République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

École Nationale Polytechnique
Département du Génie Industriel

المدرسة الوطنية المتعددة التقنيات
قسم الهندسة الصناعية

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'État en Génie Industriel

Option : Management Industriel
Option : Management de l'innovation

Étude, modélisation et implémentation d'une solution d'ordonnancement pour la gestion de la production au sein de BIOPHARM

Réalisé par :
M. BOUDJELOUAH Yaniss Rayane
M.CHELGHAM Yasser

Encadré par :
Mme.NIBOUCHE Fatima (ENP)
M.BELMOUHOUB Mohammed (BIOPHARM)

Soutenu le 11 Juillet 2021, Devant le jury composé de :

M. ZOUAGHI Iskander : MCB ENP - Président
M. BOUKABOUS Ali : MAA ENP - Examinateur

ENP 2021

République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

École Nationale Polytechnique
Département du Génie Industriel

المدرسة الوطنية المتعددة التقنيات
قسم الهندسة الصناعية

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'État en Génie Industriel

Option : Management Industriel
Option : Management de l'innovation

Étude, modélisation et implémentation d'une solution d'ordonnancement pour la gestion de la production au sein de BIOPHARM

Réalisé par :
M. BOUDJELOUAH Yaniss Rayane
M.CHELGHAM Yasser

Encadré par :
Mme.NIBOUCHE Fatima (ENP)
M.BELMOUHOUB Mohammed (BIOPHARM)

Soutenu le 11 Juillet 2021, Devant le jury composé de :

M. ZOUAGHI Iskander : MCB ENP - Président
M. BOUKABOUS Ali : MAA ENP - Examinateur

ENP 2021

ملخص

يدخل هذا العمل في إطار تصميم حل خاص بجدولة عملية الإنتاج داخل مؤسسة بيوفارم الصناعية وذلك من خلال توظيف عملية النمذجة الرياضية و الأدلة العليا، بالخصوص استعمال الخوارزميات الجينية.

الهدف من هذا الحل هو تنظيم و أتمتة و تحسين عملية الجدولة داخل شركة بيوفارم الصناعية من خلال مراعات مختلف معادلات التقييد المتعلقة بنوعية الصناعة، بطبيعة خطوط الإنتاج و القدرة على التحكم في العملية.

من أجل السماح في النهاية باكتساب قدرة انتاجية أكبر يمكن أن تترجم إلى ميزة تنافسية كبيرة و إلى نقطة قوة في يد الشركة التي تعمل في بيئة شديدة التنافسية.

كلمات مفتاحية :
الجدولة، خط إنتاج مرن، خط إنتاج ثابت، خوارزميات الحدس، خوارزميات الأدلة العليا، خوارزميات تطورية، خوارزميات جينية.

Abstract

This work is part of the design of a production scheduling solution within Biopharm Industry, by exploiting mathematical modeling and meta-heuristics, more particularly genetic algorithms.

The objective of this solution is to standardize, automate and optimize the scheduling within the production of Biopharm Industry by taking into consideration the various constraints related to the nature of the industry, production lines and process control.

In order to ultimately allow a gain in capacity that can translate into a significant competitive advantage in an ultra-competitive environment.

Keywords : Scheduling, Flexible Job Shop, Flow Shop, Genetic Algorithm, Heuristics, Metaheuristics, Evolutionary Algorithm.

Résumé

Ce travail s'inscrit dans le cadre de la conception d'une solution d'ordonnancement de la production au sein de Biopharm Industrie, et ce, en exploitant la modélisation mathématique et les meta-heuristiques plus particulièrement les algorithmes génétiques. L'objectif de cette solution est de normaliser, automatiser et optimiser l'ordonnancement

au sein de la production de Biopharm Industrie en prenant en considération les différentes contraintes liées à la nature de l'industrie, des lignes de production et la maîtrise des processus.
Afin de permettre au final, d'obtenir un gain de capacité qui peut se traduire en un avantage concurrentiel de taille dans un environnement ultra-compétitif.

Mots clés : Ordonnancement, Job Shop flexible, Flow shop, Algorithme génétique, Heuristiques , Métaheuristiques, algorithme évolutionnaire.

Dédicace

“

À mes chers parents, pour leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,

À ma famille qui a toujours été là pour moi,

À Yasser mon binôme dont j'admire le calme,

À mes amis qui ont su rester forts lorsqu'on était au plus bas, ne vous inquiétez pas nous nous reverrons au sommet,

À mes étudiants dont l'énergie m'inspire,

À IEC et aux futures générations d'indus qui redonneront un éclat au département ,

À TIMPA qui rassemble tous les phénomènes à un kilomètre à la ronde,

À moi-même, car on ne l'entend pas assez,

Merci.

”

- Yaniss

Dédicace

“

À mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,

À ma chère mère, pour ton amour pour moi, pour les sacrifices que tu consens pour rendre tes enfants heureux. tu as enduré beaucoup de peine pour mon bien-être et à ma réussite. Reçois ceci en guise de ma reconnaissance et que Dieu te garde longtemps afin que tu puisses goûter aux arbres que tu as plantés,

À mon chère père, pour le goût à l'effort qu'il a suscité en moi, de par sa rigueur, ceci est ma profonde gratitude pour ton amour, que ce rapport soit le meilleur cadeau que je te puisse t'offrire,

À ma chère petite sœur Hidaya qui a illuminé notre maison, et mes chers frères, Mouhcine, Louai et Mouatez que ce travail soit pour vous un exemple à suivre et vous incite à mieux faire, et à mon frère aîné, Taki eddine , pour son appui et son encouragement,

À toute ma famille pour leur soutien tout au long de mon parcours universitaire,

À mes chers amis qui ont été ma deuxième famille durant ce parcours, Kamel, Bilal, Ibrahim et Ibrahim, Aymen, Mohcine, Ilyas ,Soufiane, El hachmi, Zakaria, Okab, Mohammed, Elteyeb et tous les amis de Bouraoui, El calaâ et de la mosquée,

À mes chères collègues que j'ai partagé avec eux une expérience riche en apprentissage et en partage en particulier Wadoud, Belhadj, Mahiou, Rostane, Hani, Ramzy, Redouane Oussama et mon binôme Yaniss,

À tous les gens que j'ai travaillé avec dans le bénévole et dans le travail associatif pendant ces 5 dernières années, spécialement l'équipe de l'association scientifique El-Maarifa avec toutes ses sections et l'équipe IEC,

À tous les détenus d'opinions et spécialement les étudiants qui n'ont pas eu la chance de vivre ce moment qu'ils le mérite,

À mes chères grands parents, que J'ai prié toujours pour le salut de leurs âmes. Puisse Dieu, le tout-puissant, les avoir en sa sainte miséricorde !

À tous ceux qui de près ou de loin, ont contribué à la réalisation de ce travail,

À tous ceux qui me sont chers, à vous tous,

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

”

- Yasser

Remerciements

“

Louange à Dieu seul, clément et miséricordieux,

D'abord, nous adressons de vifs remerciements à notre encadreuse Mme. ***F.Nibouche****, qui nous a donné l'opportunité d'approfondir nos connaissances au sein de Biopharm,*

Nos sincères remerciements vont ensuite à Mr. ***Mohammed Belmouhoub*** *qui nous a assisté durant toute la période du projet avec son soutien moral et ses précieux conseils qui nous ont permis de mener à bien notre projet de fin d'études, dans l'espérance qu'il soit à la hauteur de ses attentes.,*

Nous exprimons notre gratitude à l'ensemble du personnel de ***BIOPHARM*** *Industrie en particulier : Salim, Emir, Abdelkrim, Chakib qui nous ont aidés et permis de travailler dans un cadre et une ambiance agréables.*

Nous saisissons cette occasion pour destiner à nos enseignants es plus sincères ainsi qu'aux membres du jury qui nous font l'honneur d'évaluer notre travail.

Que toute personne ayant participé de près ou de loin à la réalisation de notre projet, veuillez trouver ici témoignage de gratitude et de remerciements.

”

- Yaniss & Yasser

Table des matières

Liste des tableaux

Table des figures

Liste des sigles et acronymes

AG	*algorithme Génétique*
BDD	*Base de données*
BDIS	*Biopharm distribution*
BIND	*Biopharm industrie*
CA	*Chiffre d'affaire*
DG	*Directeur Général*
DL	*Degrés de liberté*
EDD	*Earliest Due Date*
FJS	*Flexible Job shop*
FS	*Flow-shop*
GPAO	*Gestion de Production Assistée par Ordinateur*
HHI	*Human Health Information*
IPM	*Identical Parallel Machines*
IT	*Innovation Technologique*
JS	*Job-shop*
LLGV	*Ligne Liquide Grand Volume*
LLPV	*Ligne Liquide Petit Volume*
MMRCPSP	*Multi-mode resource constrained project scheduling problem*

MRP	*Manufacturing Resources planning*
OA	*Ordre d'approvisionnement*
OF	*Ordre de Fabrication*
PMP	*Parallel Machine Problem*
PSE	*Process System Engineering*
SD	*Sequence Dependence*
SI	*Systèmes d'information*
SMED	*single-minute exchange of die*
S&OP	*Sales and Operations*
SPT	*Shortest Processing Time*
UfPM	*Uniform Parallel Machines*
UrPM	*Unrelated Parallel Machines*
VIU	*Value In Use*
WSPT	*Weighted Shortest Processing Time*

Introduction générale

L'industrie pharmaceutique est l'une des industries ayant le potentiel de croissance le plus élevé, en effet, en seulement vingt ans elle a connu une croissance dépassant les **220%**. Celle-ci constitua une opportunité mais aussi une menace, car les flux matériels et informationnels devinrent incontrôlables.
L'Algérie à l'instar de ces pairs, n'échappe pas aux réalités du bouleversement dans le secteur pharmaceutique.
En effet, les algériens sont de plus en plus confrontés aux pénuries de médicaments causant une déstabilisation du système de soins qui se traduit par une **perte d'indépendance** sanitaire préoccupante pour l'Algérie.
Ceci fait de l'industrie pharmaceutique une industrie stratégique pour les nations et constitue une mine d'or pour les nouveaux investisseurs d'où l'environnement ultra compétitif que subissent les entreprises pharmaceutiques au niveau national et international.
Cette compétitivité poussa les entreprises à rechercher une réactivité et une flexibilité accrues afin d'augmenter le chiffre d'affaire de l'entreprise en s'accaparant des marchés de clients exigeants.
Afin d'y parvenir, une solution s'est démarquée de par son importance, sa simplicité et l'investissement quasiment nul qu'elle constitue, cette solution n'est autre que l'**ordonnancement**.

Étant un pilier de la gestion de la production, l'ordonnancement permet de planifier chaque campagne de production dans ses moindres aspects tout en se limitant aux ressources disponibles et en optimisant un indicateur aligné aux objectifs de l'entreprise.
Le processus d'ordonnancement est alors un processus clé qui, une fois maîtrisé permet d'avoir un réel avantage en terme de :

- **délais :** l'ordonnancement permet de réduire le "lead-time" par conséquent la production est plus flexible ;
- **coûts :** l'ordonnancement permet de "diluer" les coûts fixes en augmentant la production grâce aux temps dégagés ;
- **qualité :** l'ordonnancement permet d'avoir un meilleur taux de service en réduisant les délais de livraison ;

Biopharm, l'un des leaders du secteur pharmaceutique en Algérie a conscience de ces enjeux et de l'utilité de l'optimisation de la chaîne logistique.
A cet effet et dans le cadre de sa stratégie novatrice, elle souhaite mettre en place un

système d'ordonnancement qui puisse compléter les fonctionnalités GPAO de son ERP et permettre aux équipes du département production d'avoir un planning de production optimal.
Et pour cela, la direction informatique et systèmes d'information (DSI) de Biopharm nous a confié La conception et la réalisation de ce nouveau système d'information de gestion de l'ordonnancement dans le cadre de notre projet de fin d'études. La question centrale étant :

"Comment améliorer le processus actuel d'ordonnancement au sein de BIOPHARM à l'aide de la modélisation ?"

De cette question fondamentale découlent plusieurs sous-questions :

- Quel modèle d'ordonnancement est le plus adapté pour le cas BIOPHARM ?
- Quelles sont les étapes et la démarche de développement d'une solution mathématique et informatique pour l'ordonnancement et comment la mettre en oeuvre ?

Afin de répondre à ces questionnements nous avons organisé notre travail en 4 chapitres illustréspar la **figure 1** :

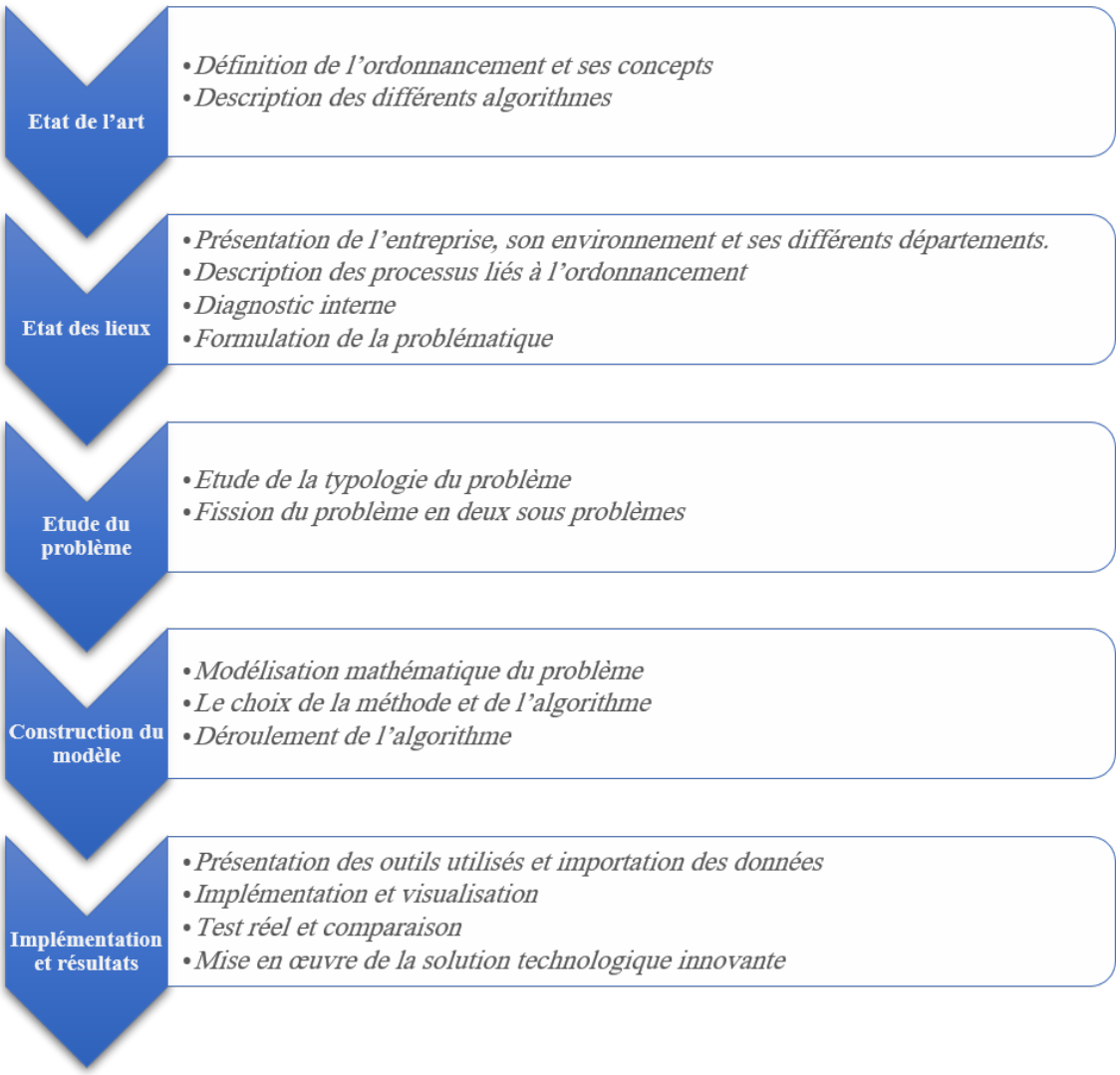


FIG. 1 : Structure du mémoire

Chapitre 1

État de l'art

1.1 Introduction

Les décennies passées ont vu la recherche opérationnelle sur les problématiques d'ordonnancement s'éloigner des méthodes exactes au profit d'autres stratégies permettant l'obtention de solutions très correctes dans un laps de temps très court. Ces méthodes sont appelées heuristiques, et elles permettent et permettent la résolution de problèmes fastidieux dans des milieux industriels à grande échelle.
Ce chapitre est alors consacré à l'état de l'art et à la revue bibliographique a pour objectif la présentation des outils et des concepts du cadre théorique utilisés dans la démarche adoptée pour les solutions proposées pour la résolution de la problématique.
Tout d'abord on va commencer par la définition du problème d'ordonnancement, de ses méthodes de résolution mentionnées dans la littérature en passant par les typologies du problème et ses classifications.
Nous présenterons ensuite on parle sur les deux méthodes de résolution, l'algorithme heuristique pour le flow shop problem (FSP) et une métaheuristique pour la résolution du flexible job shop problem (FJSP) qui ont été adoptées lors de la résolution de la problématique.
Ces outils et notions académiques nous permettront d'appuyer notre méthodologie afin d'apporter des solutions adéquates aux dysfonctionnements diagnostiqués.

Ce chapitre sera découpé en 6 sections comme illustré dans la **figure 1.1**

Fig. 1.1 : Structure du chapitre "État de l'art"

1.2 Définition et revue documentaire

Définition 1 : L'ordonnancement est un processus de prise de décision qui est utilisé régulièrement dans de nombreuses industries de services et de production. Il traite l'allocation des ressources aux tâches sur des périodes de temps données pour optimiser un ou plusieurs objectifs. (Pinedo 2016) (Castro, Grossmann et Q. Zhang 2018)

Définition 2 : L'ordonnancement consiste à organiser dans les temps la réalisation d'un ensemble de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches. Un ordonnancement décrit l'ordre d'exécution des tâches et l'allocation des ressources au cours du temps, afin de satisfaire un ou plusieurs critères d'optimisation.(Esquirol, Lopez et Lopez 1999)

Un bon programme d'ordonnancement répond de manière efficace aux besoins exprimés par l'industrie en augmentant l'efficacité des lignes de production tout en répondant aux objectifs qui peuvent être divers et variés (satisfaction des clients, les délais de livraisons, la minimisation des coûts, des changeovers ou du makespan..)(Aguirre et Papageorgiou 2018)(Harjunkoski et al. 2014)(Z. Li et Ierapetritou 2008).

L'avènement des nouvelles technologies de l'information ont fait en sorte que l'utilisation d'outils d'optimisation basés sur des modèles mathématiques et heuristiques ne soit plus une utopie que l'on trouve seulement dans les ouvrages théoriques et dans la recherche, il a permis à d'innombrables entreprises d'améliorer leurs processus de générer de la valeur et ceci est notamment vrai dans notre domaine de prédilection, la **supplychain**. (Goetschalckx 2011).

Malheureusement, les solutions sont rarement partagées car elles offrent un avantage certain par rapport aux concurrents et lorsque la publication des résultats se fait, ceux-ci sont trop spécifiques à l'industrie ou à l'entreprise en question rendant son utilisation pratiquement impossible. D'un autre côté, toute la littérature touchant de prés ou de loin l'ordonnancement est fortement **théorisée** créant un gap entre la communauté scientifique qui se consacre à des problèmes théoriques généraux et une industrie qui fait face à ses propres contraintes et à celles de son environnement.(Pierre Baptiste 2019) Dans l'ouvrage **"The complexity of flowshop and jobshop scheduling"** (Garey, Johnson et Sethi 1976) l'auteur explique que les problèmes industriels réels sont généralement testés sur des jeux de données de petite ou moyenne taille. Ceci est dû au fait que les problèmes d'ordonnancement font partie de l'ensemble de problèmes définis comme **NP-difficiles** et donc ne permettent pas une résolution rigoureuse directe.(G. P. Georgiadis, Elekidis et M. C. Georgiadis 2019).
Il est donc clair que dans ce chapitre, nous étudierons et expliciterons les méthodes heuristiques qui sont la tendance du moment, mais avant cela des définitions s'imposent.

1.3 Les éléments d'un problème d'ordonnancement

Quand on parle d'ordonnancement quatre notions principales interviennent : **les tâches, les ressources, les contraintes et les objectifs**.

1.3.1 Les tâches

Une tâche T_i est, par définition, une entité élémentaire de travail localisée dans le temps par une date de début S_i (start time) et une date de fin F_i (Finish time), dont la réalisation est caractérisée par une durée positive P_i (processing time) telle que :

$P_i = F_i$ - S_i Certaines caractéristiques relatives à l'exécution d'une tâche sont présentées ci-dessous :

- La date de début au plus tôt R_i (release date).
- La date de fin au plus tard D_i (due date).
- Un poids W_i (weight) qui représente le facteur de priorité qui dénote l'importance de la tâche i relativement aux autres (comme la profitabilité).

La figure 1.2 donne une représentation de la tâche en précisant désignant ses principales caractéristiques. On distingue deux types de tâches :

- Des tâches morcelables (Préemptibles) qui peuvent être exécutées par morceaux par une ou plusieurs ressources.
- Des tâches non-morcelables (indivisibles) qui doivent être exécutées en une seule fois et ne peuvent pas être interrompues avant qu'elles ne soient complètement achevées.

Dans notre cas le terme tâche correspond à une **opération** (granulation, compression, packaging...) et le terme **« travail »** ou **« job »** désigne le lot à fabriquer. (Habchi 2001)

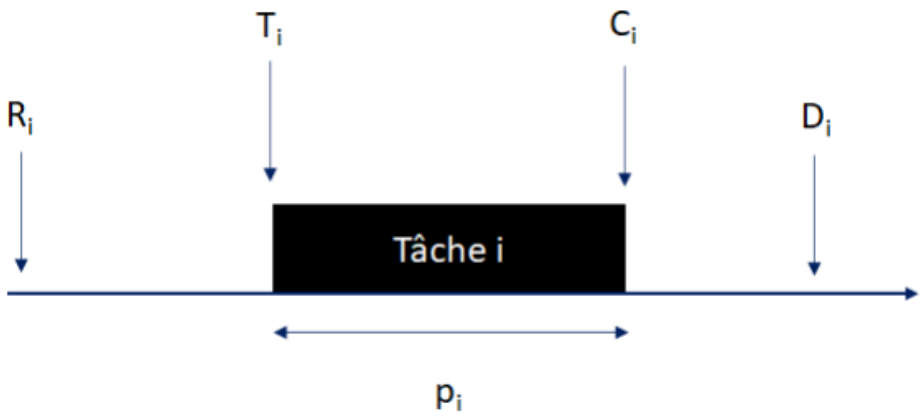


FIG. 1.2 : Caractéristiques d'une tâche

1.3.2 Les ressources

Une ressource "k" est un moyen technique ou humain nécessaire pour la réalisation d'une tâche (ESQUIROL, LOPEZ et LOPEZ 1999). Elle est disponible en quantité limitée et sa capacité est supposée constante, elle peut être :

- **Renouvelable**, si après son utilisation par une ou plusieurs tâches, elle est à nouveau disponible dans les mêmes quantités ;
- **Consommable**, si sa consommation globale au cours du temps est limitée comme le carburant par exemple ;
- **Les ressources doublement contraintes**, ces ressources combinent les contraintes liées aux deux catégories précédentes. Leur utilisation instantanée et leur consommation globale sont toutes les deux limitées. C'est le cas des ressources d'énergie.
- **Disjonctive (ou non partageable)**, si elle ne peut être allouée qu'à une tâche à la fois.
- **Cumulative (ou partageable)**, si elle peut être utilisée par plusieurs tâches simultanément.

1.3.3 Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement les variables représentant les relations reliant les tâches et les ressources. 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 la difficulté du problème d'ordonnancement augmente (Harrath 2003).
On peut diviser les contraintes en deux grandes familles (ibid.) :

1. **Les contraintes temporelles :**
   Sont des contraintes liées au temps et sa gestion, il en existe trois types :
   - **les contraintes de temps alloué :** issues généralement d'impératifs de gestion et relatives aux dates limites des tâches ou à la durée totale d'un projet.
   - **les contraintes d'antériorité :** et plus généralement les contraintes de cohérence technologique qui décrivent le positionnement relatif de certaines tâches par rapport à d'autres (les séquences)
   - **Les contraintes de calendrier :** liées au respect d'horaires de travail, etc.
2. **Les contraintes de ressources :**
   Traduisent le fait que les ressources sont disponibles en quantité limitée ou non et aussi si elles sont de nature disjonctive ou cumulative.
   - **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.

1.3.4 La fonction objectif

Tout ordonnancement est guidé par un ou plusieurs objectifs à optimiser. De manière générale la littérature propose la classification suivante (Esquirol, Lopez et Lopez 1999) :

- **Les objectifs liés au temps :** On trouve par exemple, la minimisation du temps total d'exécution (le makespan), du temps moyen d'achèvement ou des durées totales des retards.
- **Les objectifs liés aux ressources :** par exemple, maximiser la charge d'une ressource ou minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches.
- **Les objectifs liés au coût :** minimiser les coûts, de lancement, de production, de stockage, de transport, etc.
- **Les objectifs liés à l'énergie ou au débit :** réduire la consommation d'énergie et augmenter la vitesse d'exécution des tâches.

Généralement, l'objectif souhaité dans un problème d'ordonnancement quelconque définit le critère à optimiser. Lorsque cet objectif adopte un seul critère, il s'agit bien du cas d'optimisation **mono-objectif**, le cas le plus fréquent dans la littérature des problèmes

d'ordonnancement. Cependant, lorsque l'objectif adopte simultanément plusieurs critères à la fois, nous parlons dans ce cas d'optimisation **multi-objectifs**. Cette dernière est devenue de plus en plus importante face à l'évolution et à la concurrence des systèmes de production (Hentous 1999).

1.4 La typologie des problèmes d'ordonnancement

Les problèmes d'ordonnancement sont divers et variés. Il fallait alors trouver le moyen de les départager en relevant leurs différences. C'est pour cela que la littérature propose une typologie commune dépendante basée sur plusieurs paramètres tels que la nature des machines, l'ordre d'enchaînement des opérations et toutes sortes de contraintes temporelles et structurelles.

Un problème d'ordonnancement est décrit par un triplet $\alpha|\beta|\gamma$ où (Graham et al. 1979) :

1. **Le premier champ α :**
   Ce champ décrit l'environnement et l'organisation de l'atelier et ne contient qu'une seule entrée.

2. **Le second champ β :**
   Ce champ fournit des détails sur les caractéristiques et les contraintes de traitement et peut ne contenir aucune ou plusieurs entrées.

3. **Le dernier champ γ :**
   Ce champ décrit l'objectif à optimiser et contient souvent une seule entrée.

Nous allons détailler le contenu de chaque champ afin de comprendre comment utiliser la notation.

1.4.1 Les paramètres du champ α

Ce champ est très important car le type de flux traversant les lignes de production ont des contraintes propres à elles qu'il faut prendre en considération.La première question importante qui se pose lorsqu'on est dans un atelier est :

Combien a-t-on de machines/opérations sur la ligne ?

La deuxième question qui suit la première est :

Comment s'écoule le flux à travers cette ligne ?

Deux questions fondamentales auxquelles nous allons répondre dans cette section pour préciser les valeurs prises par le paramètre α.

Problème à une étape

Les problèmes à une opération sont des problèmes où on a une seule étape qui permet la transformation de la matière en produit fini.

- **Problèmes à une machine :**
  Les problèmes d'atelier à une machine (single machine problem) consistent à ordonnancer, sur une seule machine, des jobs faisant appel à une seule opération.
- **Problèmes à machines parallèles :**
  Les problèmes d'atelier à machines parallèles (PMP) sont une généralisation des problèmes à une machine où l'on transforme chaque étape en une machine virtuelle.
  Ce type d'atelier se caractérise par le fait que chaque opération peut être réalisée par n'importe quelle machine disposée en parallèle au sein d'une même étape. Le problème d'ordonnancement consiste donc à déterminer l'affectation et le séquencement des opérations sur chaque machine.
  Bien entendu, les machines au sein d'une même étape peuvent différer. C'est pour cela que l'on peut énumérer :
  - **Machines parallèles identiques (identical parallel machines) :** Toutes les caractéristiques des machines au sein d'une même étape sont identiques.
  - **Machines parallèles uniformes (uniform parallel machines) :** la durée d'exécution des opérations varie uniformément en fonction de la performance de la machine choisie.
  - **Machines parallèles indépendantes (unrelated parallel machines) :** les durées opératoires dépendent complètement des machines utilisées.

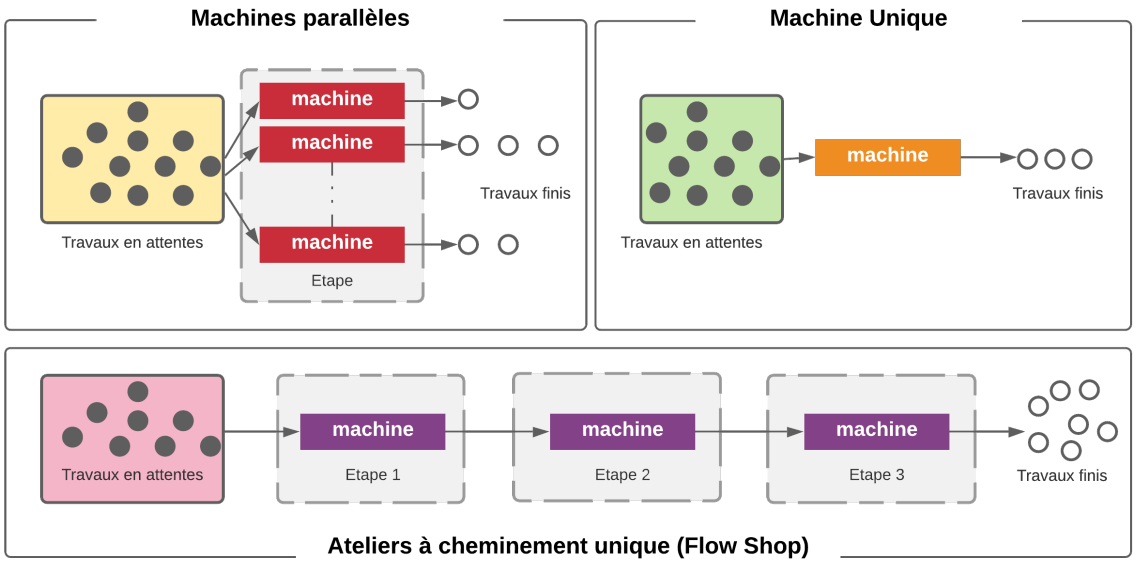


FIG. 1.3 : Typologie par machine

Problèmes à plusieurs étapes

Les problèmes de la deuxième catégorie sont dits problèmes d'atelier du fait de la nécessité du passage de chaque job sur deux ou plusieurs machines dédiées. Suivant le mode de passage des opérations sur les différentes machines, trois types d'ateliers se distinguent à savoir :

- **Problèmes Flow-Shop :**
  Les ateliers de type Flow-Shop appelés également ateliers à cheminement unique sont des ateliers constitués d'un ensemble de machines disposées en séries acheminant un **flux unidirectionnel**.
  En d'autres termes, tous les produits passants par une ligne de production organisée en FS doivent suivre un ordre de passage bien défini entre les différentes machines (contrainte de précédence).

- **Problèmes Job-Shop :**
  Dans les ateliers en Job-shop aussi appelés ateliers à cheminements multiples, chaque produit passe sur des machines/étapes dans un ordre fixé, mais contrairement au Flow-Shop, cet ordre peut être différent pour chaque produit, c'est ce qu'on appelle un **flux multidirectionnel**.

- **Problèmes Open-Shop :**
  Dans les ateliers en Open-shop aussi appelés ateliers à cheminement libre, les gammes opératoires des différents produits ne sont pas fixées a priori contrairement au problème d'atelier Job-Shop. Les opérations d'un même job peuvent donc être exécutées dans un ordre quelconque. Le problème consiste d'une part à déterminer le cheminement de chaque produit et d'autre part à ordonnancer les produits en tenant compte des gammes trouvées.

Bien entendu des formes hybrides et particulières existent comme par exemple :

- Job-Shop flexible (FJSP) où chaque étape contient plusieurs machines ;
- Flow-shop flexible (FFSP) où chaque étape contient plusieurs machines ;
- Flow-shop flexible (FFSP) où chaque étape contient plusieurs machines ;
- • Les problèmes d'ordonnancement de projets qui sont caractérisés par des ressources consommables, l'allocation de ressources rares à des activités concurrentes et des contraintes qui ajoutent une complexité significative à la planification, connue sous le nom de **"problème de planification de projets à ressources limitées" (RCPSP)** et représentant un problème NP-difficile au sens fort.

Une chose est sûre, la détermination de l'organisation de l'atelier et du le flux de produits le traversant est primordiale car elle permet de faciliter l'identification des contraintes et une recherche documentaire plus efficace, ce qui donne de solides bases pour la construction d'un modèle.

La figure 1.4 illustre les typologies des problèmes à une étape.

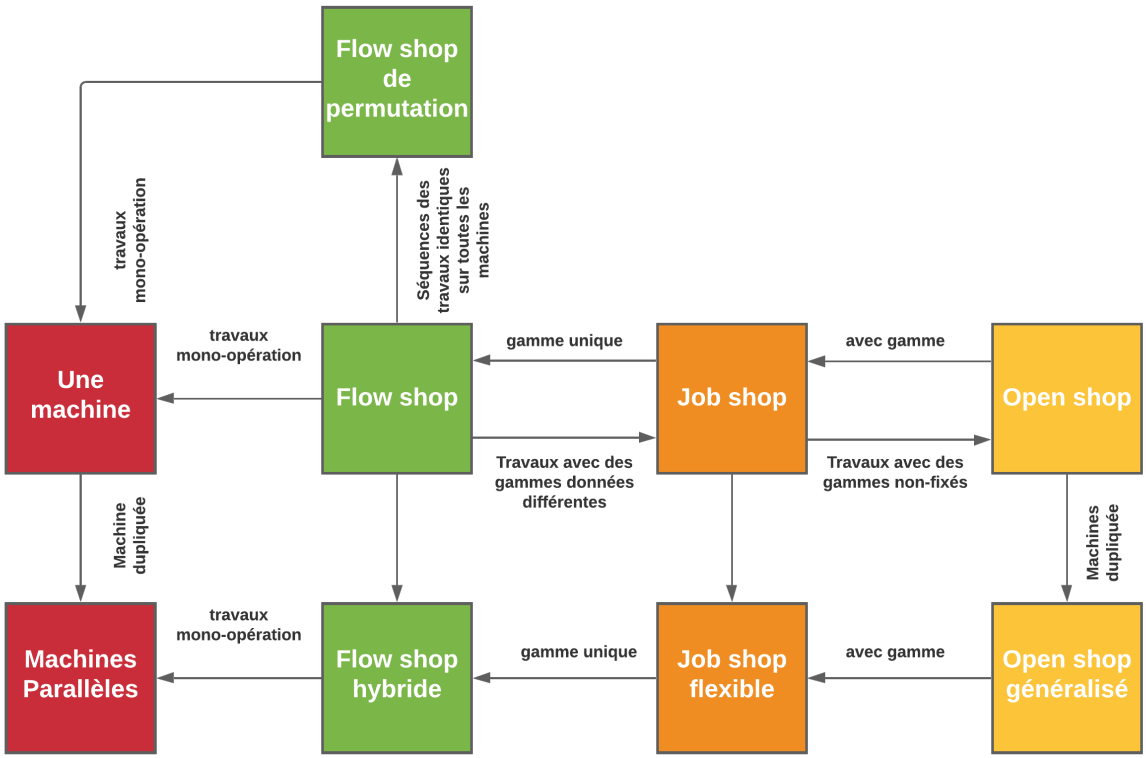


FIG. 1.4 : Typologie des problèmes d'ordonnancement

1.4.2 Les paramètres du champ β

Dans ce champ, on détermine les caractéristiques et les contraintes de la ligne de production qu'on peut énumérer comme suit :

- **La préemption** $(prmp)$ **:** La préemption implique qu'il n'est pas nécessaire de conserver un job sur une machine, une fois commencé, jusqu'à sa fin.
  Le planificateur est alors autorisé à interrompre le traitement d'une opération, à mettre un travail différent sur la machine puis revenir à l'opération interrompue plus tard.

- **Les contraintes de précédence**$(prec)$ **:** Nécessitant qu'un ou plusieurs jobs soient terminés avant que le job qui leur succède ne soit autorisé à démarrer son traitement.

- **Temps de configuration dépendante de séquence** $(S_{j,k})$ **:**Le $S_{J,K}$ représente la durée de configuration dépendante de la séquence, qui nécessite un temps de changement de série (changeover ou setup time) pour préparer la machine pour le lot prochain.

- **Familles de jobs** $(fmls)$ **:** Les jobs appartiennent dans ce cas à F différentes familles de jobs. Les travaux d'une même famille peuvent avoir des temps de traitement différents, mais ils peuvent être traités sur une machine l'un après l'autre sans nécessiter de configuration intermédiaire (pas de changeover). Cependant, si la machine passe d'une famille à une autre, disons de la famille g à

la famille h, alors une configuration est nécessaire. Si ce temps d'établissement dépend à la fois des familles g et h et dépend de la séquence, alors il est noté $s_{g,h}$.

- **Traitement par lots** ($batch(b)$) **:** Une machine peut être capable de traiter un certain nombre de jobs, disons b jobs, simultanément ; c'est-à-dire qu' elle peut aller d'un lot b jobs en même temps.

- **Pannes** ($brkdwn$) **:** Les pannes de machines impliquent qu'une machine peut ne pas être disponible en permanence.

- **Restrictions d'éligibilité des machines** (M_j) **:** Apparaît lorsque l'environnement de la machine est composé de m machines en parallèle (P_m). Lorsque M_j est présent, toutes les machines m ne sont pas capables de traiter la tâche j. L'ensemble M_j désigne l'ensemble des machines pouvant traiter le travail j. Si le champ ne contient pas M_j , le travail j peut être traité sur n'importe laquelle d'entre elles.

- **Permutation** ($prmu$) **:** Une contrainte qui peut apparaître dans l'environnement du flow shop est que les files d'attente devant chaque machine fonctionnent selon la discipline First In First Out (FIFO). Cela implique que l'ordre dans lequel les travaux passent par la première machine est maintenu pour tout le système.

- **Blocage** ($blocage$) **:** Le blocage est un phénomène qui peut se produire dans les magasins de flux. Si un magasin de flux a un stock tampon limité entre deux machines successives, il peut arriver que lorsque le stock tampon est plein, la machine en amont ne soit pas autorisée à libérer un travail terminé.

- **Recirculation** ($rcrc$) **:** La recirculation peut se produire lorsqu'un produit peut visiter une machine ou un centre de travail plus d'une fois.

1.4.3 Les paramètres du champ γ :

L'objectif à minimiser est toujours une fonction temporelle. La date de sortie du job(produit) j de la machine k est notée $C_{j,k}$.
Le temps ou le job j sort du système (produit fini) est noté C_j.
La fonction peut aussi être une fonction des échéances. Le retard (Lateness) du job j est défini comme :

$$L_j = C_j - d_j$$

Qui est positif lorsque le travail j est terminé en retard et négatif lorsqu'il est terminé en avance.
Le retard du travail j est défini comme :

$$T_j = max(C_j - d_j, 0) = max(L_j, 0)$$

Des exemples de fonctions objectif possibles peuvent être citées :

- **Makespan (C_{max})** : Le makespan, défini comme $max(C_1, ..., C_n)$, est équivalent au temps de fin d'exécution de la dernière tâche à quitter le système. Un makespan minimale de fabrication implique généralement une bonne exploitation des machines.

- **Retard maximum (L_{max})** Le retard maximum, L_{max}, est défini comme $max(L_1, ..., L_n)$. Il mesure la pire violation des dates d'échéance.

- **Temps d'achèvement total pondéré ($\sum w_j C_j$)** La somme des temps d'achèvement pondérés des n tâches donne une indication du total des coûts de détention ou d'inventaire encourus par le plan d'ordonnancement. La somme des temps de réalisation est souvent appelée dans la littérature le temps d'écoulement (flow time). Le temps d'achèvement total pondéré est alors appelé temps d'écoulement pondéré (flow time pondérée).

- **Retard total pondéré ($\sum w_j T_j$)** Il s'agit également d'une fonction de coût plus générale que le temps d'achèvement total pondéré.

1.5 Les méthodes de résolution d'un problème d'ordonnancement

Le problème d'ordonnancement étant un problème d'optimisation, une méthode d'optimisation est nécessaire pour le résoudre. Dans son ouvrage "Metaheuristics"(Talbi 2009) E.Talbi divise les méthodes de résolution en deux grandes familles :

1.5.1 Les méthodes exactes

Ces méthodes sont généralement utilisées pour résoudre des problèmes de petite taille où le nombre de combinaisons est suffisamment faible pour pouvoir explorer l'espace des solutions en un temps raisonnable.
Ces méthodes assurent toujours l'optimalité de la solution trouvée. Elles garantissent donc la complétude de la résolution mais à coût élevé. Leur inconvénient majeur est leur complexité exponentielle de résolution qui croît en fonction de la taille du problème.

Plusieurs auteurs classifient les méthodes de résolution exactes en **efficaces** et **énumératives**.

Les méthodes exactes efficaces

Ce type de méthodes garantissent, pour un problème et un critère donné, la détermination d'une solution optimale en un temps de calcul polynomial. L'application de telles méthodes n'est possible que pour des classes réduites de problèmes d'ordonnancement. Parmi les plus connues, nous citons les méthodes dédiées aux ateliers à une machine :

- La règle SPT (Shortest Processing Time).
- La règle WSPT (Weighted Shortest Processing Time).
- La règle EDD (Earliest Due Date).

Les méthodes exactes énumératives

Les méthodes de ce type les plus connues utilisées pour les problèmes d'ordonnancement sont :

- La programmation linéaire ;
- La programmation dynamique ;
- Les procédures par séparation et évaluation (Branch and Bound) ;

1.5.2 Les méthodes approchées

Ces méthodes sont utilisées pour traiter des problèmes que les méthodes optimales sont incapables de résoudre en un temps acceptable. Elles produisent généralement une solution faisable assez proche de l'optimum en un temps relativement court. Nous distinguons les heuristiques et les méta-heuristiques.

Les heuristiques

Ce sont les méthodes constructives et les méthodes de recherche locale, où dans chaque itération une solution partielle est complétée pour obtenir une solution finale.
Parmi les méthodes itératives ou de recherche locale qui ont prouvé leur efficacité dans la résolution de problèmes d'optimisation combinatoire, nous pouvons citer les méthodes ascendantes et descendantes.

Les méta-heuristiques

Les méta-heuristiques n'étant pas, à priori, spécifiques à la résolution d'un type particulier de problèmes, leur classification reste assez arbitraire. Nous pouvons cependant distinguer deux approches, la première est basée sur une solution unique et la seconde sur une population.

- Les approches basées sur une solution unique (ou trajectoire) sont des algorithmes qui partent d'une solution initiale (obtenue de façon exacte, ou par tirage aléatoire) et s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif dans l'espace des solutions. Dans cette catégorie, sont classés :
  - Le recuit simulé.
  - La méthode Tabou.

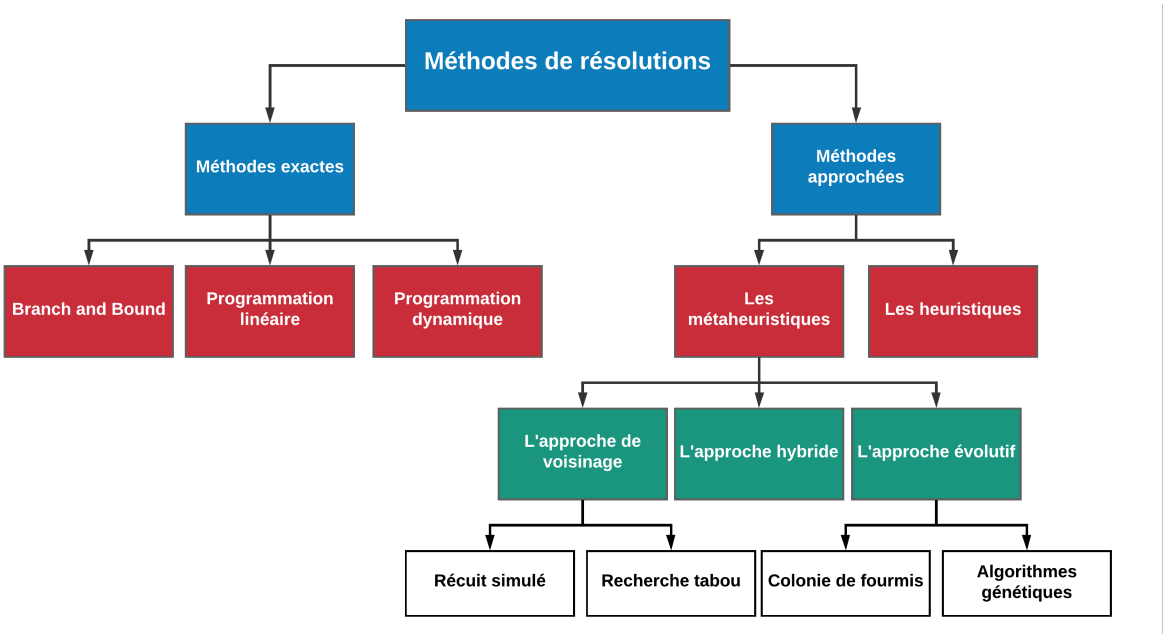


FIG. 1.5 : Méthodes de résolution de problème d'ordonnancement

  - La recherche dispersée (en anglais Scatter search).

- Les approches basées sur une population (ou révolutionnaires) quant à elles consistent à travailler avec un ensemble de solutions simultanément, que l'on fait évoluer graduellement.
  L'utilisation de plusieurs solutions simultanément permet naturellement d'améliorer l'exploration de l'espace des configurations. Dans cette seconde catégorie, on recense :

  - Les algorithmes génétiques.
  - Les algorithmes par colonies de fourmis.
  - L'optimisation par essaim particulière.
  - Les systèmes immunitaires artificiels.

1.6 Classification des problèmes d'ordonnancement (classification de Mendez)

Il y a un grand nombre d'aspects à prendre en considération afin de développer une modélisation d'ordonnancement systémique
Pour cela une feuille de route a été proposée par Mendez dans son article "review of optimization methods for short-term scheduling of batch processes" (MÉNDEZ et al. 2006) pour identifier les aspects les plus pertinents à prendre en considération lors de la modélisation.
La feuille de route montrée dans la **figure 1.6** présente 13 aspects chacun d'entre eux ayant un rôle lors de la formulation du problème.

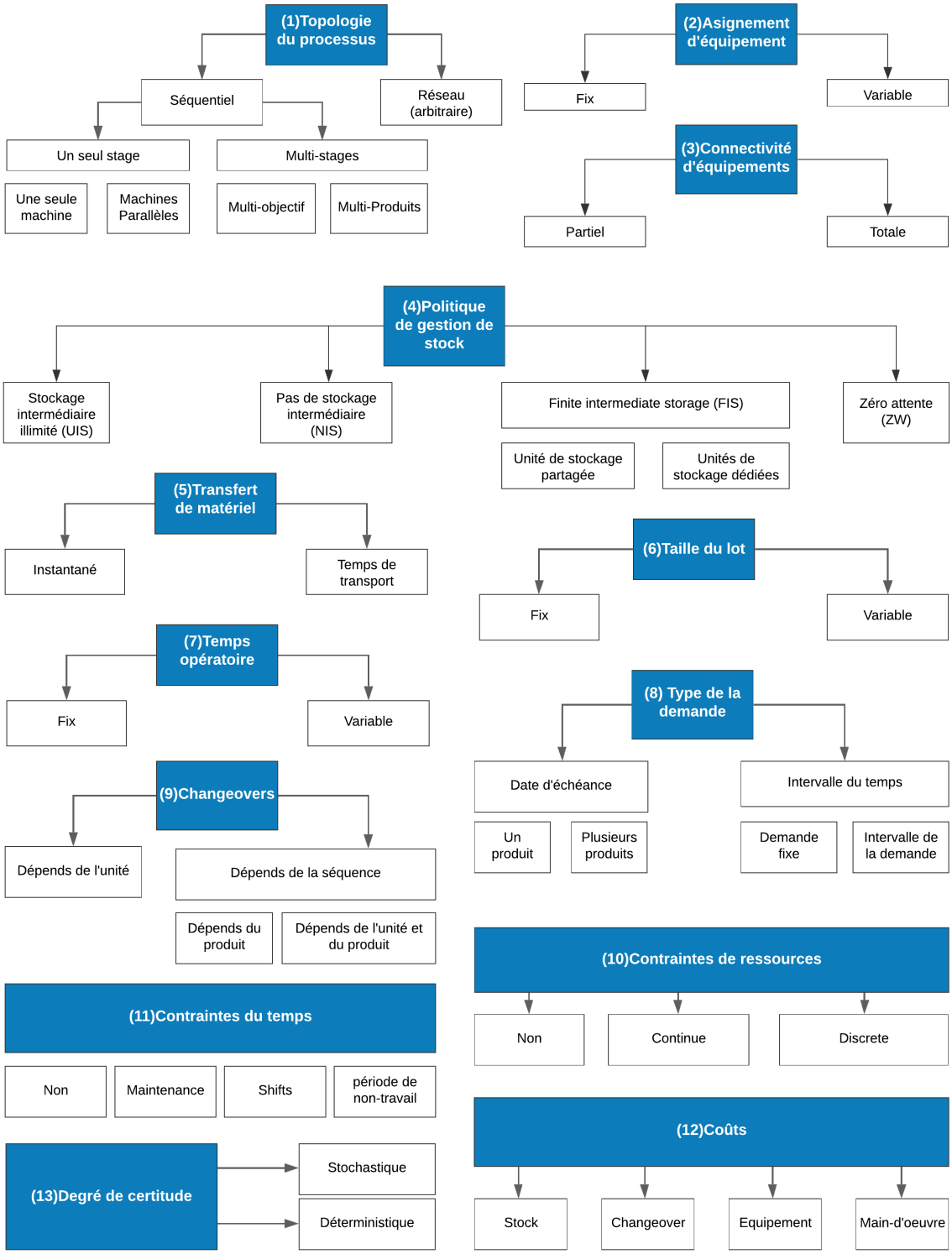


FIG. 1.6 : Feuille de route pour l'ordonnancement des lots de production.

Cette feuille de route peut se montrer très utile lors de la conception d'un modèle de production complexe, surtout du point de vu implementation.

1.7 Revue bibliographique du FJSP

Le problème d'ordonnancement d'atelier flexible (FJSP) est une extension du problème d'ordonnancement d'atelier classique (JSP) dans lequel les opérations peuvent être effectuées par un ensemble de machines candidates aptes à les effectuer.

Depuis les années 90, le FJSP a fait l'objet de nombreuses recherches.
Plusieurs chercheurs ont développé un nombre important de techniques allant des formulations mathématiques aux méta-heuristiques. Certaines formulations notables ont été examinées. En effet, Lee et al(Lee, Jeong et Moon 2002) ont été les premiers à introduire une formulation mathématique pour minimiser le temps de travail en tenant compte de la flexibilité du processus, de l'externalisation (outsourcing) et des dates d'échéance.
Plus tard, Fattahi et al. (Fattahi, Mehrabad et Jolai 2007) ont présenté un modèle de programmation linéaire mixte en nombres entiers (MILP) pour minimiser les makespan.
Özguven et al. (Özgüven, Özbakir et Yavuz 2010) ont établi deux modèles MILP. Le premier modèle traite de la minimisation de makespan sur le FJSP. Ce modèle a été comparé au au modèle présenté par Fattahi et al (Fattahi, Mehrabad et Jolai 2007). Le second modèle a été construit en analysant le FJSP et en y en ajoutant la flexibilité du processus.
Roshanaei et al. (Roshanaei, Azab et ElMaraghy 2013) ont étudié deux modèles MILP pour le FJSP. La minimisation de makespan a été utilisée comme critère d'optimisation. Ils ont analysé leurs modèles en examinant les données proposées par Fattahi et al. (Fattahi, Mehrabad et Jolai 2007), et ils ont étudié un problème industriel dans un atelier de fabrication de moules et de matrices.
Shen et al. (Shen, Dauzère-Pérès et Neufeld 2018) ont travaillé sur la construction d'un MILP pour adresser le FJSP avec des temps d'installation dépendants de la séquence pour minimiser les makespan.

Le JSP s'est avéré être un problème NP-difficile (Garey, Johnson et Sethi 1976). L'intégration du sous-problème de routage dans la JSP a augmenté la complexité du problème. En raison de la caractéristique NP-difficile du FJSP, les approches exactes n'ont pas été en mesure de trouver une solution pour les instances FJSP à grande échelle (De Giovanni et Pezzella 2010).
Pour cette raison, les méthodes méta-heuristiques ont été principalement utilisées pour résoudre les FJSP, dont beaucoup ont été présentées dans la littérature telles que les algorithmes évolutionnaires, l'intelligence en essaim, la recherche en un seul point, algorithmes hybrides (HA), etc.

La plupart de ces études ont porté sur le FJSP général ; cependant, le FJSP-SD avec les séquences dépendantes qui tiennent en compte les changeovers après chaque changement de compagne (type ou famille de produit par exemple) a été étudié par quelques chercheurs seulement. (J. C. Chen et al. 2012)(Defersha et M. Chen 2010)(Guimaraes et Fernandes 2006)(J.-q. Li et al. 2020)(G. Zhang et al. 2020).

1.8 Les algorithmes génétiques

Les algorithmes génétiques (AG) font partie de la famille des algorithmes évolutionnaires inspirés par l'évolution biologique des espèces. Ils ont été introduits par Holland de l'université du Michigan en 1975 dans l'ouvrage "Adaptation of Natural and Artificial System". L'utilisation de ces algorithmes pour résoudre des problèmes d'optimisation a été développée initialement par Goldberg et cela créa un engouement de taille au sein la communauté scientifique. Ces algorithmes génétiques se basent sur le concept d'une évolution biologique, dans laquelle la forme physique de l'individu détermine sa capacité de survivre et de se reproduire.

Afin de bien expliquer son fonctionnement, on détaillera pas à pas son déroulement :

Le codage :

Le codage est une fonction qui permet de passer de la donnée réelle du problème traité à la donnée utilisée par l'algorithme génétique. Le choix du codage est l'élément le plus important dans la conception de l'algorithme puisqu'il permet d'une part de représenter les données, les paramètres et les solutions et il influe sur la mise en œuvre des opérations génétiques tel que le croisement et la mutation qui influent directement sur le bon déroulement de l'algorithme génétique et de son convergence vers la bonne solution d'autre part. Plusieurs types de codage sont utilisés pour coder les individus, on distingue (LARIBI 2018) :

- **Codage binaire :** ce codage a été le premier à être utilisé dans les algorithmes génétiques, chaque gène peut prendre seulement les valeurs 0 ou 1;
- **Codage réel :** contrairement au codage binaire, le codage réel associe à chaque gène une valeur réelle ;
- **Codage de permutation :** dans ce codage, les chromosomes contiennent typiquement une séquence de gènes où les gènes sont des nombres entiers ou des lettres, dans laquelle l'ordre est significatif. Ce type de codage est bien adapté aux problèmes d'ordonnancement.

La population initiale :

Une fois le codage choisi, une population initiale formée de solutions admissibles du problème doit être générée.
Cette étape est très importante, car elle affecte la qualité de la solution d'une part et le nombre de générations au bout duquel nous obtenons une solution satisfaisante. Plusieurs mécanismes de génération de la population initiale sont utilisés dans la littérature. La méthode classique étant celle qui consiste à **générer aléatoirement** les individus constituant la population initiale. Cette méthode répond à la nécessité d'avoir une population variée permettant d'explorer des zones diverses de l'espace de recherche. Cependant, la population peut aussi être entièrement ou partiellement générée à l'aide

de méthodes heuristiques et cela dans le but d'accélérer la convergence de l'algorithme génétique (LARIBI 2018).

L'évaluation par la fonction fitness :

Cette partie est déroulée lors de l'initialisation et à la création de chaque nouvelle génération, c'est une sorte de **fonction d'évaluation** qui est introduite afin de mesurer la performance de chaque individu de la population. Elle permet de quantifier la capacité d'un individu à survivre en lui affectant un poids couramment appelé **« fitness »**(ibid.).

La sélection :

La sélection est un procédé qui permet d'identifier les individus répondant au mieux au problème étudié, c'est-à-dire, que chaque individu est choisi en fonction de sa valeur d'évaluation. Ce procédé ne permet pas de créer de nouveaux individus, mais de privilégier les individus en fonction de leur valeur d'évaluation, et cela, afin de se présenter au croisement et à la mutation (reproduction). Dans l'algorithme génétique, il existe de nombreuses techniques de sélection nous mentionnons (ibid.) :

- **Sélection par roulette** : pour chaque individu, la probabilité d'être sélectionné est proportionnelle à son adaptation au problème. Afin de sélectionner un individu, on utilise le principe de la roue de loterie biaisée. Cette roue est une roue de la fortune classique sur laquelle chaque individu est représenté par une portion proportionnelle à son adaptation, son "fitness". La roue étant lancée, l'individu sélectionné est celui sur lequel la roue s'est arrêtée. Cette méthode favorise les meilleurs individus, mais tous les individus conservent néanmoins des chances d'être sélectionnés.
- **Sélection par rang** : cette sélection permet de classer la population suivant la fonction d'adaptation, chaque individu de la population se voit accorder un rang. Plus l'individu est bon, plus son rang est élevé. La différence entre la sélection par roulette est que la proportion est calculée sur les rangs et non sur la valeur de la fonction d'adaptation.
- **Sélection par tournoi** : cette méthode consiste à choisir aléatoirement deux ou plusieurs individus de la population, puis à sélectionner le meilleur individu dans ce groupe en fonction de son fitness. Ce processus est répété à chaque fois, avec ou sans remise, jusqu'à ce que le nombre d'individus nécessaires pour le croisement soit atteint

Le croisement :

L'opération de croisement permet de simuler la reproduction d'individus pour créer de nouvelles solutions. Le croisement consiste donc à générer à partir de deux individus sélectionnés (parents), un ou deux nouvelles solutions fils composés chacune d'une partie des caractéristiques de leurs parents.
Le croisement se fait généralement avec des opérateurs de croisement et le choix d'un

opérateur dépend du codage adapté et des caractéristiques du problème traité. Parmi les opérateurs de croisement les plus connus dans la littérature, nous citons (LARIBI 2018) :

- **Croisement à un point** : cette opérateur consiste à diviser chacun des deux parents en deux parties à la même position p (1 p l) choisie aléatoirement avec l la longueur du chromosome et à recopier la partie inférieure du parent 1,...,p à l'enfant et à compléter les gènes manquants de l'enfant à partir de l'autre parent en maintenant l'ordre des gènes.
- **Croisement à deux points** : ce type de croisement est utilisé en choisissant aléatoirement deux points de coupure pour dissocier chaque parent en 3 fragments.
- **Croisement uniforme** : Dans un croisement uniforme, nous ne divisons pas le chromosome en segments, nous traitons plutôt chaque gène séparément. En cela, nous tirons essentiellement une pièce pour chaque chromosome pour décider s'il sera ou non inclus dans le nouveau enfant.

La mutation :

La mutation permet de maintenir la diversité qui empêche la **dégénérescence** des solutions entre elles. L'opérateur de mutation apporte donc aux algorithmes génétiques la propriété d'érgodicité de parcours d'espace de solutions. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace de recherche, sans pour autant tous les parcourir.
L'opérateur de mutation joue le rôle d'élément perturbateur qui change les individus au niveau des gènes de manière stochastique (avec une certaine probabilité p_m appelée la probabilité ou le taux de mutation). Nous discernons : (ibid.) :

- **Mutation par échange** : cet opérateur de mutation permet de sélectionner deux gènes et les inter-changer.
- **Mutation par insertion** : cet opérateur consiste à sélectionner au hasard un gène et une position dans le chromosome à muter, puis à insérer le gène en question dans la position choisie.
- **Mutation par inversion** : cet opérateur consiste à choisir aléatoirement deux points de coupure et inverser les positions des gènes situés au milieu.

Remplacement :

L'opérateur de **remplacement** consiste à choisir les individus qui vont constituer la génération suivante. Deux principaux types sont distingués dans la littérature (ibid.) :

- **Remplacement stationnaire**
- **Remplacement élitiste**

L'algorithme continue son déroulement jusqu'à atteindre un critère d'arrêt ou un état de stagnation de la population (boucle fermée).(ibid.)

1.9 Conclusion

Dans ce chapitre nous avons présenté les problèmes d'ordonnancement, la formulation du problème, la typologie, les méthodes de résolution et leurs classifications selon les fonctionnalités.
Nous avons aussi découvert que nous pouvions le classer parmi l'ensemble des problèmes NP-Difficile, ce faisant certaines solutions sont adaptées du point de vu complexité tandis que d'autres sont inapplicables.
Dans ce mémoire nous aurons recours au concepts vu dans ce chapitre afin de cerner le type de problème qui sera observé au sein des ateliers de BIOPHARM, et choisir la solution la plus adaptée.
Mais avant cela, il faut d'abord analyser le contexte de l'entreprise et ses objectifs. Une étude de l'existant s'impose !

Chapitre 2

État des lieux

2.1 Introduction

Dans le présent chapitre nous aurons un développement en entonnoir où nous commencerons par introduire le secteur pharmaceutique en général puis nous étudierons le cas particulier de Biopharm Algérie. Passant après par un diagnostic interne et externe contenant une étude de l'existant, une analyse des processus cibles afin d'assimiler les contraintes, le savoir-faire et détecter les besoins. Nous finirons par une formulation une problématique qui sera traitée dans les trois chapitres qui suivent.
La **figure 2.1**

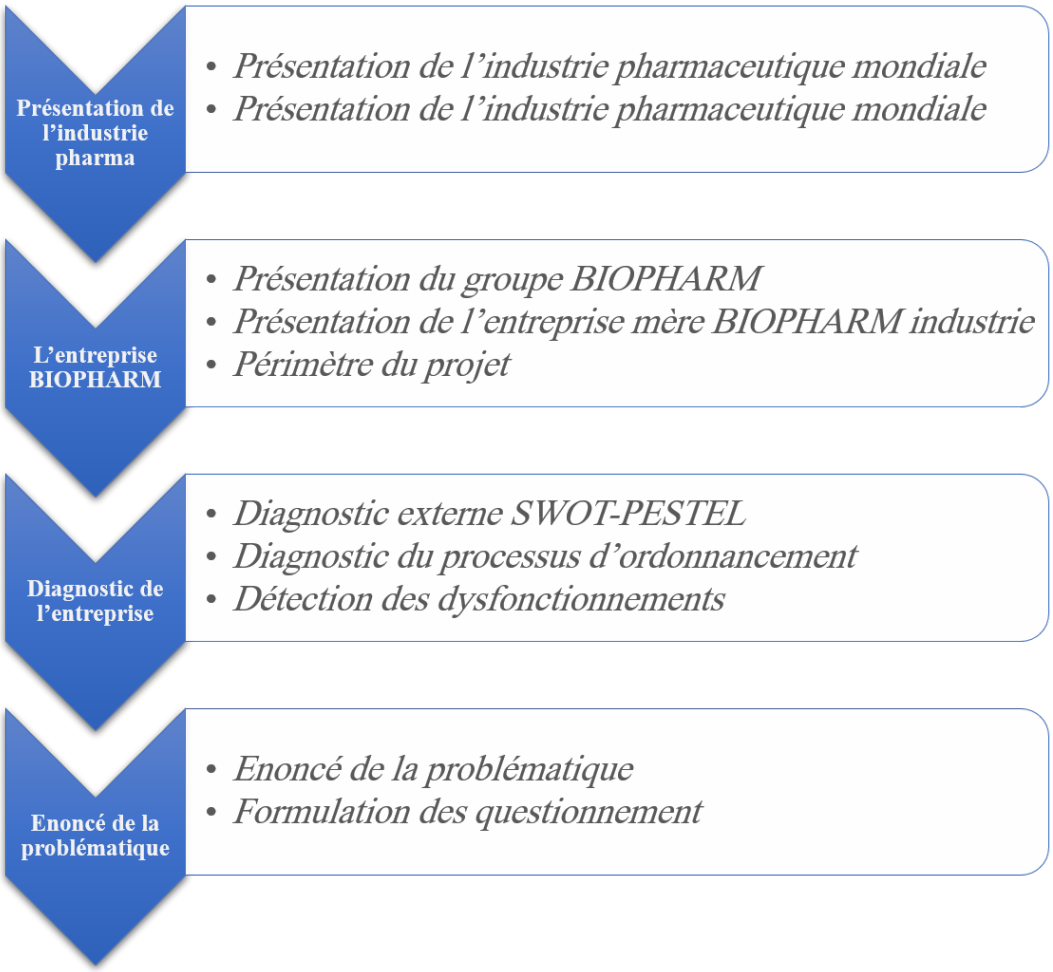


Fig. 2.1 : Structure du chapitre "État des lieux"

2.2 Présentation de l'industrie

2.2.1 L'industrie pharmaceutique mondiale

L'industrie pharmaceutique est un secteur économique et stratégique qui regroupe les activités de recherche, de fabrication et de commercialisation des médicaments pour la médecine humaine. Ayant un potentiel de développement toujours revu à la hausse, cette

industrie est vue comme l'une des industries les plus rentables économiquement.

Pour illustrer le potentiel de croissance de cette industrie, nous nous baserons sur une étude de Statista (Statista p. d.) qui précise le chiffre d'affaires du secteur pharmaceutique durant les deux dernières décennies comme présenté sur l'histogramme de la figure 2.2. En 2019 ce chiffre est évalué à 1250 Mds de dollars soit 3.7 pourcent de plus qu'en 2018.

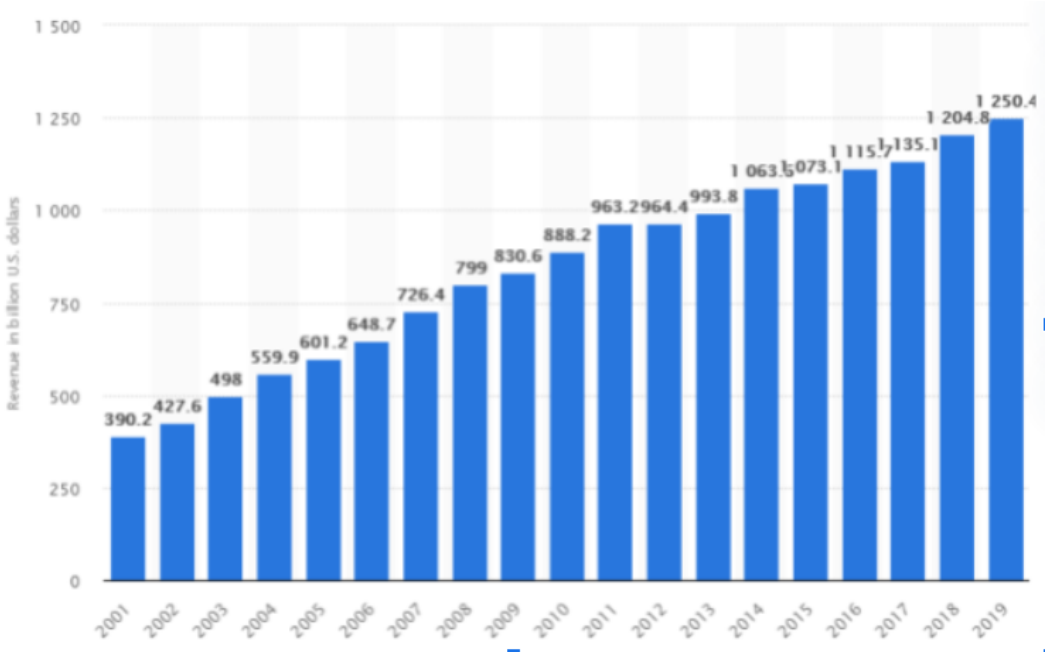


Fig. 2.2 : Chiffre d'affaire globale par année pour l'industrie pharmaceutique (Statista p. d.)

Son marché est en expansion en raison de l'amélioration de l'accès aux soins, de l'apparition de nouvelles maladies plus complexes, et de l'augmentation de l'espérance de vie qui fait apparaître de nouveaux besoins médicaux pour une catégorie de population de plus en plus importante. Cette industrie est très mondialisée, elle est dominée par des groupes d'envergure mondiale, appelés « global players » ou « big pharmas » qui poursuivent une stratégie de développement mondial en utilisant des stratégies collaboratives comme c'est le cas avec Biopharm, ou des stratégies de pénétration de marché afin de s'accaparer plus de parts.

2.2.2 L'industrie pharmaceutique en Algérie

En Algérie, le secteur de l'industrie pharmaceutique est porteur. Il est dominé par des grands laboratoires multinationaux dont les investissements les plus importants sont présentés dans le tableau 2.1 :

Tab. 2.1 : Investissements pharmaceutiques les plus importants en Algérie

Entreprise	Sanofi	Hikma pharm	Saidal	GSK	Novartis	Pfizer
Investissement (M$)	320	165	146	142	129	110

Son marché est classé 1er marché pharmaceutique au sud du bassin méditerranéen et deuxième sur le continent africain après l'Afrique du Sud, avec **1,37 M** de Dollars d'importations et une production locale ne couvrant que 30% des besoins (l'Office National des Statistiques algérien, 2015). C'est pour cela que l'Algérie a déployé des stratégies pour développer l'industrie pharmaceutique locale et avoir une certaine indépendance de production et ceci est particulièrement vrai dans la production des génériques qui permet de réduire les frais d'importation représentant la part du lion.

L'une des stratégies adoptées par l'état Algérien est l'incitation et l'encouragement des agents économiques à investir dans l'industrie pharmaceutique **privée** et accorder à ceux-ci plus de liberté contrairement à leur homologue public (Saidal), Une multitude d'entreprises privées verront alors le jour et parmi elles une entreprise qui se démarqua par rapport aux autres, Biopharm.

2.3 L'entreprise Biopharm

2.3.1 Présentation générale

BIOPHARM, laboratoire pharmaceutique algérien, est un groupe industriel et commercial qui a investi au début des années 1990 dans le secteur pharmaceutique et qui dispose aujourd'hui d'une unité de production aux normes internationales et d'un réseau de distribution aux grossistes et aux pharmacies.

Depuis sa création, BIOPHARM n'a cessé de progresser pour s'affirmer comme un **acteur de premier plan** du secteur pharmaceutique, et plus généralement, de la santé publique en Algérie.
Grâce à un réseau commercial dense et couvrant même les régions les plus reculées du pays, constitué de **14 centres de distribution**, de plus de **150 grossistes** et **3000 officines** pharmaceutiques, Biopharm est actuellement en mesure de délivrer **4000** types de produits pharmaceutiques sur l'ensemble du territoire algérien.

Ce réseau est également le support de relations solides et pérennes que Biopharm a tissée progressivement avec plus de **50 laboratoires internationaux** parmi les plus réputés (Abbott ; Alcon ; Astrazeneca ; Bayer..).
Son réseau de distribution grossiste, tourné vers la satisfaction de ses clients, est certifié depuis 2008 selon le Référentiel qualité **ISO 9001.**

2.3.2 La structure de "Biopharm group"

La structure organisationnelle du groupe Biopharm présentée sur la figure 2.3 a été adaptée aux différents métiers du groupe :

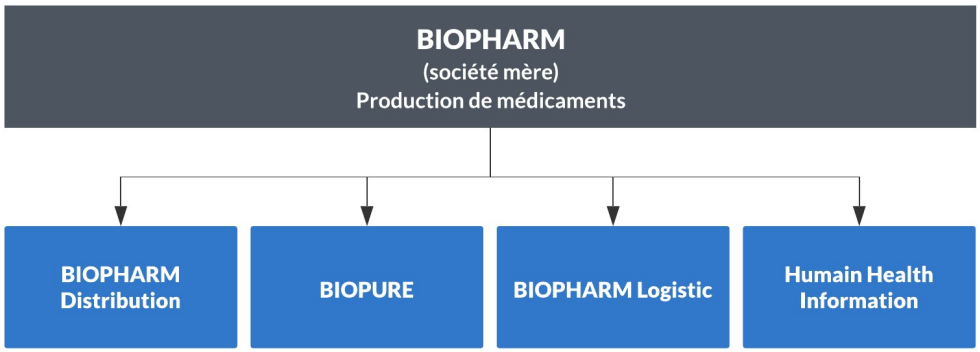


Fig. 2.3 : La structure du groupe Biopharm

- **Biopharm :**
  Est la société mère et le **noyau du groupe**. Elle est spécialisée dans la production de médicaments.

- **Biopharm distribution :**
  Est une filiale de Biopharm industrie spécialisée dans la **distribution** de médicaments aux grossistes.
- **Biopure :**
  Est une filiale de Biopharm industrie spécialisée dans la **répartition** aux officines(pharmacies).
- **Human Health information (HHI) :**
  Est une filiale de Biopharm industrie spécialisée dans l'**information** et la **promotion** médicale auprès de la communauté scientifique, médicale et professionnelle en Algérie.
- **Biopharm Logistics :**
  Est une filiale de Biopharm industrie spécialisée dans la **logistique**. Elle s'occupe de la logistique de bout en bout contrairement à **Biopharm distribution** qui elle s'occupe seulement de la distribution vers les grossistes (logistique aval)

2.3.3 La structure interne de Biopharm industrie

Biopharm a commencé par adapter progressivement sa structure organisationnelle en tant que groupe à ses différents métiers. La société mère Biopharm industrie a deux directions principales :

La direction industrielle

Le **cœur de métier** de l'entreprise, c'est la direction qui s'occupe de la supplychain de bout en bout en plus de la production.

La direction administrative et financière (DAF)

Chargée des affaires administratives, de la comptabilité et de la gestion des ressources humaines, elle englobe la majorité des processus support de l'entreprise.

Nous pouvons illustrer cette structure par l'organigramme présenté dans la figure 2.4.

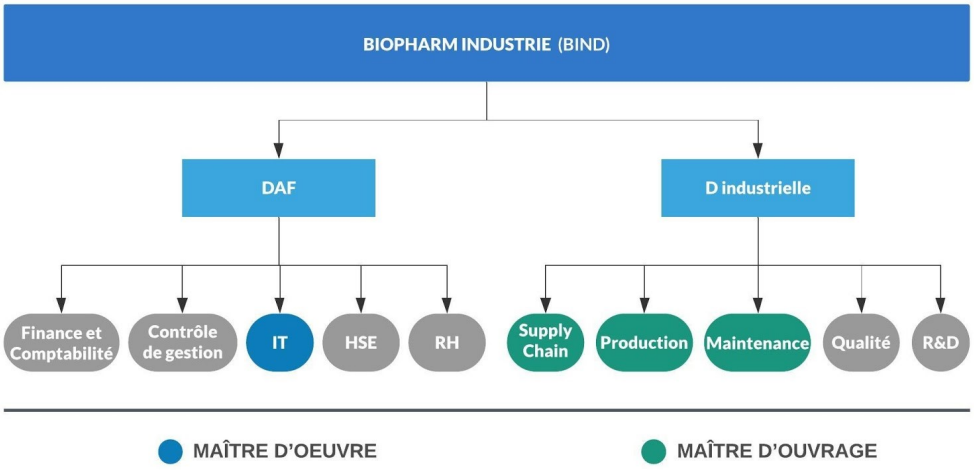


FIG. 2.4 : Organigramme Biopharm

2.3.4 Périmètre du projet au sein de l'entreprise

Le périmètre du stage proposé par l'entreprise concerne trois départements savoir la production, la supplychain et la DSI. .
Une relation **maître d'oeuvre - maître d'ouvrage** 2.5 se crée autour du stage dans le sens où les départements production/supplychain expriment un besoin et le département IT se charge de mener à bien le projet tel que présenté dans la figure 2.5.

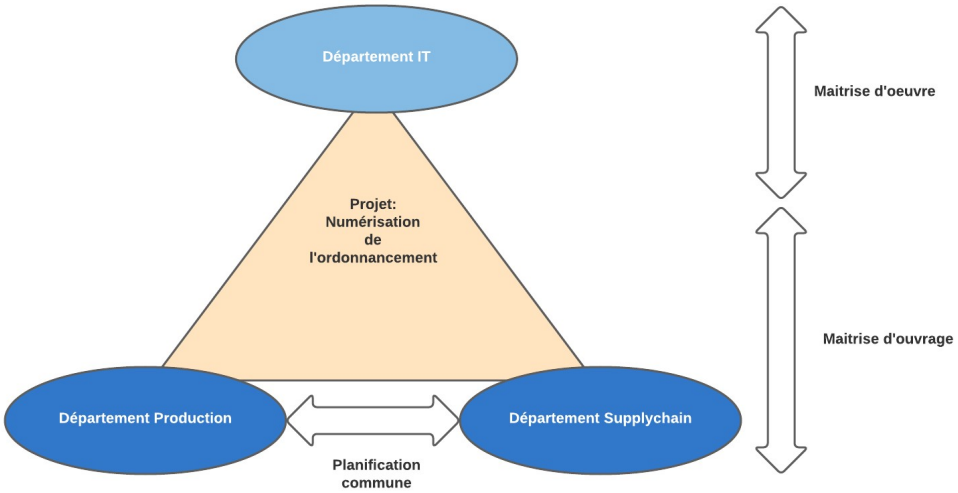


FIG. 2.5 : Relation maîtrise d'oeuvre - maîtrise d'ouvrage

Étudions alors ces trois départements de telle façon à comprendre leur fonctionnement et leur apport à l'entreprise.

La Direction IT et SI

La direction des Systèmes d'Information se charge de la gestion et de la maintenance du système d'information afin d'assurer son bon fonctionnement.

De plus, elle a pour mission de mener à terme des projets IT et de développer des solutions informatiques selon le besoin interne des différents départements. Cette direction englobe les services énumérés ci-dessous :

- Service réseau et systèmes ;
- Service systèmes d'information ;
- Service support et maintenance ;
- Service projets d'entreprise

Le service projets d'entreprise étant bien évidemment le service responsable de tous les projets IT y compris celui de la **numérisation de l'ordonnancement** qui est le sujet du mémoire actuel.

Le département Supplychain

Sa mission principale est de suivre le processus qui est généré lorsqu'un client passe une commande jusqu'à ce que le produit ou le service soit livré.
C'est pourquoi, la Supply Chain comprend la planification, l'exécution et le contrôle de toutes les activités liées aux flux de matières et d'information lors de l'achat, de la transformation et de la livraison finale. La supplychain au sein de BIOPHARM se subdivise 3 étapes principales :

1. **L'approvisionnement :** il s'agit de savoir comment, où et quand les matières premières sont obtenues et fournies pour la fabrication des produits.
2. **La production :** Il s'agit de planifier la production de telle manière à répondre au besoin final dans les temps .
3. **La distribution :** Il s'agit d'acheminer le produit final vers le client à travers un réseau de grossistes et d'entrepôts

L'ordonnancement étant une tâche assignée à ce département, l'inclusion de ce **"client final"** lors de l'implémentation de la solution est alors primordiale.
La figure 2.6 résume les flux gérés par la supplychain au sein de BIOPHARM.

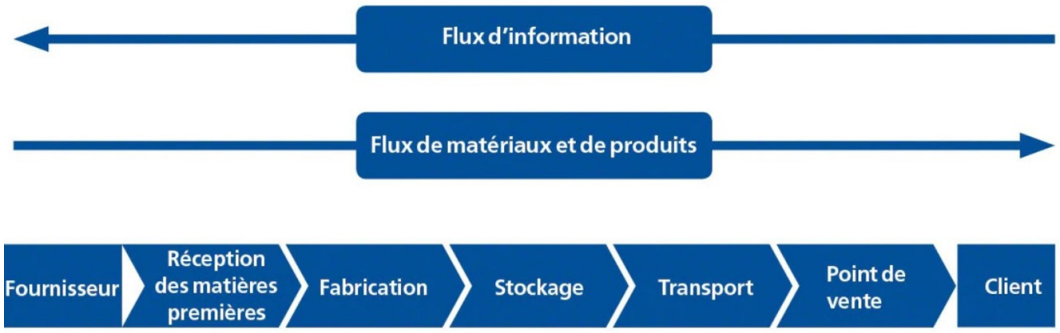


FIG. 2.6 : Les étapes de la supplychain

Le département Production

Sa fonction est de gérer la transformation des matières premières en produits finis. Son objectif principal est la mise en place d'un **système de production** adapté permettant de réduire considérablement les **coûts** de production tout en gardant une certaine **flexibilité** afin d'assurer un certain niveau de service imposé par la demande.
Au sein de BIOPHARM, une démarche d'**amélioration continue** a été entamée par la mise en place d'outils du **lean manufacturing** (5S, SMED..). Ce département doit être pris en considération car il représente l'application réelle du programme sur le terrain, de ce fait il délimite le projet de ses contraintes. Il s'agit en effet des tâches ci-dessous :. Nous énumérons alors les tâches du département production en relation avec l'ordonnancement afin de mieux cerner le périmètre du problème :

1. Étude de la **faisabilité** du plan de production ;
2. La **validation** du plan d'ordonnancement ;
3. L'élaboration du plan de **maintenance** ;
4. L'assignation et l'ordonnancement des **équipes** ;
5. La mise à jour et la **modification** du plan d'ordonnancement de la production.

La **délimitation** du problème est cruciale afin de ne pas empiéter sur les tâches et responsabilités des autres départements et assurer une implémentation structurée répondant au besoin.

2.4 Diagnostic de l'entreprise

Dans ce qui va suivre, nous allons procéder en deux temps :

1. Un premier diagnostic **externe** utilisant une analyse **SWOT-PESTEL** afin de cerner le positionnement de BIOPHARM par rapport à son environnement.

2. Un second diagnostic **interne** étudiant le processus de **planification de production** grâce à un raisonnement en entonnoir qui donnera lieu une analyse du processus d'ordonnancement.

2.4.1 Diagnostic interne & externe

Comme nous l'avons vu précédemment, l'industrie pharmaceutique subit l'influence de plusieurs **facteurs** dynamiques de nature technique, sociale et économique.

L'industrie subit une forte croissance qui la met face à plusieurs **défis** entraînant des variations conséquentes et imprévisibles de la demande comme celles observées lors des manifestations populaires de 2019 et la propagation du COVID-19.
Il est alors crucial d'étudier les facteurs environnementaux de l'entreprise à travers des **outils de diagnostic**.

Ce diagnostic a pour but de préciser **valeur ajoutée** que peut retirer BIOPHARM de l'ordonnancement sachant qu'elle évolue dans un environnement hautement compétitif.

Les différents facteurs détectés lors du diagnostic externe sont rassemblés dans les figures 2.7 et 2.8, qu'il faudra considérer d'un point de vue **producteur**.
L'impact des facteurs (positif/négatif) est déterminé en fonction des possibilités d'**augmentation** des ventes de produits pharmaceutiques, générant une augmentation du chiffre d'affaire.

Analyse SWOT

Dans cette analyse interne et externe, nous allons pouvoir détecter et énumérer les forces et faiblesses de l'entreprise en interne et les opportunités et les menaces en externe. Les résultats de l'analyse sont présentés sur la figure 2.7.

De cette analyse nous pouvons résumer chaque champs en une phrase :

- **Forces :**
  La force de BIOPHARM peut se résumer en une volonté d'amélioration et la maîtrise de son coeur de métier ;

- **Faiblesse :**
  La faiblesse principale de BIOPHARM est son manque d'utilisation des outils qui sont déjà mis à disposition (légère résistance au changement technologique) ;

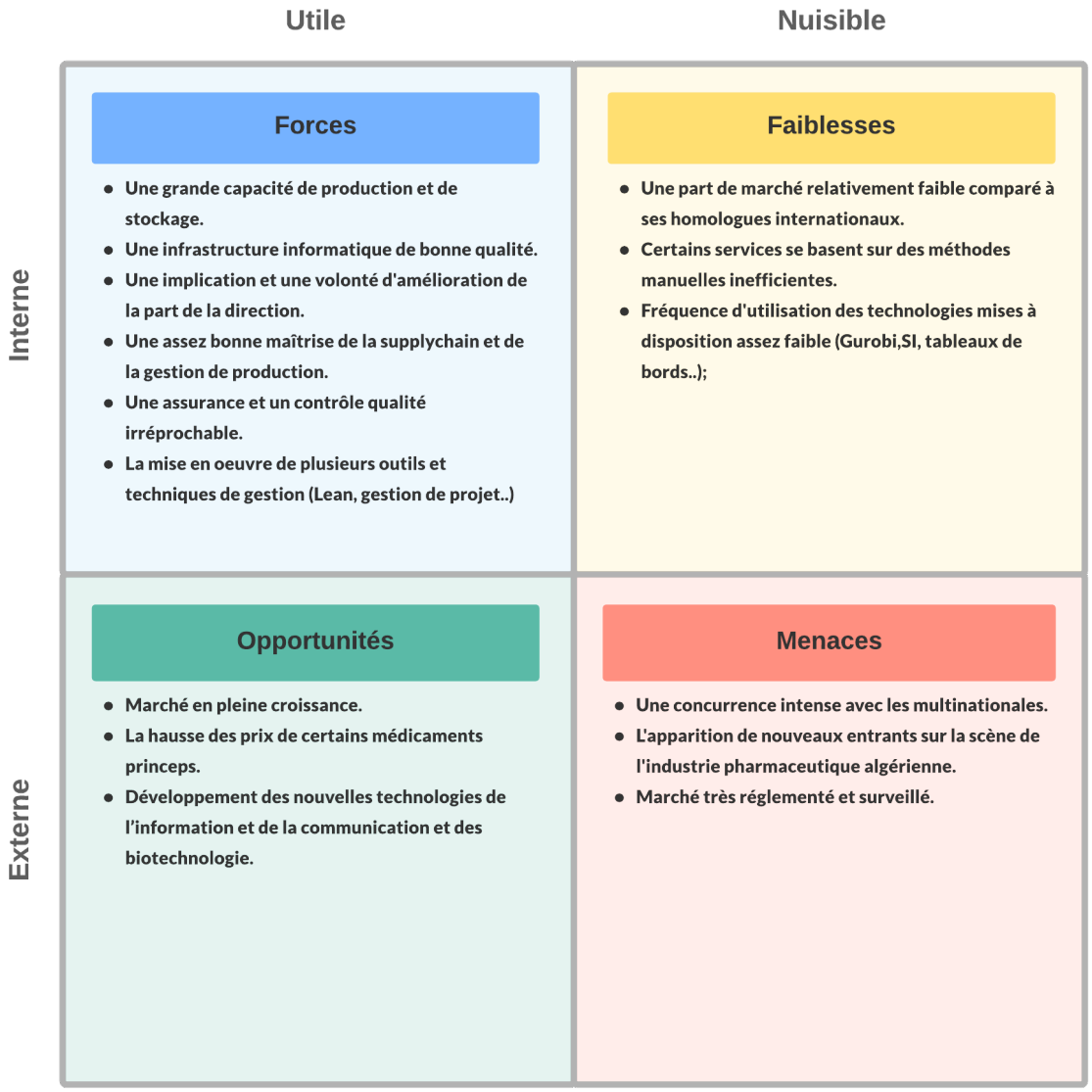


Fig. 2.7 : Analyse SWOT

- **Opportunité :**
  L'opportunité principale dont pourrait profiter BIOPHARM est la croissance conséquente du marché .

- **Menaces :**
  La menace la plus nuisible que doit affronter BIOPHARM est l'apparition de nouveaux concurrents sur la scène du pharmaceutique local.

Analyse PESTEL

Dans cette analyse nous allons pouvoir cerner le macro environnement de l'entreprise. Les résultats obtenus à l'issue de l'analyse PESTEL sont présentés dans la figure 2.8.

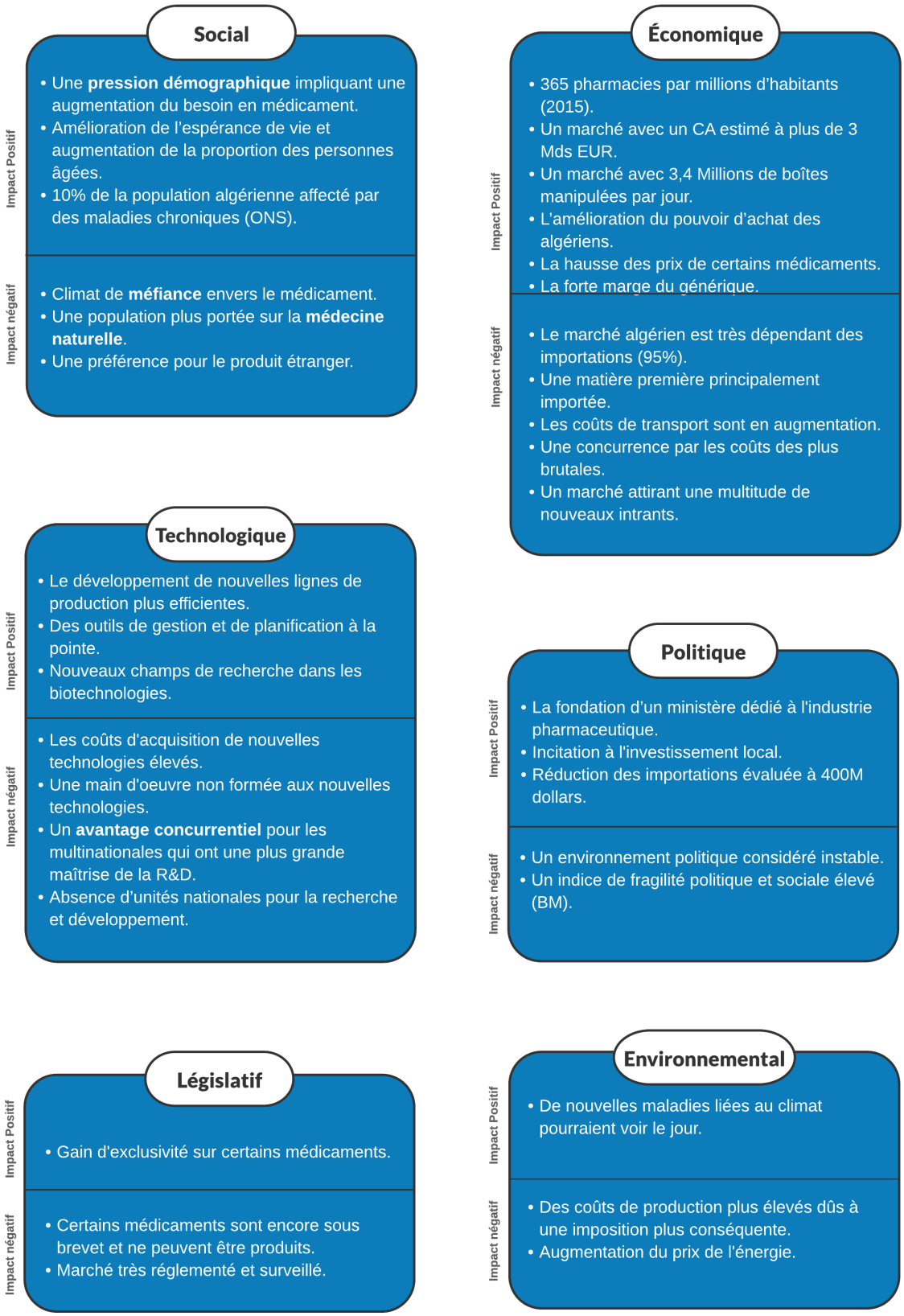


Fig. 2.8 : Analyse PESTEL

De cette analyse nous pouvons résumer chaque champs en une phrase :

- **Politique :**
  La politique d'importation de médicament restreint les multinationales et encourage la production locale ;
- **Économique :**
  Le marché pharmaceutique est à très fort potentiel de développement malgré une dépendance à l'importation ;
- **Technologique :**
  Des outils de gestion et des moyens de productions chers mais accessibles pouvant dégager un avantage comparatif si la main d'oeuvre est formée.
- **Social :**
  La croissance démographique et une croissance du taux de maladies augmentent le chiffre d'affaire malgré un climat de méfiance envers le produit local.
- **Environnemental :**
  Une augmentation du prix de l'énergie et l'éventualité d'une taxe carbone.
- **Législatif :**
  L'industrie pharmaceutique est très réglementée.

2.4.2 Diagnostic du processus de planification

L'ordonnancement est une partie (sous-processus) d'un processus plus complexe **"La prévision et la planification de la production"** que l'on déroulera 2.12 afin de mieux comprendre le fonctionnement de l'existant.

Nous découperons le processus de planification en trois grandes parties suivant un ordre séquentiel comme décrit sur la figure 2.12, ensuite on le déroulera afin de mieux comprendre le fonctionnement de l'existant.

La prévision (Demand planning)

Afin d'estimer la demande future, HHI qui fait office de département marketing émet des prévisions en se basant sur des demandes fermes, des modèles mathématiques, des historiques et la situations socio-economique en Algérie.
La figure 2.9 illustre le processus de prévision.

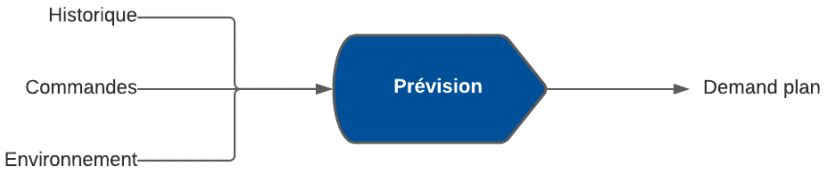


FIG. 2.9 : Processus de prévision de la demande

Le plan de livraison des ventes

Après la validation de la prévision, une réunion est organisée par HHI et BIND afin de planifier la production annuelle et de la décliner en productions mensuelles..
Ayant la **prévision** et les **capacités** de production, un outil informatique développé par McKinsey baptisé **VIU (Value In Use)** est utilisé afin d'obtenir en output un **plan annuel de livraison** qui sera revisité et validé par toute l'équipe concernée et le DG au début de chaque mois d'exercice.
La figure 2.9 illustre le processus de planification des ventes.

FIG. 2.10 : Processus de planification des ventes

Le plan de production

Suite à la validation du plan de livraison, VIU est une fois de plus utilisé afin de lancer les ordres d'approvisionnement si nécessaire.
Un ordonnancement de la production est enfin effectué, afin de pouvoir lancer les ordres de fabrication et de ce fait satisfaire la demande journalière.

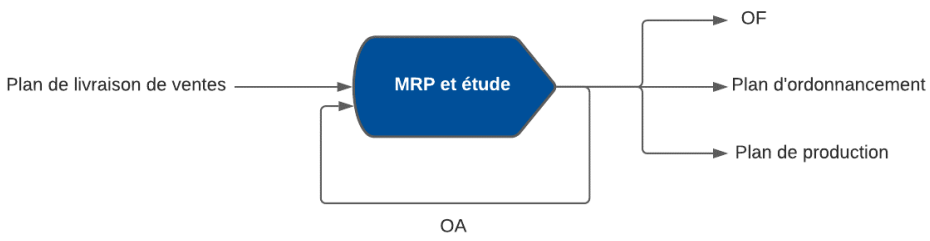


FIG. 2.11 : Processus de planification de la production

La cartographie des processus présentée sur la figure 2.12 explicite le déroulement d'une planification de la production sur le terrain.

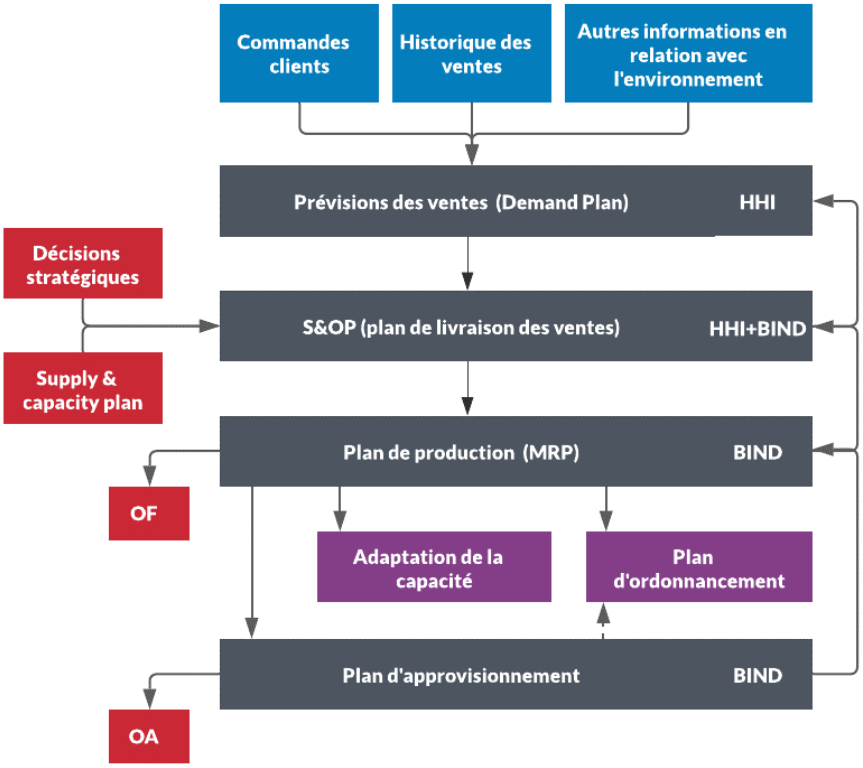


FIG. 2.12 : Macro-Processus de planification de la production

L'ordonnancement

Le processus d'ordonnancement est un processus clé permettant à l'entreprise d'organiser sa production de manière plus efficiente afin d'en dégager un profit substantiel et un **avantage concurrentiel** par rapport aux autre entreprises de ce secteur. La figure 2.13 présente les input et les output du modèle d'ordonnancement actuel.

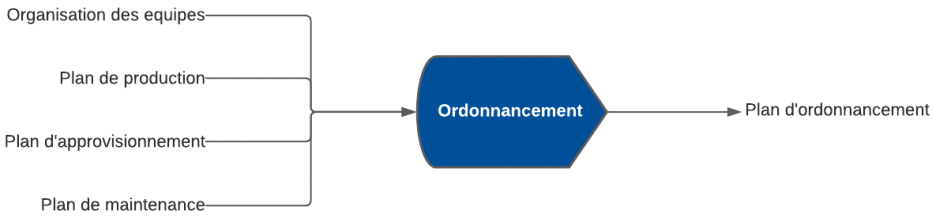


FIG. 2.13 : Processus d'ordonnancement

Après plusieurs entretiens avec les différents acteurs du processus d'ordonnancement, un brainstorming et une comparaison entre l'existant et un programme d'ordonnancement efficient comme définit dans l'état de l'art. Nous avons pu relever les dysfonctionnements de ce processus sur lesquels il faudra agir, on les classera en trois catégories selon leur enchaînement séquentiel :

- **Dysfonctionnements observés dans les inputs :**

  1. Un retard dans le transfert de données entre les différents départements ;
  2. Des écarts par rapport aux **délais** entre le plan d'approvisionnement théorique et les réalisations ;
  3. Une structure d'inputs **non normalisée** ;
  4. Des données d'entrée parfois erronées ou inexistantes ;

- **Dysfonctionnements observés lors de la planification :**

  1. Une planification **manuelle** soumise à l'erreur humaine ;
  2. Une planification longue et **fastidieuse** (durée entre 2-4 jours) immobilisant des ressources humaines et matérielles ;
  3. La solution obtenue est faisable mais **non optimale** ne tirant pas profit des solutions technologiques mises à disposition ;
  4. La méthode de planification ne suit aucune **stratégie** préétablie, elle est le résultat de l'instinct du planificateur ;
  5. Processus dépendant de l'employé chargé de le mener à bien, une formation est donc nécessaire à chaque changement de responsabilité ;
  6. La solution adoptée par l'entreprise ne prend pas en compte la maintenance ;
  7. La solution ne prend pas en compte l'approvisionnement ;

- **Dysfonctionnements observés dans l'output :**

  1. L'output est sous forme de table excel surchargée de données ;
  2. Une lecture de l'output difficile et parfois illisible ;

3. Aucun **suivi** de la solution, la mise à jour ne se fait que dans le cas des urgences ;
4. Une mise à jour lente et fastidieuse ;
5. Présence d'erreurs humaines ;
6. Manque de graphiques rendant la solution plus **intuitive** ;

Afin de mieux structurer les dysfonctionnements, nous avons utilisé le **diagramme d'Ishikawa** que nous avons présenté sur la figure 2.14. Ce diagramme va permettre de déterminer la nature des dysfonctionnements et de choisir les dysfonctionnements qu'il va falloir prendre en charge dans le cadre de notre projet.
Nous explicitons ci-dessous les éléments du digramme d'Ishikawa.:

- **Main d'oeuvre :** inclus les différents dysfonctionnements liés aux ressources humaines ;
- **Matière :** dans un projet IT la matière première peut être considérée comme étant les données d'entrée du processus, cette catégorie inclut donc les problèmes de données ;
- **Méthode :** inclus les problèmes liés à l'exécution des différentes étapes du processus ;
- **Milieu :** dans un projet IT le milieu peut être considéré comme l'environnement d'exécution ou l'environnement au sein de l'entreprise, nous considérons le second car il est plus approprié ;
- **Matériel :** inclus les dysfonctionnements induits par le matériel informatique et l'ERP principalement ;
- **Management :** inclus les manquements en gestion et suivi ;

Du diagramme l'on peut en déduire que notre solution aura un rôle important dans la construction de la procédure et la structuration des données d'entrées/sortie qui sont en **périphérie** de l'objectif principal qui est la formulation mathématique et l'implémentation d'un modèle répondant au besoin d'ordonnancement.

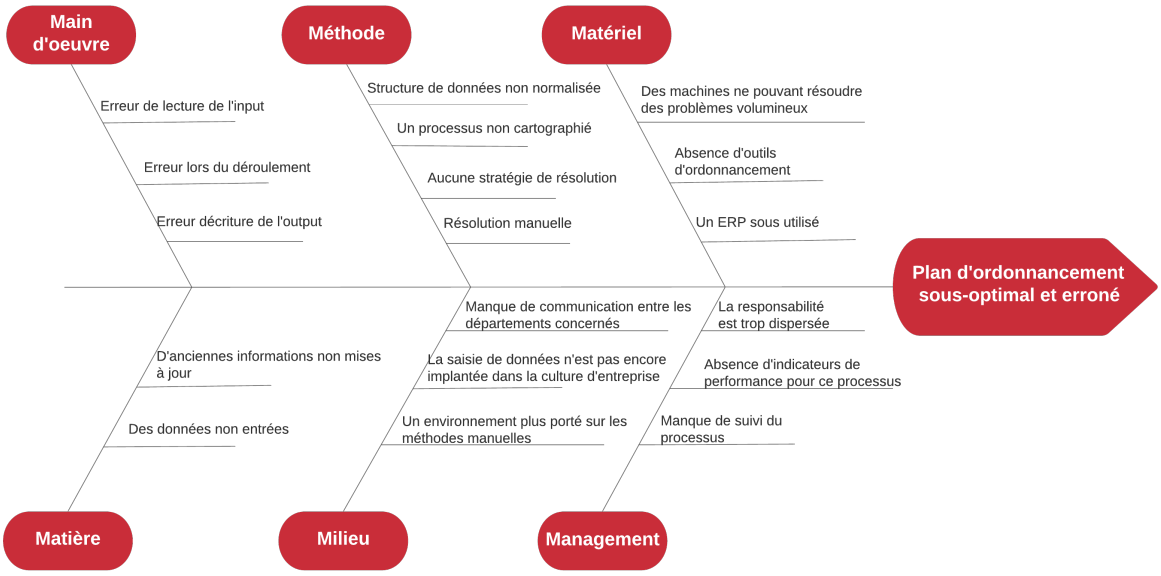


FIG. 2.14 : Diagramme d'Ishikawa pour le problème d'ordonnancement

2.4.3 Synthèse du diagnostic

Les diagnostics interne et externe nous permettent de justifier notre orientation vers une solution d'ordonnancement.
En premier lieu nous fixerons les objectifs de la solution :

1. L'obtention d'un avantage concurrentiel en augmentant le taux de service, réduisant les délais et augmentant le chiffre d'affaire afin de s'approprier plus de parts de marché ;
2. La construction d'une structure de planification concrète ;
3. Faire un pas de plus vers une automatisation de la gestion ;
4. Faciliter la communication inter-départementale ;

En second lieux, nous voudrons cibler les dysfonctionnements à réduire ou éliminer :

1. Réduire les risques d'erreur humains ;
2. Réduire le temps de planification ;
3. Optimiser le résultat de la planification ;
4. Rendre le processus d'ordonnancement indépendant de l'employé résponsable ;
5. Améliorer l'output et simplifier sa lecture ;

Cette synthèse rend la formulation d'une problématique plus aisée.

2.5 Énoncé de la problématique

Suite aux résultats du diagnostic que nous avons réalisé et qui met en relief l'intensité concurrentielle qu'observe l'industrie pharmaceutique en Algérie. Nous pouvons mettre le doigt sur l'importance d'une bonne gestion de la production qui peut s'avérer être un atout stratégique permettant de gagner un avantage concurrentiel par rapport à aux concurrents algériens et étrangers.

Tout processus aussi simple soit-il a son importance dans une organisation, mais lorsque celui-ci fait office de maillon faible toute la chaîne de valeur vascille, ceci est particulièrement le cas pour la SCM et plus précisément l'ordonnancement qui est le **métronome** qui orchestre la production de médicament, le **coeur de métier** de l'entreprise.
Une bonne orchestration permettrait de dégager de la capacité et de ce fait produire une plus grande quantité pouvant répondre à une demande perpétuellement croissante tout en augmentant le chiffre d'affaire de l'entreprise afin de réduire l'écart qui se creuse entre les entreprises nationales et multinationales.

Par ailleurs, la numérisation des données et l'automatisation de certains processus reste un défi pour l'entreprise Algérienne. En effet, la difficulté réside dans l'adoption des bonnes pratiques et leur intégration dans la culture de l'entreprise, un des points fondamentaux explicité dans le diagramme d'Ishikawa fig2.14.

A l'issue du diagnostic étant riche en idée, nous avons pu en concertation avec les responsables du projets au sein de l'entreprise sélectionner les objectifs les plus importants :

1. L'obtention d'un avantage concurrentiel en augmentant le taux de service, réduisant les délais et augmentant le chiffre d'affaire afin de s'approprier plus de parts de marché ;
2. La construction d'une structure de planification concrète ;
3. Faire un pas de plus vers une automatisation de la gestion ;
4. Faciliter la communication inter-départementale ;

Et les dysfonctionnements auxquels nous souhaitons remédier :

1. Réduire les risques d'erreur humains ;
2. Réduire le temps de planification ;
3. Optimiser le résultat de la planification ;
4. Rendre le processus d'ordonnancement indépendant de l'employé responsable ;
5. Améliorer l'output et simplifier sa lecture ;

Ces objectifs et cette volonté d'éliminer les dysfonctionnements cités nous conduisent vers l'énoncé d'une question centrale :

"Comment améliorer le processus actuel d'ordonnancement au sein de BIOPHARM à l'aide de la modélisation ?"

De cette question fondamentale découlent quelques sous-questions :

- Quel modèle d'ordonnancement est le plus adapté pour le cas BIOPHARM ?
- Quelles sont les étapes et la démarche de développement d'une solution informatique pour l'ordonnancement ?

Toutes ces questions trouveront réponse tout au long de ce mémoire.

2.6 Conclusion

Dans ce chapitre dédié à l'étude de l'existant nous avons introduit le secteur ainsi que l'entreprise et les avons diagnostiqués de telle manière à localiser ses forces et faiblesses et mettre le doigt sur les divers dysfonctionnements visibles et implicites au sein des différents départements concernés.

Le contexte et les objectifs étant cernés, les prochains chapitres porteront sur l'analyse détaillée du problème d'ordonnancement et l'implémentation d'une solution permettant d'atteindre les objectifs que nous avons énumérés précédemment.

Chapitre 3

Étude du problème

3.1 Introduction

Dans le chapitre précédent nous avons suivi un raisonnement en entonnoir, commençant par l'industrie pharmaceutique mondiale et l'environnement qu'elle engendre jusqu'à atteindre le processus d'ordonnancement au sein de BIOPHARM.
Le processus étant connu, nous voudrons maintenant connaître les spécificités du problèmes afin de pouvoir le classer comme vu précédemment dans l'état de l'art.

Ce chapitre sera découpé en 4 sections comme résumé dans la **figure 3.1**

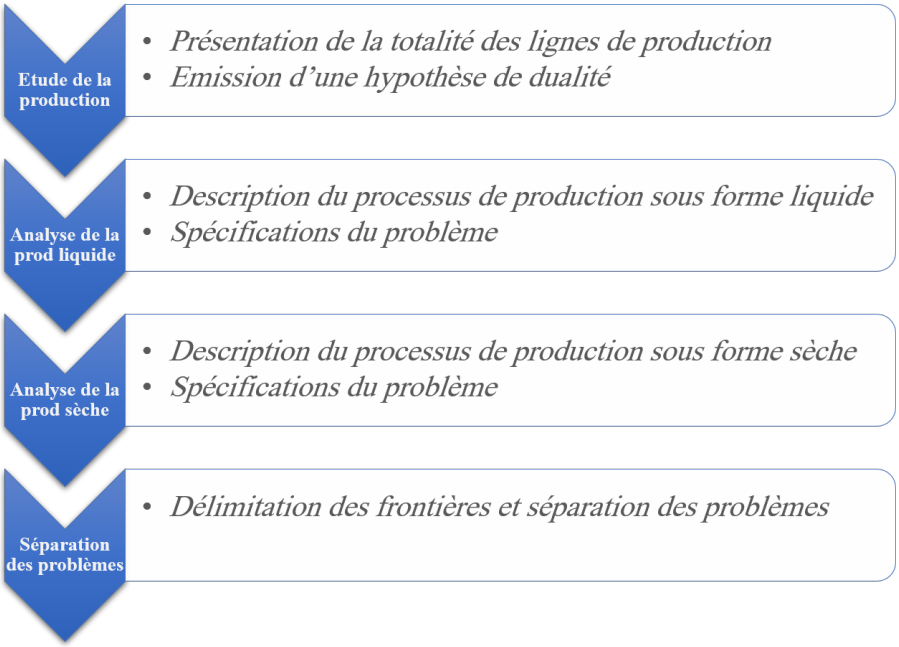


FIG. 3.1 : Structure du chapitre "Étude du problème"

3.2 La production au sein de BIOPHARM

Durant nos visites dans les ateliers nous avons pu observer les lignes de production de plus prés, comprenant ainsi le processus de fabrication des médicaments.
Cela nous a permis de dessiner un schéma3.2 englobant l'ensemble des lignes de production.

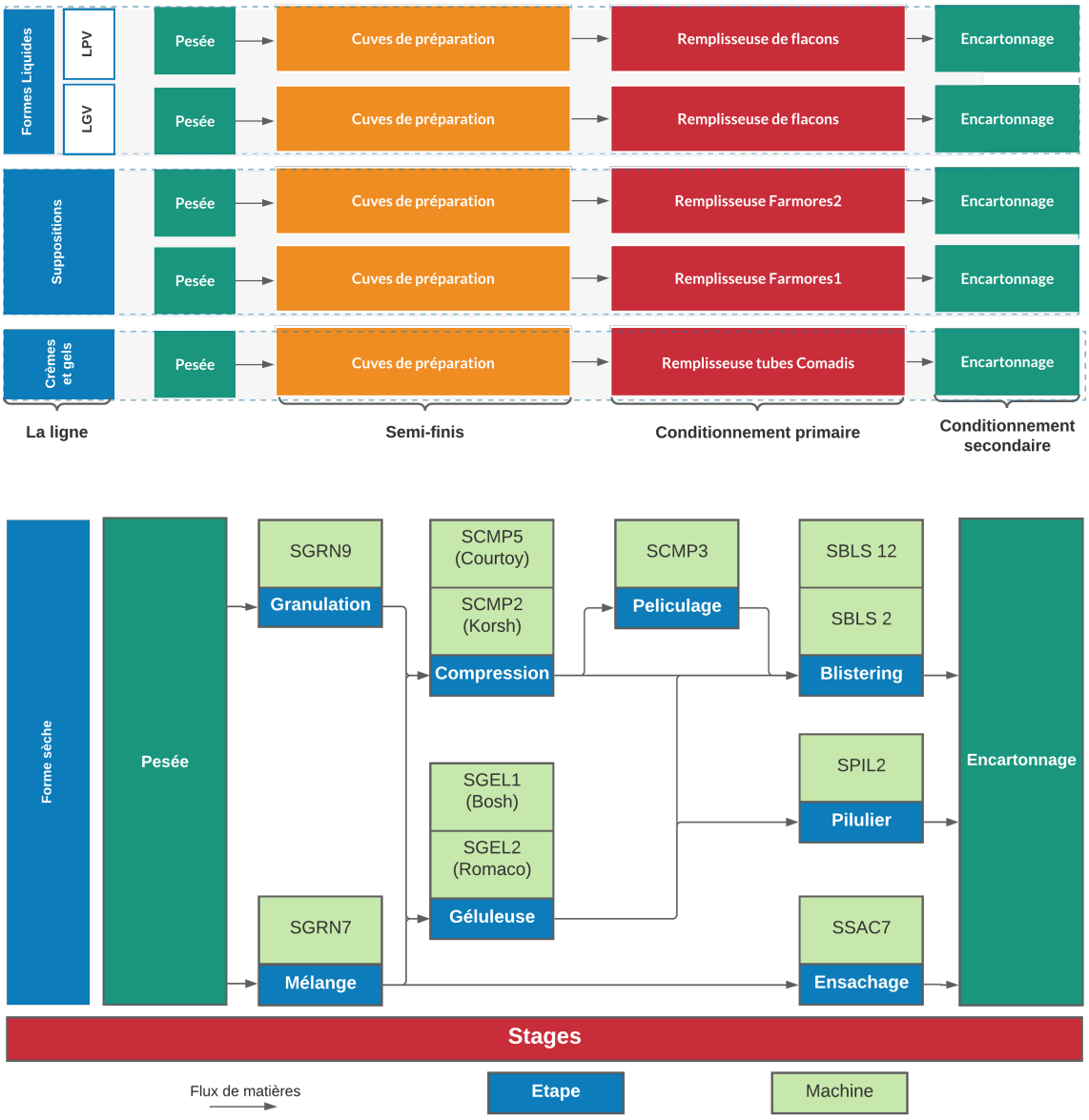


FIG. 3.2 : Schéma englobant l'ensemble des lignes de production

Néanmoins nous en sortirons avec deux questionnements majeurs :

1. Est-ce-que toutes les machines subissent les mêmes contraintes ?
2. Si ce n'est pas le cas comment les distinguer ?

Ayant déjà une intuition, nous avons décidé de mener des entretiens avec les responsables de productions qui confirme notre hypothèse, certaines lignes ne subissent pas les mêmes contraintes et ceci est dû à l'état de la matière.
Tandis que l'état solide présente certaines caractéristique lui permettant d'être stocké sous forme d'en-cours, la forme liquide quant à elle doit être acheminée de bout en bout.

En plus des entretiens et des visite effectués, le processus en lui-même est séparée selon l'état de la matière.

Cette première séparation peut sembler intéressante mais nécessitera une vérification rigoureuse avant d'être appliquée.

3.3 Analyse des lignes de production sous forme liquide

3.3.1 Le processus de production des lignes sous forme liquide

Les lignes de production de médicament sous format liquide sont relativement simples lorsqu'on les compare avec les lignes des formes sèches, mais tout le challenge se se trouve dans la formulation du problème.

Premièrement, il n'y a pas une ligne **unique** mais plutôt **quatre** lignes indépendantes ayant toutes les mêmes caractéristiques mais des types de produits différents.
L'on peut résumer les lignes et leur type dans la figure ci-dessous :

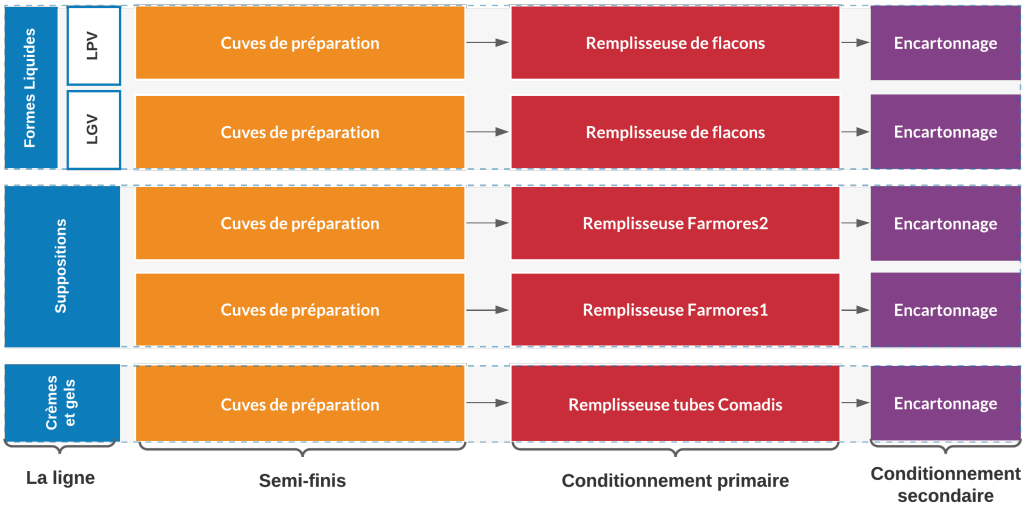


Fig. 3.3 : Les lignes de production sous forme liquide

Le fonctionnement d'une ligne de production liquide est simple, et se décompose en 4 étapes :

1. **La pesée :**
   L'on pèse la matière première dans un local stérilisé afin de prélever la quantité nécessaire pour lancer la production d'un ou plusieurs lots ;

2. **Le mixage (manufacturing) :**
   L'étape durant laquelle le médicament est préparé ;

3. **Le stockage en cuve :**
   Une fois que le médicament est préparé il est stocké dans une cuve en tant que produit intermédiaire afin de pouvoir lancer la production d'un autre lot ;

4. **Le conditionnement :**
   L'étape durant laquelle le médicament est conditionné et ceci inclut le conditionnement primaire et secondaire ;

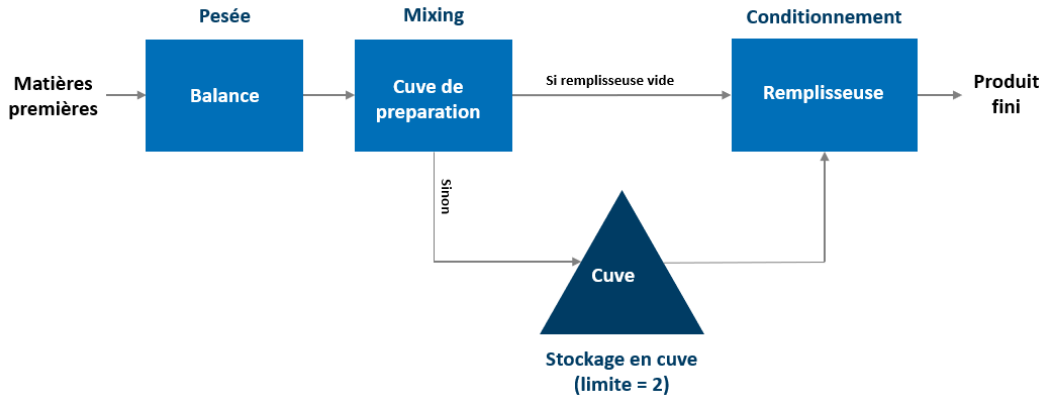


FIG. 3.4 : Schéma explicatif d'une ligne de production liquide

3.3.2 Spécifications du problème

Afin de pouvoir se lancer dans la recherche de solution, il faudra classifier notre problème selon les conventions et ceci en étudiant les différents paramètres suivants comme détaillé dans "Scheduling, Pinedo" (PINEDO 2016) :

- **L'organisation de l'atelier :**
  Pour les formes liquides nous avons deux types d'organisation d'atelier :

  1. Nous avons une seule ligne de 3 machines en série pour les formes liquides, crèmes et gels l'atelier est alors organisé en Flowshop F_3.
  2. Nous avons deux lignes identiques de 3 machines pour les suppositoires l'atelier est alors organisé en Flexible Flowshop FF_3.

- **La grandeur à optimiser :**
  Voulant dégager de la capacité afin de profiter d'une plus grande production et donc d'un CA plus conséquent, il faudra alors minimiser le **Makespan** que l'on peut approximer par la date de fin du dernier produit du mois.

$$C_{max}$$

- **La date de sortie (release date) :**
  La contrainte temporelle de sortie pour les produits étant inexistante nous pouvons la négliger.

$$\cancel{R_i}$$

- **La préemption :**
  Vu que nous nous trouvons dans une organisation en flowshop, les ressources matérielles ne permettent pas l'arrêt des machines en plein production, on peut donc mettre la préemption de côté.

$$\sout{PRMP}$$

- **La précédence :**
  Nous avons une contrainte de précédence entre les machines mais prise en compte dans la typologie Flowshop, elle n'est donc pas prise en considération afin d'éviter la redondance.

$$\sout{PREC}$$

- **Le setup time s_{jk} :**
  Ayant des temps de changement de série différents il faudra les prendre en considération c'est ce qu'on appellera par la suite temps de nettoyage ou de changement de série que l'on notera :
  - pour l'étape de manufacturing :

$$t_{ij}^{m}$$

  - pour l'étape du packaging :

$$t_{ij}^{p}$$

- **Les familles de produits :**
  La transition d'un produit à l'autre n'étant pas définie pour des familles de produits, nous n'aurons pas à la prendre en considération.

$$\sout{FMLS}$$

- **La production en lot (batch(b)) :**
  L'on ne peut produire qu'un seul lot à la fois et il est impossible de jumeler ou effectuer plusieurs taches simultanément.

$$b = 1$$

- **Les temps d'arrêts (breakdowns) :**
  Lors de notre planification il faut prendre en considération la période allouée à la maintenance préventive.

$$\text{brkdwn}$$

- **Le blocage (blocking) :**
  Puisque la production n'a que deux cuves entre le mixage et le conditionnement, nous pouvons considérer cette étape comme un buffer de taille 2 lots.

Block

- **La recirculation :**
  Les produits ne circulent qu'une seule fois sur chaque machine rendant la recirculation impossible.

~~*RCRC*~~

L'on peut résumer le tout dans un tableau récapitulatif :

Tab. 3.1 : Tableau récapitulatif des contraintes du problème

Paramètre	Situation
L'organisation de l'atelier	Flowshop F_3
La grandeur à optimiser	Le makespan C_{max}
La date de sortie	Non spécifiée
La préemption	Impossible
La précédence	Déjà prise en considération dans l'organisation
Le setup time	Pris en considération
Les familles de produits	Non spécifié
La production en lot	Un seul lot
Les temps d'arrêts	Pour maintenance
Le blocage	Stock limité
La recirculation	Impossible

Ayant étudié les différents paramètres d'un problème d'ordonnancement et en supposant que les lignes parallèles identiques sont considérées comme des lignes à part entière nous pouvons définir la typologie du problème à travers une notation appropriée :

Notation pour le problème d'ordonnancement (forme liquide) :

$$F_3|s_{jk}, brkdwn, block|C_{max}$$

3.4 Analyse des lignes de production sous forme sèche

3.4.1 Le processus de production des lignes sous forme sèche

Les lignes de production de médicament sous forme sèche sont relativement complexes lorsqu'on les compare avec les lignes des formes liquides, ceci est dû aux différents chemins qu'un produit peut prendre afin d'être considéré comme produit fini.
De plus la présence d'en-cours de production ajoute une couche de difficulté supplémentaire que l'on ne retrouve pas dans la partie liquide.
Nous avons pu schématiser les lignes de production sous forme sèche dans la 3.5.

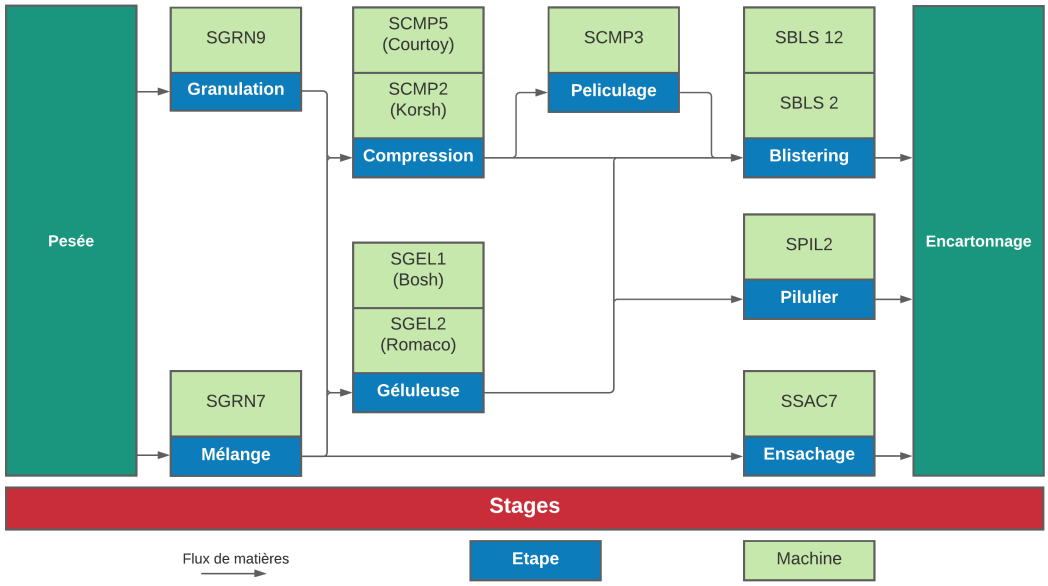


FIG. 3.5 : Mapping des lignes de production sous forme sèche

La production des produits semi-finis sur les lignes sèches suit 4 étapes principales **figure 3.6** :

1. **La pesée :** L'on pèse la matière première dans un local stérilisé afin de prélever la quantité nécessaire pour lancer la production d'un ou plusieurs lots ;
2. **La granulation ou le mélange :** pour produire les poudres.
3. **La compression ou la géluleuse :** pour produire les comprimés ou les gélules.
4. **Le pelliculage :** pour enrober les comprimés nus.

Les produits semi-finis montrés dans **la figure 3.6** en vert passeront sur l'une des 4 lignes de conditionnement décrites dans la **figure 3.7** afin d'obtenir des produits finis.
Chaque produit a son propre processus de fabrication qui suit une certaine séquence d'opérations, d'où la nécessité d'inclure l'ERP pour l'obtention de **données statiques** définies au préalable.

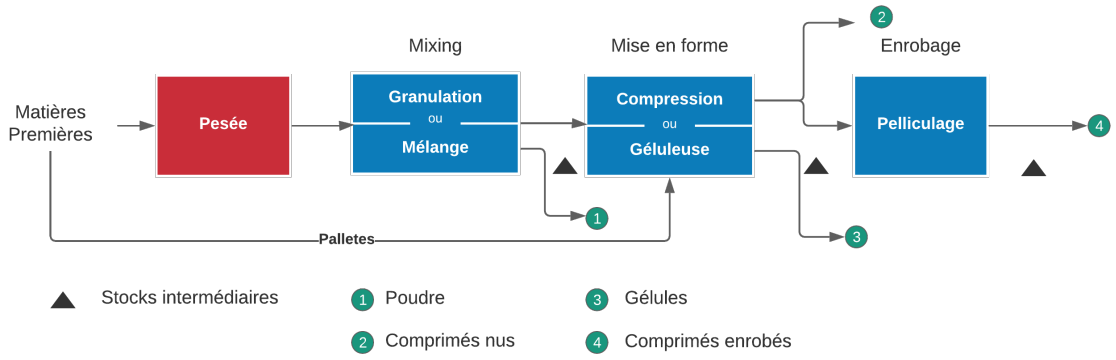


FIG. 3.6 : Les étapes de production des produits semi-finis

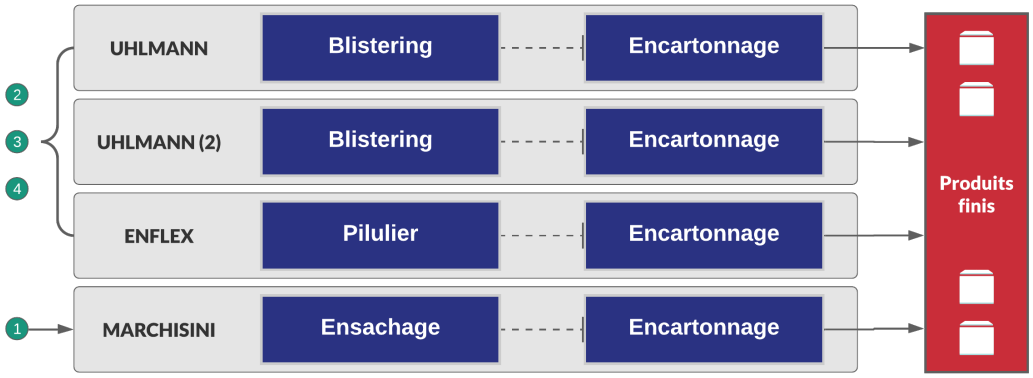


FIG. 3.7 : Étapes de production des produits finis

3.4.2 Spécifications du problème

Afin de pouvoir se lancer dans la recherche de solution, il faudra classifier notre problème selon les conventions et ceci en étudiant les différents paramètres suivants comme détaillé dans l'ouvrage (PINEDO 2016) :

- **L'organisation de l'atelier :**
  Pour les formes sèches l'atelier est de type FJS à 8 étapes FJ_8 où des tâches peuvent être traitées par plus d'une machine par étape.

- **La date de sortie (release date) :**
  La contrainte temporelle de sortie pour les produits étant inexistante nous pouvons la négliger.

$$\not{R_i}$$

- **La préemption :**
  La préemption des taches est impossible, toute opération, une fois commencée ne peut être interrompue.

$$\not{PRMP}$$

- **La précédence :**
  Nous avons une contrainte de précédence entre les machines, chaque lot a sa propre séquence d'opérations.

$$PREC$$

- **Le setup time s_{jk} (le changeover) :**
  Ayant des temps de changement de série différents il faudra les prendre en considération c'est ce qu'on appellera par la suite temps de nettoyage ou de changement de série (changeover) qui dépendent seulement de la machine et que l'on notera :

$$co_k$$

- **Les familles de produits :**
  La transition d'un produit à l'autre n'étant pas définie pour des familles de produits mais la notion existe car les lots de même produit n'impliqueront pas un changement de série.

$$FMLS$$

- **La production en lot (batch(b)) :**
  Les machines ne peuvent traiter qu'un seul lot simultanément.

$$b = 1$$

- **Les temps d'arrêts (breakdowns) :**
  Lors de notre conception il faudra prendre en considération la période allouée à la maintenance préventive et aux week-ends, les machines seront alors à l'arrêt.

$$brkdwn$$

- **Le blocage (blocking) :**
  Pour les lignes sous forme sèche, nous n'avons aucun blocage contrairement aux lignes sous forme liquide.

$$\not{Block}$$

- **La recirculation :**
  Les produits ne circulent qu'une seule fois sur chaque machine rendant la recirculation impossible.

$$\not{RCRC}$$

Tab. 3.2 : Tableau récapitulatif des contraintes du problème de la forme sèche

Paramètre	Situation
L'organisation de l'atelier	Job shop flexible FJ_8
La grandeur à optimiser	$\sum(C_j)$
La date de sortie	Dans un mois
La préemption	Impossible
La précédence	Pris en considération
Le setup time	Pris en considération
Les familles de produits	Pris en considération
La production en lot	Pris en considération
Les temps d'arrêts	Pour la maintenance et pour les weekends
Le blocage	Pas de blockage
La recirculation	Impossible

L'on peut résumer le tout dans un tableau récapitulatif :

Ayant étudié les différents paramètres d'un problème d'ordonnancement, nous pouvons définir la typologie du problème à travers une notation appropriée :

Notation pour le problème d'ordonnancement (forme sèche) :

$$FJ_8|prec, s_{jk}, fmls, block|\sum(C_j)$$

Problème NP-difficile au sens fort (Pinedo 2016)

Une autre notation mentionnée dans la littérature est : FJSP-SD (Flexible Job Shop with Sequence Dependent), le problème consiste donc à affecter chaque opération à une machine et d'ordonner les opérations dans le but d'atteindre l'optimum.

3.5 Séparation des modèles

Notre étude détaillée du problème et de sa typologie nous permet d'affirmer qu'il peut être décomposé en deux :

1. **Un problème d'ordonnancement spécifique aux lignes sous forme liquide :**
   C'est un problème d'ordonnancement d'atelier de type Flow-shop qui est relativement simple et qui ne nécessitera pas des méthodes de modélisation et de résolution très complexes.

2. **Un problème d'ordonnancement spécifique aux lignes sous forme sèche :**
   C'est un problème d'ordonnancement d'atelier de type FJS-SD qui est complexe et qui nécessitera probablement des heuristiques développées afin d'aboutir à une solution satisfaisante.

La frontière entre les différentes lignes et leurs typologies étant définie, une construction mathématique séparée s'impose, néanmoins l'implémentation réussira à fusionner les deux modèles et en faire une solution complète.

3.6 Conclusion

Dans ce chapitre, l'intuition d'une **dualité** de problèmes fut confirmée à travers une étude approfondie des lignes de production et de leurs contraintes, une conclusion déterminante en est alors extraite.
Il sera primordial de diviser la construction mathématique en deux parties, la première pour la partie liquide et la seconde pour la partie sèche, nous les consoliderons par la suite à l'aide d'une implémentation commune.

Chapitre 4

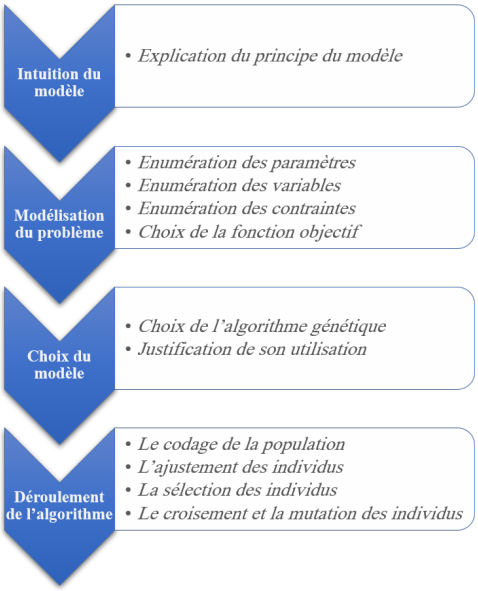
Construction des modèles d'ordonnancement

4.1 Introduction

Dans le chapitre précédent nous avons séparés le problème d'ordonnancement dans les ateliers de production de BIOPHARM en deux tout en définissant les particularités de chacun.
L'étape suivante consisterait alors à exploiter ces résultats afin de construire un modèle mathématique pouvant être implémenté et répondant à la réalité du terrain.

Nous commencerons par le modèle d'ordonnancement sous forme liquide à cause de sa simplicité puis nous entamerons la construction du modèle pour les lignes sous forme sèche, la structure du chapitre est illustré dans la figure 5.7 :

(a) Structure de la construction du modèle sous forme liquide



(b) Structure de la construction du modèle sous forme sèche

FIG. 4.1 : Structure adoptée pour la construction des modèles

Partie A :
Construction du modèle d'ordonnancement pour la forme liquide

4.2 Construction du modèle d'ordonnancement pour la forme liquide

4.2.1 L'intuition derrière le modèle

Cette solution consiste à définir une fonction objectif à minimiser puis utiliser une heuristique exhaustive afin de trouver une solution exacte ou approchée.
Le modèle gagne en simplicité et en efficacité en se basant sur un concept **naïf** qui est la **permutation**.
L'idée principale est de prendre un ordre de produits au hasard, puis permuter entre ceux-ci afin d'obtenir de meilleurs résultats, l'utilisation d'un algorithme approprié comme l'algorithme « génétique » ou l'algorithme de « branch and bound » ou même une **amélioration mathématique** est de rigueur afin de réduire la complexité algorithmique.

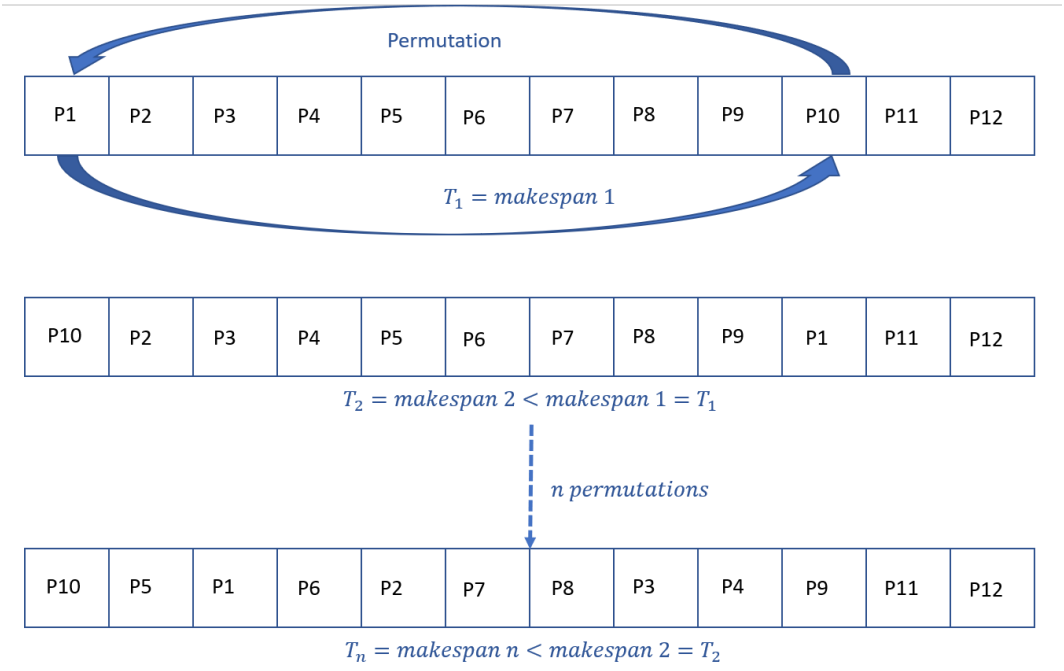


FIG. 4.2 : La permutation dans l'ordonnancement

4.2.2 Les paramètres et les variables du modèle

Les ensembles

- $D \in$ **Produits** $= \{1, 2, \ldots, \mathrm{d}\}$: ensemble des médicaments, qui représente les familles des lots a fabriquer ;
- $J \in$ **Lots** $= \{J_1, J_2, J_3, ..., J_i, ..., J_N\}$: Ensemble des lots a produire (jobs J) ;
- $E \in$ **Étapes** $= \{\mathrm{Manufacturing}, \mathrm{Stockage}, \mathrm{Packaging}\}$: les étapes de la production en flowshop ;

La modélisation

- **Le type de ligne $type_i$:**
  L'on notera $type_i$ le type de ligne liquide par lequel passe le produit i (LGV,LPV,suppo..).

- **La demande D_i :**
  L'on notera D_i la demande mensuelle du produit i issue du MRP.

- **Le nombre de lots L_i :**
  L'on notera L_i le nombre de lots du produit i à livrer durant ce mois, on peut l'obtenir à travers l'équation suivante :

$$L_i = \frac{D_i}{Lotsize_i}, \ \forall i \in I \tag{4.1}$$

- **Le temps opératoire d_i :**
  L'on notera d_i le temps que met un lot du produit i à être produit et plus particulièrement :

  - pour le manufacturing : d_i^m
  - pour le packaging : d_i^p

- **Le temps de transition/nettoyage t_{ij} :**
  L'on notera t_{ij} le temps mis pour passer d'un produit i vers un produit j sur l'une des machines et plus particulièrement :

  - pour le manufacturing : t_{ij}^m
  - pour le packaging : t_{ij}^p

  à partir des informations obtenues, l'on peut construire une matrice de temps de transitions respectivement :

  - pour le manufacturing :

$$T^m = \begin{bmatrix} t_{11}^m & \cdots & t_{1n}^m \\ \vdots & \ddots & \vdots \\ t_{n1}^m & \cdots & t_{nn}^m \end{bmatrix}$$

  - pour le packaging :

$$T^p = \begin{bmatrix} t_{11}^p & \cdots & t_{1n}^p \\ \vdots & \ddots & \vdots \\ t_{n1}^p & \cdots & t_{nn}^p \end{bmatrix}$$

Les variables de décision

La date de début de production d'un lot C_i :
L'on notera C_i la date de début de production du lot i et plus particulièrement :

- pour le manufacturing : C_i^m
- pour le packaging : C_i^p

et à partir de cette information nous pouvons déduire les dates de fin :

$$F_i^m = C_i^m + d_i^m, \ \forall i \in I$$
$$F_i^p = C_i^p + d_i^p, \ \forall i \in I$$

L'on peut tout résumer dans un seul tableau récapitulatif **??** :

TAB. 4.1 : Les paramètres du modèle de production de la forme liquide

Symbole	Description
J	Ensemble des lots a produire qui sont les jobs J
E	Ensemble d'étapes
i, j	L'indice des lots J
m	L'indice désignant manufacturing
p	L'indice désignant packaging
D_i	La demande du produit i
$Lotsize$	La taille de lot pour chaque type de produit D
L_i	Le nombre de lots à produire pour le produit i
d_i^m	Le temps que passe le produit i en manufacturing
d_i^p	Le temps que passe le produit i en packaging
t_{ij}	Le temps de transition/nettoyage
T	la de transition/nettoyage

Les contraintes

- **La contrainte de stockage δ_{it} :**
  Vu que l'on ne peut stocker plus de deux lots à cause du nombre de cuves disponibles, il faut vérifier qu'à tout instant t il y a au maximum 2 produits dans des cuves.

$$\sum_{i \in I} \delta_{it} \leq 2$$

  Cette contrainte étant difficile à implémenter sur la machine, nous avons décidé d'utiliser une formule de récurrence de **pas h=2** nous permettant de réduire la complexité en diminuant un **degrés de liberté** et contraignant complètement la variable.

- **Les contraintes sur les dates de début :**
  Nous avons plusieurs contraintes qu'il faut prendre en considération lors de la constriction du modèle :

  – Le premier produit doit se positionner sur **l'origine des temps** $t = 0$;

  – Deux contraintes **Fin-Début** car le manufacturing du produit i doit commencer après la fin du manufacturing du produit $i-1$, la deuxième contrainte étant un peu plus complexe car elle contient une récurrence de second ordre qui est possible grâce à l'organisation de l'atelier en flowshop (nous pourrons l'appeler la contrainte FIFO car le produit $i-2$ doit sortir de sa cuve de stockage pour céder la place au produit i) ;

$$C_i^m \geq C_{i-1}^m + d_{i-1}^m + t_{(i-1)\to i}^m \tag{4.2}$$

$$C_i^m \geq C_{i-2}^p - d_i^m \tag{4.3}$$

  – Deux contraintes **Fin-Début** car le packaging du produit i doit commencer après la fin du manufacturing du produit i et la fin du packaging du produit $i-1$ qui lui cède la place ;

$$C_i^p \geq C_{i-1}^p + d_{i-1}^p + t_{(i-1)\to i}^p \tag{4.4}$$

$$C_i^p \geq C_i^m + d_i^m \tag{4.5}$$

  Recherchant les dates de début minimales, il est évident que nous allons choisir la valeur minimale d'entre les valeurs possibles, ce qui fait que nous allons saturer les contraintes.

$$\boxed{C_i^m = max(C_{i-2}^p - d_i^m; C_{i-1}^m + d_{i-1}^m + t_{i-1\to i}^m)} \tag{4.6}$$

$$\boxed{C_i^p = max(C_i^m + d_i^m; C_{i-1}^p + d_{i-1}^p + t_{(i-1)\to i}^p)} \tag{4.7}$$

La fonction objectif

L'objectif du modèle est de dégager de la capacité sur les lignes de production, la fonction objectif la plus adéquate est la minimisation du makespan :

$$\boxed{\min(\text{Makespan}= \textstyle\sum_{i\in I} C_i^p \approx C_n^p + d_n^p)} \tag{4.8}$$

4.2.3 Le déroulement de l'algorithme

Dans cette section nous déroulerons l'algorithme recherchant l'ordre optimal des produits ainsi que leurs dates de début de production :

1. **Le calcul des dates :**
   Ayant un ordre de départ aléatoire, nous utilisons les formules récursives afin d'obtenir les dates :

   - L'initialisation :

$$C_0^m = 0$$

$$C_0^p = 0 + d_0^p$$

   - La récursivité :

$$C_i^m = max(C_{i-2}^p - d_i^m; C_{i-1}^m + d_{i-1}^m + t_{(i-1)\to i}^m)$$

$$C_i^p = max(C_i^m + d_i^m; C_{i-1}^p + d_{i-1}^p + t_{(i-1)\to i}^p)$$

2. **Test d'optimalité :**
   Voulant minimiser le makespan un calcul de celui-ci s'impose afin de trouver l'ordre des produits optimal :

$$Makespan = \sum_{i \in I} C_i^p \approx C_n^p + d_n^p$$

   à chaque permutation la valeur di makespan et de l'ordre peut être écrasée si le makespan est meilleur.

3. **Permutation et boucle :**
   à partir de là nous pouvons utiliser divers algorithmes de permutation, nous avons opté pour l'algorithme de **heap** dont la complexité est assez faible.

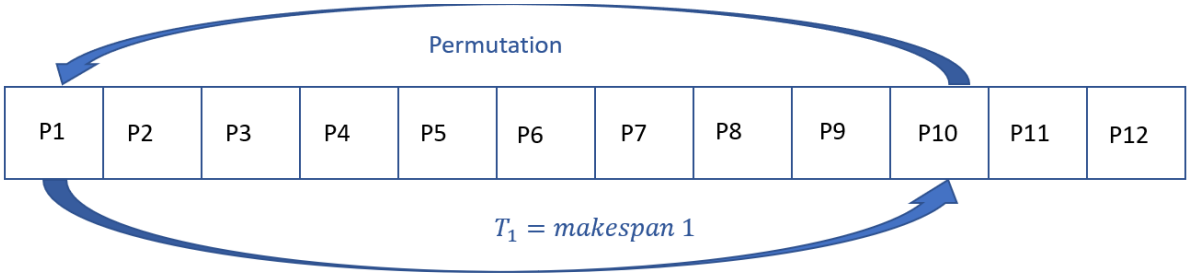


Fig. 4.3 : Permutation

Une fois la permutation effectuée, toutes les étapes de calcul sont répétées, et le makespan est mis à jour de telle façon à choisir la valeur minimale.
La boucle s'arrête lorsque toutes les permutations sont testées.

4.2.4 Critique de la solution

La solution proposée est adéquate au problème mais comporte certains problèmes conséquents si ils ne sont pas étudiés et pris en considération.

Points négatifs :

- La **complexité algorithmique** de la permutation est de l'ordre du $n!$ ce qui rend l'algorithme particulièrement **lent**, pour avoir un ordre d'idée la permutation de 10 produits nous donne $10! = 3628800$ cas possibles à étudier ce qui peut prendre environ 10 heures ;
- Ne prend pas en considération la maintenance préventive au préalable ;
- Ne prend pas en considération le nombre de shifts et les week-ends ;

Points positifs :

- Le code est simple, dynamique et flexible et représente une base solide pour toute future amélioration ;
- La solution est adéquate en terme de complexité car en analysant l'historique de production nous remarquant que chaque mois nous avons un maximum de 6 médicaments produits par ligne ce qui donne un cas limite de $6! = 720$ cas qui peuvent être traités en quelques secondes.
- l'ordonnancement est à horizon modifiable et peut être appliqué pour un mois, une semaine ou même une année ;
- La maintenance peut être prise en compte et peut même être ordonnancée si elle est entrée en tant que produit, transformant une faiblesse de l'algorithme en une option à valeur ajoutée ;

La solution étant déjà adéquate, nous voulons pousser le raisonnement mathématique plus loin ce sera donc le sujet de la section suivante.

4.3 Amélioration de l'algorithme

4.3.1 Un problème de temps

Notre algorithme ayant une complexité exponentielle, le temps d'exécution se voit allongé jusqu'à atteindre des sommets.
Dans cette partie nous allons proposer certaines piste en approfondissant la piste sur l'amélioration mathématique.

1. **Algorithmes metaheuristiques :**
   L'une des solutions proposées est l'utilisation d'algorithmes de recherche plus puissants tels que la méthode de **recuit simulé** ou un algorithme **génétique**.
   Nous utiliserons l'une des méthodes pour les lignes forme sèche car la complexité du problème est élevée alors que pour les lignes liquides une simple simplification mathématique pourrait réduire le temps d'exécution de manière drastique.

2. **Programmation linéaire :**
   Ayant un problème NP-complet, il est possible de le simplifier en utilisant un algorithme de programmation linéaire, c'est une méthode intuitive que nous allons développer par la suite.

3. **Simplification mathématique et intuitive :**
   Par fois, il est inutile d'attaquer un problème de complexité moyenne avec des méthodes complexes. Ceci est le cas pour notre problème actuel alors nous sommes parvenus à trouver des simplifications mathématiques qui permetteraient de réduire la complexité algorithmique du problème.

Ayant énuméré ces solutions possibles à notre problème de complexité nous allons nous approfondir notre analyse pour la dernière méthode.

4.3.2 Amélioration mathématique de l'algorithme

Cette approche consiste à utiliser des tests mathématiques afin de trouver un optimum local (recherche du chemin le plus court Djikstra).
L'amélioration de l'algorithme se fait en deux temps :

1. **Le choix du premier produit :**
   Le choix du produit est assez simple, il suffit de trouver le produit dont les transitions vers lui ont le minimum le plus grand, car en le choisissant nous gagnons inéluctablement la transition au coût le plus élevé au départ, en effet le processus d'ordonnancement est **sans mémoire** et donc le premier produit n'a pas de prédécesseur.

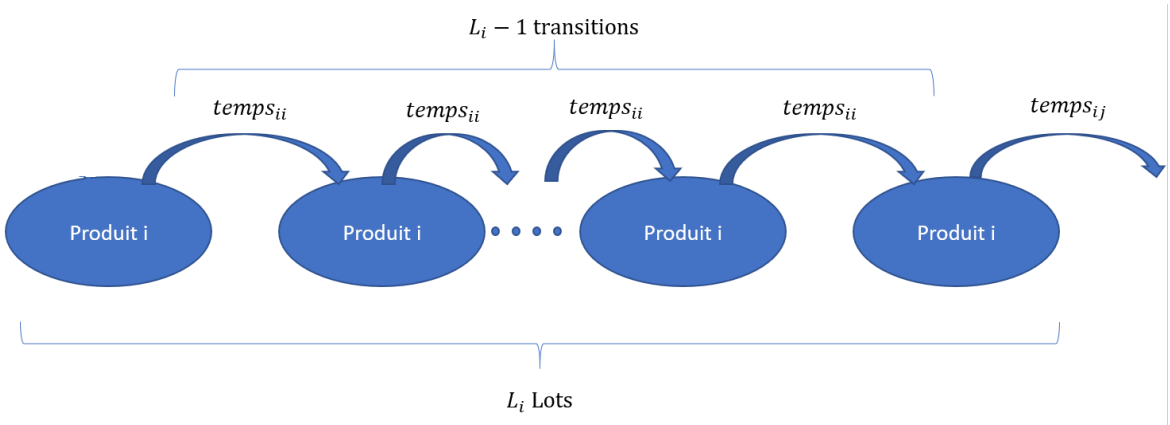


FIG. 4.4 : Schéma explicitant l'absence d'antécédent pour le premier produit

On choisira alors le produit vérifiant :

$$\boxed{max_j(min_i(t^m_{i\to j}))}$$

Le produit vérifiant cette simple formule se verra classé premier.

2. **Le choix du produit suivant :**
   Le choix du produit est assez simple et s'inspire des **chaines de markov**, il suffit de trouver le produit i ayant un temps de transition minimal en connaissant le produit qui le précède $i-1$.
   Il suffira alors de chercher le produit vérifiant :

$$\boxed{min_i(t^m_{(i-1)\to i})}$$

Partie B :
Construction du modèle d'ordonnancement pour la forme sèche

4.4 Construction du modèle d'ordonnancement pour la forme sèche

4.4.1 L'intuition derrière le modèle

Ayant déjà vu la construction d'un modèle mathématique dans la précédente partie, nous allons nous pencher sur le cas des lignes sous forme sèche en utilisant une metaheuristique citée précédemment dans l'état de l'art, **l'algorithme génétique.**
Le principe est de commencer par générer une population initiale (des parents) puis sélectionner les enfants ayant la meilleure mutation et qui deviendront à leur tour des parents.
Ce type d'algorithme n'est pas exact mais il permet d'obtenir des solutions satisfaisantes et d'apprécier l'amélioration de la solution au fil des générations.

4.4.2 Les paramètres et variables du modèle

Nous avons un ensemble de jobs N qui représente les lots à ordonnancer $J = \{J_1, J_2, J_3, ..., J_i, ..., J_N\}$ chaque lot J_i consiste en un ensemble m d'opérations séquentielles $O_{i,j} = \{O_{i,1}, O_{i,2}, O_{i,3}, ..., O_{i,m}\}$ et chaque opération peut être opérer sur un ensemble de M machines $M = \{M_1, M_2, M_3, ..., M_k, ..., M_M\}$, le temps opératoire de l'opération j du lot i dans la machine k est $p_{i,j,k}$

Les ensembles :

- $D \in$ **Produits** $= \{1, 2, \ldots, \mathrm{d}\}$: Ensemble des médicaments représentant les familles des lots a fabriquer ;
- $J \in$ **Lots** $= \{J_1, J_2, J_3, ..., J_i, ..., J_N\}$: Ensemble de lots a produire qui représentent les jobs J.
- $E \in$ **Étapes** $= \{\mathrm{Pesée}, \mathrm{Granulation}, \mathrm{Mélange}, \mathrm{Compression}, \mathrm{Géluleuse}, \mathrm{Pelliculage}, \mathrm{Pilulier}, \mathrm{Encartonnage}\}$, Ensemble d'étapes de production
- $M \in$ **Machines** $= \{M_1, M_2, M_3, ..., M_k, ..., M_M\}$: Ensemble de machines où chaque opération O est traitée par une machine m appartenant à une étape E
- $O_{i,j} \in$ **Opérations** $= \{O_{i,1}, O_{i,2}, O_{i,3}, ..., O_{i,m}\}$: Ensemble d'opérations O du lot J avec leurs séquences tel que $O_{i,1}$ est la première opération et $O_{i,m}$ la dernière
- $S_k \in$ **shifts** $= S_1, S_2, \ldots, S_M$ Le nombre d'équipes possible pour chaque ligne tel que :
$$S_k = 1 + \frac{24}{shift},\ \forall k \in M \tag{4.9}$$
avec $shift = 12,\ 8\ ou\ 6$ qui nous donne $3,\ 4\ ou\ 5$ respectivement.
- PF : Ensemble des produits finis
- PI : Ensemble des produits intermédiaires

Les paramètres

$p_{i,j,k}$: Le temps opératoire de l'opération j du lot i dans la machine k ;

SD : Date début de l'ordonnancement (Généralement le 15 de chaque mois) ;

RD : Date fin de l'ordonnancement (Généralement le 15 du mois prochain) ;

K : Le nombre de machine $m \in M$ disponible dans chaque étape $e \in E$;

ST_j : Temps de lancement (start time) de la tâche $O_{i,j} \in O$;

FT_k : Temps de fin de fonctionnement de la machine m ;

FT_j : Temps de fin (finish time) de l'opération $O_{i,j} \in O$;

co_k : Temps de changeover de la machine k ;

$Lotsize$: La taille de lot pour chaque type de produit D ;

teo : Temps d'attente entre les opérations O_i du même lot J ;

$shift \in (360, 480, 720)$: La durée de chaque shift S_k (6,8,12 heures) ;

$nshift \in (3, 4, 5)$: Le nombre d'équipes nécessaires pour la ligne ;

En résumant l'ensemble des paramètres dans le tableau 4.2

TAB. 4.2 : Les paramètres du modèle de production de la forme sèche

Symbole	Description
i	L'indice des lots J
j	L'indice des opérations O
k	L'indice des machines M
ID	L'indice des médicaments D
D	Ensemble des médicaments, qui représente les familles de lots à fabriquer
J	Ensemble des lots à produire qui sont les jobs J
E	Ensemble d'étapes (Granulation, Compression ... etc)
M	Ensemble des machines
$O_{i,j}$	Ensemble d'opérations O du lot J avec leur séquence
S_k	Le nombre d'équipes possible pour chaque ligne
PF	Ensemble des produits finis
PI	Ensemble des produits intermédiaires
$p_{i,j,k}$	Le temps opératoire de l'opération j du lot i dans la machine k
SD	Date début de l'ordonnancement
RD	Date fin de l'ordonnancement
K	Le nombre de machine $m \in M$ disponible dans chaque étape $e \in E$
ST_j	Temps de lancement de la tâche $O_{i,j} \in O$
FT_k	Temps de fin de fonctionnement de la machine m
FT_j	Temps de fin (finish time) de l'opération $O_{i,j} \in O$
co_k	Temps de changeover de la machine k
$Lotsize$	La taille de lot pour chaque type de produit D
teo	Temps d'attente entre les opérations O_i du même lot J
$shift$	La durée de chaque shift S_k (6,8,12 heures)
$nshift$	Le nombre d'équipes nécessaires pour la ligne

4.4.3 Modélisation

Formulation des notions principales

1. **Les Tâches**

   Les tâches dans le problème sont les opérations de chaque lot qui doivent être traitées par une seule machine, par exemple : L'opération de granulation du lot 2 qui peut être traitée par la machine "SGRN9".

   La pesée ne va pas être prise en considération dans la modélisation parce que son temps opératoire et négligeable par rapport aux autres étapes et elle n'a pas de changeovers, elle va être planifiée selon le plan de l'ordonnancement et selon la séquence des lots à fabriquer, pour préparer la matière première.

2. **Les Ressources**

Les seuls ressources qui existent sont les ressources renouvelables qui sont les machines M et les équipes (shifts) de chaque ligne.

3. **Les Contraintes**

   - Ordonner les tâches selon l'ordre de fabrication (la séquence).
   - Le nombre de machines est limité dans chaque étape.
   - Une machine ne peut opérer qu'une seule opération à la fois.
   - Une machine ne peut traiter qu'une seule opération à la fois.
   - Après chaque changement de type de produit dans une ligne, un temps de passage (changeover) est nécessaire pour préparer la machine pour le prochain produit.

4. **La fonction objectif** Dans un problèmes réel de type flow-shop et flow-shop flexible plusieurs types de fonctions objectif peuvent être utilisés et sont choisis selon la complexité du problème, le nombre de contraintes et les objectifs de l'entreprise elle-même Pinedo 2016.

   Dans le cas de BIOPHARM industrie , la production mensuelle commence le 15 de chaque mois afin d'aligner la production avec le plan de livraison et gagner en flexibilité.

   L'objectif de l'entreprise est de respecter le plan de production tout en optimisant l'utilisation des lignes de production, c'est pour cela que nous avons utilisé deux fonctions objectifs :

   - La 1ère fonction utilisée est la fonction de minimisation de la somme des temps d'achèvement totale (Cj)

     $$Min \sum_{k \in M} C_k$$

     $$\forall k \in M$$

     Car elle permet de dégager de la charge par ligne et donc minimise les temps de changeover et les temps d'arrêts. Si le résultat retourne un plan qui dépasse l'horizon de planification nous utiliserons une seconde fonction qui est le makespan.
   - **Le makespan** va charger d'autres lignes et machines pour réduire le chemin critique du projet et terminer dans les délais.

     Min C_{max}

4.4.4 Choix de la méthode de résolution

Pour la solution du problème de FJSP-SD nous avons choisi un algorithme génétique (métaheuristique basé sur une population) après une recherche bibliographique approfondie.
La **complexité du problème** étant élevée, l'algorithme génétique est l'un des meilleurs choix Zaghdoud 2015pour la résolution de ce type de problème, de plus :

- L'utilisation d'une fonction objectif (fitness) indépendamment de sa nature ; convexe, continue et dérivable ce qui lui donne plus de souplesse et un large domaine d'application.
- La solution est une population de taille N, ce qui permet la génération d'une forme de parallélisme.
- La probabilité de croisement et de mutation permet parfois d'éviter de tomber dans un optimum local et se diriger vers l'optimum global.

Le déroulement de l'algorithme génétique

Inspiré de **l'évolution biologique** des espèces l'algorithme génétique fait partie de la famille des **algorithmes évolutionnaires** où une population est générée et dont chaque individu (aussi dit chromosome) est évalué par une fonction **fitness** qui déterminera la probabilité avec laquelle il risque d'être sélectionné.
Dans la phase de **sélection**, les parents les plus performants sont choisis puis sont croisés à l'aide d'un opérateur de **croisement**afin de générer des enfants.
Les nouveaux individus sont alors **mutés** afin de modifier leur code génétique. Ainsi, seuls les individus les plus performants sont gardés pour former la population de la **génération** suivante.
Cette métaheuristique converge vers une solution optimale au fil des générations, le processus est alors répété jusqu'à satisfaire un critère d'arrêt.

1. **Codage et décodage**

   L'individu est représenté par un chromosome, qui est un vecteur qui porte l'ensemble des indices de chaque opération O_j de chaque lot J_i avec l'ensemble des machines qui peut le traiter M_k.
   Parce que le problème est de type SD d'autres travaux ont proposé des chromosomes en deux parties, l'une pour représenter la séquence d'opérations et l'autre pour l'affectation des machines.

   Le tableau : 4.3 présente un exemple de codage utilisé par l'algorithme génétique. Le tableau contient 3 lots avec leurs séquences d'opérations et les machines possibles, le lot 1 à 3 opérations, l'opération 2 peut être traitée par deux machines 7 et 18 avec un temps opératoire de 400 et 695 minutes respectivement.

   Un chromosome est alors un vecteur de taille **k**, égale au nombre de lignes, généré aléatoirement dans lequel chaque élément a un indice qui mappe une ligne.

Tab. 4.3 : Exemple d'un input de l'algorithme génétique

indice	ID	lot	opération	machine	proc-time	Changeover
0	AT192	1	1	13	420	260
1	AT192	1	2	7	400	810
2	AT192	1	2	18	695	480
3	AT192	1	3	8	794	210
4	BF153	2	1	5	540	180
5	BF153	2	2	18	919	480
6	BF153	2	3	8	1580	210
7	BF153	2	4	8	788	210
8	BX207	3	1	18	30	480
9	BX207	3	2	11	240	210
10	BX207	3	3	8	20	210

Ce chromosome contient tous les indices de l'input, ce qui facilite la mutation et le croisement qui prennent en considération l'ordre des lots. Mais afin de réduire le nombre de combinaisons qui augmente exponentiellement avec le nombre d'opérations, chaque vecteur contenant un indice machine est codé. Le vecteur final est le résultat de la combinaison des vecteurs selon l'ordre des étapes.

Pour le cas précédent voilà l'exemple d'un chromosome codé généré par l'algorithme :

- Pour la machine 5 le vecteur est : [4]
- Pour la machine 13 le vecteur est : [0]
- Pour la machine 18 le vecteur est : [2,8,5]
- Pour la machine 7 le vecteur est : [1]
- Pour la machine 11 le vecteur est : [9]
- Pour la machine 8 le vecteur est : [3,10,6,7]

La combinaison des sous-vecteurs étant effectuée, l'individu résulte de la combinaison de l'ensemble selon l'ordre des étapes (Granulation ensuite la compression ensuite l'encartonnage par exemple) :

$$[4, 0, 2, 8, 5, 1, 9, 3, 10, 6, 7]$$

Ce type de codage est appelé **codage de permutation.**

TAB. 4.4 : Individu codé

indice	4	0	2	8	5
mappe	$j_($2, 1, 5)	$j_($1, 1, 13)	$j_($1, 2, 18)	$j_($3, 1, 18)	$j_($2, 2, 18)
1	9	3	10	6	7
$j_($1, 2, 7)	$j_($3, 2, 11)	$j_($1, 3, 8)	$j_($3, 3, 8)	$j_($2, 4, 8)	$j_($2, 3, 11)

Réparation de l'individu

Afin que la solution soit faisable, le chromosome doit respecter les contraintes suivantes :

- Le séquençage des opérations ;
- Une opération est traitée par une et une seule machine ;

Tant que le chromosome est généré aléatoirement, la réparation du chromosome suit les règles suivantes :

- L'ordre de priorité des opérations sur le vecteur généré va de gauche à droite où l'opération se trouvant à l'extrémité gauche est prioritaire ;
- La séquence d'opération doit être respectée et donc leurs indices doivent être ordonné selon un ordre croissant (ex : J(0,1,4), J(0,2,2), J(0,3,13)) si l'une des opérations ne suit pas un ordre croissant alors une permutations a lieu (ex : J(0,3,13), J(0,2,2), J(0,1,4) devient J(0,1,4), J(0,2,2), J(0,3,13)) et l'ordre est remplacé dans le vecteur ;
- Si une opération existe déjà dans le vecteur elle sera supprimée ;

Voici le résultat de la réparation du chromosome précédent :

$$[0, 2, 8, 9, 3, 4, 10, 5, 7, 6]$$

Le Job 1 (J(1,2,7)) a été supprimé car il existe déjà dans le vecteur sous forme de Job 2 (J(1,2,18)) qui a déjà été parcourue dans l'algorithme.
Les autres opérations ont été modifiés selon la séquence qui doit être respectée.

Après la réparation de l'individu il serait décodé et on passe à la création de tableau des tâches selon les règles suivantes :

- Le chromosome est parcouru de gauche à droite ;
- Une opération $O_{i,0,k}$ commence dès que la machines est disponible ;
- Une opération $O_{i,j,k}$ ne peut commencer que si l'opération antécédente est achevée $O_{i,j-1}$
- Une opération s'achève au bout de son temps opératoire sur une machine ;

- Le temps de changeover est ajouté pour le cas des changements de série

$$ST_j = max\{FT_k, FT_{j-1}\} \qquad \forall j, k \in O, M \tag{4.10}$$

$$FT_j = ST_j + p_{j,k} \qquad \forall j, k \in O, M \tag{4.11}$$

$$if(J_i \neq J_{i-1}) : ST_j = max\{FT_k + co_{k\,j-1}\} \qquad \forall i, j, k \in J, O, M \tag{4.12}$$

Le tableau 4.5 illustre le procédés :

Tab. 4.5 : Le chromosome après le décodage

indice	ID	lot	operation	machine	proc-time	Changeover	ST	FT
0	AT192	1	1	13	420	260	0	420
2	AT192	1	2	18	695	480	0	695
8	BX207	3	1	18	30	480	1175	1205
9	BX207	3	2	11	240	210	0	240
3	AT192	1	3	8	794	210	0	794
4	BF153	2	1	5	540	180	0	540
10	BX207	3	3	8	20	210	1004	1024
5	BF153	2	2	18	919	480	1685	2604
7	BF153	2	3	11	1580	210	450	2030
6	BF153	2	4	8	788	210	1234	2022

2. **La population initiale**

   La population initiale composée de n individu est générée aléatoirement en prenant l'ensemble des indices de la table d'inputs et les mélangeant.

3. **L'évaluation (fitness function)**

   Une fois la population initialisée, une nouvelle population est créée. Une fonction d'évaluation est introduite afin de mesurer les performances de chaque individu de la population qui correspond à une solution donnée du problème à résoudre. Dans notre cas et comme nous avons dit dans **4.3.2.4**, 3 fonctions fitness ont été testées. Pour le cas précédent la somme des temps d'arrêts des machines a donné :

$$\sum_{k \in M} \max C_k = 420 + 2604 + 2030 + 2022 + 540 = 7616$$

4. **La sélection**

   La sélection est un procédé qui permet d'identifier les individus répondant au mieux au problème étudié, c'est-à-dire, que chaque individus est choisi en fonction de sa valeur d'évaluation. Ce procédé ne permet pas de créer de nouveaux individus, mais de privilégier les individus en fonction de leur valeur d'évaluation, et cela, afin de

les exposer au croisement et à la mutation (reproduction).
Dans notre cas nous avons choisi la méthode de sélection **par tournoi** qui consiste à choisir aléatoirement k individus de la population, puis à sélectionner le meilleur individu.
Ce processus est répété à plusieurs fois, avec ou sans remise, jusqu'à ce que le nombre d'individus nécessaires pour le croisement soit atteint.

5. **Le croisement**

   L'opération de croisement permet de générer des individus afin de créer de nouvelles solutions. Le croisement consiste donc à générer à partir de deux individus sélectionnés (parents), plusieurs nouvelles solutions chacune composée d'une partie des caractéristiques de leurs parents.

   Le croisement se fait généralement avec des opérateurs de croisement et le choix d'un opérateur dépend du codage adopté et des caractéristiques du problème.
   Dans notre cas nous avons utilisé **le croisement uniforme à correspondance partielle** Cicirello 2006. Le choix des points du croisement entre les deux parents est effectué d'après des probabilités P_i dont i et la position du point et la probabilité représente la longueur de la partie allant être croisée. 4.5.
   Nous définissons aussi une probabilité P_c qui permet de décider si les parents seront croisés entre eux ou s'ils seront tout simplement recopiés dans la population suivante, est associe a l'algorithme dans le but de garder quelques individus parents dans la prochaine population dans le cas où ceux-ci aient une performance élevée.

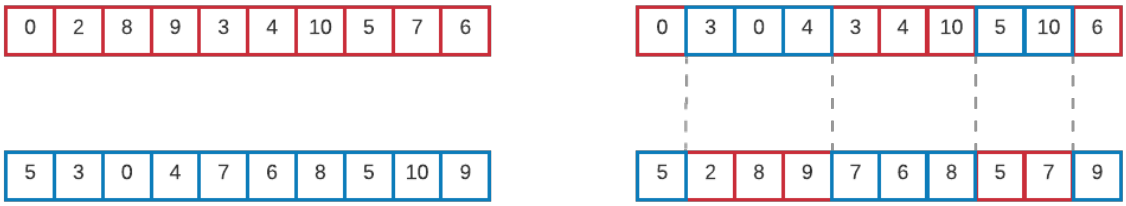


Fig. 4.5 : Croisement uniforme à correspondance partielle

6. **La mutation**

   La mutation permet de maintenir la diversité et empêche la dégénérescence des solutions entre elles dans la population.
   L'opérateur de mutation apporte une propriété d'**érgodicité de parcours d'espace de solutions** aux algorithmes génétiques.
   Cette propriété indique que l'algorithme génétique est capable d'atteindre tout les points de l'espace de recherche, sans pour autant les parcourir dans le processus de résolution Durand 2004.
   L'opérateur de mutation joue le rôle d'un élément perturbateur qui consiste en un changement au niveau des gènes d'un individu choisi avec une certaine probabilité P_m appelée la probabilité ou le taux de mutation. Dans notre cas nous avons

appliqué la méthode des **indices aléatoires** car le problème est un problème de séquence et d'ordre.

4.5 Conclusion

Dans ce chapitre, le modèle fut divisé en deux afin de pour adapter la construction à chaque ligne de production.
Un modèle mathématique simple fut conçut pour le problème des lignes liquides dû à sa simplicité et ses contraintes.
Le problème des lignes sèches quant à lui nécessite une résolution plus abstraite et approfondie dû à sa complexité, l'utilisation de l'algorithme génétique est alors justifiée.

Bien que la modélisation pour l'ensemble des lignes ne soit pas la même, la solution est unique.
La prochain chapitre portera sur l'implémentation informatique de la solution qui rassemblera les modèles construits afin de présenter des résultats utilisables en industrie.

Chapitre 5

Implémentation et résultats

5.1 Introduction

La partie théorique et mathématique étant aboutie, une implémentation s'impose. Ce chapitre portera alors sur la conception de la solution informatique d'ordonnancement et ceci en détaillant les étapes d'implémentation sur machine.
La démarche que nous adopterons pour ce chapitre est celle illustré par la figure5.1

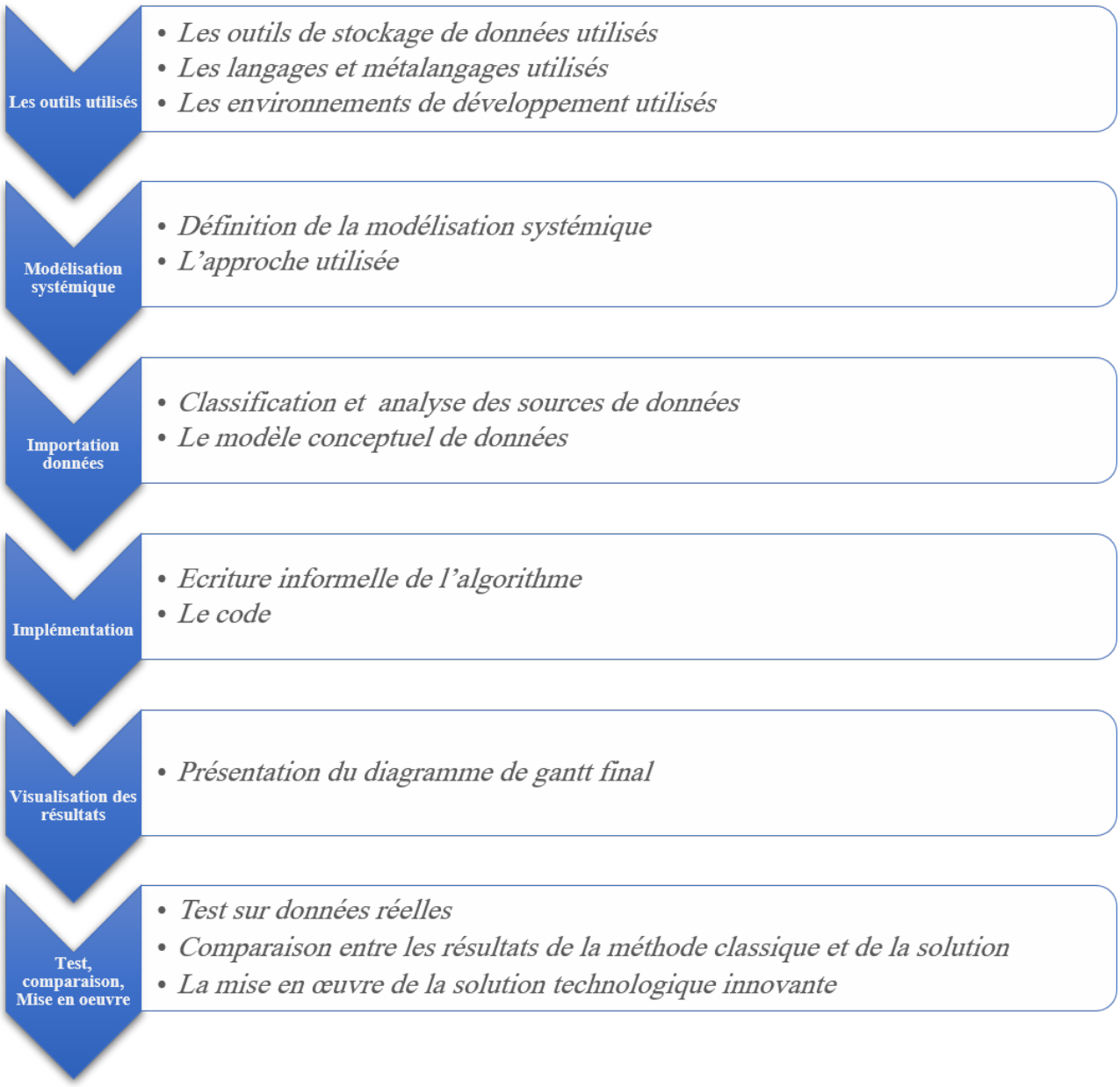


FIG. 5.1 : Structure du chapitre "Implémentation et résultats"

5.2 Les outils informatiques utilisés

5.2.1 Outils de stockage de données

- **L'ERP Sage X3**
  BIOPHARM Industrie utilise actuellement l'ERP Sage X3 pour la gestion des flux informationnels internes, nous l'avons utilisé afin d'extraire les données statiques liées aux lignes de production.

- **Microsoft Excel**
  Les employés au sein de BIOPHARM Industrie maîtrisent parfaitement ce tableur, nous avons eu recours à lui afin d'éviter toute résistance au changement causée par un changement d'outil.

5.2.2 Les langages et meta-langages de programmation

- **Python 3.0**

  L'implémentation du programme a été entièrement conduite sous Python 3.0 et ses différentes bibliothèques.

- **SQL**

  L'importation des données statiques à partir de l'ERP a dû se faire sous forme de requêtes SQL.

5.2.3 Les environnements de développement

- **Google colaboratory**

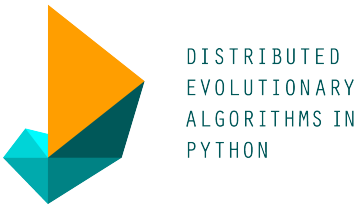
  Google Colab est un environnement de développement entièrement en ligne qui facilite le travail colaboratif, nous l'avons utilisé au départ afin de pouvoir continuer à travailler à distance.

- **Jupyter**

  Jupyter est un environnement de développement que nous avons utilisé après Google Colab, car permet l'installation des solutions sur des serveurs locaux ce qui permet la construction d'interfaces utilisateurs.

5.2.4 Les bibliothèques

- **Deap**
  ”Deap” est une bibliothèque open-source qui offre un nouveau cadre de calcul évolutif pour le prototypage rapide et le test d’idées. Il cherche à expliciter les algorithmes et à rendre les structures de données transparentes.
  Il fonctionne en parfaite harmonie avec les mécanismes de parallélisation tels que le multitraitement et le SCOOP. Parmi les fonctionnalités qu’il comprend : les algorithmes génétiques.



- **Plotly**

  Plotly est une bibliothèque sur Python permettant le dessin de graphiques, nous l’avons utilisé afin d’obtenir une visualisation des résultats sous forme de diagramme de Gantt.

5.3 Modélisation systémique

La décision d’ordonnancement de la production dans les industries de transformation suit les étapes suivantes montré dans **la figure 5.2** G. P. Georgiadis, Elekidis et M. C. Georgiadis 2019. Cette approche systémique est introduite après la spécification du problème pour présenter les étapes nécessaires pour arriver aux résultats les contraintes étant déjà étudiées lors de la sous-section précédente.

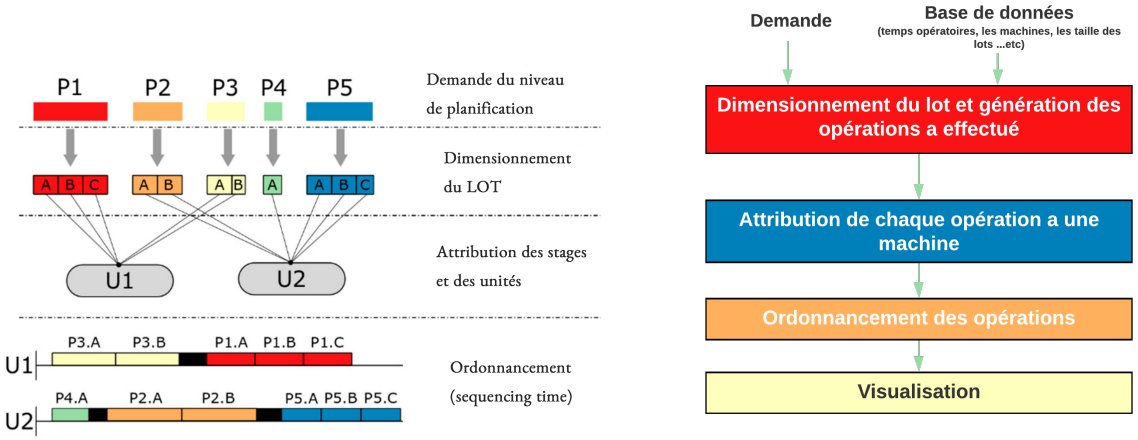


Fig. 5.2 : Les étapes de l’ordonnancement

Cette approche étant intéressante mais impliquerait des difficultés inutiles, nous choisirons de la simplifier en une démarche en trois temps :

1. L’importation et la structuration des données ;

2. L'implémentation de la solution ;
3. La visualisation ;

5.4 L'importation et la structuration des données

Le besoin en donnée étant spécifié lors de la construction du modèle (voir tableau 4.1 et 4.2), nous pouvons alors classer les données en deux familles :

- **Des données statiques :**
  Des données importées directement de l'ERP car elles sont définies par le système de production, par le constructeur ou même par des normes industrielles, celles dont on aura besoin sont :
  - Les produits ;
  - Les machines ;
  - Les lignes ;
  - La cadence/temps de cycle des machines ;
  - Le temps de changeover et de nettoyage ;
  - Les tailles de lot pour chaque produit ;
- **Des données dynamiques :**
  Des données entrées par l'utilisateur au début de chaque exercice, elles sont issues du MRP et de l'étude du marché :
  - La demande pour chaque produit ;
  - Le nombre de lot à produire pour chaque produit ;

Les sources de données étant connues, nous avions conçu un modèle conceptuel de données (figure5.3) permettant de formaliser le besoin en donnée, de normaliser le format des données et d'optimiser l'extraction en évitant les redondances.

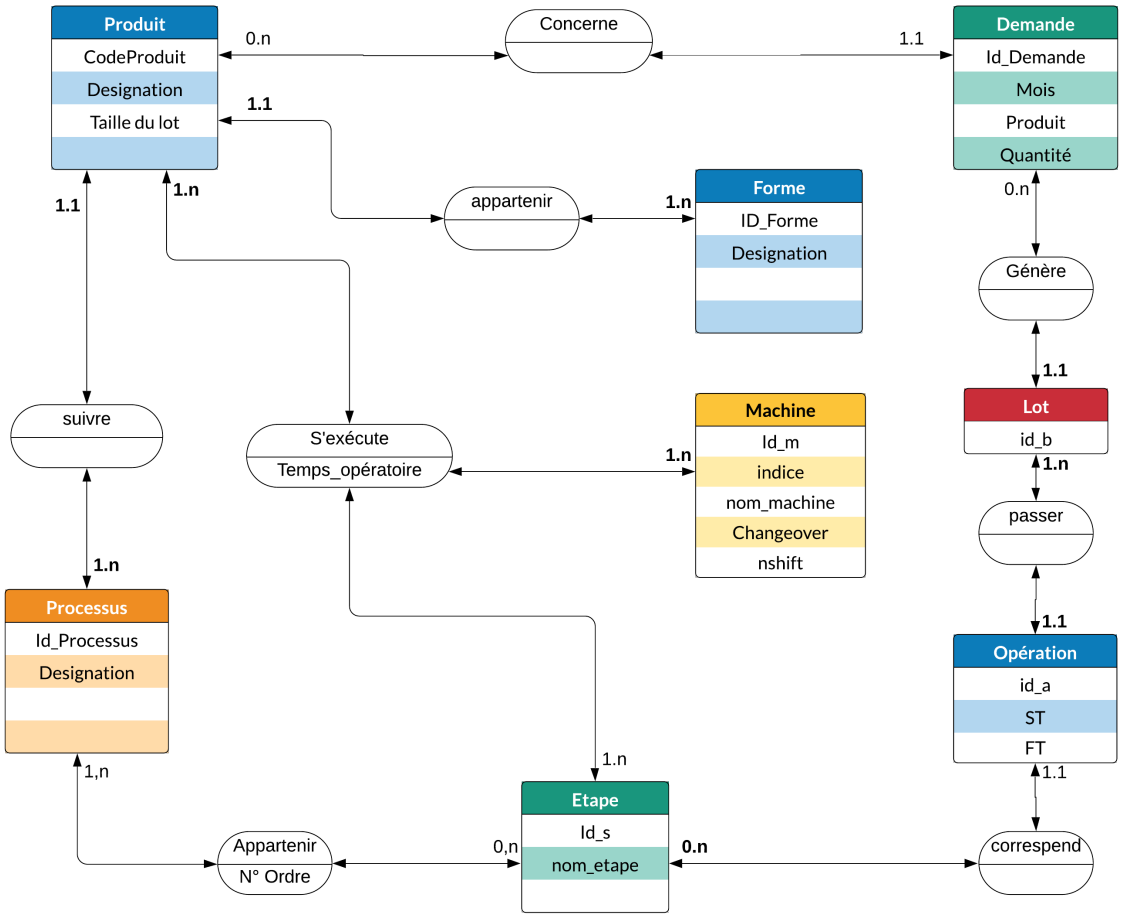


FIG. 5.3 : Le modèle conceptuel des données du problème d'ordonnancement

Les données une fois récupérées sont nettoyées, triées et regroupées dans un tableur facilitant la lecture (Excel car est l'outil de prédilection du département supplychain) et sont par la suite exportées dans un environnement de développement afin d'être exploitées.

5.5 L'implémentation

Les données étant prêtes, l'implémentation peut commencer.
Cette partie étant en grande partie un projet IT, nous suivrons les bonnes pratiques en rigueur en commençant par écrire une description informelle de l'algorithme.

5.5.1 L'algorithme

Un algorithme est une description d'une succession d'étapes nécessaires afin d'obtenir les résultats voulus, nous l'avons illustré dans la **l'algorithme1** afin de

faciliter l'implémentation de la solution sur machine.

Algorithm 1 : Structure générale de l'algorithme

Input : Demande,Produit,Machine,Process,Etape,Calendrier
Output : Lots, opérations, plan d'ordonnancement

```
1  Début
2  Générer les lots
3  Générer la table des opérations à ordonnancer
4  Début de l'algorithme de permutation
5  Initialiser les paramètres de l'algorithme
6  while (n ≤ (nbreproduits) !) do
7      Calcul Calcul des dates de début C_i
8      Evaluation de la performance de la combinaison à l'aide du makespan
9      if NewMakespan ≤ Makespan then
10         Remplacer l'ancien Makespan
11         Remplacer la combinaison optimale
12     else
13         Passer ;
14 Retourner La meilleure combinaison
15 Retourner Le programme de production
16 Retourner Le diagramme de Gantt
17 Fin de l'algorithme de permutation
18 Début de l'algorithme génétique
19 Initialiser les paramètres de l'algorithme génétique
20 Générer la population une population de taille pop = 50
21 while (ngen ≤ 100) do
22     Évaluation des individus
23     Sélection par tournoi de taille tournsize = 5
24     Croisement uniforme à correspondance partielle avec indpb=0.05 et
       cxpb=0.3
25     Mutation avec indpb = 0.05 et mutpb = 0.1
26     Remplacement de la population
27 Retourner Le meilleur individu
28 Retourner Le programme de production
29 Retourner Le diagramme de Gantt
30 Fin de l'algorithme génétique
31 Fin
```

5.5.2 Le code

Afin de ne pas surcharger notre mémoire de code, nous avons séparé les code des lignes sous forme liquide que nous avons mis en annexe A.0.3 des lignes sous forme sèche que nous avons mis en annexe A.0.3.
Ceci est justifiable car le principe et les bibliothèques utilisées sont très différentes, une annexe spéciale permet au lecteur intéressé par un algorithme particulier de le retrouver

facilement. Pour l'algorithme génétique nous avons travailler avec les paramètres présentés dans **5.1**

Tab. 5.1 : Paramètres de l'algorithme génétique

La taille de l'individu	(nombre de lots)(nombre d'opérations)
Population	50
Générations	100
Tournoi	k = 5
Probabilité de croisement	30%
Probabilité de croisement de gènes individuels	5%
Probabilité de mutation	10%
Probabilité de mutation de gènes individuels	5%

5.6 Résultat et visualisation

Après l'obtention de l optimale, il suffit de mettre le plan d'ordonnancement sous forme de graphique afin que celui-ci soit intuitif. Nous avons opté pour un **diagramme de gantt** pour chaque ligne/machine pour sa simplicité et sa clarté. La figure 5.4 illustre le résultat en prenant comme exemple la ligne liquide crème et gel :

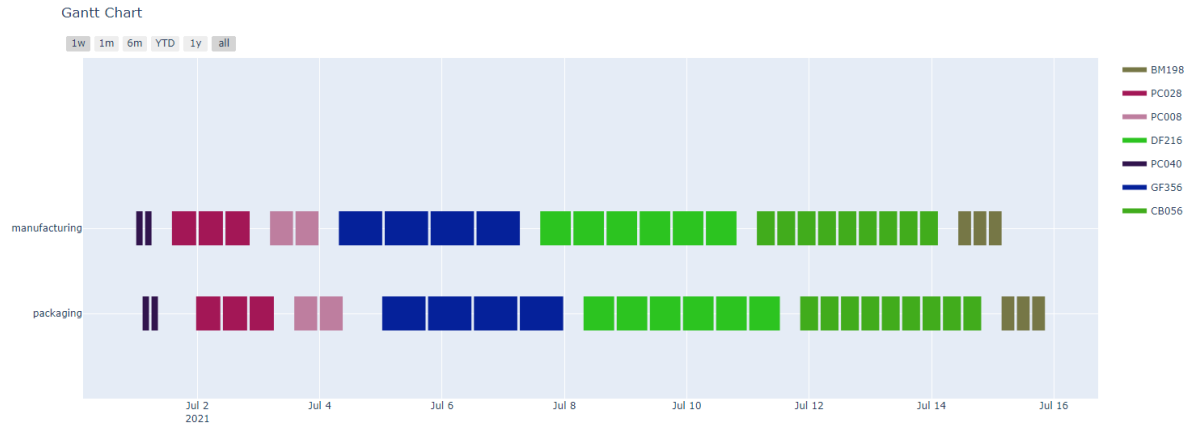


Fig. 5.4 : Diagramme de Gantt de la ligne crème et gel pour le mois de Juin

Afin de mieux illustrer la visualisation du plan d'ordonnancement d'un aspect pratique, nous avons décidé d'appliquer la solution pour les lignes sèches et les résultat sont présentés dans le diagramme de Gantt de la figure5.5

Bien évidemment, la figure précédente ne fait qu'illustrer la complexité de l'ordonnancement, pour obtenir un outil visuel compréhensible il suffit de séparer le diagramme selon les lignes de production et leurs machines.

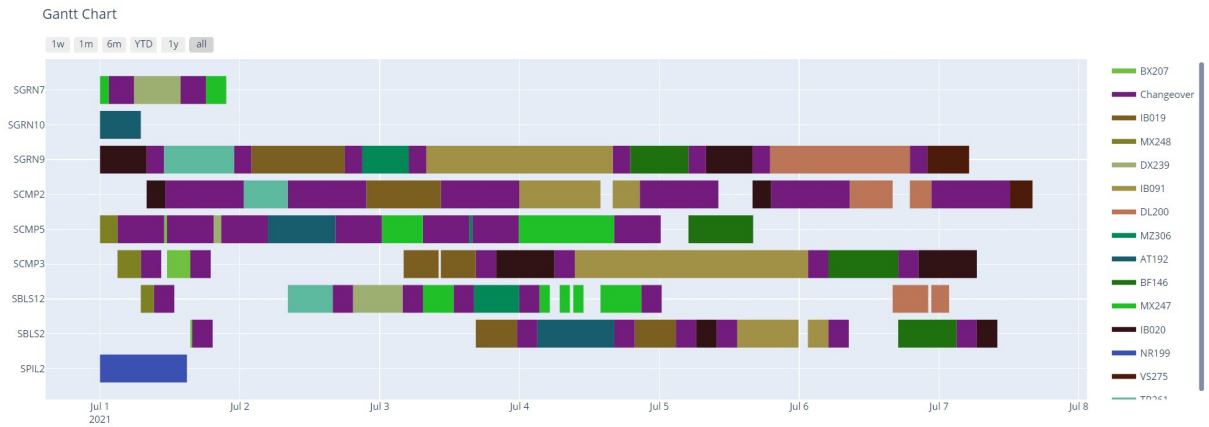


FIG. 5.5 : Diagramme de Gantt global des lignes sèches pour le mois de Juin

5.7 Comparaison des résultats

Afin d'apprécier la valeur ajoutée de la solution, un test a été effectué en utilisant la demande du mois de Juin.

Les données

Nous utiliserons le plan du production du mois de juin (figure5.7)afin d'avoir une bonne base de comparaison.
La liste contient **30** produits avec les quantités demandées, cette liste a généré : **74 lots**, qui ont généré 243 opérations (un tableau avec 243 lignes. l'algorithme va ajouter les changeovers et on peut aller vers plus de **300 opérations** a ordonnancer (opérations plus les changovers).

order	ID	reqdemand	month
601	AT191	222564	6
603	BD152	51426	6
604	BF146	60000	6
605	BF149	106666	6
606	BF153	55172	6
607	BN224	26315	6
609	BN226	24677	6
613	CO237	165000	6
617	CT141	160000	6
621	DZ238	10000	6
625	GX178	83332	6
629	IB019	29629	6
630	IB020	44442	6
631	IB072	59258	6
632	IB091	29628	6
633	IB095	44442	6
634	LM011	1041660	6
635	LT195	130228	6
640	NC051	480000	6
641	NM258	33333	6
642	NV257	16666	6
645	OZ233	2500	6
646	OZ282	11250	6
648	PG143	19696	6
652	RP243	20832	6
653	RV105	9166	6
654	RV231	5833	6
657	SP242	30000	6
658	SP252	18750	6
665	ZL255	53520	6

Les résultats

Le résultat de l'ordonnancement de 74 lots sur les lignes de production de l'entreprise est illustré sur la **figure5.6**.

Comparaison des résultats

L'ordonnancement étant fait en entreprise pour le mois de juin, nous allons comparer les résultats obtenus avec ce qui a été fait.

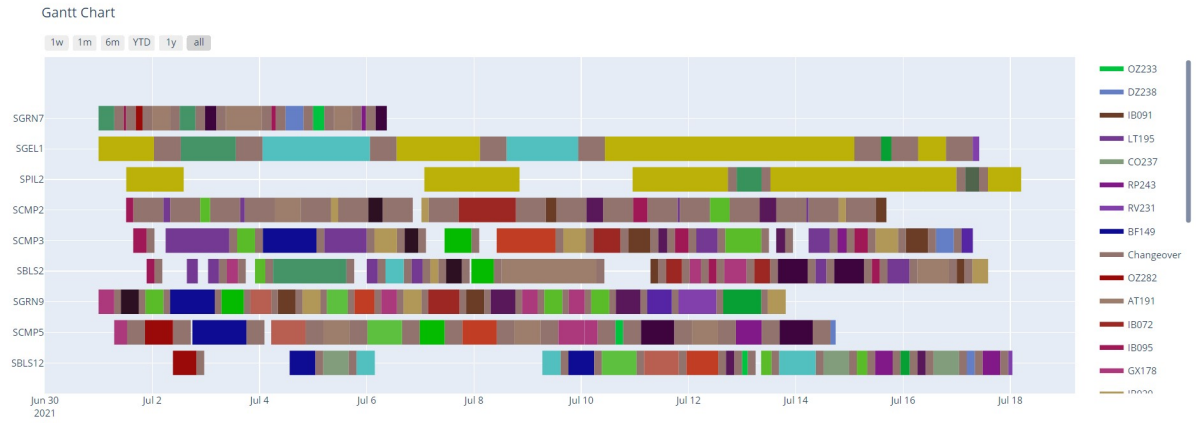
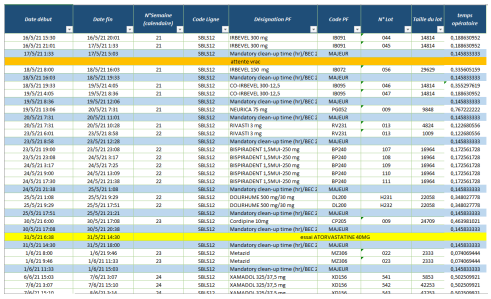


FIG. 5.6 : Ordonnancement de la production du mois de Juin



(a) Plan d'ordonnancement actuel de BIOPHARM(destiné à la lecture)

ID	lot	operation	machine	changeover	Start	Finish
LT195	49	4	5	0	2021-07-01 01:00:00	2021-07-01 10:30:00
LT195	52	4	5	0	2021-07-01 10:30:00	2021-07-01 20:00:00
Changeover	21	5	5	210	2021-07-01 20:00:00	2021-07-01 23:30:00
BP240	21	5	5	0	2021-07-02 08:47:00	2021-07-02 16:47:00
Changeover	5	5	5	210	2021-07-02 16:47:00	2021-07-02 20:17:00
BD152	5	5	5	0	2021-07-04 04:55:00	2021-07-04 12:55:00
BD152	2	5	5	0	2021-07-04 12:55:00	2021-07-04 20:55:00
BD152	6	5	5	0	2021-07-04 20:55:00	2021-07-05 04:55:00
Changeover	42	5	5	210	2021-07-05 04:55:00	2021-07-05 08:25:00
IB091	42	5	5	0	2021-07-05 08:25:00	2021-07-05 18:25:00
Changeover	46	6	5	210	2021-07-05 18:25:00	2021-07-05 21:55:00
IB095	46	6	5	0	2021-07-06 20:56:00	2021-07-07 02:56:00
Changeover	37	5	5	210	2021-07-07 02:56:00	2021-07-07 06:26:00
IB020	37	5	5	0	2021-07-08 07:29:00	2021-07-08 17:29:00
Changeover	4	5	5	210	2021-07-08 17:29:00	2021-07-08 20:59:00
BD152	4	5	5	0	2021-07-09 01:21:00	2021-07-09 09:21:00
BD152	3	5	5	0	2021-07-09 09:21:00	2021-07-09 17:21:00
Changeover	44	6	5	210	2021-07-09 17:21:00	2021-07-09 20:51:00
IB095	44	6	5	0	2021-07-10 23:38:00	2021-07-11 05:38:00
Changeover	19	5	5	210	2021-07-11 05:38:00	2021-07-11 09:08:00
BP240	19	5	5	0	2021-07-11 09:08:00	2021-07-11 17:08:00
Changeover	10	5	5	210	2021-07-11 17:08:00	2021-07-11 20:38:00

(b) Plan d'ordonnancement en généré par la solution (n'est pas destiné à la lecture)

FIG. 5.7 : L'output de l'ordonnancement avant et après la solution

Une simple lecture des résultats permet de relever les avantages de la solution :

TAB. 5.2 : Tableau de comparaison entre les résultats retournés par la méthode classique et la solution

Paramètre	Méthode classique	Solution
La démarche	Absence de démarche	Une démarche claire
Utilisation	L'ordonnancement se fait manuellement	L'ordonnancement est automatisé
Le résultat	N'est pas optimal	Est optimal et a permis de dégager 2 jours de capacité sur certaines lignes et une libération des ressources sur d'autres
Le temps	L'ordonnancement se fait au bout de 3 ou 4 jours	L'ordonnancement se fait en 1h
Les erreurs	Risque d'erreur dû au facteur humain élevé	Aucun risque d'erreur lié au facteur humain
La visualisation	La visualisation se fait directement sur un sheet surchargé	Une visualisation intuitive grâce au diagramme de Gantt
La mise à jour	Fastidieuse, complexe et manuelle	Une simple mise à jour de la demande permettrait d'obtenir un nouvel ordonnancement
L'amélioration	Impossibilité d'améliorer un processus non structuré	Possibilité d'améliorer les algorithmes et la démarche

5.8 La mise en œuvre de la nouvelle ressource technologique

5.8.1 la création et l'acquisition

Ce travail et un travail qui suit le processus d'innovation interne de l'entreprise, il consiste à concevoir un nouvel outil technologique pour l'entreprise. La création et l'acquisition des nouveaux outils technologiques constituent un processus à grand risque d'échecs s'il n'est pas bien piloté, l'entreprise a déjà fait appel à un cabinet de conseil (Mckinsey) pour résoudre le même problème, la solution proposée par le cabinet n'a pas été utilisée car elle n'était pas adaptée au problème. C'est pour cela que nous avons suivi une démarche MRT en mobilisant plusieurs ressources de l'entreprise pour que la solution réponde au besoin des différents départements concernés.

les principales ressources internes à l'organisation qui ont été mobilisés sont principalement :

- Le top management
- Le service de production
- Le service Supplychain
- La direction IT

Nous nous sommes entretenus avec plusieurs employés de divers départements afin de cerner le besoin exprimé par chacun et d'émerger les idées car la création et l'acquisition de nouvelles technologies se fait à travers une organisation en interne (chapitre II) et l'acquisition d'idées et de technologies en externe (chapitre I)

5.8.2 La mise en oeuvre

Nous avons cerné l'étape de la création par l'implication de plusieurs acteurs, mais l'étape de la mise en oeuvre et une étape cruciale et fait partie du processus IT, pour que l'utilisation de l'outil soit avec un avantage plus durable comme montré dans **la figure 5.8**(Corbel 2010).

Les aspects psychologiques et l'alignement avec l'innovation font partie des grands enjeux auxquels doit faire face la mise en oeuvre. C'est pour cela qu'un processus de suivi de mise on oeuvre doit être mis en place par l'entreprise afin de protéger ses investissements et veiller à son bon déroulement.

Ce processus peut être géré par un groupe de pilotage qui va :

- Tester l'outil dans des conditions réels avec des données réelles
- Faire émerger les bonnes pratiques
- Motiver le personnel à utiliser l'outil

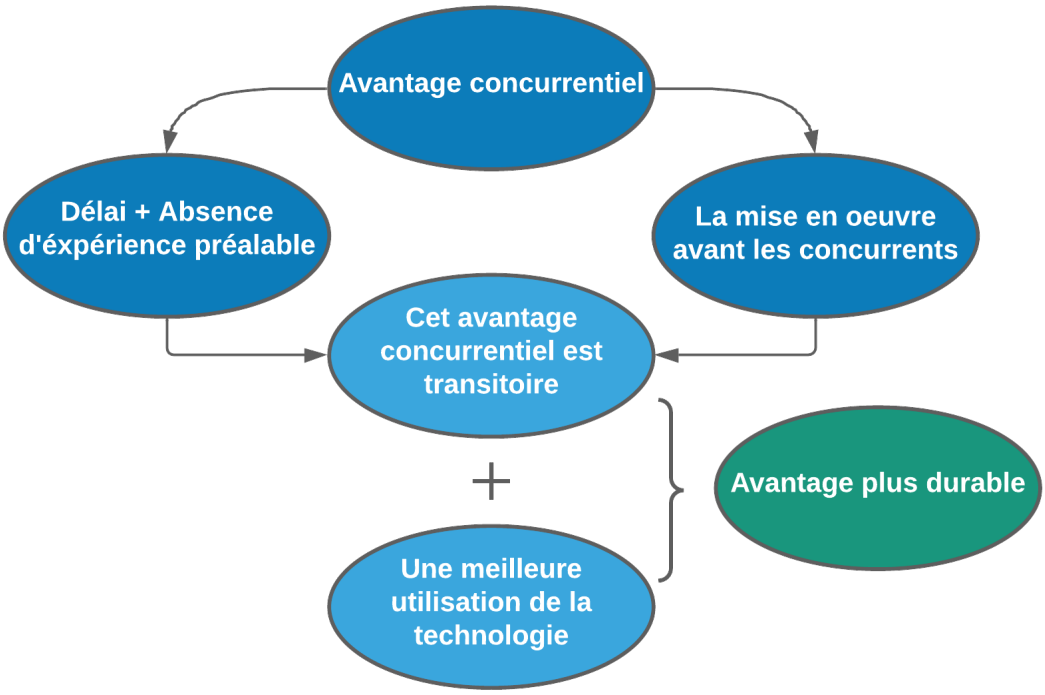


FIG. 5.8 : Les avantages de la mise on oeuvre des nouveaux ressources technologiques

Lors de cette phase de pilotage il faut impliquer le changement par les bonnes pratiques managériales et accompagner les utilisateurs de l'outil tout au long du processus. À la fin de la phase de pilotage un déploiement de masse sera effectué pour que l'outil soit enfin mis en place et puisse être profitable à l'entreprise.

5.9 Conclusion

Ce chapitre marque l'aboutissement de la solution d'ordonnancement après une étude et une construction théorique.
L'implémentation complète de la solution sur machine a permis de la tester et apprécier sa valeur ajoutée.

La mise on oeuvre de la solution nécessite un pilotage, l'étape du pilotage est assez importante et fait partie du processus IT et de ce projet, afin que l'entreprise récolte les avantages de la solution.

Conclusion et perspectives

Conclusion générale

Au regard de l'environnement à haute intensité concurrentielle dans lequel évoluent les entreprises du marché pharmaceutique, se différencier devient plus que nécessaire.
Pour ce faire, chacune d'entre elles doit se fixer des objectifs stratégiques lui permettant d'une part créer un avantage concurrentiel et d'une autre part d'atteindre une performance globale satisfaisante. Et cela dans le but de renforcer son positionnement vis-à-vis ses concurrents, et assurer la pérennité de son activité.

Le management de la production sous tous ses aspect est alors un facteur clé de différenciation et de génération de valeur. Une bonne gestion de la production peut être une source de profitabilité considérable. Cette constatation nous a mené à nous questionner sur les facteurs de réussite de la gestion de production avec une attention particulière portée sur notre thématique qui la planification et l'ordonnancement.

Une étude bibliographique s'est imposée afin de nous imprégner de la théorie autour de la planification de la production et de nous instruire sur les différents types de problèmes et leur résolution afin de les projeter sur l'entreprise, le premier chapitre s'est alors porté vers la définition des divers concepts relatifs à l'ordonnancement et les typologies rencontrés en pratique en industrie.

Afin de projeter les concepts théoriques sur la réalité, la compréhension du contexte de l'entreprise doit être irréprochable, c'est pour cela que nous avions mené un diagnostic complet de BIOPHARM et son environnement à travers des outils tels que l'analyse SWOT-PESTEL et le diagramme d'Ishikawa (chapitre 2).
Ce diagnostic nous permit d'identifier les dysfonctionnements de la planification et à partir desquelles nous avions compris le rôle fondamental de l'ordonnancement au sein de la production, apportant ainsi une réponse à la question posée initialement.

Cela nous a finalement permis d'orienter notre étude sur une problématique soulevée par les services Supply Chain et Production de BIOPHARM Industrie concernant l'ordonnancement de la production dans le but d'enrichir l'entreprise d'une solution informatique permettant d'automatiser la planification de la production et d'éliminer les lacunes observées dans le processus actuel.

Afin de répondre au besoin émis par l'entreprise, nous devions analyser le problème en profondeur en allant directement sur le terrain et prélever les contraintes liées aux machines, à l'approvisionnement, aux normes et au processus.
Cette analyse détaillée dans le "chapitre 3" nous permit de séparer la production en deux lignes présentant des caractéristiques différentes, cette séparation est primordiale car elle élargit le champs des possibilités en donnant la liberté de choisir plusieurs modèles selon leur adéquation avec le problème des lignes respectives.

L'analyse du problème fut tout de suite suivie d'une modélisation. Le problème des lignes de production sous forme liquide étant plus simple que son homologue sous forme sèche, il est entamé en premier.
La documentation étant assez maigre pour les problèmes de Flow-shop, le modèle s'est

alors construit sur un principe simple qui est la permutation et le parcours des différentes combinaisons afin de localiser la meilleure d'entre elles.
La solution répond au besoin émis mais présente quelques failles qui ont été corrigées à l'aide de formules mathématiques et d'algorithmes préalablement crées (Djikstra).

Le problème des lignes de production sous forme sèche fut modélisé, il était alors temps d'entamer la forme sèche qui est un problème de Job-shop dont la complexité est due à son flux de matière multidirectionnel.
Fort heureusement, l'algorithme génétique put être adapté au problème de l'entreprise sans aucune perte de performance tout en retournant un résultat hautement satisfaisant voir optimal.

Le dernier chapitre fut consacré à l'implémentation informatique de la solution et sa cohabitation avec les divers outils existants déjà au sein de l'entreprise.
L'implémentation achevée, nous décidâmes d'effectuer un test de performance afin de nous assurer de la conformité de la solution aux exigences imposées par l'entreprise.
Les résultats étant concluants, la solution est alors déployée au sein du département supplychain qui l'accueille avec engouement.

La solution étant conçue de bout en bout, des améliorations sont à prévoir selon l'évolution de l'entreprise.
Avec la diversification des produits, d'autres usines contenant d'autres lignes de production pourraient voir le jour rendant l'amélioration algorithmique que nous avons proposé moins performantes voir obsolètes .
Afin de pallier à ce problème de dégénérescence de la complexité, nous proposons une évolution vers un système d'ordonnancement central, dynamique, hebdomadaire et automatisé.

- **Central :** Commun à toutes les usines ;
- **Dynamique :** Automatiquement alimenté à partir de l'ERP ;
- **Hebdomadaire :** Un ordonnancement hebdomadaire permet de réduire la complexité de l'ordonnancement ;
- **Automatisé :** Élimine l'intervention humaine ;

S'ajoute à cela l'évolution fulgurante de l'IA et ses applications en milieu industriel, aura-t-il un rôle à jouer dans l'ordonnancement ?

Dans tous les cas, nous pouvons affirmer que ce projet, malgré sa grande complexité, fut une expérience enrichissante qui nous permit de capitaliser les connaissances acquises durant notre cursus universitaire et d'acquérir de nouvelles compétences en IT, en planification et même en gestion de projet.
La beauté derrière le pilotage de projets industriels est le sentiment d'accomplissement que l'on ressent après une création qui sera utile et dont pourront profiter les générations suivantes.

L'ordonnancement est un domaine très vaste touchant le monde académique et industriel, il laisse l'essence même du génie industrie s'exprimer à travers des

formulations mathématiques, une implémentation informatique et une interprétation managériale et fait briller la spécialité de toute sa splendeur.

Bibliographie

Aguirre, Adrian M et Lazaros G Papageorgiou (2018). "Medium-term optimization-based approach for the integration of production planning, scheduling and maintenance". In : *Computers & Chemical Engineering* 116, p. 191-211.

Castro, Pedro M, Ignacio E Grossmann et Qi Zhang (2018). "Expanding scope and computational challenges in process scheduling". In : *Computers & Chemical Engineering* 114, p. 14-42.

Chen, James C et al. (2012). "Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm". In : *Expert Systems with Applications* 39.11, p. 10016-10021.

Cicirello, Vincent A (2006). "Non-wrapping order crossover : An order preserving crossover operator that respects absolute position". In : *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, p. 1125-1132.

Corbel, P. (2010). *Mettre en œuvre les nouvelles technologies*.

De Giovanni, Luigi et Ferdinando Pezzella (2010). "An improved genetic algorithm for the distributed and flexible job-shop scheduling problem". In : *European journal of operational research* 200.2, p. 395-408.

Defersha, Fantahun M et Mingyuan Chen (2010). "A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups". In : *The international journal of advanced manufacturing technology* 49.1, p. 263-279.

Durand, Nicolas (2004). "Algorithmes Génétiques et autres méthodes d'optimisation appliqués à la gestion de trafic aérien". Thèse de doct. INPT.

Esquirol, Patrick, Pierre Lopez et Pierre Lopez (1999). *"L'ordonnancement"*. Economica Paris.

Fattahi, Parviz, Mohammad Saidi Mehrabad et Fariborz Jolai (2007). "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems". In : *Journal of intelligent manufacturing* 18.3, p. 331-342.

Garey, Michael R, David S Johnson et Ravi Sethi (1976). "The complexity of flowshop and jobshop scheduling". In : *Mathematics of operations research* 1.2, p. 117-129.

Georgiadis, Georgios P, Apostolos P Elekidis et Michael C Georgiadis (2019). "Optimization-based scheduling for the process industries : from theory to real-life industrial applications". In : *Processes* 7.7, p. 438.

Goetschalckx, Marc (2011). *Supplychain engineering*. Springer.

Graham, Ronald Lewis et al. (1979). "Optimization and approximation in deterministic sequencing and scheduling : a survey". In : *Annals of discrete mathematics*. T. 5. Elsevier, p. 287-326.

Guimaraes, Kairon Freitas et Marcia Aparecida Fernandes (2006). "An approach for flexible job-shop scheduling with separable sequence-dependent setup time". In : *2006 IEEE International Conference on Systems, Man and Cybernetics*. T. 5. IEEE, p. 3727-3731.

Habchi, Georges (2001). "Conceptualisation et Modélisation pour la Simulation des Systèmes de Production". In : *Rapport d'habilitation à diriger des recherches, Université de Savoie*.

Harjunkoski, Iiro et al. (2014). "Scope for industrial applications of production scheduling models and solution methods". In : *Computers & Chemical Engineering* 62, p. 161-193.

Harrath, Youssef (2003). "Contribution à l'ordonnancement conjoint de la production et de la maintenance : Application au cas d'un Job Shop." Thèse de doct. Université de Franche-Comté.

Hentous, Hamid (1999). "Contribution au pilotage des systèmes de production de type Job Shop". Thèse de doct. Lyon, INSA.

LARIBI, Imane (2018). "Résolution de problèmes d'ordonnancement de type Flow-Shop de permutation en présence de contraintes de ressources non-renouvelables". Thèse de doct. Thèse de doctorat.

Lee, Young Hae, Chan Seok Jeong et Chiung Moon (2002). "Advanced planning and scheduling with outsourcing in manufacturing supply chain". In : *Computers & Industrial Engineering* 43.1-2, p. 351-374.

Li, Jun-qing et al. (2020). "An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times". In : *Knowledge-Based Systems* 200, p. 106032.

Li, Zukui et Marianthi Ierapetritou (2008). "Process scheduling under uncertainty : Review and challenges". In : *Computers & Chemical Engineering* 32.4-5, p. 715-727.

Méndez, Carlos A et al. (2006). "State-of-the-art review of optimization methods for short-term scheduling of batch processes". In : *Computers & chemical engineering* 30.6-7, p. 913-946.

Millman, K Jarrod et Michael Aivazis (2011). "Python for scientists and engineers". In : *Computing in Science & Engineering* 13.2, p. 9-12.

Özgüven, Cemal, Lale Özbakir et Yasemin Yavuz (2010). "Mathematical models for job-shop scheduling problems with routing and process plan flexibility". In : *Applied Mathematical Modelling* 34.6, p. 1539-1548.

Pierre Baptiste Robert pellerin, Michel Guevremont (2019). "L'ORDONNANCEMENT DE LA PRODUCTION : BIEN PLUS QU'UNE QUESTION DE CALCUL". In :

Pinedo, ML (2016). *Scheduling : theory, algorithms, and systems. 5-th ed. Cham*. Springer.

Roshanaei, V, Ahmed Azab et H ElMaraghy (2013). "Mathematical modelling and a meta-heuristic for flexible job shop scheduling". In : *International Journal of Production Research* 51.20, p. 6247-6274.

Shen, Liji, Stéphane Dauzère-Pérès et Janis S Neufeld (2018). "Solving the flexible job shop scheduling problem with sequence-dependent setup times". In : *European Journal of Operational Research* 265.2, p. 503-516.

Talbi, El-Ghazali (2009). *Metaheuristics : From Design to Implementation*. Wiley.

Zaghdoud, Radhia (2015). “Hybridation d’algorithme génétique pour les problèmes des véhicules intelligents autonomes : applications aux infrastructures portuaires de moyenne taille”. In :

Zhang, Guohui et al. (2020). “An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints”. In : *Swarm and Evolutionary Computation* 54, p. 100664.

Webographie

STATISTA (p. d.). *Revenue of the worldwide pharmaceutical market from 2001 to 2020*. URL : https://www.statista.com/statistics/263102/pharmaceutical-market-worldwide-revenue-since-2001/.

Annexes

Annexe A

Présentation des outils

A.0.1 Python

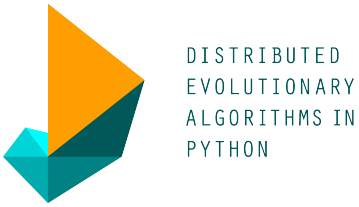
Python est un langage open source créé par le programmeur Guido van Rossum en 1991. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. C'est l'un des langages les plus utilisés par la communauté scientifique pour les calculs et applications sur machines. Son haut niveau d'interactivité avec le codeur et le nombre important de bibliothèques de fonctions scientifiques en libre utilisation font qu'il est de plus en plus utilisé pour un usage académique ou en industrie Millman et Aivazis 2011.

A.0.2 Jupyter Google colab

Nous avons utilisé **JupyterLab** qui est un environnement de développement interactif basé sur le Web pour les blocs-notes, le code et les données. JupyterLab est flexible pour la configuration et l'organisation de l'interface utilisateur pour prendre en charge un large éventail de flux de travail en data science, en calcul scientifique et en machine learning, il est aussi extensible et modulaire.
Et pour le travail collaboratif et à distance nous avons utilisé **Google Colaboratory** qui offre les avantages suivants : aucune configuration requise, accès gratuit aux GPU, partage facile.

A.0.3 La bibliothèque Deap

Nous avons utilisé la bibliothèque **Deap** (open source) développée au Laboratoire de vision et systèmes informatiques (CVSL) de l'Université Laval, à Québec, Canada. C'est est un nouveau cadre de calcul évolutif pour le prototypage rapide et le test d'idées. Il cherche à rendre les algorithmes explicites et les structures de données transparentes. Il fonctionne en parfaite harmonie avec les mécanismes de parallélisation tels que le multitraitement et le SCOOP. Parmi les fonctionnalités qu'il comprend : les algorithmes génétiques.



Sage X3 (anciennement Sage ERP X3) est un progiciel de gestion intégré (ou ERP) destiné aux PME-PMI et filiales de grands groupes, édité par la société britannique éditrice de logiciels Sage. Il est la suite de l'ERP Adonix X3 de la société Adonix, rachetée en octobre 2005 par Sage. Adonix X3 est lui-même le fruit de la fusion des ERP Prodstar 2 de Prodstar (rachetée par Adonix) et d'Adonix Entreprise V2. La première version d'Adonix fut créée par la société Spemi en 1979 à l'initiative de Bertrand Yvinec, Luc Verna et Freddy Salama.

Annexe B

Solution des lignes de production sous forme liquide

Dans cette annexe nous présentons le code implémente pour le problème de l'ordonnancement des lignes sous forme liquide, les parties sont divisées par des commentaires afin de gagner en lisibilité.

```
nstallation
ip install --upgrade plotly
ip install pandas
ip install openpyxl
ibraries
port numpy as np
port pandas as pd
port random
port math
port sys
port matplotlib.pyplot as plt

mporting data from excel
ste_lignes=['LLGV','LLPV','PCRG','SUPP']

r ligne in Liste_lignes :
file = 'Ordonnancement_liquide.xlsx'
LLGV_original = pd.read_excel(file,sheet_name='LLGV')
LLGV_original.drop(LLGV_original[LLGV_original['Nombre de lots'] ==
↪  0].index, inplace=True)
print(LLGV_original)

if LLGV_original.empty :
  print('La ligne '+ligne+' ne produit rien')
  pass
else :
  #Creating the change/over matrix for the manufacturing machine
  Nb_prod=LLGV_original.shape[0]
  COM1=[]
  COM=[]
  for i in range(Nb_prod) :
    for j in range(Nb_prod) :
      if j==i :
        COM1.append(LLGV_original.iloc[i]['C/O same prod (min)1'])
      else :
        COM1.append(LLGV_original.iloc[i]['C/O manufacturing (min)'])
    COM.append(COM1)
    COM1=[]
  COM = pd.DataFrame(COM, columns = LLGV_original['CODE_PF'],
  ↪  index=LLGV_original['CODE_PF'])
  #print(COM)
```

```
#Creating the change/over matrix for the manufacturing machine
Nb_prod=LLGV_original.shape[0]
COP1=[]
COP=[]
for i in range(Nb_prod) :
  for j in range(Nb_prod) :
    if j==i :
      COP1.append(LLGV_original.iloc[i]['C/O same prod (min)2'])
    else :
      COP1.append(LLGV_original.iloc[i]['C/O packaging (min)'])
  COP.append(COP1)
  COP1=[]
COP = pd.DataFrame(COP, columns = LLGV_original['CODE_PF'],
↪  index=LLGV_original['CODE_PF'])
#print(COP)

#Establish all the permutations possible for the products
from itertools import permutations
perm = permutations(LLGV_original.index)
perm=list(perm)
LLGV= LLGV_original

makespan=float('inf')
l=0
#Start permuting
for permus in perm :
  LLGV=LLGV.reindex(permus)

  l=l+1
  #print(l)

#Creating the products data frame by multiplying it by the number of
↪  batches
  Produits=[]
  code=0
  for i in range(Nb_prod) :
      for j in range(LLGV.iloc[i]['Nombre de lots']) :
        P=[code , LLGV.iloc[i]['CODE_PF'] , LLGV.iloc[i]['T
        ↪  manufacturing (min)'] , LLGV.iloc[i]['T packaging (min)'] ,
        ↪  0, 0, 0 , 0 ]
        code=code+1
        Produits.append(P)
  Produits

#Transforming it into a dataframe
```

```
    Produits_original = pd.DataFrame(Produits, columns = ['Clé', 'Code
    ↪  PF','T manuf','T pack','Start time manuf','Finish time manuf'
    ↪  ,'Start time pack','Finish time pack'])
    Prod=Produits_original

  #Finding the start and finish times for all batches
    for i in range(Prod.shape[0]) :
      if i==0 :
        Prod.at[i , 'Start time manuf'] =0
        Prod.at[i , 'Finish time manuf']=Prod.iloc[i]['Start time manuf']
        ↪  + Prod.iloc[i]['T manuf']
        Prod.at[i , 'Start time pack']= Prod.iloc[i]['Start time manuf']
        ↪  + Prod.iloc[i]['T manuf']
        Prod.at[i ,'Finish time pack']=Prod.iloc[i]['Start time pack'] +
        ↪  Prod.iloc[i]['T pack']
      elif i==1 :
        Prod.at[i , 'Start time manuf'] = Prod.iloc[i-1]['Start time
        ↪  manuf']+Prod.iloc[i-1]['T manuf'] + COM[Prod.iloc[i-1]['Code
        ↪  PF']][Prod.iloc[i]['Code PF']]
        Prod.at[i , 'Finish time manuf'] = Prod.iloc[i]['Start time
        ↪  manuf'] + Prod.iloc[i]['T manuf']
        Prod.at[i , 'Start time pack'] = max(Prod.iloc[i-1]['Start time
        ↪  pack']+Prod.iloc[i-1]['T pack']+COP[Prod.iloc[i-1]['Code
        ↪  PF']][Prod.iloc[i]['Code PF']],Prod.iloc[i]['Start time
        ↪  manuf']+Prod.iloc[i]['T manuf'])
        Prod.at[i ,'Finish time pack'] = Prod.iloc[i]['Start time pack']
        ↪  + Prod.iloc[i]['T pack']
      else :
        Prod.at[i , 'Start time manuf'] = max(Prod.iloc[i-1]['Start time
        ↪  manuf']+Prod.iloc[i-1]['T manuf']+COM[Prod.iloc[i-1]['Code
        ↪  PF']][Prod.iloc[i]['Code PF']],  Prod.iloc[i-2]['Start time
        ↪  pack'] - Prod.iloc[i]['T manuf'])
        Prod.at[i , 'Finish time manuf'] = Prod.iloc[i]['Start time
        ↪  manuf'] + Prod.iloc[i]['T manuf']
        Prod.at[i , 'Start time pack'] = max(Prod.iloc[i-1]['Start time
        ↪  pack']+Prod.iloc[i-1]['T pack']+COP[Prod.iloc[i-1]['Code
        ↪  PF']][Prod.iloc[i]['Code PF']], Prod.iloc[i]['Start time
        ↪  manuf']+Prod.iloc[i]['T manuf'])
        Prod.at[i ,'Finish time pack'] = Prod.iloc[i]['Start time pack']
        ↪  + Prod.iloc[i]['T pack']

    iteration_makespan = Prod.iloc[Prod.shape[0]-1]['Finish time pack']
    #print(iteration_makespan)

    if iteration_makespan<= makespan :
      makespan=iteration_makespan
```

```
    Optimal_order= Prod

#importing a date time framework
from datetime import *

date_time_str = '01/07/21 00:00:00'
b= datetime.strptime(date_time_str, '%d/%m/%y %H :%M :%S')

#Converting finish and start time to date time type
Optimal_order['startdate manuf'] = b
Optimal_order['finishdate manuf'] = b
Optimal_order['startdate pack'] = b
Optimal_order['finishdate pack'] = b
Optimal_order['startdate manuf']  = Optimal_order['startdate manuf'] +
↪  pd.to_timedelta(Optimal_order['Start time manuf'],unit='m')
Optimal_order['startdate pack']  = Optimal_order['startdate pack'] +
↪  pd.to_timedelta(Optimal_order['Start time pack'],unit='m')
Optimal_order['finishdate manuf']  = Optimal_order['finishdate manuf']
↪  + pd.to_timedelta(Optimal_order['Finish time manuf'],unit='m')
Optimal_order['finishdate pack']  = Optimal_order['finishdate pack'] +
↪  pd.to_timedelta(Optimal_order['Finish time pack'],unit='m')

#Separatating manufacturing and packaging
m=[]
manuf1 = Optimal_order[['Code PF','startdate manuf','finishdate
↪  manuf']]
for i in range(manuf1.shape[0]) :
  m.append('manufacturing')
manuf1['Task']=m
manuf = manuf1.rename(columns={'startdate manuf' :'Start','finishdate
↪  manuf' :'Finish'})

m=[]
pack1 = Optimal_order[['Code PF','startdate pack','finishdate pack']]
for i in range(pack1.shape[0]) :
  m.append('packaging')
pack1['Task']=m
pack = pack1.rename(columns={'startdate pack' :'Start','finishdate
↪  pack' :'Finish'})

gf=manuf.append(pack)

#Generate colors for gantt diagram
r = lambda : random.randint(0,200)
colors = ['#%02X%02X%02X' % (r(),r(),r())]
for i in range(Prod.shape[0]) :
```

```
    colors.append('#%02X%02X%02X' % (r(),r(),r()))
    #Draw gantt diagram
import plotly.figure_factory as ff
fig = ff.create_gantt(gf, colors=colors, index_col='Code PF',
 ↪  show_colorbar=True,title='Diagramme de Gantt pour la ligne'+ligne,
 ↪  group_tasks=True, bar_width=0.2,showgrid_x=True, showgrid_y=True)
fig.show()
```

Annexe C

Solution des lignes de production sous forme sèche

Dans cette annexe nous présentons le code implémenté pour le problème de l'ordonnancement des lignes sous forme sèche, les parties sont divisées par des commentaires afin de gagner en lisibilité.

```
artie 1 : génération des lots et des opérations :
port numpy as np
port pandas as pd
om datetime import datetime
rom xl2dict import *

le = 'Input.xlsx'
tput = 'output_Solide.xlsx'
ead and create the dataframes

ug = pd.read_excel(file,sheet_name='Drug')
ep = pd.read_excel(file,sheet_name="Step")
ocess = pd.read_excel(file,sheet_name="Process")
chine = pd.read_excel(file,sheet_name="Machine")
mande = pd.read_excel(file,sheet_name="Demand")
ug_mach = pd.read_excel(file,sheet_name="Couple Machine-Produit")
te = pd.read_excel(file,sheet_name="Dates")

sm = pd.merge(step,machine)
dd = pd.merge(drug,demande)

batch creation
= []
r i in dfdd.index :
m.append(np.ceil(dfdd.at[i,'reqdemand'] / dfdd.at[i,'batchsize']))
dd['nbatch']= m
= 0
```

```
= 1
ch = []
 = []
tch = pd.DataFrame()
= 1
r index, row in dfdd.iterrows() :
  j = row['nbatch']
  while i <= j :
      ID.append(row['ID'])
      btch.append(k)      #btch.append( i*100000 + index * 100 +
      ↪  row['month'] )
      i +=1
      k +=1
  i = 1
tch['ID'] = ID
tch['lot'] = btch    # dataframe of batches with the product name
tch=pd.merge(batch,drug)
tch=pd.merge(batch,process)

tch=batch.sort_values('lot').reset_index()
tch = batch.drop(['index'], axis=1)

reate activity
= 1
= []
 = []
t = []
tegory =[]
ape = []
lding=[]
eration =[]
7

tivity = pd.DataFrame()
rint(batch.to_string())
r index , row in batch.iterrows() :
  while i <= j :
      s = 6 + i
      m = str(batch.iloc[index][s])
      if m  != 'nan'  :
          #a.append(row['lot']*100 + i )
          ID.append(row['ID'])
          lot.append(row['lot'])
          category.append(row['category'])
          etape.append(m)
          holding.append(row['holdingtime'])
```

```
            operation.append(i)
        i += 1
    i = 1

ctivity['id_a'] = a
tivity['ID'] = ID
tivity['lot'] = lot
tivity['operation'] = operation
tivity['step'] = etape
tivity['holdingtime'] = holding

tivity["order"] = np.arange(len(activity))

tivity = activity.merge(step, how='left', on='step')

tivity = activity.merge(machine, how='left', on='step')
↪  #.set_index("order").iloc[np.arange(len(activity)), :]
rint(activity.to_string())

tivity = pd.merge(activity,drug_mach)

tivity = activity.sort_values("order")
rint(activity.to_string())

chines = machine.set_index(['id_m'])
nom = activity #.drop(['nmachine','id_a','category','n_machine','id_m'],
↪  axis=1)
nom = genom[['ID','lot','operation', 'machine', 'proc-time',
↪  'holdingtime','changeover']]

iter = pd.ExcelWriter(output)
nom.to_excel(writer,'data')
chine.to_excel(writer,'machine')
te.to_excel(writer,'Agenda')
iter.save()
int('DataFrame is written successfully to Excel file.')



artie 2 : l'algorithme génétique

port pandas as pd
port os
port random
port numpy as np
om deap import algorithms
```

```
om deap import base
om deap import creator
om deap import tools
port math
port sys
port matplotlib.pyplot as plt

lire le fichier excel contenant l'ensemble de données (changer le nom de
↪  la feuille pour changer l'entrée)
EET_NAME = 0
T_NUMBER = 1
w_df = pd.read_excel(os.path.join('output_Solide.xlsx'),
↪  sheet_name=SHEET_NAME)
tput = 'Output_genetic.xlsx'

EET_NAME = 1
chine = pd.read_excel(os.path.join('output_Solide.xlsx'),
↪  sheet_name=SHEET_NAME)
chine = machine[['machine','id_m']]

our la préparation de la population
p = pd.read_excel('output_Solide.xlsx')
p =pop.reset_index()
ch = pop['machine']

dataframe contenant tous les jobs
bs_df = raw_df[['ID','lot', 'operation', 'machine',
↪  'proc-time','changeover']]
bs_df = jobs_df.dropna()
bs_df[['lot', 'operation', 'machine','changeover']] = jobs_df[['lot',
↪  'operation', 'machine','changeover']].astype(int)
int(jobs_df.head())

quelques paramètres des travaux à effectuer
TS = jobs_df['lot'].nunique()
ERATIONS = jobs_df['operation'].nunique()
CHINES = jobs_df['machine'].nunique()
BS = len(jobs_df.index)

onction qui génére les individues

responsable de la réparation individuelle
1. Supprime les opérations répétées en privilégiant l'ordre d'apparition
↪  sur le chromosome
2. s'assure que les opérations respectent les prérequis, en privilégiant
↪  l'ordre d'apparition sur le chromosome
```

```
# 3. Il ajoute le temps de changeover selon l'ordre des opérations

f fix_individual(individual) :
# Créer un nouveau individual :
individual = shuffle(mach)

# crée une base de données pour représenter l'individu d'origine
individual_df = pd.DataFrame(columns=['lot','operation','machine'])

# remplit la base de données de l'individu d'origine
for i in individual :
  lot = jobs_df.loc[i, 'lot']
  operation = jobs_df.loc[i, 'operation']
  machine = jobs_df.loc[i, 'machine']

  # vérifie si l'opération est déjà dans le dataframe de l'individu
  is_already = not individual_df[(individual_df['lot'] == lot) &
  ↪  (individual_df['operation'] == operation)].empty
  if(is_already) :
    continue

  individual_df.loc[i, ['lot', 'operation', 'machine']] = lot, operation,
  ↪  machine
individual_df = (individual_df.reset_index()).drop('index', axis=1)

# créer un dataframe pour représenter l'individu réparé
fixed_df = pd.DataFrame(columns=['lot','operation','machine'])

# réparer l'individu
for i in individual_df.index :

  if(not (i in individual_df.index)) :
    continue

  lot = individual_df.loc[i, 'lot']
  operation = individual_df.loc[i, 'operation']
  machine = individual_df.loc[i, 'machine']

  # vérifie si l'opération de ce lot est déjà dans le dataframe
  is_already = not fixed_df[(fixed_df['lot'] == lot) &
  ↪  (fixed_df['operation'] == operation) & (fixed_df['machine'] ==
  ↪  machine)].empty
  if(is_already) :
    continue

  # vérifie si cette opération peut être effectuée
```

```
    prev_lot_op = fixed_df.loc[fixed_df['lot'] == lot]['operation'].max()

    if(math.isnan(prev_lot_op)) :
      prev_lot_op = 0

opération ne peut pas être effectuée, les prérequis doivent être
↪  effectués avant
    if(operation - prev_lot_op  != 1) :

      # recherche de prérequis
      lot_req_op = (individual_df.loc[(individual_df['lot'] == lot) &
                                      (individual_df['operation'] <
                                      ↪  operation)]).sort_values(by='operation',
                                      ↪  ascending=True)

      indexes = lot_req_op.index.values

      # supprime les prérequis du cadre de données de l'individu d'origine
      individual_df = individual_df.drop(indexes)

      # ajouter des prérequis au cadre de données fixe
      fixed_df = (fixed_df.append(lot_req_op, ignore_index=True))

    # ajouter l'opération au dataframe de l'individu fixe
    fixed_lst = [[lot, operation, machine]]
    df = pd.DataFrame(fixed_lst, columns = ['lot', 'operation', 'machine'])
    fixed_df = fixed_df.append(df, ignore_index=True)

    # supprimer l'opération de bloc de données de l'individu
    individual_df = individual_df.drop(i)

# remplit l'individu construit selon le dataframe
fixed_individual = []
for i in fixed_df.index :
    lot = fixed_df.loc[i, 'lot']
    operation = fixed_df.loc[i, 'operation']
    machine = fixed_df.loc[i, 'machine']

    index = jobs_df.loc[(jobs_df['lot'] == lot) &
                        (jobs_df['operation'] == operation) &
                        (jobs_df['machine'] == machine)].index.values[0]

    fixed_individual.append(index)

return fixed_individual
```

```
ndividual
N indexes that are mapped to jobs_df

f decode(individual) :

# ajuster l'ordre des indices en fonction de l'ordre des opérations
individual_fixed = fix_individual(individual)

# créer un cadre de données pour indiquer l'application des opérations
↪  sur les machines
schedule_df =
↪  pd.DataFrame(columns=['ID','lot','operation','machine','changeover'
,'start','finish'])
start = 0
last_finish_machine = 0
last_finish_prev_op = 0
# remplit toute la base de données avec les informations de l'individu
for i in individual_fixed :
  ID = jobs_df.loc[i,'ID']
  lot = jobs_df.loc[i, 'lot']
  operation = jobs_df.loc[i, 'operation']
  machine = jobs_df.loc[i, 'machine']
  changeover = jobs_df.loc[i, 'changeover']

  proc_time = jobs_df.loc[(jobs_df['lot'] == lot) &
                          (jobs_df['operation'] == operation) &
                          (jobs_df['machine'] ==
                          ↪  machine)].reset_index().loc[0, 'proc-time']



  # vérifier la fin de la dernière opération de la machine
  last_finish_machine = schedule_df.loc[schedule_df['machine'] ==
  ↪  machine]['finish'].max()
  if(math.isnan(last_finish_machine)) :
    last_finish_machine = 0

  # vérifier quand la fin de la dernière opération batch a été
  last_finish_prev_op = schedule_df.loc[schedule_df['lot'] ==
  ↪  lot]['finish'].max()
  if(math.isnan(last_finish_prev_op)) :
    last_finish_prev_op = 0

  #récupérer l'index de la derniere opération effectuer dans la
  ↪  machine :
```

```
max_index = schedule_df.loc[schedule_df['machine'] ==
↪  machine]['finish']
if(max_index.empty == True) :
  max_index = i
  last_product = jobs_df.loc[max_index,'ID']
else :
  max_index = pd.to_numeric(max_index)
  max_index = max_index.idxmax()
  last_product = schedule_df.loc[max_index,'ID']
#print('id :', i, 'max_index :', max_index)
#récupere le ID du dernier produit ffabriqué par la machine on
↪  utilisant l'index :
#last_product = scedule_df.loc[max_index,'ID']
#print(last_product)
#print(jobs_df.loc[i]['ID'])

#L'operation commence une fois la machine sera libre est la derniere
↪  etape sera terminé
#L'operation aussi commence aprés le changeover qui dépend du type de
↪  produit

if jobs_df.loc[i,'ID'] == last_product :          #Si on va fabriqué un
↪  produit de meme type :
  start = max(last_finish_machine , last_finish_prev_op)  # pas de
  ↪  changeover
  #print('The same')
  change = 0
else :
  start = max(last_finish_machine + changeover , last_finish_prev_op)
  ↪  #sinon on ajoute le changeover
  change = changeover
 # print('Different')
#print('start :',start, 'last_finish_machine :', last_finish_machine,
↪  'last finish prev operation' , last_finish_prev_op)
#print('process_time :',proc_time ,'changeover : ',changeover)

 #Add changeover table :
if change > 0 :
  change_lst = [['Changeover', lot, operation, machine, changeover,
  ↪  last_finish_machine , last_finish_machine + changeover ]]
  cf =pd.DataFrame(change_lst, columns = ['ID','lot', 'operation',
  ↪  'machine','changeover', 'start', 'finish'])
  schedule_df =schedule_df.append(cf,ignore_index = True)
  #changeover_df = changeover_df.append(cf,ignore_index= True)
```

```
  #Create the schedule list of all the operations
  schedule_lst = [[ID, lot, operation, machine, 0 , start, start +
  ↪  proc_time]]
  df = pd.DataFrame(schedule_lst, columns = ['ID','lot', 'operation',
  ↪  'machine','changeover', 'start', 'finish'])
  schedule_df = schedule_df.append(df, ignore_index=True)
  #print('lot :',lot,'operation :',operation,'machine :',machine)

return schedule_df



f objective_function(individual) :
schedule_df = decode(individual)
grouped_df = schedule_df.groupby("machine")
maximums = grouped_df.max()
maximums = maximums.reset_index()
somme = sum(maximums['finish'])
makespan = schedule_df['finish'].max()
return makespan,


eator.create("FitnessMin", base.Fitness, weights=(-1.0,))        #
↪  fonction objectif : nom, type(d.o.), poids de chaque objectif (dans
↪  ce cas un seul objectif)
eator.create("Individual", list,  fitness=creator.FitnessMin)   #
↪  individuel
reator.create("FitnessMulti", base.Fitness, weights=(-1.0, -1.0 )) #s'il
↪  est multi objectifs
reator.create("Individual", list,  fitness=creator.FitnessMulti)


olbox = base.Toolbox()
Initialiseur individuel et de population

olbox.register("indices", random.sample, range(JOBS), JOBS)
olbox.register("individual", tools.initIterate, creator.Individual,
↪  toolbox.indices)
olbox.register("population", tools.initRepeat, list, toolbox.individual)
↪  #liste d'individus


Initialiseur d'opérateurs
olbox.register("evaluate", objective_function)
↪  # Function objective
olbox.register("mate", tools.cxUniformPartialyMatched, indpb=0.05)
```

```
olbox.register("mutate", tools.mutShuffleIndexes, indpb=0.05)
olbox.register("select", tools.selTournament, tournsize=5)

p = toolbox.population(n=100)                                    # Initialisation
↪  de la population
f = tools.HallOfFame(1)                                         # Meuilleur
↪  individu
ats = tools.Statistics(lambda ind : ind.fitness.values)  # Statistiques
ats.register("avg", np.mean)
ats.register("std", np.std)
ats.register("min", np.min)
ats.register("max", np.max)

p, log = algorithms.eaSimple(pop, toolbox, cxpb=0.3, mutpb=0.2, ngen=200,
↪  stats=stats, halloffame=hof, verbose=True)

Meilleure solution
int("Meuilleur solution :")
= hof[0]
int(hof[0])
int(decode(hof[0]))
int(decode(hof[0])['finish'].max())
=decode(hof[0])

Meilleur résultat de la fonction objectif
int("Meilleur résultat de la fonction objectif :")
int(objective_function(hof[0])[0])

hedule_df = df
kespan = schedule_df['finish'].max()
= schedule_df[schedule_df['ID']=='Changeover'].index
=len(n)
chine_last = pd.concat([schedule_df[schedule_df["machine"]==
↪  2],schedule_df[schedule_df["machine"]== 3],
↪  schedule_df[schedule_df["machine"]==
↪  12],schedule_df[schedule_df["machine"]== 14] ], ignore_index=True)
ouped_last = machine_last.groupby("machine")
ouped_df = schedule_df.groupby("machine")
ouped = schedule_df.groupby("ID")

xm = grouped_last.max()
xm = maxm.reset_index()

ximums = grouped_df.max()
ximums = maximums.reset_index()
```

```
x =grouped.max()
x = max.reset_index()

mme = sum(maximums['finish'])
m = sum(max['finish'])
mm = sum(maxm['finish'])

int('makespan = ', makespan)
int('changeovers =', n)
int('Total completion time of packaging machines =',somm)
int('Total completion time of all machines = ',somme)
int('Total completion time of all jobs = ',som)

n, min, avg, max = log.select('gen', 'min', 'avg', 'max')
t.plot(gen, min)
t.plot(gen, avg)
lt.plot(gen, max)
t.xlabel('generation')
t.legend(['minimum makespan', 'average makespan','maximum makespan'])

réation des dates selon le planning de l'entreprise
om datetime import *

te_time_str = '01/07/21 00:00:00'
 datetime.strptime(date_time_str, '%d/%m/%y %H :%M :%S')

['startdate'] = b
['finishdate'] = b


['startdate']  = df['startdate'] + pd.to_timedelta(df['start'],unit='m')
['finishdate']  = df['finishdate'] +
↪  pd.to_timedelta(df['finish'],unit='m')

= df.rename(columns={'machine' :'Task','startdate' :
↪  'Start','finishdate' :'Finish'})
l = list(gf['lot'])
r i in range(len(lol)) :
lol[i] = str(lol[i])
['lot']=lol

= lambda : random.randint(0,200)
lors = ['#%02X%02X%02X' % (r(),r(),r())]
r i in range(1, gf.lot.nunique()+1) :
colors.append('#%02X%02X%02X' % (r(),r(),r()))
```

```
 = pd.merge(df,machine)
= gd.rename(columns={'id_m' :'Task','startdate' :
↪  'Start','finishdate' :'Finish'})
f= df.rename(columns={'machine' :'Task','startdate' :
↪  'Start','finishdate' :'Finish'})
l = list(gf['lot'])
r i in range(len(lol)) :
lol[i] = str(lol[i])
['lot']=lol

port plotly.figure_factory as ff
port plotly.express as px
port plotly
port pandas as pd

Assign Columns to variables
sks = gf['Task']
art = gf['Start']
nish = gf['Finish']
mplete = gf['lot']

Create Gantt Chart
g = px.timeline(gf, x_start=start, x_end=finish, y=tasks, color=complete,
↪  title='line Overview')

Upade/Change Layout
g.update_yaxes(autorange='reversed')
g.update_layout(
      title_font_size=42,
      font_size=18,
      title_font_family='Arial'
      )

Interactive Gantt
g = ff.create_gantt(gf,colors=colors, index_col='ID', show_colorbar=True,
↪  group_tasks=True, bar_width=0.4,showgrid_x=True, showgrid_y=True)
g
Save Graph and Export to HTML
otly.offline.plot(fig, filename='Task_Overview_Gantt.html')

iter = pd.ExcelWriter(output)

.to_excel(writer,'data')
.to_excel(writer,'data')
iter.save()
int('DataFrame is written successfully to Excel file.')
```