction de pm

Remplacement : Les nouveaux individus produits représentent la nouvelle population

Algorithme 3 : Un algorithme génétique standard

3.1. Codage et population initiale

3.1.1. Codage binaire

Soit la fonction représentée par la Figure 8, la fonction quantifiée sur la même figure permet de discrétiser la fonction en échantillons quantifiés de niveaux horizontaux égaux. Chaque variable $F(x_i) = y_i$à une coordonnée quantifiée sur lafonction quantifiée qui correspond à une valeur élevée, moyenne, ou basse. Pour cet exemple, on prend 4 variables de $f(x)$: 0.55, 0.11, 0.95, et 0.63, leur coordonnées quantifiées sur la fonction quantifiée sont

enregistrées sur la Figure.9. En général les meilleures valeurs de ces cordonnées correspondent aux valeurs quantifiées moyennes [32]. 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.

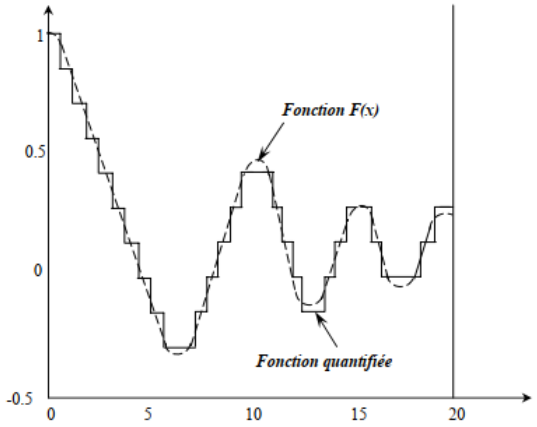

Figure 8 : Fonction F(x) et sa fonction quantifiée. [33]

		Valeurs des Variables			
		0.55	0.11	0.95	0.63
1.000 – 0.875	111			*		– 0.9375
0.875 – 0.750	110					– 0.8125
0.750 – 0.625	101				*	– 0.6875
0.625 – 0.500	100	*				– 0.5625
0.500 – 0.375	011					– 0.4375
0.375 – 0.250	010					– 0.3125
0.250 – 0.125	001					– 0.1875
0.125 – 0.000	000		*			– 0.0625
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 9 : Exemple de la représentation binaire des valeurs quantifiées [33]

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

- **Codage**

$$P_{norm} = \frac{p_n - p_{lo}}{p_{ni} - p_{lo}} \quad (3.1)$$

$$gene[m] = round\{p_{norm} - 2^{-m} - \sum_{p=1}^{m-1} gene[2p]^{-p}\} \quad (3.2)$$

- **Décodage**

$$P_{norm} \sum_{p=1}^{m-1} gene[m] 2^{-m} + 2^{-(m+1)} \quad (3.3)$$

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

Pour chaque cas :

- $\boldsymbol{p_{norm}}$ **:** Variable normalisée, $0 \leq P_{norm} \leq 1$.
- $\boldsymbol{p_{lo}}$**:** La plus petite valeur
- $\boldsymbol{p_{hi}}$**:** La plus grande valeur
- $\boldsymbol{gene[m]}$**:** Version binaire de p_n
- $\boldsymbol{round\{\ \}}$**:** Arrondissage vers un entier proche
- $\boldsymbol{p_{quant}}$**:** Version quantifiée de p_{norm}
- $\boldsymbol{q_n}$**:** 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 $N_{gene} = 10\ bits$

$$chromosome = \left[\underbrace{1111001001}_{géne1} \underbrace{0011011111}_{géne2} \dots\dots\dots \underbrace{0000101001}_{géne3} \right]$$

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

3.1.2. Codage non binaire

Avec les développements récents, d'autres types de représentation plus sophistiquée sont apparus [26]. Ainsi, un chromosome peut être une liste des villes à parcourir pour le problème de voyageurs de commerce [7], 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 4.

3.2. La Population initiale

La population initiale peut être obtenue en générant aléatoirement les chaînes de l'espace de recherche. Cette méthode à 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 génère en utilisant la fonction aléatoire rand sous Matlab [35].

$$\boldsymbol{pop = round\left(rand\left(N_{pop,}N_{bits}\right)\right)} \qquad (3.5)$$

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.

3.3. 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 [23] :

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

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 :

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

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 [8] :

$$\boldsymbol{fitness}[\boldsymbol{pop_i}] = \frac{\boldsymbol{F}[\boldsymbol{pop_i}]}{\sum_{i=1}^{n} \boldsymbol{F}[\boldsymbol{pop_i}]} \qquad (3.8)$$

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

3.4. La sélection

La sélection permet de choisir quels chromosomes vont se reproduire afin défaire 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 [28].

3.4.1. La méthode roulette (Roulette Wheel Selection)

Pour chaque individu, la probabilité d'être sélectionné est proportionnelle à son adaptation au problème. Le principe de Roulette Wheel selection est celui de la roue de la fortune biaisée. Cette roue est une roue de la fortune classique sur laquelle on associe à chaque individu un segment dont la longueur est proportionnelle à sa fitness. On effectue ensuite un tirage aléatoire utilisé dans les roulettes de casinos avec une structure linéaire. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent adressés que les petits. Nous avons utilisé cette méthode pour résoudre le problème dans le chapitre 4.

3.4.2. 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 [27].

3.5. Modèle de reproduction

Au cours de la phase de reproduction, les chromosomes sont sélectionnés et leur La structure est modifiée pour générer de nouveaux individus qui formeront la prochaine génération. Différentes stratégies existent pour maintenir la population continue. Les parents sélectionnés à la roulette sont introduits dans le pool d'élevage (mating pool, considéré comme

la population intermédiaire), où ils seront à nouveau sélectionnés au hasard pour voir leurs chromosomes subir la transformation de l'opérateur génétique. Les deux principaux opérateurs fondamentaux sont le croisement et la mutation. Le croisement effectue des opérations binaires (ou sexuée) et nécessite deux parents. La mutation est une opération unaire (ou asexuée) utilisée pour introduire de petits changements dans la solution ou changer la direction de la recherche.

3.5.1. Le croisement

Après avoir fait une sélection des individus « Genitor », il nous faut générer une nouvelle population, afin d'enrichir la diversité de la population. Le croisement permet cela en manipulant la structure des chromosomes. Il existe plusieurs manières d'effectuer un croisement soit par cross-over où l'on a besoin de deux parents « Genitor » qui génèrent à la fin du croisement deux enfants, soit par copie où l'on n'a besoin que d'un seul parent « Genitor » qui nous donnera à la fin du croisement un enfant qui est le parent lui-même (sa copie).

- **cross-over :** d'une manière générale le chromosome de chaque parent est découpé en deux parties qui sont recombinées pour former les descendants (voir exemple1), mais il se peut qu'il y ait plusieurs points de croisement.

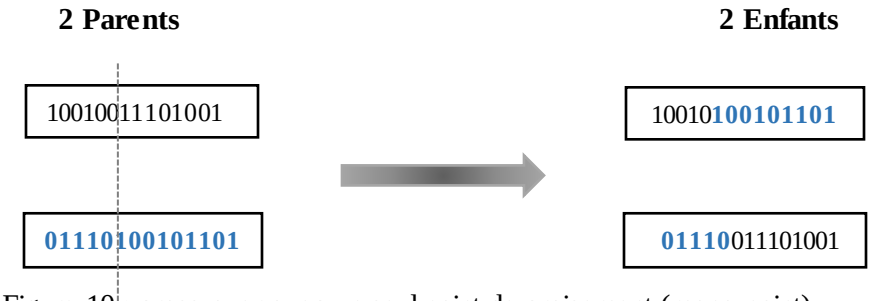

Figure 10 : cross-over avec un seul point de croisement (mono-point).

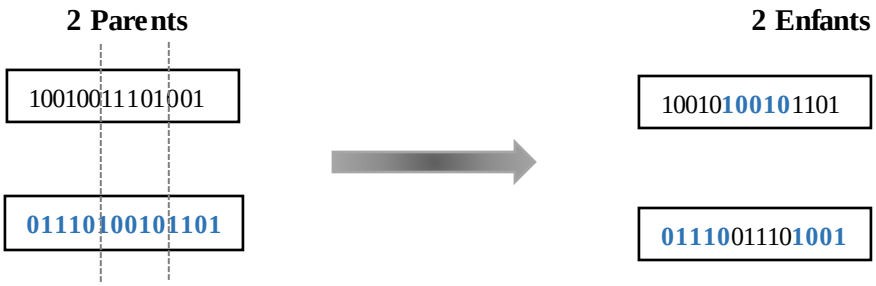

Figure 11 : cross-over avec plusieurs points de croisement (multipoints de croisement)

Le choix du nombre de points de croisement diffère selon l'algorithme.

- **copie :** appelé aussi cross-over uniforme, où le nouvel individu hérite du même gène que le parent « Genitor ».

3.5.2. La mutation

Permet aux gènes d'être modifiés (remplacés) d'un chromosome à un autre de manière aléatoire, très similaire au croisement à première vue. La différence est que le croisement essaie de converger vers la meilleure solution pour lui (intéressé par la qualité), mais la mutation permet la diversité. Dans une certaine mesure, la mutation est utilisée pour éviter une convergence prématurée de l'algorithme. Par exemple, la mutation est utilisée pour éviter la convergence vers un optimum local lors de la recherche d'une solution optimale. Il faut définir le taux de mutation lors du changement de population, généralement compris entre 0,001 et 0,01. Une valeur relativement faible doit être choisie pour ce taux afin de ne pas rester coincé dans une recherche aléatoire et de garder les principes de sélection et d'évolution.

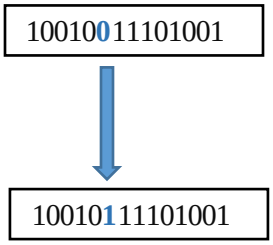

Figure 12 : Représentation schématique d'une Mutation dans un Chromosome

3.5.3. L'élitisme

L'élitisme est une opération spéciale [26], 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 conseillé 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 [27].

4. Choix des valeurs des paramètres

Les valeurs des paramètres d'un Algorithme Génétique doivent être choisies avec les considérations suivantes [26] :

4.1. 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. [33]

4.2. 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% [33], [26].

4.3. 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. [33]

4 Autres paramètres

Les opérateurs de l'algorithme génétique sont guidés par un certain nombre de paramètres fixés à l'avance. La valeur de ces paramètres influence la réussite ou non d'un algorithme génétique. Ces paramètres sont les suivants :

- La taille de la population, *N*, et la longueur du codage de chaque individu *l* (dans le cas du codage binaire). Si *N* est trop grand le temps de calcul de l'algorithme peut s'avérer très important, et si *N* est trop petit, il peut converger trop rapidement vers un mauvais chromosome. Cette importance de la taille est essentiellement due à la notion

de parallélisme implicite qui implique que le nombre d'individus traité par l'algorithme est au moins proportionnel au cube du nombre d'individus.

- La probabilité de croisement P_{cro}. Elle dépend de la forme de la fonction de fitness. Son choix est en général heuristique (tout comme pour P_{mut}). Plus elle est élevée, plus la population subit de changements importants. Les valeurs généralement admises sont comprises entre 0.5 et 0.9.
- La probabilité de mutation P_{mut}Ce taux est généralement faible puisqu'un taux élevé risque de conduire à une solution sous-optimale.

5 Conclusion

Les algorithmes génétiques correspondent à de nouvelles méthodes d'optimisation dont les processus permettent d'arriver à de meilleures solutions (optimales) basées sur les principes de l'évolution génétique. Leur fonctionnement est très différent des algorithmes d'optimisation traditionnels. Il y a 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. Par conséquent, comme le prouve la théorie, l'opérateur de croisement a une place importante dans le processus d'exploration de l'espace des solutions. Quant à la mutation, elle ne joue qu'un rôle mineur par rapport au croisement, mais son impact sur l'exploration de l'espace des solutions ne peut être ignoré.

Chapitre 04

Etude pratique

Chapitre 04 Etude pratique

1. Introduction

Dans ce chapitre nous allons étudier un atelier de fabrication du plastique par extrusion ainsi que la démarche de résolution proposée.

Le processus de fabrication de cette chaîne continue est divisé en plusieurs étapes. Différentes machines se succèdent, dans le cadre de cette étude nous nous intéressons aux chaînes de production, et non à chaque machine individuellement, qui tournent 24h/24 toute l'année, sauf panne, maintenance, surcapacité (en période de faible demande peut être anormale).

Sur une même ligne de production, la production peut varier d'une référence à l'autre, selon la commande et les besoins du client, le format, la couleur, la densité ou son décor esthétique du film plastique pouvant être commandé.

Ces changements prennent plus ou moins de temps, selon les commandes de la chaîne et le personnel qualifié ajustant l'extrudeuse. Ils changent de paysage en quelques minutes et de formats en heures. Il faut donc comprendre que, d'une part, ces changements de production peuvent sérieusement affecter la production, car ils introduisent du temps non productif (perte de temps machine), et d'autre part, en raison du besoin de spécialistes supplémentaires et de produits qui sont rejetés selon divers paramètres.

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. L'environnement de développement

Notre travail est écrit dans L'environnement de développement Matlab version 7.10 sous Windows 10 Professionnel. Et nous avons utilisé comme élément matériel un ordinateur **DELL** qui possède les propriétés suivant :

Figure 13 : les propriétés de notre PC

2.1. Matlab*« Mat (matrix) et laboratory (lab) »*

Est un langage de programmation de quatrième génération et un environnement de développement ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran. Les utilisateurs de MATLAB (environ un million en 2004) sont de milieux très différents comme l'ingénierie, les sciences et l'économie dans un contexte aussi bien industriel que pour la recherche. Matlab peut s'utiliser seul ou bien avec des toolbox (« boîte à outils »).

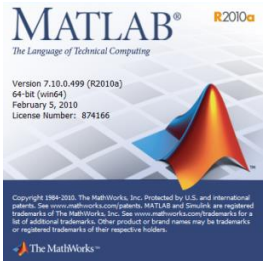

Figure 14 : Logo MATLAB (version R2010a)

2.1.1. Les avantages de Matlab

L'environnement de programmation Matlab possède de nombreux avantages :

- Développement rapide pour le calcul et pour l'affichage,
- Un environnement facile d'approche pour un débutant,
- Un éditeur intégré,
- Une librairie riche,
- La possibilité d'intégrer un programme en C++/C++,
- Une documentation bien faite.

Parmi ses principaux inconvénients, Matlab affiche un temps de calcul beaucoup moins rapide qu'en C/C++, jusqu'à cent fois inférieur à tâche équivalente. En revanche, il permet un temps de développement beaucoup plus rapide que ce dernier.

2.1.2. Les différents usages de MATLAB

Les cas d'usage de Matlab sont nombreux. Le langage de programmation est notamment utilisé dans :

- Système de contrôle.
- Machine Learning.
- La maintenance prédictive.
- Le traitement du signal et les séries temporelles.
- L'automatisation des tests.
- les systèmes de télécommunication.
- la robotique.

2.1.3. Comment télécharger Matlab

Matlab est téléchargeable sur le site officiel de MathWorks. Une version d'essai de 30 jours est disponible ici. Pour l'achat d'une licence, ce lien permet d'accéder aux différentes offres de prix et composants Matlab proposés. À noter qu'il est possible d'obtenir des outils complémentaires, pour parfaire les fonctionnalités de Matlab. C'est le cas des produits Simulink, des modules de data science et de l'accès aux rapports de données, par exemple.

3. Formulation du problème

3.1.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 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.

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 les plastiques 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. [33]

3.2.Description du System à étudier

L'objectif de notre travail est de trouver, les meilleures solutions d'ordonnancement multi objectif en utilisant l'algorithme génétique pour 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", qui respect les contraintes imposés.

3.3. Fonction objectif

Minimisation du " Coût"et "Makespan"

3.4.Les contraintes

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 prises en considération.

4. Démarche de résolution

A partir des chapitres que nous avons vus précédemment, nous proposons de souligner dans cette sous-section l'apport des Algorithmes Génétiques (AG) à la résolution de notre problème d'ordonnancement.

Nous décrivons d'abord les techniques de codage que nous utilisons et les différentes approches propres aux algorithmes génétiques. Enfin, nous montrons des résultats inhérents au problème que nous traitons.

4.1. Codage du problème en chromosome

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.

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 silos 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 : [33]

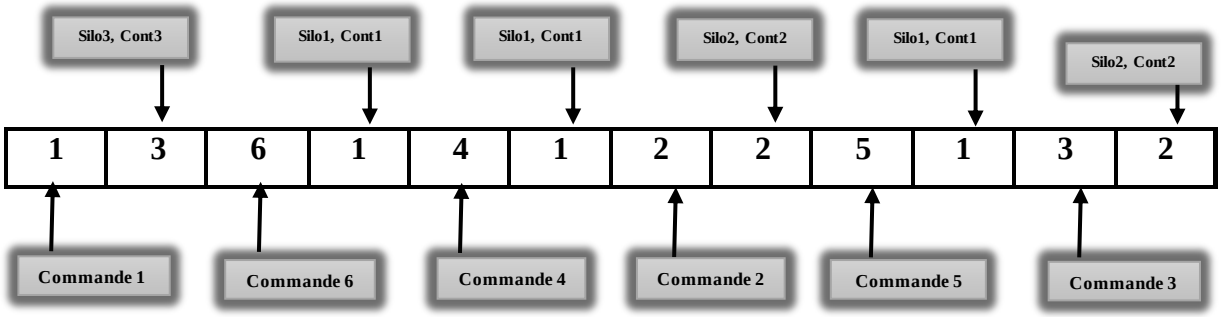

Figure 15 : Représentation d'une solution possible d'Ordonnancement.

Machine	Commandes 'Cm'/Temps 'Minutes								
Silo1	Cm4	Cm5	Cm6						
Silo2	Cm2	Cm3							
Silo3	Cm1								
Conteneur1		Cm4	Cm5	Cm6					
Conteneur2		Cm2	Cm3						
Conteneur3		Cm1							
Extrudeuse			Cm2	Cm1	Cm4	Cm3	Cm5	Cm6	

Figure 16 : Ordonnancement représenté par le codage 13, 61, 41, 22, 51, 32

4.2.Création de Population initiale

La population initiale qui constitue l'ensemble des chromosomes est générée d'une façon aléatoire. La population initiale contient des solutions possibles du problème.

```
1.Debut
2.Pour n de 1 a taille de la population
     permuter les numeros de 1 a nombre de commande
     charger le vecteur resultant dans les positions impaires du chromosome n
     generer un vecteur aleatoire avec des valeurs allant de 1 a 3
     charger le vecteur resultant dans les positions paires du chromosome n
  Fin Pour
3.Fin
```

Algorithme 4 : algorithme de la création de population.

4.3.Makespan et coût de transitions

4.3.1. Fonction multi-objective et fonction *Fitness*

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. [33]

Cette fonction est définie comme suit :

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

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 :

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

Notez que la valeur maximale de F(x) provient de la valeur minimale de f(x).

4.4.Méthode de sélection

4.4.1. La sélection par roulette

Il s'agit de la méthode la plus courante. Les individus parents sont sélectionnés proportionnellement à leur performance. Meilleur est le résultat fourni par l'évaluation d'un individu, plus grande est sa probabilité d'être sélectionné.

Le nombre de fois qu'un individu sera sélectionné est égal à son évaluation divisée par la moyenne de l'évaluation de la population totale. Plus exactement, la partie entière représente le nombre de fois qu'il sera sélectionné, et la partie flottante la probabilité qu'il aura d'être sélectionné à nouveau.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 [35] :

- **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.

$$f(x)_{moy} = \frac{\sum f(x)}{taille\ de\ la\ poulation=24} \qquad (4.3)$$

- **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.

$$Nombre\ de\ copies(x) = \frac{f(x)}{f(x)_{moy}} \qquad (4.4)$$

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

$$Probabilité\ de\ selection(x) = \frac{Nombre\ de\ copies\,(x)}{taille\ de\ la\ population} \qquad (4.5)$$

- **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.

$$Probabilité\ cumulée(x) = Probabilité\ de\ selection(x) + Probabilité\ cumulée(x-1) \quad (4.6)$$

- **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.

4.4.2. Algorithme de Sélection :

```
1. Debut
2. Pour chaque ordonnancement n de 1 a taille de population faire
          calculer le cout C
          calculer le makespan M
          calculer la fonction objectif F(n) = 100/(c*Alpha+M*Beta)
   Fin pour
3. Inistialiser somme par 0
4. Pour chaque ordonnancement n de 1 a a taille de population
           somme = somme+F(n)
   Fin Pour
5. calculer la moyenne F_Moyenne=somme/taille de population
6. pour chaque ordonnancement n de 1 a taille de population faire
          nbr_copies(n)F(n)/F_moyenne
          Pr_select(n) = nbr_copies(n)/taille de population
   Fin pour
7. Initialiser Pr_cumul(1) par Pr_select(1)
8. Pour chaque ordonnancement n de 2 a taille de population faire
          Pr_cumul(n) = Pr_cumul(n-1)+Pr_select(n)
   Fin pour
9. Pour n de 1 a taille de population
          choisir un nombre aleatoire R entre 0 et 1
          Initialiser n par 1
          tant que Pr_cumul(n)<R faire
               n=n+1
          fin tant que
       placer ordonnancement n dans la population intermediaire
   Fin Pour
10. Fin
```

Algorithme 5 : Algorithme de sélection.

4.5.Croisement et Mutation

4.5.1. Le croisement de la solution possible :

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 [26].

Parent 1	**1**	**3**	**6**	**1**	**4**	**1**	**2**	**2**	**5**	**1**	**3**	**2**
Parent 2	**4**	**2**	**1**	**3**	**2**	**3**	**6**	**1**	**3**	**2**	**5**	**3**
Enfant 1	**1**	**3**	**6**	**3**	**2**	**3**	**6**	**1**	**3**	**1**	**3**	**2**
Enfant 2	**1**	**3**	**6**	**1**	**4**	**1**	**2**	**2**	**5**	**1**	**3**	**2**

Figure 17 : 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. Algorithme de croisement

```
1.Debut
2.Decouper la population en 2 sous population pop1 et pop2 aleatoirement de taille egales
3.Pour chaque ordonnancement de 1 a n/2
      choisir ch1(n) de pop1
      Choisir ch2(n) de pop2
      tout les gens de l'enfant 1 recoit les genes de ch1
     les genes numero (4 a 2*nbr_cmd-3) de l'enfant 1 recoit les genes (4 a 2*nbr_cmd-3) de ch2
     les genes numero (4 a 2*nbr_cmd-3) de l'enfant 2 recoit les genes (4 a 2*nbr_cmd-3) de ch1
  Fin pour
4.Fin
```

Algorithme 6 : Algorithme de croisement

4.5.3. Algorithme de mutation

Quand un ordonnancement enfant s'est créé à la population, 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, Cela revient à sélectionner aléatoirement une commande dans l'ordonnancement et changer sa machine de démarrage. Dans l'exemple ci-dessous, la commande 6 a été sélectionnée aléatoirement pour être fabriquée sur le Silo 1, Conteneur 1, au lieu de Silo 3, Conteneur 3.

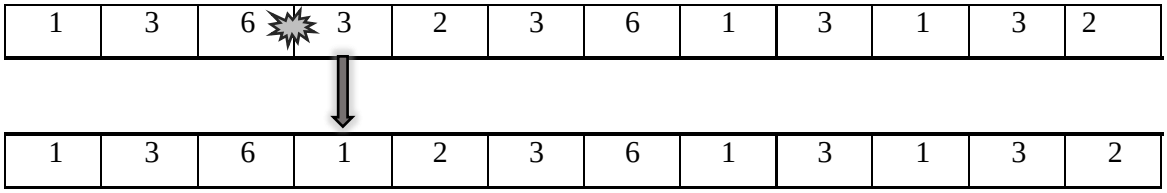

Figure 18 : exemple de mutation dans un ordonnancement enfant

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

```
1.Debut
2.choisir un numero aleatoire N_mut entre 1 et nombre de commande
3.choisir un nombre aleatoire M_mut entre 1 et 3
4.remplacer le gene dans la position N_mut*2 par M_mut
5.Fin
```

Algorithme 7 : Algorithme de mutation.

4.6. Régénération

Une fois toutes les étapes de l'AG de première génération terminées, on entre dans la nouvelle génération en répétant le même processus, la seule différence est que la dernière population de la première génération devient automatiquement la première population de la deuxième génération, car notre exemple. L'algèbre est fixée à 66 générations. Lors de l'exécution de notre programme de calcul, nous n'avons pas été en mesure de suivre la dynamique d'évolution d'une population de taille 24 X 12 sur 66 générations (66 itérations), en revanche d'observer différentes statistiques. Les mesures statistiques sont importantes pour régler les paramètres et mesurer les performances de notre programme.

5. Les données utilisées

Les deux tableaux suivants représentent successivement le temps de chaque machine associée à chaque type de commandes, les valeurs des coûts de transitions en euros par kilogramme lors du passage d'une commande à une autre.

Tableau 1 : Temps d'exécution tâches (Min). [33]

	Commande1	**Commande2**	**Commande3**	**Commande4**	**Commande5**	**Commande6**
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

Tableau 2 : Coût associé aux transitions entre commandes en Euros. [33]

De/à	**Commande1**	**Commande2**	**Commande3**	**Commande4**	**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	

6. Le programme

6.1.Comment ça marche

Le contenue de code du programme consiste un ensemble des M_Files le code général « ordon.m »

Les étapes de la solution sont :

- Définir les paramètres d'entrée (les paramètres d'AG et paramètres machines 'cost.mat','time.mat').
- Initialiser la population (gen=0) et créer la population initial matrice d'une taille 24*12 avec six gènes en utilisant la fonction de MATLAB « randper » qui permet de crée les chromosomes à partir de 1 à 6 avec permutation chaque fois.
- Créer « cout_mat » qui calcul cout de transition entre les commandes pour chaque ordonnancement et « Makespan » qui calcul makespan pour chaque ordonnancement
- La sélection par roulette cette étapes fait le calcul fitness de chaque individu et la probabilité cumulée en et créer 'Maiting.pool'.
- Une fois 'Maiting.pool' a été remplie les chromosomes sont aléatoirement repartir en couple et appliquer le croisement entre deux pères qui formes un seul couple.
- Enfin la mutation appliquer sur un chromosome choisi chaque fois,cela créera une nouvelle génération.
- Apres ces itérations, les résultats seront obtenues et afficher l'ordonnancement optimal avec les meilleures solutions sont affichées ci-dessus.

  La figure suivante représente les ensembles des M-files qui l'on utilisés sous Matlab

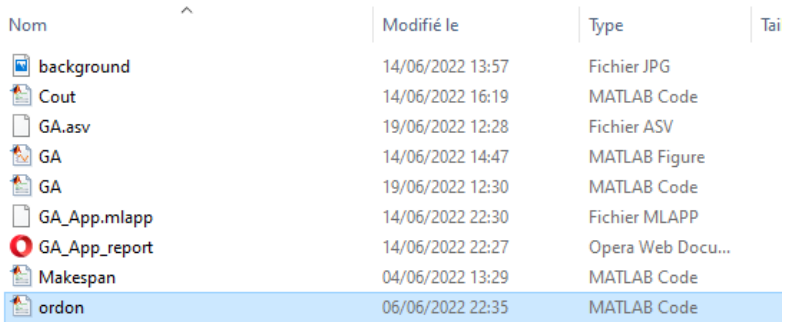

Figure 19 : les ensembles des M-files qui l'on utilisés sous Matlab

6.2.Le schéma de résolution du problème

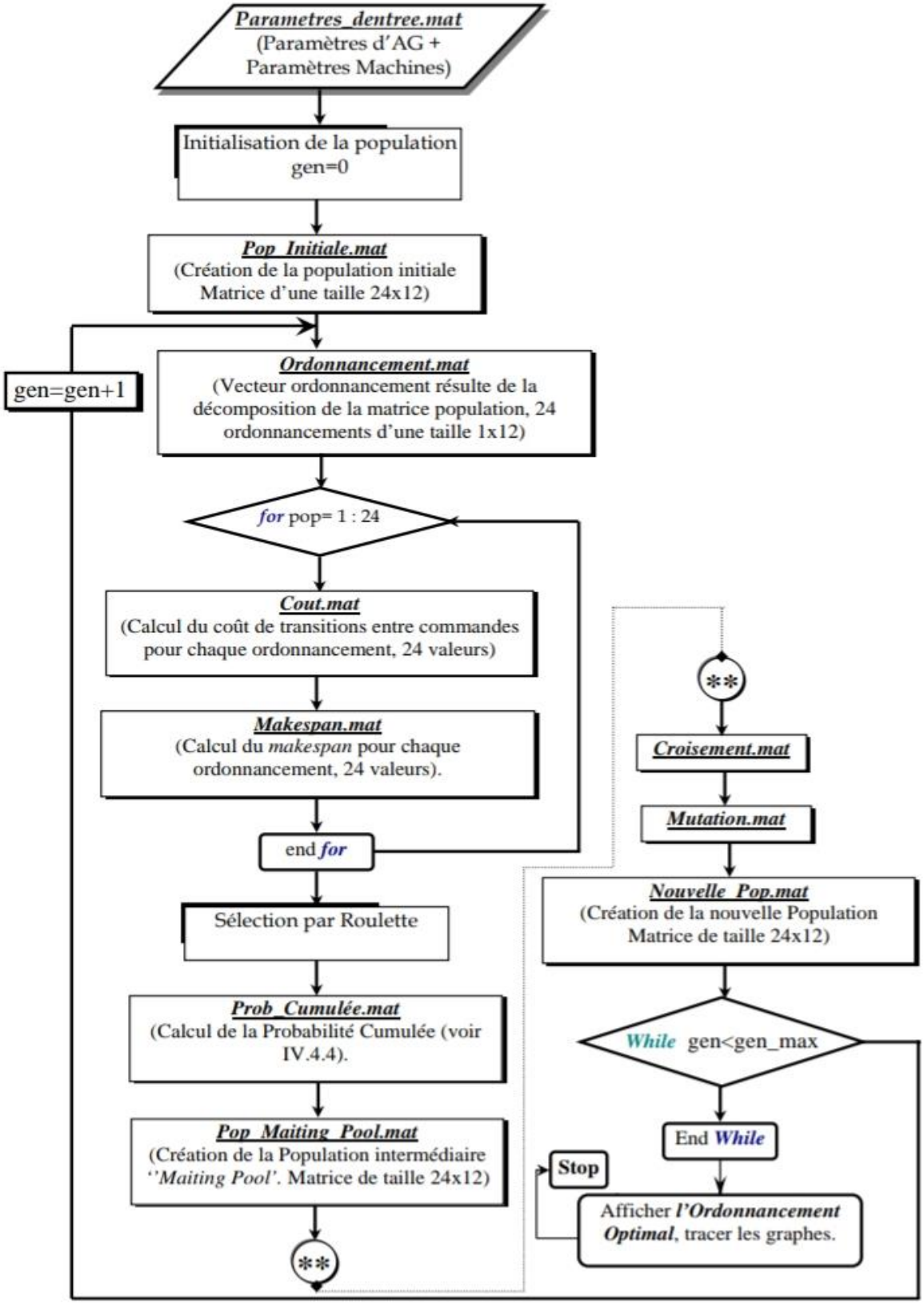

Figure 20 : Le schéma de résolution du problème

6.3. Résultats obtenus par le programme de calcule

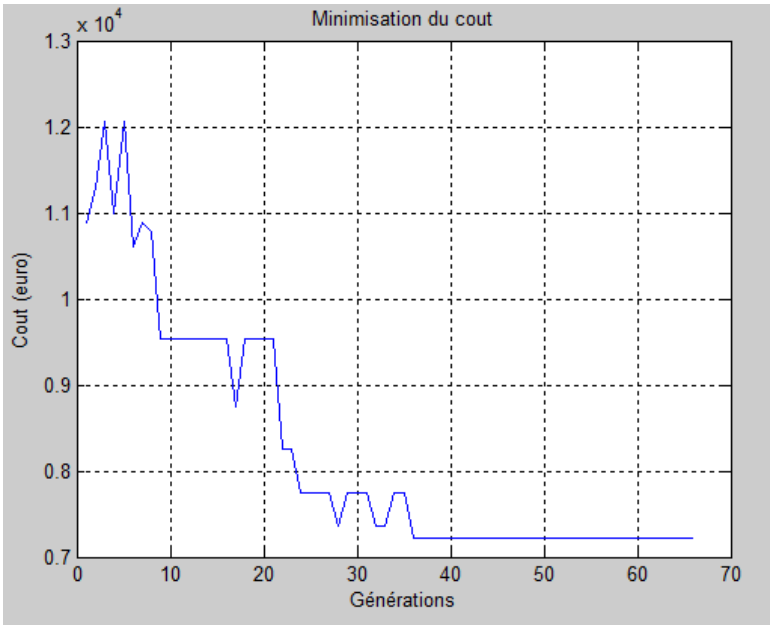

Figure 21 : Résultat d'exécution (Minimisation du cout)

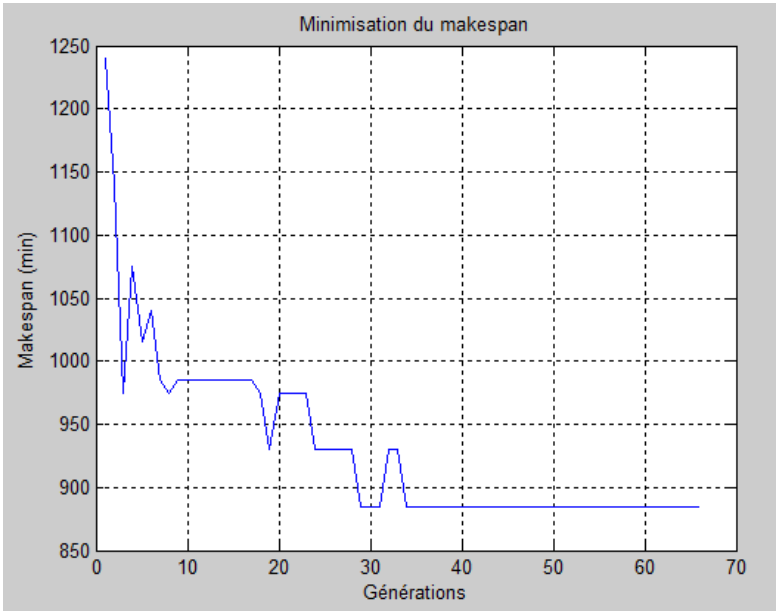

Figure 22 : Résultat d'exécution (Minimisation du Makespan)

Figure 23 : Résultat d'exécution (Maximisation du Fitness)

6.4.Illustration de logiciel

Notre Logiciel se compose d'une fenêtre permet de saisir Nombre de commandes et la taille de population et le nombre de générations, les données sont saisies de manière aléatoire dans les conditions mentionnées précédemment.

Maintenant, nous vous montrons un ensemble des figures qui affichent le contenu de notre programme.

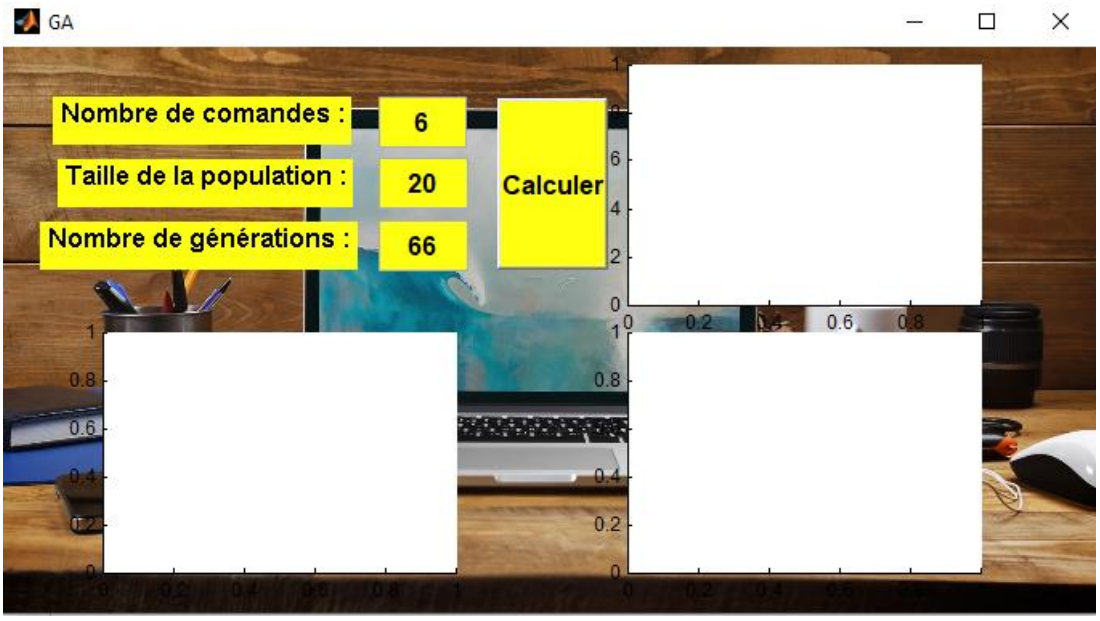

Figure 24 : L'interface de logiciel

Lorsque vous remplissez les données (d'après l'exemple précédent), cliquez sur "Calculer" pour afficher les graphes suivant :

- **Minimisation de la fonction coût.**
- **Minimisation de la fonction temps (Makespan).**
- **Maximisation de la fonction *Fitness*.**

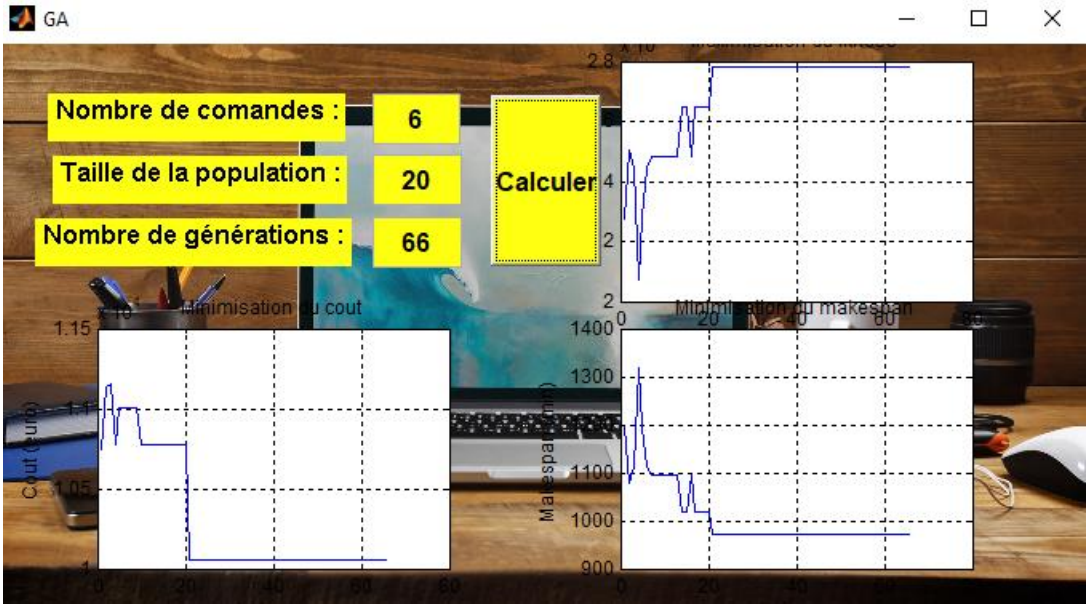

Figure 25 : Les résultats obtenus

Lorsque vous remplissez les données aléatoirement, ce qui suit apparaît

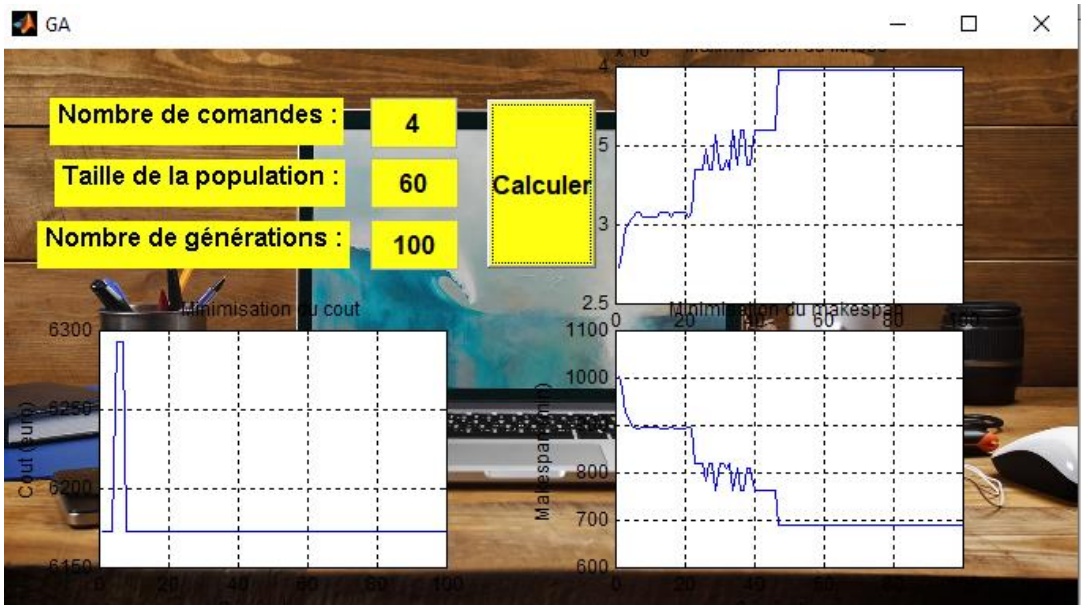

Figure 26 : Les résultats obtenus (données aléatoire)

Pour ce problème d'ordonnancement multi objectif nous avons remarqué que le programme nous permet de trouver un ordre de passage optimal de six commandes sur trois entrées possible un Makespan optimal et un cout de transition optimal.

7. Conclusion

Nous avons présenté dans ce chapitre le programme développé, le langage et ses outils utilisés pour résoudre le problème que nous avons couvert dans ce mémoire.

Ensuite, nous avons basée sur les procédures d'optimisation en particulier sur l'algorithme génétique que nous proposons dans ce travail est très efficace pour générer des séquences de démarrage pour différentes commandes. Notre travail s'inscrit dans le processus global d'aide à la décision, les décideurs Oui, en changeant certains paramètres (surtout dans client : temps et coût de fabrication) pour trouver la meilleure solution. Cette solution correspond à un ordonnancement optimal, qui permet au décideur de réduire Temps associé à la production (Makespan) et coûts associés au passage d'un produit à l'autre Ordre, et on a présenté l'implémentation de notre programme et nous avons considéré un exemple avec les résultats obtenus.

Conclusion générale

Le problème d'ordonnancement multi objectif est l'un des problèmes d'ordonnancement intensivement étudiés. C'est un problème extrêmement complexe. Il est lassé parmi les problèmes combinatoires difficiles au sens fort. Cette complexité est due à l'explosion combinatoire du nombre de solutions qui croît exponentiellement avec la taille du problème.

Le travail exposé dans ce mémoire s'intéresse à l'étude d'une problématique industrielle concernant l'ordonnancement dans une entreprise d'extrusion plastique, nous avons choisi d'appliquer les Algorithmes Génétiques à notre atelier.

Les algorithmes génétiques basés sur des processus évolutifs naturels appartiennent aux techniques d'optimisation basées sur l'exploration aléatoire de l'espace des solutions, et ils recherchent l'optimum à partir d'un ensemble de points plutôt qu'un point unique. L'algorithme GA de base est divisé en 4 parties : sélection, croisement, mutation et réplication. Ils sont plus proches de la formulation du problème d'ordonnancement, chaque chromosome correspond à une solution d'ordonnancement possible, et le meilleur chromosome obtenu après plusieurs générations correspond à la solution d'ordonnancement optimale. Nous avons montré que les algorithmes génétiques appliqués à l'ordonnancement sont efficaces car ils trouvent de meilleurs résultats et atteignent l'optimum, et nous calculons cette solution représentée par l'ordonnancement optimal, qui découle de l'amélioration du choix des différents paramètres. 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 procédure de calcul établit l'ordre de passage optimal de 6 ordres sur 3 entrées possibles (Silos), et le résultat de l'ordonnancement donne également le Makespan optimal et le coût de transfert optimal.

Nous pouvons dire que les algorithmes génétiques peuvent apporter des solutions aux problèmes complexes tels que celui de notre étude.

Le travail réalisé présente un avantage majeur qui est le calcul du makespean de l'ordonnancement des tâches. Il reste encore du travail à faire, tel que le calcul du coût des opérations.

Références bibliographiques

[1] Alain Berro, 18 décembre 2001, Optimisation multi objectif et stratégies d'évolution en environnement dynamique, Université de Toulouse I

[2] Algorithmes de construction de graphes dans les problèmes d'ordonnancement de projet, mouloud nasser.

[3] 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).

[4] BELLACHE Nour El Islam, Développement et implémentation d'un solveur bio inspirépour la résolution d'un problème d'ordonnancement d'atelier, Diss, Université de M'sila, 2017.

[5] BENFERDI MOSTAFA, «Etude Et Simulation D'un Détecteur De Gaz : Gaz De Pétrole Liquéfie GPL», mémoire De Master En Informatique Industrielle, Université d'OEB ,2014.

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

[7] 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.

[8] C. Hamzacebi and F. Kutay. "A heuristic approach for finding theglobal minimum : Adaptive random search technique. Applied Mathematicsand Computation", 173 :1323–1333, 2006.

[9] D. E. Goldberg & K. Deb. A comparative analysis of selection schemesused in genetic algorithms. In Foundations of Genetic Algorithms, pp. 69–93. MorganKaufmann, 1991.

[10] D. Goldberg. Genetic algorithms with sharing for multimodal function optimization.In Proceedings of the Second International Conference on GeneticAlgorithms, pages 41–49, 1987.

[11] D.E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning.AddisonWesley, 1989.

[12] E. A. Feigenbaum and J. Feldman. (Edirors). Computers and thought. McGraw-Hill Inc. pp.192. New York, 1963.

[13] E. A. Feigenbaum, j. Feldman, (Edirors). Computers and thought. McGraw-Hill Inc. pp.6. New York, 1963.

[14] Fred Glover and Manuel Laguna, 1997, Tabu search, Kluwer Academic Publishers.

[15] GARE 79 M. R. GAREY et D. S. JOHNSON, Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[16] Groupe GOTHA, Modèle et algorithmes en ordonnancement, Ellipses, 2004, p 227

[17] H. Hentous, contribution au pilotage des systèmes de production de type Job Shop, Doctorat, LYON, 1999.

[18] H.-P. Schwefel. Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of organic evolution. Annals of Operations Research, 1 :165–167, 1984.

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

[20] J.H. Holland, editor. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. MITPress/Bradford books, Cambridge, MA, 1992. Second edition.

[21] J.H. Holland. Adaptation in natural and artificial systems. Technical report, University of Michigan, Ann Arbor, 1975.

[22] 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.

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

[24] Lerman, I. & Ngouenet, F. (1995), Algorithmes génétiques séquentiels et parallèles pour une représentation affine des proximités, Rapport de Recherche de l'INRIA Rennes - Projet REPCO 2570, INRIA.

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

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

[27] 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.

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

[29] NADIR Ramla. Un problème d'ordonnancement de type Job Shop dans un environnement dynamique. Université de Msila, 2019.

[30] Optimisation Combinatoire, M1 GADM, Dr. N. KHERICI, UBMA, 2020/2021 p 3,5

[31] R. L. Solso. Cognitive psychology. Harcourt Brace Jovanovich, Inc., New York. Pp. 436, 1979.

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

[33] T.BENIKHLEF Optimisation des Problèmes d'Ordonnancement Multi-Objectifs dans les Systèmes de Production, Mémoire de Magister, Université de Boumerdès, 2005.

[34] T.zoulikha, L'optimisation multi objectifs pour l'ordonnancement de tâches dans le cloud computing, universite m'sila, 2019.

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

[36] THOMAS VALLEE ET MURAT YILDIZOGLU, « Présentation Des Algorithmes Génétiques Et De Leurs Applications En Economie», Université De Nantes, LEA-CIL 2001

[37] Thomas Vallée] et Murat Yıldızoglu, Université de Nantes, Chemin de la Censive du Tertre et NANTES Thomas.Vallee, Université Montesquieu Bordeaux 7 septembre 2001.

[38] V.Giard, Gestion de la Production, 2ème édition, Economica, PARIS, 1988.

[39] Vincent Barichard-Jin-Kao Hao, 2003, Une approche hybride Pour l'optimisation multiobjectif sous contraintes, Faculté des sciences-Université d'Angers.

[40] Vincent Giard, Gestion de la production et des Flux, Paris Economica, 2003, p 122

ملخص

في هذه الرسالة عملنا على حل مشكلة الجدولة في أنظمة الإنتاج الخاصة بـ "ورشة بثق البلاستيك". تواجه هذه الورشة مشاكل كبيرة تتعلق بالمواد الخام باهظة الثمن. من خلال تقليل مادة الانتقال بين الأوامر المختلفة والتكلفة المرتبطة بها، في هذا النوع من المشاكل لا توجد طرق دقيقة تعطي حلولاً متبادلة. هنا، الطرق التقريبية بدورها تحل مشاكل درجة NP الصعبة واخترنا من بينها الخوارزمية الجينية لاستخراج جميع الحلول الممكنة واستخراج Makespan والتكلفة.

Résumé

Dans ce mémoire on a travaillé sur la résolution du problème d'ordonnancement dans les systèmes de production d'un "Atelier d'extrusion plastique". Cet atelier fait face aux grands problèmes liés aux matières premières, qui sont coûteuses. En réduisant la matière de transition entre les différentes commandes et le coût associé, Dans ce type de problème il n'y a pas de méthodes exactes qui donnent des solutions réciproques. Ici, les méthodes approchées résolvent à leur tour ces problèmes d'un degré NP difficile et nous avons choisi parmi elles l'algorithme génétique pour extraire toutes les solutions possibles, extraire du Makespan et du coût.

Abstract

In this thesis, we worked on the resolution of the scheduling problem in the production systems of a "Plastic extrusion workshop". This workshop faces major problems related to raw materials, which are expensive. By reducing the material of transition between the various commands and the associated cost, in this type of problem there are no exact methods, which give reciprocal solutions. Here, the approximate methods in turn solve these problems of NP-hard degree and we chose among them the genetic algorithm to extract all possible solutions, extract Makespan and cost.