République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi ouzou
Faculté des sciences
Département des Mathématiques

Mémoire de fin d'études

en vue de l'obtention du diplôme de master en Recherche opérationnelle

Thème : Flow optimization in local manufacturing Tizi-Ouzou

Présenté par :

Mlle. Dalila Meradji **Mlle. Célia Miloudi**

Soutenu publiquement le 30/06/2024, devant le jury :

Mr KASDI.K	MAA	UMMTO	Président
Mr AOUANE.M	MAA	UMMTO	Rapporteur
Mr TALEM.D	MCB	UMMTO	Examinateur
Mr BENMESSAOUD.M	Invité	Novo nordisk	Encadreur entreprise

Promotion : 2023/2024

Remerciements

On remercie Dieu de nous avoir accordé la force et le courage de mener jusqu'à la fin de ce travail.
On tient à remercier M. AOUANE enseignant au département de mathématique, pour son encadrement, son soutien et son aide dans la réalisation de ce travail, on lui en est très reconnaissantes.
Nous adressons également, nos sincères remerciements à Monsieur BENMESSAOUD , Manager du département supply chain de Novo Nordisk site LMTO, pour son engagement et pour la confiance qu'il nous a accordé dès notre arrivée au sein de son équipe.Nous le remercions aussi pour tous les conseils qu'il nous a prodigués et qui nous ont aidé à chaque stade de notre travail.
Nos vifs remerciements vont à l'égard des membres de jury pour avoir accepter de présider et de juger ce modeste travail.

Nous remercions également, l'équipe de LMTO, pour nous avoir accueillis et intégrés au sein de leur grande famille. Nous tenons à remercier également les membres de nos familles, nos amis et tous ceux qui nous ont épaulé tout au long de ces années d'études.

Dédicace

J'ai le plaisir de dédier ce travail à :
Mes chers parents, que nulle dédicace ne puisse témoigner de ma gratitude, pour tous les sacrifices qu'ils ont consentis pour mon bien-être, leur patience inépuisable, leur encouragement et leur aide.
Ma sœur *Massiva* , mon frère *Lamine* qui me sont très chers.
Ma chère, binôme*Célia* avec laquelle j'ai passé des moments inoubliables, et qui sans elle ce travail n'aurait pas pu être réalisé.
Mes chères amies *Dehbia, Anais, Mélissa, Nina et Ryma* qui sans leur encouragement ces pages n'auraient jamais vu le jour.
Et à toute ma famille et tous ceux que j'aime.

Dalila

Dédicace

DIEU tout-puissant merci d'être toujours auprès de moi

Je dédie ce projet aux êtres les plus chers à mon cœur que j'aime énormément :

Mes parents source d'amour et de courage, qui ont sacrifié pour que je puisse arriver jusque-là et qui m'ont beaucoup soutenu durant mon cursus, mais surtout pour leurs confiances en moi, et que DIEU les garde et les protège toujours pour nous.

À mes fréres Ouamer, Lounes et Ghiles qui me sont très chères et surtout leurs enfants que j'admire beaucoup ainsi que leurs femmes.

À ma très chère sœur Aziza et sa petite princesse que j'aime très fort.

À tous mes amis (es) qui m'ont toujours soutenu, et à tous ceux que j'aime et ceux qui m'ont aidé.

Une spéciale dédicace pour ma meilleure et binôme avec laquelle j'ai passé des moments inoubliables.

Célia

Table des matières

Table des figures

Liste des algorithmes

Introduction générale

La vie moderne mène l'homme actuel à se confronter à des situations problématiques d'un système de relations sociales et économiques très complexe. De plus en plus, d'éléments doivent être pris en compte, lors des prises des décisions concernant une action donnée, qui deviennent l'objet de véritables recherches, qui ne peuvent être menées sans l'aide d'outils mathématiques appropriés. C'est ainsi que s'est développé, un domaine des mathématiques centré sur l'activité de décision : la Recherche Opérationnelle. Cette discipline est définie comme étant l'approche scientifique des problèmes complexes rencontrés dans la direction et la gestion de grands systèmes. La théorie des graphes, la programmation linéaire, l'optimisation combinatoire, les processus aléatoires, la théorie des jeux, les files d'attentes, etc sont les principales techniques mathématiques auxquelles fait recours la recherche opérationnelle. La recherche opérationnelle (RO) est une discipline des mathématiques appliquées qui traite des questions d'utilisations optimales des ressources dans l'industrie et dans le secteur public. Depuis des dizaines d'années, le champ d'application de la RO [1] s'est élargi à des domaines comme l'économie, les finances et la planification d'entreprise.

Même si ses origines remontent à très loin, mais ses premiers champs d'application remontent à la Seconde Guerre mondiale, des efforts conjugués d'éminents mathématiciens (Neumann, Dantzig, Blackett), à qui il avait été demandé de fournir des techniques d'optimisation des ressources militaires. Le premier succès de cette approche a été obtenu en 1940, par le prix Nobel de physique Patrick Blackett, qui résolut un problème d'implantation optimale de radars de surveillance. La dénomination est restée par la suite, même si le domaine militaire n'est plus le principal champ d'application de cette discipline, le mot opérationnel prenant alors le sens d'effectif. Ce sont donc ces mathématiciens qui ont créé une nouvelle méthodologie, caractérisée par les deux mots-clés modélisation et optimisation.

1. RO :Recherche opérationnelle

À partir des années 50, la RO fait son entrée dans les entreprises, et les universités et les grandes écoles commencent à l'enseigner. Puis, au milieu des années 70, sans doute à cause d'un excès d'enthousiasme, au départ et à l'inadéquation des moyens informatiques à l'application des méthodes de la RO, la discipline a stagnée. A partir du milieu des 90, on assiste à un retour en force de la RO, les outils informatiques étant maintenant à la hauteur des méthodes proposées par la discipline. On assiste depuis, à une explosion du nombre de logiciels commerciaux et à l'apparition de nombreuses boites de conseils. Si l'on cherche à trouver des précurseurs à la recherche opérationnelle, on peut penser à Alcuin ou à Euler, qui se sont, tous deux, intéressés à des problèmes de type RO, bien qu'aucune application n'ait motivé leur travail. Sans que l'on en soit toujours conscient, la théorie des graphes est aujourd'hui très présente dans notre société moderne. Cette branche des mathématiques, dont on fait remonter l'origine à Euler. L'histoire de la théorie des graphes débute, peut-être, avec les travaux d'Euler au 18ème siècle et trouve son origine dans l'étude de certains problèmes, tels que celui des ponts de Königsberg (les habitants de Königsberg se demandaient s'il était possible, en partant d'un quartier quelconque de la ville, de traverser tous les ponts de la ville, une et une seule fois et de revenir à leur point de départ). Les problèmes de domination est parmi les problèmes les plus importants et les plus étudiés en théorie des graphes grâce à son intérêt pratique dans la modélisation des problèmes réels issue de l'industrie.

Dans le cadre de ce mémoire, nous nous intéressons particulièrement á l'étude des flux de Novo Nordisk, site LMTO [2].Cette dernière qui est une entreprise pharmaceutique renommée mondialement, spécialisée dans le domaine du diabète.

À l'instar de toutes les industries, l'une des problématiques principales à laquelle LMTO doit faire face est l'optimisation de ses flux, notamment au niveau de la production. Elle conditionne le succès des projets établis.Elle consiste à piloter et à optimiser toutes les opérations liées à la fabrication, à la réception et à la distribution de divers produits tout au long de la chaîne logistique, afin de maximiser la rentabilité et l'efficacité de cette entreprise.

L'objectif de ce mémoire donc est de modéliser cette problématique tout en proposant une solution efficace qui va résoudre le problème en question.

Pour cela, on va faire appel à des méthodes mathématiques de résolution telles que l'ordonnancement, la programmation linéaire, la programmation par contraintes, PNE [3], Branch and Bound ou encore les algorithmes heuristiques et les différents modèles de la gestion des stocks. Ces outils permettent d'analyser et de résoudre de manière efficace les problèmes exposés, en prenant en compte de nombreuses contraintes et variables.

Notre étude est structurée en 5 chapitres :

1. Le premier chapitre, intitulé ***"État des lieux"***, est devisé en sous-chapitres, principalement :
   - Historique de Novo Nordisk.
   - Novo Nordisk en Algérie.
   - Présentation de LMTO.
   - Les missions de Novo Nordisk.
   - Structures et Organisations.
   - Les différents départements de LMTO et son organigramme.
   - Présentation du département Supply Chain.

2. LMTO : Local Manufacturing Tizi-Ouzou
3. PNE : la programmation en nombre entier mixte

2. Le deuxième chapitre est consacré essentiellement aux définitions préliminaires des notions de la théorie des graphes les plus utilisés dans ce mémoire et à la PLNE et gestion des stocks.
3. Dans le troisième chapitre, on va traiter les concepts principaux de la complexité algorithmique.
4. On parlera dans le quatrième chapitre de la problématique de notre travail, de la modélisation et des méthodes de résolutions.
5. Le cinquième chapitre est consacré aux résultats et à l'implémentation, où on exposera notre application conçue pour la résolution du modèle construit, avec des outputs clairs.
6. Nous terminerons notre mémoire par une petite conclusion

En conclusion, ce mémoire va apporter un éclairage approfondi sur LMTO , en mettant en évidence les défis spécifiques de l'industrie pharmaceutique. Il fournira une analyse des processus existants, tout en proposant des solutions pour optimiser ses flux et améliorer la gestion de la chaîne d'approvisionnement.

Chapitre 1

Etats des lieux

1.1 Historique de Novo Nordisk

Novo Nordisk est une entreprise danoise (multinational et leader dans le traitement du Diabète). Elle est connue par ses produits, notamment l'insuline. Elle a été fondée en 1923 à Copenhague, au Danemark, par deux frères Danois, August et Marie Krogh, ainsi que leur collègue, Hans Christian Hagedon. L'entreprise a commencé comme une petite entreprise de production d'insuline, avec comme objectif de fournir des traitements de qualité pour des patients atteints de diabète.

Novo Nordisk se concentre sur la recherche et le développement de nouveaux traitements pour les maladies chroniques, en utilisant notamment la biotechnologie pour développer de nouvelles molécules.

Au fil des années, Novo Nordisk a développé de nouvelles technologies pour la production d'insuline qui est extrait du pancréas des vaches et porcs, y compris la production de l'insuline humaine recombinante à partir des cellules bactériennes génétiquement modifiées. Cette avancée a permis à l'entreprise de produire des traitements plus efficaces et plus sûrs pour les malades.

Dans les années 1970, Novo Nordisk s'est diversifiée en produisant des traitements pour d'autres maladies chroniques telles que l'hémophilie, l'obésité et les troubles de la croissance. La société a également étendu sa présence à l'international, en ouvrant des filiales dans plusieurs pays d'Europe et en Amérique du Nord.

Au cours de la dernière décennie, Novo Nordisk a continué à se développer et à innover, en investissant dans la recherche et le développement de nouveaux traitements pour les maladies chroniques et en adoptant des technologies de pointe pour améliorer la qualité et l'efficacité de ses produits.

Aujourd'hui, Novo Nordisk est l'une des plus grandes entreprises pharmaceutiques du monde, avec des activités dans plus de 170 pays et un portefeuille de produits comprenant des traitements pour le diabète, l'hémophilie, l'obésité et les troubles de croissance.

1.2 Novo Nordisk en Algérie

Novo Nordisk est présente en Algérie depuis très longtemps. En effet, la première cargaison d'insuline a été envoyée en 1936. C'est des années plus tard que le bureau de représentation fut implanté (1992), puis l'entité légale a suivie en 1994 sous le nom d'ALDAPH [1]. Novo Nordisk Algérie compte aujourd'hui plus de 600 employés dont 240 basés à l'usine de production située à Tizi-Ouzou.

1. ALDAPH : Algerian Danish Pharmaceutic

Novo Nordisk Algérie a 2 sites de fabrication, un à Tizi-Ouzou et un autre à Blida, avec une capacité de production de plus de 82 millions d'unités.

1.3 Les missions de Novo nordisk :

— Améliorer la vie des personnes atteintes des maladies chroniques :

Novo Nordisk s'engage à améliorer la vie des personnes atteintes de maladies chroniques, plus précisément le diabète. Les traitements fournis et développés par l'entreprise ont pour but d'aider les patients et les mener à une vie épanouissante.

– Être un leader mondial dans le domaine du diabète : Novo nordisk vise à être reconnu comme le numéro 1 dans ce domaine, tout en fournissant des solutions thérapeutiques de qualité et en investissant dans la recherche et le développement de nouvelles options de traitement (améliorer ses produits pour répondre aux besoins des patients).

— Promouvoir la durabilité :

Novo Nordisk s'est engagée à intégrer la durabilité dans toutes ses activités. Elle met en place des initiatives visant à réduire son impact sur l'environnement, à promouvoir une utilisation responsable des ressources et à contribuer au bien-être des communautés dans lesquelles elle opère. Novo Nordisk cherche à intégrer les principes de durabilité dans sa chaîne d'approvisionnement, sa production et ses pratiques commerciales.

— Favoriser la recherche et l'innovation :

Novo Nordisk investit dans la recherche et l'innovation pour développer de nouveaux traitements et des solutions thérapeutiques avancées.

L'entreprise collabore avec les universités, les institutions de recherche et d'autres partenaires pour repousser les limites des connaissances médicales.

— Assumer une responsabilité sociale et éthique :

Novo Nordisk s'engage à agir de manière éthique et responsable dans toutes ses activités. L'entreprise vise à être un modèle d'intégrité et de transparence, en respectant les normes éthiques les plus élevées et en se conformant aux réglementations en vigueur. Novo Nordisk s'efforce également de favoriser l'accès aux soins de santé dans les régions ou il est le plus nécessaire.

1.4 Structures et Organisation :

— **Structure organisationnelle :** Novo Nordisk a une structure organisationnelle matricielle, ce qui signifie qu'elle combine à la fois une structure fonctionnelle et une structure de divisions.

L'entreprise est divisée en plusieurs divisions et unités fonctionnelles, chacune ayant ses propres responsabilités et domaines d'expertise. Les principales divisions de Novo Nordisk sont :

- Division du diabète : Responsable de la recherche, du développement et de la commercialisation des produits liés au diabète.

- Division de l'obésité et des troubles métaboliques : Axée sur les traitements de l'obésité et d'autres troubles métaboliques.

- Division de l'hémophilie : Dédiée aux traitements de l'hémophilie et des troubles de la coagulation.

- Division des biopharmaceutiques : Impliquée dans la recherche et le développement des médicaments biologiques.

— **Hiérarchie :** Novo Nordisk a une structure hiérarchique traditionnelle avec un conseil d'administration qui supervise les activités de l'entreprise. Le conseil d'administration nomme un président du groupe CEO [2] charge de la gestion quotidienne de l'entreprise. Le CEO est soutenu par une équipe de direction composée de cadres supérieurs responsables des différentes divisions et fonctions clés.

— **Recherche et développement :** Novo Nordisk accorde une grande importance à la recherche et au développement. L'entreprise investit massivement dans la découverte de nouveaux traitements pour le diabète et d'autres maladies métaboliques. Elle dispose de centres de recherche et développement dans plusieurs pays, où des scientifiques et des chercheurs travaillent sur des projets innovants.

— **Fabrication et chaine d'approvisionnement :** Novo Nordisk possède des installations de fabrication dans plusieurs pays, où les médicaments sont produits conformément aux normes de qualité et de sécurité. L'entreprise gère également une chaîne d'approvisionnement mondiale pour distribuer ses produits aux patients du monde entier.

— **Ventes et marketing :** Novo Nordisk dispose d'équipes de vente et de marketing qui assurent la promotion et la commercialisation de ses produits. Ces équipes travaillent en étroite collaboration avec les professionnels de la santé pour informer sur les avantages des traitements de Novo Nordisk et assurer un accès approprié aux patients.

— **Engagement social :** Novo Nordisk est engagée dans des initiatives sociales et responsables, notamment dans le domaine de la sensibilisation au diabète et de l'accès aux soins de santé. L'entreprise soutient des programmes éducatifs, des campagnes de prévention et des initiatives pour améliorer les soins aux patients atteints de diabète.

1.5 Présentation de LMTO(Local Manufactring Tizi-Ouzou)

L'usine Novo Nordisk-LMTO / Aldaph SPA est située à Oued Aissi, Zone industrielle Aissat Idir, Wilaya de Tizi-Ouzou. Unique unité de production de forme sèche d'antidiabétiques oraux du groupe Novo Nordisk dans le monde, l'usine produit de la novoformine pour répondre aux besoins du marché Algérien et à des capacités d'exploration sur d'autres marchés sous forme de trois dosages, à savoir ;

2. CEO : Chief Executive Officer

— Novoformine 500mg :

FIGURE 1.1 – NOVOFORMINE 500mg

— Novoformine 850mg :

FIGURE 1.2 – NOVOFORMINE 850 mg

— Novoformine 1000mg :

FIGURE 1.3 – NOFORMINE 1000mg

– Les principales caractéristiques de LMTO

L'unite de LMTO est caractérisée par :

— Une expérience dans le domaine de la production pharmaceutique malgré sa période d'existence relativement courte (2006-à ce jour).

— Une capacité de production très importante(1 Milliard de comprimés/ans).

— Un savoir-faire en constante évolution dans le domaine de la production, contrôle et analyse, maintenance, support commercial et logistique et résolution des problèmes.

— Une grande capacité de surface de stockage.

- Les objectifs stratégiques de LMTO

— Garantir la disponibilité de ses produits pour ses patients à tout moment et de qualité.

— Viser le marché de l'export et le développer.

— Contribuer significativement à la protection de l'environnement (utilisation de l'énergie solaire, recyclage, ...)

1.6 Les différents départements de LMTO et son organigrame

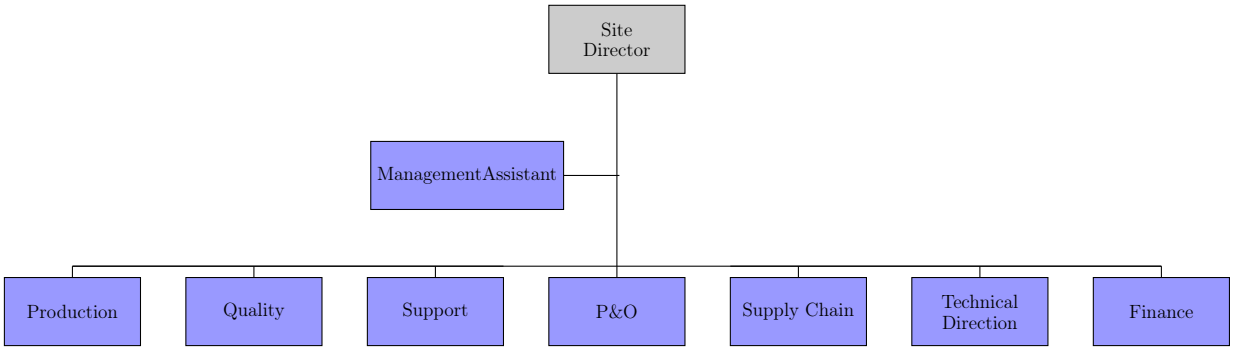


FIGURE 1.4 – Départements du site LMTO

1.7 Présentation du département d'accueil : Supply Chain

Supply Chain :

Le département, Supply Chain, autrement dit, département de chaîne d'approvisionnement, est chargé de la planification, de la coordination et de l'exécution de toutes les activités d'approvisionnement de l'entreprise, de la production à la livraison des produits finis au client.

Ses principales responsabilités comprennent :

- L'analyse des besoins en matières premières et en produits finis.
- La planification de la production.

– La gestion des stocks, la gestion des entrepôts et des transports.

– La gestion des achats, ainsi que la gestion des relations avec les fournisseurs et les clients.

Le département de la chaîne d'approvisionnement travaille en étroite collaboration avec les autres départements pour assurer la qualité, l'efficacité et la rentabilité de l'ensemble du processus d'approvisionnement et de distribution. Il peut rencontrer plusieurs défis spécifiques dans l'industrie pharmaceutique, notamment :

1. Gestion des stocks : En raison de la nature des produits pharmaceutiques, la gestion précise des stocks est cruciale pour éviter les pénuries ou les excédents.

2. Conformité réglementaire : L'industrie pharmaceutique est soumise à des réglementations strictes en matière de fabrication, de transport et de distribution. Le département doit s'assurer de respecter ces réglementations pour garantir la sécurité et la qualité des produits.

3. Gestion de la chaîne du froid : Certains médicaments nécessitent des conditions de stockage spécifiques, notamment des températures contrôlées et humidifiées. La gestion de la chaîne du froid est un défi logistique majeur pour le département.

4. Complexité des commandes : Les commandes de produits pharmaceutiques peuvent être complexes en raison de différentes formulations, dosages et emballages. Le département doit gérer efficacement ces demandes tout en assurant une livraison précise et en temps voulu.

Flux de travail de la Supply Chain :

La Supply Chain assure l'optimisation des mouvements d'entrées et de sorties des informations, de la matière première pour livrer au client son produit fini en temps et en bonne qualité et quantité.Comme le montre cette figure :

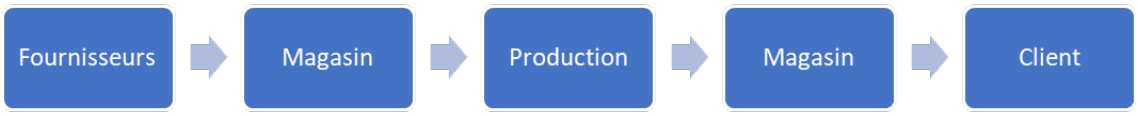


FIGURE 1.5 – flux de LMTO

1.8 Organnigramme du département Supply Chain :

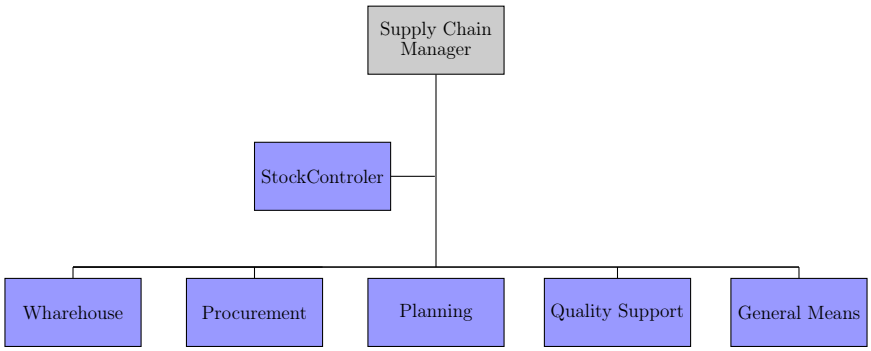


FIGURE 1.6 – Département supply chain

Chapitre 2

Concepts de base

2.1 Introduction :

Dans ce chapitre, nous nous intéressons aux notions fondamentales de la théorie des graphes et ses applications tels que le problème d'ordonnancement et ses méthodes de résolution. Ensuite, nous allons aborder les notions de la programmation linéaire en nombres entiers, et enfin les fondamentales de la gestion des stocks.

2.2 Théorie des graphes :

L'histoire veut que la théorie des graphes ait commencé avec l'article du mathématicien suisse Leonard EULER, sur le problème des ponts de Königsberg, néanmoins il faudra attendre deux siècles pour que le premier livre paraisse sur le sujet. Celui-ci a été écrit par Dénis KONIG en 1936.

Depuis cette époque, la théorie des graphes s'est largement développée. Elle est devenue l'un des domaines les plus féconds et les plus dynamiques des mathématiques et de l'informatique ; cette évolution constante est sans doute due au large spectre des applications telles que : l'électricité, la chimie, la sociologie. . ..

Dans ce chapitre, on va aborder les définitions de bases nécessaires pour faciliter la lecture de ce mémoire.

2.2.1 Quelques notions de bases :

1. **Un Graphe (Graphe orienté) :**
   Un graphe orienté est un couple formé d'un ensemble fini non vide de sommets X et un ensemble d'arcs U ; chaque arc étant associé à un couple de sommets ou une partie d'entre eux selon une direction représentée par une flèche. Le graphe G ainsi obtenu est noté $G = (X, U)$.
   - L'arc u noté $u = (x, y)$ où x est l'extrémité initiale de u notée $I(u) = x$ et y est l'extrémité terminale notée $T(u) = y$
   - T et I sont les applications extrémités initiales et terminale de U dans X.
   - Si un arc u est de la forme (x, x) on parle alors d'une boucle. [5]

Exemple :

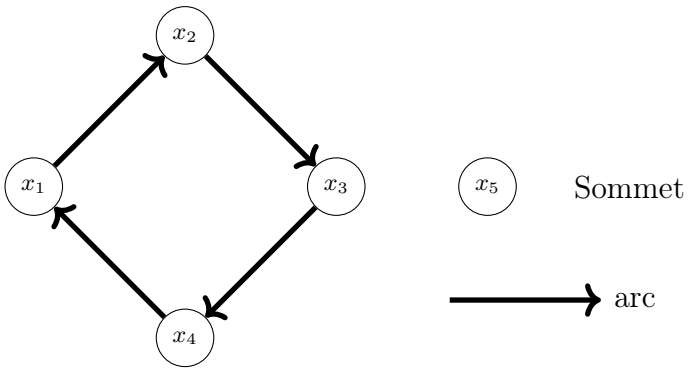
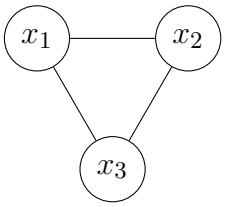


FIGURE 2.1 – Graphe orienté

2. **Un graphe non-orienté :**

   Un graphe non orienté est un schéma constitué par un ensemble X fini non vide de points $(x_1, \ldots, x_n)$ dits sommets et un ensemble E, d'éléments $(e_1, \ldots, e_m)$ dites arêtes reliant les couples de ces sommets. Le couple (X, E) définit le graphe $G = (X, E)$.
   Si G est un graphe alors $|X| = n$ est dit ordre de G et $|E| = m$ le nombre de ses arêtes, sa largeur. [3]
   Exemple :



Ce graphe contient 3 arêtes x_1x_2, x_2x_3, x_3x_1.

FIGURE 2.2 – Graphe non orienté

3. **Adjacence :**

   Deux sommets x et y sont dits adjacents si (x, y) est une arête ou un arc de G. Deux arêtes ou arcs sont dits adjacents s'ils ont au moins une extrémité commune.

4. **Graphe complet :**
   Un graphe $G = (X.U)$ est complet, si ses sommets sont deux à deux adjacents. Un graphe complet d'ordre n est noté K_n.

5. **Les successeurs et les prédécesseurs :**

   Si (x, y), est un arc de G, on dit que y est un successeur de x et x est un prédécesseur de y.

   • L'ensemble des successeurs est donc l'ensemble noté :

$$\overset{+}{\Gamma}(x) = \{y \in X \mid (x, y) \in U\}$$

• L'ensemble des prédécesseurs est noté :

$$\bar{\Gamma}(x) = \{y \in V \mid (y, x) \in U\}$$

Dans un graphe orienté, l'ensemble des voisins du sommet x est égal à la réunion de l'ensemble de ses prédécesseurs et ses successeurs :

$$\Gamma(x) = (\overset{+}{\Gamma}(x) \cup \bar{\Gamma}(x))$$

6. **Arcs incidents à un sommet :**

   Un arc u est incident intérieurement à un sommet $x \in X$ si $I(u) = x$
   Et il est incident extérieurement à un sommet $x \in X$ si $T(u) = x$.

7. **Le voisinage d'un sommet :**

   Un sommet x est dit voisin à un autre sommet y si x est adjacent à y, ainsi :

   L'ensemble des voisins ouverts de x est défini par :

   $$N(x) = \{y \in X \mid (x, y) \in E\}$$

   L'ensemble des voisins fermés de x est défini par :

   $$N[x] = N(x) \cup \{x\}$$

8. Un graphe est simple, s'il est sans boucle et si entre chaque paire de sommet il y a au plus une arête
9. **Le Voisinage :**

   Si $e = (x, y)$ est une arête de G, on dit que y et x sont voisins dans G et qu'ils forment les extrémités de e.
   On définit le voisinage d'un sommet x dans un graphe G comme l'ensemble de ses voisins.

10. **Degré d'un sommet :**

    Si G est simple, on appelle degré d'un sommet x dans G noté, $d_G(x)$ ou simplement $d(x)$ le nombre défini par $d(x) = |N(x)|$, sinon $d(x)$ est le nombre d'arêtes ou arc dont x est une extrémité. Bien entendu, si G est simple et orienté, on a

    $$d(x) = |N(x)| = \left|\Gamma^+(x) \cup \Gamma^-(x)\right| = d^+(x) + d^-(x)$$

    où $d^+(x)$ et $d^-(x)$ représentent respectivement le demi-degré extérieur et le demi-degré intérieur de x.

11. **Chaîne et cycle :**

    Une chaîne est une séquence successive de sommets arêtes de la forme :

$P_n = (v_0, e_1, v_1, e_2, v_2, \ldots, e_{n-1}, v_n)$, v_0 et v_n sont dites les extrémités de la chaîne, la taille ou la longueur d'une chaîne est le nombre de ses arêtes, ainsi P_n ainsi définie est de taille $n-1$.

• Une arête reliant deux sommets non consécutifs d'une chaîne est dite corde.
• Un cycle est une chaîne dont les extrémités sont confondues.

12. **Chemin et circuit :**
Un chemin (respectivement un circuit) est une chaîne (respectivement un cycle) dont les arêtes sont toutes orientées dans le même sens.
• Un C est élémentaire (respectivement simple) s'il n'utilise pas deux fois le même arc ou arête (respectivement sommet). C peut-être une chaîne, un cycle, un chemin ou un circuit.

Exemples :

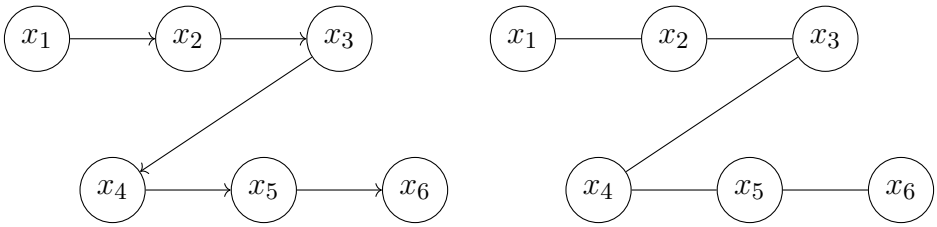


FIGURE 2.3 – Chemin et Chaine

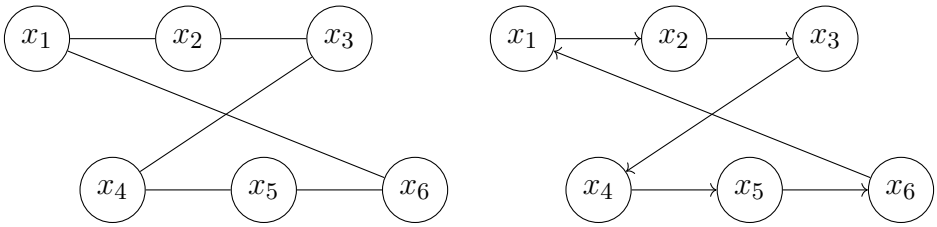


FIGURE 2.4 – Cyrcle et circuit

13. **Distance :**
La distance entre deux sommets x et y notée $d_G(x, y)$ ou simplement $d(x, y)$ est la longueur de la plus courte chaîne reliant x à y dans G. Il est clair que cette distance satisfait aux axiomes topologique de la distance. En effet, on a facilement : $d(x, y) = 0 \implies x = y$, $d(x, y) = d(y, x)$ et $d(x, z) \leq d(x, y) + d(y, z)$. Ainsi, on peut définir les voisins d'un sommet x comme étant la boule ouverte de centre x et de rayon 1 ($N(x) = B(x, 1)$).

14. **Le voisinage d'un ensemble :**
le voisinage d'un ensemble S, noté $N_G(S)$ est l'ensemble de tous les sommets adjacents aux sommets de S , mais qui est n'y sont pas inclus. on le note par : $N_G(S) = N(S) = \bigcap_{v \in S} N(v)$

15. **Excentricité :**
L'excentricité du sommet $v \in V$ noté $e(v)$ est définie comme étant la distance la plus longue qui peut exister du sommet v à n'importe quel autre sommet de G, on la note : $e(v) = \max_{u \in V}\{d(u, v)\}$

16. **Rayon :**
le rayon de G noté $rad(G)$ est la plus petite excentricité en G : $rad(G) = \min_{v \in V}\{e(v)\}$

17. **Diamètre d'un graphe :**
Le diamètre de G noté $diam(G)$ est la plus grande excentricité dans G : $diam(G) = \max_{v \in V}\{e(v)\}$

2.2.2 Représentation matricielle d'un graphe en machine.

(a) **Matrice d'adjacence :**
Soit G un graphe d'ordre n ($X = n$ et $E = m$), on associe à G une $(n \times m)$ matrice M d'éléments

$$a_{ij} = \begin{cases} 1 & \text{si les sommets } x_i \text{ et } x_j \text{ sont adjacents} \\ 0 & \text{sinon} \end{cases}$$

si le graphe est non orienté.
Et et d'éléments :

$$a_{ij} = \begin{cases} 1 & (x_i, x_j) \text{ est un arc} \\ 0 & \text{sinon} \end{cases}$$

si le graphe n'est pas orienté.

Exemples :

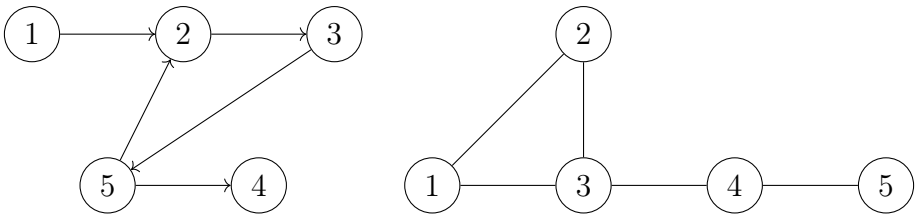


FIGURE 2.5 – Les graphes de G et G' respectivement

La matrice M, ainsi définie est dite matrice d'adjacence associée à G.
Dans les exemples au dessous M et M' sont respectivement les matrices d'adjacence de G et G'.

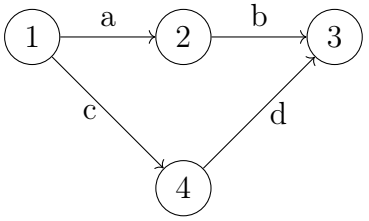
$$M = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad \text{Et} M' = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(b) **Matrice d'incidence :**

On peut aussi représenter un graphe orienté par sa matrice d'incidence sommets arcs d'éléments

$$a_{ij} = f(x) \begin{cases} 1 & \text{si } I(u_j) = i \\ -1 & \text{si } T(u_j) = i \\ 0 & \text{sinon} \end{cases}$$

Exemple :



$$M = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

FIGURE 2.6 – Graphe orienté et sa matrice d'incidence

18. **Sous graphes :**

• Un sous graphe de G induit par une partie $\varnothing \neq A \subset X$ de sommets est le graphe défini par $G = (A, E_A)$ où E_A est l'ensemble des arêtes dont les deux extrémités sont dans A.

• un graphe partiel engendré par une partie $\varnothing \neq E' \subset E$ d'arêtes est le graphe défini par $G = (X, E')$.

• Le sous-graphe partiel engendré par une partie $\emptyset \neq A \subset X$ de sommets et une partie $\emptyset \neq E' \subset E$ d'arêtes est le graphe défini par $G = (A, E'_A)$ où E'_A est l'ensemble des arêtes de E' dont les deux extrémités sont dans A.

• Un sous graphe de G dont les sommets sont deux à deux adjacents est dit une clique de G.

19. **Graphe connexe :**
Un graphe $G = (X, E)$ est connexe si pour chaque couple de sommets il y a au moins une chaîne les reliant.

20. **Graphe fortement connexe :** Il existe une procédure de recherche des CFC, donc de reconnaissance d'un graphe FC qui peut être résumé dans les étapes suivantes :
— Choisir un sommet $x_0 \in X$ et le noter par pm.
- Marquer alors tous les prédécesseurs même les non immédiats de x par - et tous les successeurs même les non immédiats de x par $+$, les sommets marqués à la fois par $+$ et - forment une CFC contenant x_0, notée $\dot{x}_0$. - Poser $X := X - \dot{x}_0$ et refaire le travail tant que $X \neq \phi$.
Une fois la procédure appliquée ; on reconnaît que G est fortement connexe ou non à l'aide de la définition suivante : Un graphe est ainsi fortement connexe si et seulement s'il ne possède qu'une seule CFC.

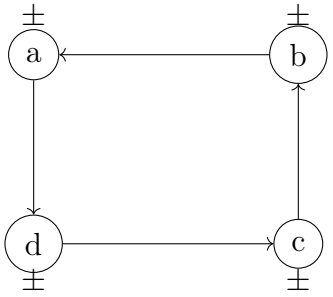


FIGURE 2.7 – Graphe fortement connexe avec DFS et temps de fin.

Exemple : Application de la procédure pour un graphe fortement connexe
L'application de l'algorithme de Kosaraju sur ce graphe confirme qu'il est fortement connexe, avec une seule CFC contenant tous les sommets : $\{a, b, c, d\}$. En effet, en commençant par le sommet a on marque par $+$ les sommets d, c, b, dans cet ordre, car ils sont des successeurs non immédiats de a et on marque par $-$ les sommets b, c, d, dans cet ordre, car ils sont des prédécesseurs non immédiats de a, ainsi, a, b, c, d forment une seule CFC, donc G est fortement connexe.
Si G n'est pas fortement connexe, alors la procédure donne par exemple :
Exemple : Application de la procédure pour un graphe non fortement connexe Dans ce second exemple on voit bien qu'en commençant par

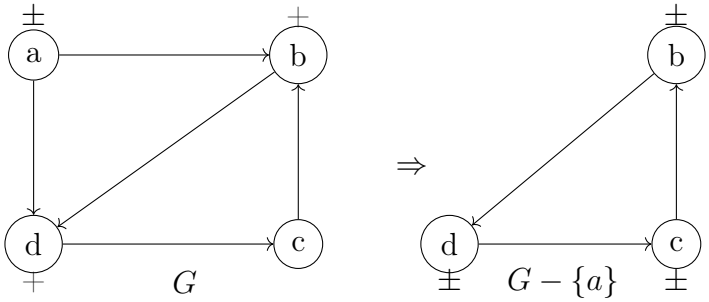


FIGURE 2.8 – Graphe fortement connexe avec DFS et temps de fin.

a, on obtient que a comme CFC, donc on soustrait ce sommet de G. On obtient le sous graphe induit, $G - \{a\}$, pour lequel on applique la procédure en commençant du sommet b et on obtient comme seconde CFC $\{b, c, d\}$.

21. **Arbre et arborescence :**
Un graphe G est un arbre s'il est connexe et sans cycle, autrement dit, il possède exactement $n - 1$ aretes.
Une racine dans un graphe G est un sommet $x \in X$ tel qu'il existe un chemin de x vers $y, \forall y \in X$
Une arborescence est un arbre muni d'une racine.
Il est clair qu'un graphe peut avoir plusieurs racines, mais la racine d'une arborescence est toujours unique. En effet, si une arborescence A possède plus d'une racine alors elle possédera forcement un cycle ou un circuit ce qui est impossible car A est un arbre.

22. **Réseau :**
Si P (respectivement d) est une application de X (respectivement de E ou U) dans R qui à chaque sommet x (respectivement e ou u) associe un poids $p(x)$ (respectivement une distance $d(e)$ ou $d(u)$) fait de G un graphe à sommets pondérés ou à arêtes ou arcs valués.
Une entrée est un sommet de G n'ayant aucun prédécesseur et une sortie est un sommet n'ayant aucun successeur.
Un réseau est un graphe à arêtes ou arcs values ayant une entrée et une sortie, il est noté $R : (X, E(ouU), d)$.
L'optimisation dans les réseaux consiste en la recherche de sous-ensembles de sommets, d'arêtes ou d'arcs, autrement dit un sous graphes, un graphe partiel et/ou un sous graphe partiel.

2.2.3 Problème du plus court chemin :

Étant donné un réseau R on cherche un plus court chemin C d'une racine e vers n'importe autre sommet $y \in X$ (première version), ou bien étant donné deux sommets x et y de X on cherche un plus court chemin de x vers y (deuxième version). La longueur d'un chemin est la somme des arcs le constituant.

Conditions d'existence :
Un circuit C est dit absorbant si sa distance $d(C) < 0$.
Exemple :

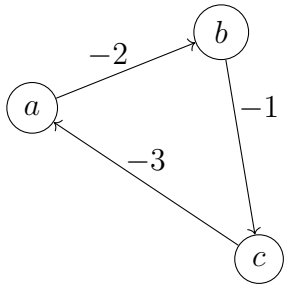


FIGURE 2.9 – Circuit absorbant

Dans cet exemple, la longueur du circuit est -6 qui est la somme des arcs AB, BC et CA.

Si la solution des deux versions précédentes existe, alors forcément G ne possède pas de circuit absorbant, sinon en allant de x vers y il suffit de passer infiniment de fois sur le circuit absorbant et la distance de x à y aller tendre vers moins l'infini et le plus court chemin n'existera jamais.
Ajouter à cette condition nécessaire, la première version admet une solution si e est une racine et la deuxième version exige que G soit fortement connexe.
Il existe plusieurs algorithmes de résolution de ce problème de plus court chemin, tels que l'algorithme de Bellman, Dijkstra, Dansing...etc
Nous nous intéressons, ici, qu'à la première version du problème, et qu'à un seul algorithme de résolution. Ces choix seront justifiés dans la suite.

• Algorithme de Bellman pour la recherche du plus court chemin dans un réseau sans circuit à distances positives.

Cet algorithme est basé sur le principe fondamental de l'optimisation séquentielle ou de la programmation dynamique de Bellman énoncé dans ce qui suit : [4]

Important

Toute sous politique d'une politique optimale est optimale.

Démonstration : : évidente, il suffit de supposer le contraire et d'aboutir à une contradiction, en effet, si P' est une sous politique non optimale de la politique optimale P il suffit alors de remplacer P' par une autre sous politique optimale P'' est en insérant cette dernière dans P, on améliore alors P ce qui est impossible, car P est optimale.

• Algorithme de Bellman avec un exemple.

Énoncé :

– Données : un réseau $R = (X, U, d)$ sans circuit avec $d(u) \in R$.
– Résultat : Arborescence de plus courtes distances A.

(0) Initialisation :
— Soit x un sommet de X, on note par $\pi(x)$ la plus petite distance qui sépare x de e.
— on note par S, l'ensemble des sommets dont π est trouvée.
— on pose $S = \{e\}$ et $\pi(e) = 0$; $A = \emptyset$

(1) Chercher un sommet x hors de S dont tous les prédécesseurs sont dans S.
– Si un tel sommet n'existe pas ;
-Si $S = X$, terminerla solution est trouvée
- Si $S \neq X$, stop pas de solution, le sommet e n'est pas une racine dans R.
– Sinon, si un tel sommet existe ; aller en (2)

(2) On pose : $\pi(x) = Min((I(u)) + d(u), u \in U, T(u) = x$ Où $I(u)$.
On pose :

$$\pi(x) = \min\left(\pi(I(u)) + d(u) \mid u \in U, T(u) = x\right)$$

Soit u' l'arc pour lequel $\pi(x) = \pi(I(u')) + d(u')$

$$A := A \cup \{u'\}; \quad S := S \cup \{x\}; \text{aller à (1)}$$

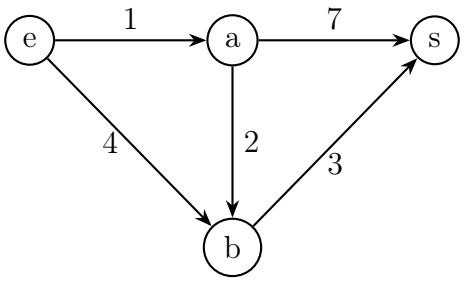


FIGURE 2.10 – La recherche d'un PCC

Exemple
Initialisation :
$\pi(e) = 0, S = \{e\}, A = \{.\}$
On a le seul sommet dont les prédécesseurs sont dans S est a, on calcule alors $\pi(a) = \pi(e) + d(ea) = 1$ donc $A = \{ea\}$. Comme $S \neq X$, donc on continue, maintenant le sommet b est tel que $\Gamma^{-}(b) \subset S$ donc on calcule $\pi(b) = \min\{\pi(e) + d(eb), \pi(a) + d(ab)\} = \min\{3, 4\} = 3 = \pi(a) + d(ab)$ donc $A = \{ea, ab\}$, comme $S \neq X$ alors on continue de cette façon jusqu'à avoir marqué tous les sommets. A la fin on obtient l'arborescence $A = \{ea, ab, bs\}$

2.2.4 Problème ordonnancement :

L'ordonnancement est un ensemble de méthodes et techniques qui ont pour but d'aider à la planification d'un projet pour minimiser sa durée totale. En effet, ordonnancer comme son nom l'indique est le fait de déterminer l'ordre dans lequel devront être exécutées les différentes tâches de manière à optimiser une certaine fonction objective (par exemple rendre la durée d'exécution totale du projet aussi brève que possible) tout en satisfaisant certaines contraintes (précedences de certaines taches par rapport à d'autres ,disponibilité limitée de biens nécessaires à l'exécution du projet..) [4]
- Pour résoudre un problème d'ordonnancement , il ne suffit pas de juste déterminer un calendrier d'exécutions des différentes tâches qui permettent de raccourcir la durée totale de réalisation mais tout d'abord de :

— Déterminer ***les tâches critiques*** c'est à dire celles sur lesquelles il convient de porter l'effort en priorité car un retard dans leur réalisation entraînerait un retard dans l'exécution du projet. [4]
— Contrôler le déroulement de l'exécution du projet et savoir, à chaque instant, mettre à jour les prévisions initiales.
[4]

Parmi les méthodes de résolution les plus connues, on trouve la méthode PERT [1] et CPM [2] . Dans ce qui suit nous nous interessons en particulier à la méthode PERT.

1. PERT :Program Evaluation and Review Technique
2. CPM :Critical Path Method

La méthode PERT (Program Evaluation and Review Technique) :

La méthode PERT est une méthode de résolution des problèmes d'ordonnancement qui vise à minimiser la durée totale d'un projet par une analyse détaillée des tâches ou activités élémentaires et de leur enchaînement. On étudie les délais sans prendre en compte les charges. Cette méthode s'appuie en grande partie sur une représentation graphique nommée « réseau PERT » noté $R = (X, U, d)$.
[4]

Dans cette représentation :

- Les tâches seront des arcs t_{ij} du graphe, telle que i est l'évènement début et j et fin de la tâche t_{ij}. [4]
- La longueur de chaque arc sera la durée d_{ij} de la tâche correspondante notée juste en dessous de la tache. [4]
- Le début et la fin d'une tâche seront des dits évènements. [4]
- L'entrée de R doivent être une racine c'est à dire $a \in X$ tel que il existe un chemin de a vers x, $\forall x \in X \setminus \{a\}$. [4]
- Chaque étape x sera définie par un ensemble de tâches ayant déjà été effectuées. Notons qu'on ne peut pas commencer l'exécution d'une tâche (réelle ou fictive) avant que l'étape x ne soit atteinte. Cette dernière est atteinte lorsque toutes les tâches correspondantes à des arcs dont x est l'extrémité terminale sont achevées. [4]
- Le réseau ainsi définit doit être sans circuit. Sinon, les tâches du circuit (les arcs) chacune attendez l'autre pour démarrer et le projet sera bloqué. [4]

Quelques notions de base dans un réseau PERT :

On adopte les notations suivantes :

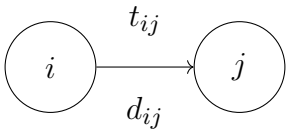


FIGURE 2.11 – Présentation de deux évènements et une tache

Tâche fictive : est une tâche de durée nulle qu'on ajoute pour faire un réseau et respecter les précédances.
Il est à signaler que ces tâches fictives sont ajoutées essentiellement dans deux cas :
[3]

- Si certaines tâches ne sont pas reliées par un chemin à la fin du projet, on ajoute alors des tâches fictives reliant la fin de ces tâches à la fin du projet. [3]
- Si une ou plusieurs tâches sont précédées par des tâches n'ayant pas les mêmes évènements fin de tâches du fait qu'elles précèdent des tâches différentes, voir exemple FIGURE 2.12 page 28, A précède B et E précède G mais E ne précède pas B, ainsi la tâche fictive X. [3]

Date au plutôt et au plut tard :

Pour chaque tâche t_{ij}, on définira la date au plus tôt d_{t_i} de réalisation comme étant la longueur du plus long chemin entre le début du projet et l'événement début de cette tâche. [3]

On définira aussi la date au plus tard d_{T_i} associée à chaque tache t_{ij} comme étant la longueur du chemin critique moins la longueur d'un plus long chemin du début de la tache à la fin du projet.
[3]

La marge d'une tâche : le responsable du projet peut se pencher aussi sur le degré de liberté dont il dispose pour éventuellement augmenter la durée d'une tâche sans compromettre la durée totale du projet. Elle est calculée par :

$$M(t_{ij}) = d_{Tjk} - d_{ij}d_{tij}1001[3]$$

Exemple :

Soit P un projet dont les taches, leurs précédances et durées sont dans le tableau suivant :

Tache	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Précédances	-	A	B,J	-	D	E,H,I	E,A	G	G	H,I	H,I,J	K	L,F	-	N
Durée	1/2	3	1/2	1	1	1/2	2	2	3	1	1/2	1	4	2	1

Pour respecter les précedénces des tâches, on a ajouté 3 tâches fictives X, Y, Z de durées nulles.

Premièrement, on doit dessiner le graphe du projet (qui est différent du réseau PERT) pour déterminer le plus long chemin c'est à dire le chemin critique pour concevoir le réseau PERT.

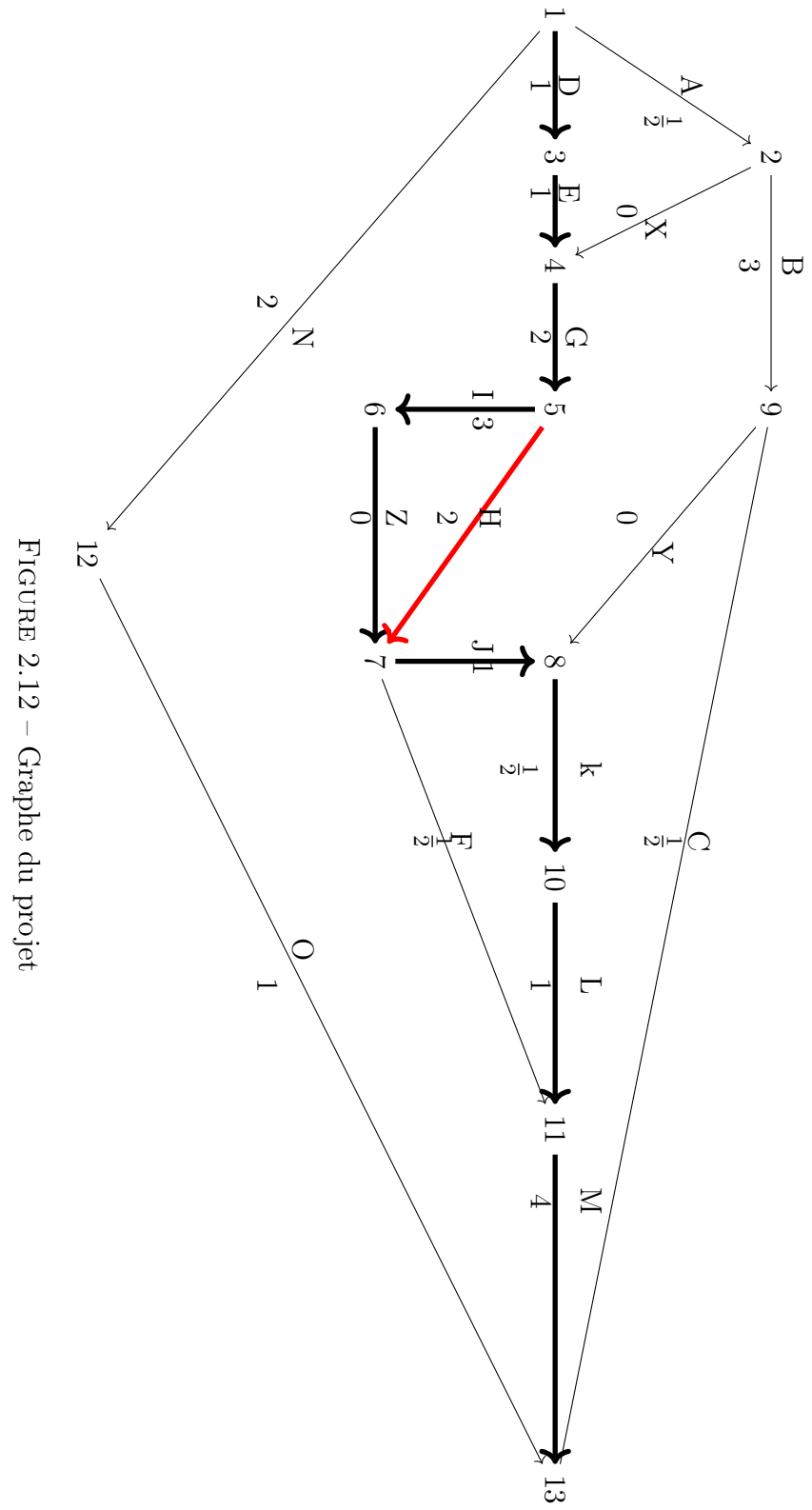


FIGURE 2.12 – Graphe du projet

1. Calculant le chemin critique de ce projet :
   On va appliquer l'algorithme de Bellman au lieu de d'appliquer avec le min on va l'appliquer avec le max, donc :
   Le chemin critique est constitué des tâches D, E, G, I, J, K, L et M,(le chemin en gras) de longueur 13,5 calculé par la somme des durée des tâches donc $C = 1 + 1 + 2 + 3 + 0 + 1 + 1/2 + 1 + 4 = 13,5$. La tâche H est aussi une tâche critique.
2. Calculant les dates au plutôt et au plus tard de toutes les tâches pour construire un réseau PERT :

Tache	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Date au plus tard	0	0,5	8	0	1	7	2	4	4	7	8	8,5	9,5	0	2
Date au plus tôt	0	2	13	0	1	7	2	4	4	7	8	8,5	9,5	0	12,5

Pour représenter le projet ordonnancé dans le réseau PERT ; on adopte la représentation, de la méthode PERT, suivante :

— Chaque tache est un arc, son nom est porté au dessus de l'arc et sa durée au dessous, FIGURE 2.11.
— Chaque évènement est représenté par un cercle dont est porté son nom, la date au plus tard et la date au plus tôt de la tâche ayant comme début cet évènement, comme représenté dans la FIGURE 2.13.

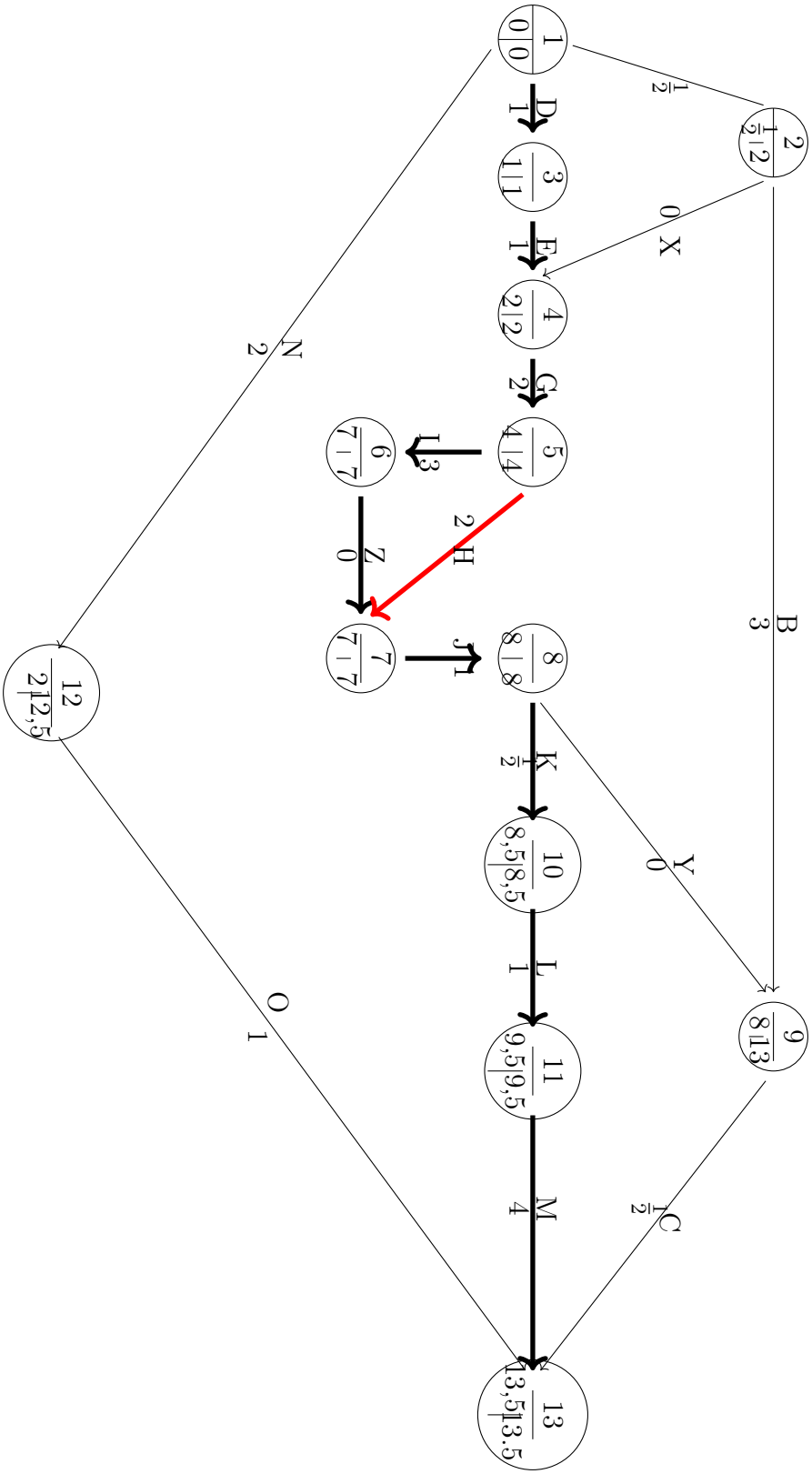


FIGURE 2.13 – Réseau PERT

2.3 Programmation linéaire en nombres entiers :

2.3.1 Définitions :

La Programmation Linéaire en Nombres Entiers (PLNE) n'est pas uniquement un outil de modélisation, mais également un ensemble de méthodes de résolution pour divers problèmes d'optimisation. Elle a pour objet l'étude et la résolution des problèmes d'optimisation dans les quels la fonction objective aussi bien que les contraintes sont exprimées de manière linéaire, elle cherche a déterminer la meilleur solution possible sous certaines contraintes. En particulier, elle évoque des solutions où un grand nombre de ressources telles que la main d'œuvre, les machines de divers manière disponibles doivent être combinées afin d'obtenir des produits de toute nature.

Un problème d'optimisation en nombre entier est un problème où tout les variables sont contraintes à ne prendre que des valeurs entière, ce programme linéaire en nombre entier est écrit de la manière suivante :

$$\begin{cases} \min Z = cx \\ \quad Ax \geq b, \quad \text{(PLNE)} \\ x \geq 0, x \in \mathbb{N}^n \end{cases}$$

Si on examine un programme linéaire en nombre entier on constate qu'il est composé :

* D'une fonction objective qu'on cherche à minimiser
* D'un système d'équations linéaire $Ax \geq$b
* D'un système de contrainte d'appartenance à N sur les variables $x \in \mathbb{N}$

Résoudre ce programme revient à déterminer les valeurs des variables entières qui minimisent la fonction Z.

Dans le cas particulier où les contraintes $x \in \mathbb{N}$sont remplacées par $x \in \{0, 1\}$, on dit qu'on a un «Programme linéaire binaire».

On donnera plus loin un exemple d'un PLNE binaire, qui le poblème du sac à doc.

2.3.2 Méthodes de résolutions (PLNE) :

Un grand nombre de méthodes de résolutions existe en recherche opérationnelle pour l'optimisation. La nature des variables, domaines de définitions et des critères à optimiser influent le choix de la méthode d'optimisation à utiliser. Un certain nombre de problèmes faciles ont été identifiés et étudiés (problème du plus court chemin). S'agissant des problèmes difficiles, il existe deux grandes familles : les méthodes exactes et les méthodes approchées.

Méthodes exactes :

Les méthodes exactes sont entre autres des algorithmes de résolutions des problèmes, qui assurent l'identification de la solution optimale, si elle existe. Cependant, leur utilisation est souvent difficile quand elles sont appliquées à des problèmes complexes nécessitant des ressources considérables, en termes de temps de calcul et de traitement ce qui les rendent quasi inapplicables.

Ces méthodes comme : "Branch and Bound", "Branch and Cut", "Branch and Price", "Branch, Cut and Price". D'autres méthodes sont moins générales, comme : "La programmation dynamique", "simplexe", "La programmation linéaire en nombre entiers"...etc, donnent en général de très bon résultats pour de petites instances mais s'arrêtent dès qu'elles sont des instances considérables.

Ainsi, lorsque les instances deviennent trop grandes, ces méthodes exactes deviennent inefficaces, d'où la nécessité d'appliquer des méthodes approchées.

Méthodes approchées :

Méthodes souvent inspirées de mécanismes d'optimisation rencontrés dans la nature. Elles sont utilisées pour les problèmes où on ne connaît pas d'algorithmes de résolution en temps polynomial et pour lesquels on espère trouver une solution approchée de l'optimum global.

Typiquement ce type de méthodes, dites heuristiques permettent d'obtenir des solutions "suffisamment bonne" en un temps de calcul raisonnable, elles sont particulièrement utiles pour résoudre des problèmes difficiles sur des instances numériques de grande taille lorsque, là où les méthodes exactes, connues, échouent. Elles peuvent aussi être utilisées afin d'initialiser une méthode exacte (Branch and Bound).

Elles sont fondées principalement sur des heuristiques, qui peuvent être spécifiques à un type de problème. Parmi ces heuristiques, on trouve les méta heuristiques qui fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes. Cette dernière catégorie est subdivisée en deux grandes familles :

les méthodes de recherche locale à solution unique telle que : la méthode de descente,le Recuit Simulé et la recherche Tabou.

les méthodes évolutionnaires : à base de population de solutions comme : les algorithmes génétiques.

Pour plus de détails sur ce sujet, le lecteur est invité à lire le chapitre 3.

2.4 Gestion des stocks

La gestion des stocks c'est l'ensemble des méthodes scientifiques principalement mathématiques de gestion d'un stock d'une entreprise, magasin, ... dont le but ultime est de répondre à la problématique fondamentale qui est Quand ? et Combien commander ? c-à-d elle consiste principalement à déterminer à quel moment et en quelle quantité un article devra être renouvelé, que ce soit pour les matières premières, les pièces de rechange des équipements de production ou pour les produits finis. [6]

2.5 Définitions :

1. **Le stocks :**
   Un stock est une quantité de biens ou d'articles mis en réserve pour une utilisation future. Le stock est utilisé soit pour faciliter ou pour assurer la continuité de l'activité pour satisfaire une demande intérieure formulée par l'un des services de l'entreprise. [6]
2. **Les différents type des Stocks :**
   Le stock des matières premières : Il représente les produits achetés par l'entreprise, par l'intermédiaire de ses fournisseurs, et destinés à la production. [6]
   Le stock de produits semis-finis : Il s'agit de produits non finis qui ne peuvent être mis en vente qu'après avoir subi une certaine transformation au niveau de la production. [6]

Le stock de produits finis : Ce sont les articles qui ont subi une transformation afin d'atteindre leur niveau final de fabrication. Ils sont désormais prêts à la vente. [6]

Stock à commandes régulières : c'est les articles régulièrement consommés. Ils sont commandés par le gestionnaire des stocks chaque fois que le stock minimum est atteint. [6]

Stock de projet : il prend naissance au début d'un projet et disparaît à la fin de ce dernier, Les commandes de tels articles sont faites directement par le chef de projet qui seul maîtrise ses besoins. [6]

3. **Lead time :**

   Lead time ,appelé aussi le délai de livraison, renvoie au temps d'écoulement dont un produit prend depuis l'entrée de la matière première jusqu'à l'expédition du client. Il s'agit du temps de traversée d'un produit tout au long de la chaîne de valeur complète de l'usine, et ce, de la préparation des commandes jusqu'aux livraisons.

   Toute entreprise a comme enjeu d'avoir un Lead time très court afin de rester réactive et être compétitive, donc pour le limiter il est essentiel de définir les principales phases de la Supply Chain comme : la fabrication, le transport, le temps de traitement en entrepôt, et leurs Leads Times afin de déterminer les éléments à améliorer. Notons que l'entrepôt joue un rôle important dans l'organisation de la Supply Chain. Maillon indispensable entre l'offre et la demande, il est donc essentiel que vous choisissiez une gestion efficace pour réduire votre Lead Time.

   Et de plus en plus de responsables logistiques optent pour des relations sur le long terme quand ils doivent choisir un fournisseur. Cela leur permet de standardiser leurs processus, de mieux calculer leur Lead Time et d'utiliser les méthodes du juste-à-temps (le Lean Manufacturing).

   Pour le calculer on utilise la formule suivante :

   $$\text{Lead time} = \text{Date de fin du processus} - \text{Date du début de processus}$$

4. **Les niveaux des stocks :**

   Stock minimum : C'est le niveau de stock pour couvrir la consommation de matière pendant le délai de livraison, il permet de poursuivre une activité normale pendant le délai de réapprovisionnement (entre la date de commande et la date de la livraison), comme il permettra de ne pas descendre en dessous ce niveau et ainsi éviter la rupture de stock. [10]

   $$\text{Stock minimum} = \text{Consommation moyenne quotidienne} \times \text{Nombre de Jours}$$

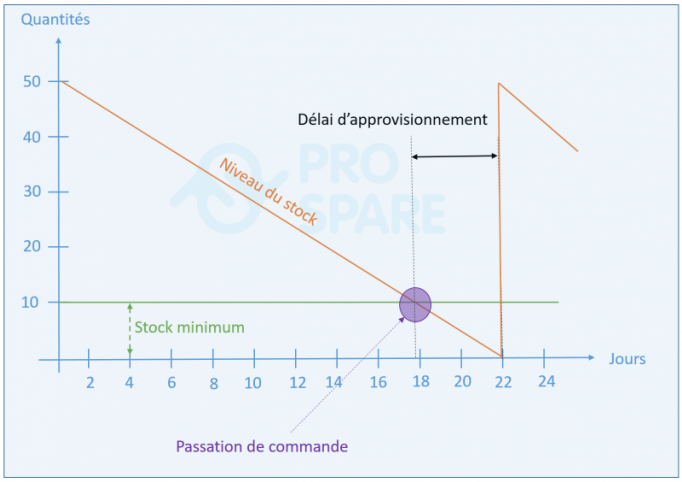


FIGURE 2.14 – Représentation graphique du stock minimum

Stock de sécurité : C'est une quantité d'un article qui, en plus du stock minimum, est gardée dans le magasin afin de pallier les ruptures de stock. C'est un stock « dormant » qui doit être reconstitué dès lors qu'il est entamé afin qu'il puisse jouer son rôle. [10]

Stock de sécurité = (Délai maximal garanti - Délai de livraison habituel) x Demande habituelle du produit

[10]

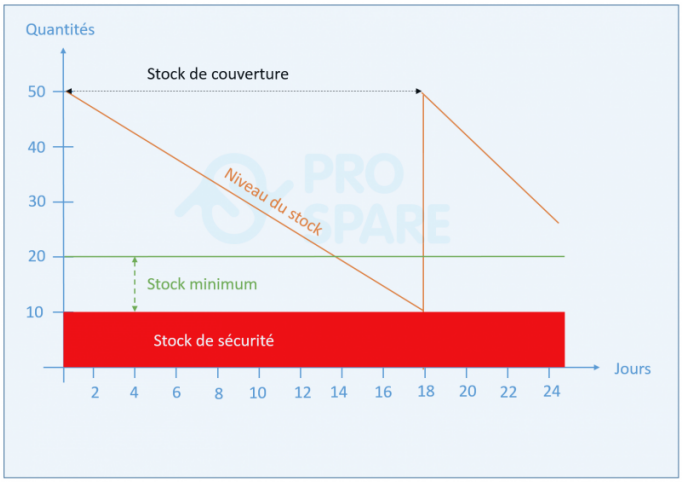


FIGURE 2.15 – Représentation graphique du stock de sécurité

Stock maximum : Il est fixé par les gestionnaires de l'entreprise et représente une limite supérieure des stocks que l'entreprise ne peut dépasser. Il ne faut pas conclure de ce qui précède que le stock détenu par l'entreprise doit pour des raisons de sécurité, être le plus élevé possible, le stock maximum est un stock suffisamment bas pour éviter le sur stockage et le gaspillage. [10]

Stock maximum = Consommation moyenne quotidienne × Délai d'Approvisionnement +Stock de Sécurité

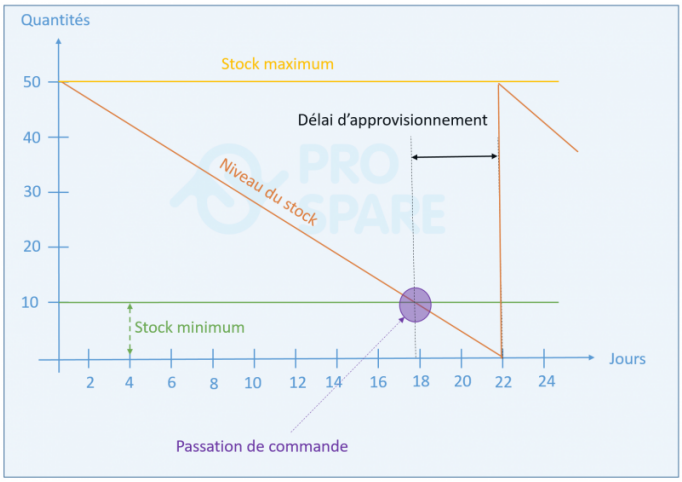


FIGURE 2.16 – Représentation graphique du stock maximum

Stock d'alerte : C'est le niveau de stock prédéfini par le gestionnaire, supérieur au stock de sécurité qui déclenche le réapprovisionnement. [10]

Stock d'alerte = Stock minimum + Stock de sécurité

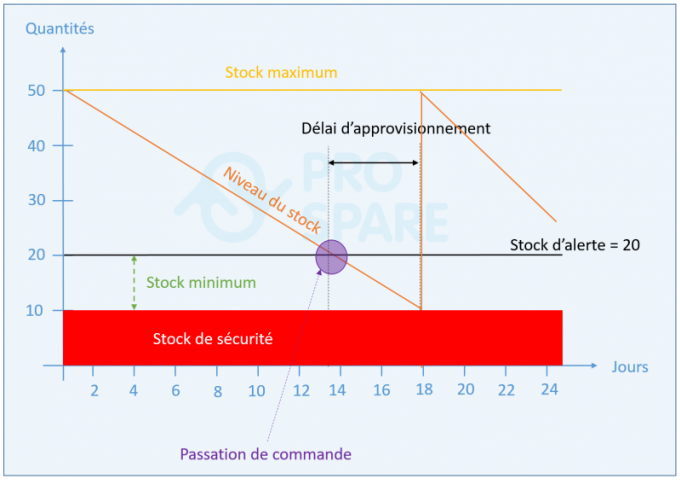


FIGURE 2.17 – Représentation graphique du stock d'alerte

5. **Un coût :**
   Un coût représente le montant à payer dû à la présence d'unité en stock. On a 5 types :
   Les coût de commande :
   Les coûts de commandes coûts fixes ou le sont des coûts engendrés par le processus de commande.En effet, ils incluent les frais du processus de la commande elle même et ceux de la logistique amont. Ils sont indépendants du nombre d'unités commandées. Pour chaque commande on le note par (K),c'est-à-dire $C_f = K \times$ nombre de commande. [6]
   Les coûts variable :
   Les coûts variables sont proportionnels à la quantité commandée. c.-à-d. si on prend (C) est le coût unitaire de revient et si on commande (Q) unités alors les coûts variables sont $C_v = c * Q$. [6]
   Les coûts de stockage :
   Les coûts de stockage sont des coûts qu'on doit payer pour l'entretien du stock comme la location, l'électricité,... on les note C_s. Et (h) est le coût unitaire de stockage par unité d'article et unité de temps. [6]
   Les coûts de pénurie :
   Les coûts de pénurie sont des coûts qui sont dû à chaque fois qu'une demande n'est pas satisfaite. On note (P) le coût unitaire de pénurie par unité d'article et unité de temps. [6]
   Les coûts totaux :
   Les coûts totaux sont la somme de tous les coûts cités précédemment. [6]
6. **Modèles de gestions des stocks :**
   Selon la demande connue ou non, on distingue deux types de modèles de gestion des stocks. [6]
   (a) Modèles déterministe :
   Un modèle déterministe est un système de gestion dans lequel les éléments sont supposés non soumis au hasard (la demande et le délai d'approvisionnement sont connus à l'avance). [6]
   (b) Le modèle du lot économique de Wilson :
   appelé également modèle de la quantité économique de la commande (EOQ : Economic Order Quantity). C'est l'un des premiers modèles développés pour la gestion des stocks, mais qui reste toujours très utilisé et ce pour sa simplicité et la stabilité des solutions fournies par cette méthode. Dans ce modèle on suppose que : [6]
   - La demande est connue et uniforme au court du temps, et on ne suppose que μ unités sont retirées du stock chaque unités de temps ; [6]
   - Les livraisons ou les productions sont immédiates (L=0) ; [6]
   - Les pénurie ne sont pas autorisés ; [6]
   - Pour chaque unité en stock et chaque unité de temps un coût $h \geq 0$ doit être payer. [6]
   - Un coût $K \geq 0$ doit être payer chaque fois qu'un ordre est émis ainsi que un coût variable $C_v(Q) = c * Q$ proportionnel au nombre d'unités Q commandés et c un coût unitaire [6] ;

   Comme le montre le diagramme précédent, l'évolution des stocks dans le lot économique est périodique. En effet, au début de chaque période, un réapprovisionnement de Q unités est effectué pour satisfaire la demande jusqu'au début de la prochaine période où une nouvelle livraison va être reçue, et ainsi de suite.

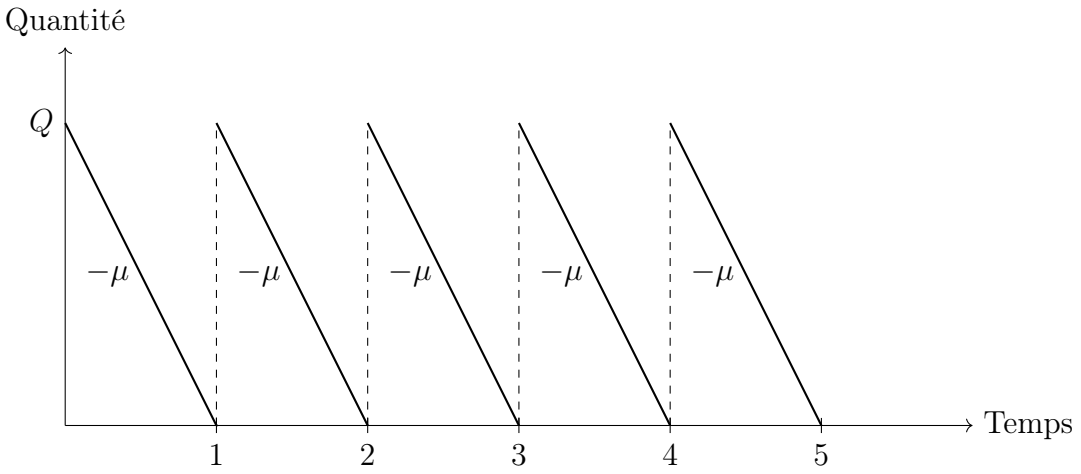


Diagramme en dents de scie de l'évolution des stocks dans le modèle du lot économique

Dans ce modèle la politique optimale est cyclique et consiste à la commande de $Q^* = \sqrt{\frac{2\times\mu\times K}{h}}$ unités d'articles chaque $T^* = \sqrt{\frac{2\times K}{h\times\mu}}$ de temps tel que h est le coût de stockage et Q^* et T^* sont dites les formules de Wilson. [6]
Pour rendre ce modèle plus pratique c'est à dire plus proche de la réalité, on distingue 2 cas :

Cas 1 : Si L$\leq T$:
Dans ce cas le point de commande est le moment où le niveau des stocks atteint le stock d'alerte, l'ordre de commande doit être émis immédiatement car pendant L unités de temps on consomme exactement ce qui est équivalent au stock d'alerte.
[6]
Cas 2 : Si L>T :
Dans ce cas, le nombre d'unité nécessaires pour couvrir le délai de livraison est supérieure à Q^*, donc le point de commande est égal au reste de la division entière de $\frac{\mu\times L}{Q^*}$ [6]

2.6 Conclusion :

Dans ce chapitre, on a introduit la théorie des graphes et ses notions de base dont on a besoin. En suite on a aborder le problème d'ordonnancement avec une méthode de résolution et les notions fondamentales de la gestion des stocks. Ce derniers est dédié à être utiliser dans l'outil développer qu'on va exposer dans le chapitre 5.
En fin,nous avons présenté la programmation linéaire en nombre entier utilisés pour la modélisation de notre problème.

Chapitre 3

Complexité algorithmique

Introduction

Dans ce chapitre on va traiter des concepts principaux de la complexité algorithmique en précisant sur les notions fondamentales des classes P (polynomial) et NP (non déterministe polynomial) puis on va présenter quelques définitions.

3.1 La théorie de la Complexité

La théorie de la complexité est un domaine des mathématiques et de l'informatique qui étudie la difficulté des problèmes afin de les classer par des classes de complexité. [7]
Cette classification se base sur la quantité des ressources (temps espace mémoire) qu'un algorithme a besoin pour résoudre un problème algorithmique. [7]
En sachant la complexité d'un algorithme, cela aide le décideur à choisir l'algorithme le plus efficace pour la résolution d'un problème donné. On va essayer de donner quelques définitions afin d'assurer une meilleure compréhension de la complexité algorithmique. [7]

3.2 Concepts de base

3.2.1 Définitions :

• Un problème est une situation ou un sujet qui requière une solution. [7]
• Un problème mathématique consiste à rechercher une entité mathématique donnée qui puisse satisfaire les conditions du problème. [7]
• Une instance est une application numérique du problème, ou la forme sous laquelle on peut présenter les données de notre problème,dans la complexité algorithmique on prend l'instance dans le pire des cas (les données sont très grandes) pour évaluer l'efficacité d'un algorithme. [7]
• La taille d'une instance représente la quantité de données nécessaires pour la décrire. [7]

Exemple : on peut citer comme exemple le problème du sac à dos.

Étant donné plusieurs objets possédant chacun un poids et une valeur et étant donné un poids maximum qui est la capacité du sac, la question est quels objets faut-il mettre dans le sac de manière à maximiser la valeur totale sans dépasser le poids maximal autorisé pour le sac. La formulation de ce problème commence par l'énoncé des données. Nous avons le sac à dos de poids maximum P, l'instance de ce problème est les n objets. Pour chaque objet i, nous avons un poids p_i et une valeur v_i. On définit la variable x_i associé à un objet i de la façon suivante : $x_i = 1$ si l'objet i est mis dans le sac, et $x_i = 0$ si l'objet i n'est pas mis. La solution de ce problème doit vérifier la contrainte suivante : $\sum_{i=1}^{i=n} p_i x_i \leq P$ tout en maximisant la fonction objectif $\max \sum_{i=1}^{i=n} x_i v_i$.

L'instance de ce problème est le nombre n d'objets et la capacité P du sac. Ainsi, si n et P sont relativement grands la résolution exacte de ce PLNE devient quasi-impossible, vu le nombre important de cas à explorer par les méthodes exactes.

• Un algorithme est une méthode de résolution générale d'un problème, qui s'exécute étape par étape, il prend en entrée une instance du problème et retourne en sortie la solution, qui est la réponse à la question du problème appliquée à l'instance. [7]

• Problème d'optimisation : il consiste à trouver la meilleure solution parmi toutes les solutions possibles. [7]

•**La complexité :** on distingue deux types de complexité, complexité temporelle et complexité spatiale. [7]

(a) La complexité temporelle est la complexité de calcul qui décrit le temps nécessaire à l'exécution d'un algorithme. Elle est généralement estimée en comptant le nombre d'opérations élémentaires (opérations arithmétiques, comparaison, affectation....) effectuées par l'algorithme. [7]

(b) La complexité spatiale est une mesure de l'espace utilisé par un algorithme en fonction de propriétés de ses entrées. L'espace compte le nombre maximum de cases mémoire utilisées simultanément pendant un calcul. [7]

•**Notation grand** O

Soient f et g deux fonctions définies de $\mathbb{N} \to \mathbb{R}^+$. On dit que $f = O(g)$ (qui se prononce f grand O de g) s'il existe une constante k telle que :

$$\forall n \geq M, f(n) \leq kg(n).$$

Exemple : Soit $f(n) = 10n$ et $g(n) = n^2$. Choisissons $M = 10$, alors pour tout $n \geq 10$, $f(n) = 10 \times n \leq n^2 = g(n)$. En prenant la constante $k = 1$, on constate que la relation $f = O(g)$ est vérifiée (c'est-à-dire qu'à un facteur constant près, f ne croit pas plus rapidement que g).

3.2.2 Différents classes des problèmes :

Classe P :

Un problème $\mathcal{P}$ de décision est dans la classe P, s'il existe une constante c tel que $\mathcal{P}$ peut être résolu en temps polynomial ; en temps n^c. Cette classe correspond à l'ensemble des problèmes faciles. [7]

Classe NP :

Un problème de décision est de la classe NP si on peut vérifier sa solution en un temps polynomial, bien que trouver une solution peut s'avérer très difficile. [7]

3.3 Conclusion :

Dans ce chapitre, on a exploré les fondamentaux de la complexité algorithmique, et ses différents concepts de base pour la compréhension et l'évaluation des performance des algorithmes.

Chapitre 4

Modélisation du problème d'optimisation des flux de production, Méthodes de résolutions

4.1 Introduction :

Dans le domaine de l'industrie pharmaceutique, où la qualité, la sécurité et l'efficacité des produits sont primordiales, la gestion des flux de production revêt une importance capitale. En effet, l'optimisation de ces flux constitue un défi majeur pour les entreprises du secteur, confrontées à des contraintes telles que les exigences réglementaires strictes, les variations de la demande du marché et les impératifs de rentabilité.

Ce processus de modélisation implique l'analyse et la représentation des différents composants du système de production pharmaceutique, tels que les lignes de production, les équipements, les matières premières, les produits intermédiaires et finis, ainsi que les flux logistiques associés. Il s'agit également de prendre en compte les contraintes spécifiques à l'industrie pharmaceutique, telles que BPF(les bonnes pratiques de fabrication), les normes de qualité, les exigences de traçabilité et les impératifs de conformité réglementaire.

Ce qui nous mène à modéliser le problème sous forme d'un modèle mathématique qui vise à maximiser l'utilisation des ressources disponibles, à minimiser les coûts de production, et aussi permettent de prendre des décisions éclairées concernant la planification de la production, la distribution des produits et l'optimisation des schémas d'ordonnancement.

En somme, la modélisation du problème d'optimisation des flux dans une industrie pharmaceutique représente un domaine de recherche et de pratique essentiel pour améliorer l'efficacité opérationnelle, réduire les coûts et garantir la qualité des produits pharmaceutiques. En développant des modèles et des algorithmes adaptés aux besoins spécifiques du secteur, les entreprises peuvent renforcer leur compétitivité et leur capacité à répondre aux exigences croissantes du marché pharmaceutique et surtout de maximiser leur productivité. Alors :

Quelle est la problématique à laquelle nous devons faire face ?

4.2 Position du problème :

LMTO se fixe un objectif de satisfaire leurs demandes mensuelles, tout en respectant les normes et bonnes pratiques de fabrication. Pour s'y faire , les planificateurs au niveau du département de la Supply Chain mettent en évidence un plan(suivi) de production généré qui calcule les durées de fabrication pour chaque produit, les quantités journalières a produire, sur quelle machine produire. Aussi, ils prennent en compte la gestion de leurs stocks c'est a dire Quand ? et Combien commander ? Tout ça dans le but d'optimiser les flux de production et assurer l'efficacité et la rentabilité.

Cela soulève des questions liées à l'efficacité de ce processus :

Est-ce-que les délais de fabrication sont toujours respectés ? En cas de retard qu'entraînent ces retards ? Et comment peut-on optimiser ces flux de production pour mieux respecter les délais ?

Pour répondre a ces questions , on doit mener une analyse approfondie de ces différents processus de production.

Cette analyse consiste d'abord à bien définir les différents produits, ateliers et processus de la production. Une fois le problème est repéré, on passe à la description de la procédure à suivre pour optimiser les flux, en vue d'obtenir une solution améliorée et efficace.

Commençons d'abord par définir ces différents produits, ateliers, et processus qui constituent les flux de production de LMTO.

4.3 Les différents produits, ateliers et processus :

4.3.1 Type de produits utilisés :

Matières premières :

Les matières premières sont des éléments fondamentaux qui marquent le début du processus de fabrication, que ce soit dans l'industrie pharmaceutique ou dans d'autres secteurs industriels. Elles sont habituellement transformées ou associées à d'autres matériaux tout au long du processus de production afin de générer des produits finis. Sur le site LMTO, les matières premières telles que le Granulat, la Paraffine et l'Opadry sont utilisées pour démarrer le processus de fabrication.

Articles de conditionnement :

Les articles de conditionnement constituent des éléments cruciaux de l'emballage des produits, étant spécifiquement conçus pour garantir la protection du produit tout au long de son stockage, de son transport et de sa distribution, tout en préservant son intégrité et sa sécurité. Ces articles sont généralement regroupés en deux principales catégories :

— Articles de conditionnement primaire :

Les articles de conditionnement sont les composants de l'emballage qui entrent directement en contact avec le produit. Ils sont souvent conçus pour protéger le produit contre les dommages physiques, chimiques ou biologiques, tout en assurant son intégrité et sa sécurité.

Au niveau du site LMTO on trouve comme articles de conditionnements primaire : Aluminium et PVC.

— Articles de conditionnement extérieur : ils constituent la couche externe de l'emballage servent souvent d'étiquettes ou de moyen de communication pour attirer l'attention du consommateur et fournir des informations sur le produit.

Au niveau du site LMTO on trouve comme article de conditionnement extérieur ; Notice et étuis.

4.3.2 Les processus et ateliers de production :

Les processus de production dans l'industrie pharmaceutique englobent un ensemble d'étapes et d'opérations essentielles visant à transformer les matières premières en produits finis de haute qualité, répondant aux normes de sécurité et d'efficacité les plus strictes. Ces processus comprennent généralement des étapes telles que la formulation, le mélange, la compression, le pelliculage, l'emballage et le contrôle qualité. Chaque étape est soigneusement planifiée et exécutée pour garantir la fiabilité, la stabilité et l'efficacité des médicaments produits,afin de garantir la sécurité des patients et la conformité aux exigences réglementaires.

Au sein du site LMTO, trois processus fondamentaux sont engagés pour garantir une production efficace et de haute qualité des produits pharmaceutiques. Ces étapes impliquent la compression, le pelliculage et le conditionnement des médicaments. Chacun de ces processus joue un rôle déterminant dans la création de produits sûrs et efficaces pour les patients, et nous allons maintenant détailler chaque étape, en commençant par la compression.

(a) La compression :

La compression, également connue sous le nom de tableting, est une étape cruciale du processus de fabrication des comprimés pharmaceutiques. Elle implique la compression de la matière première, généralement sous forme de granulés, pour former des comprimés solides. Les machines utilisées pour cette opération, telles que les presses à comprimé A04 et A03, sont essentielles pour garantir la précision et la qualité des comprimés produits.

Dans le tableau ci-dessous , on trouve les informations associes a ce processus :

Machines	Dosages	Capacités
A04	500 mg	36 750 boites par jour
A04	850 mg	37 625 boites par jour
A04	1000 mg	22 500 boites par jour
A03	850 mg	37 625 boites par jour

Après la finalisation du processus de compression des comprimés, des échantillons de chaque dosage sont prélevés pour des tests visant à garantir que les produits non enrobés sont conformes aux normes établies. Une fois ces étapes terminées, on passe au processus suivant, à savoir le pelliculage.

(b) Le pelliculage :

Le pelliculage, également connu sous le nom Coating (enrobage), implique l'application d'une couche externe de paraffine et d'Opadry sur les comprimés à l'aide des enrobeuses A02 et A05, toutes deux qualifiées pour les trois dosages respectifs. Cette étape vise à masquer le goût ou l'odeur du médicament et à faciliter la déglutition en rendant les comprimés plus lisses.

Dans le tableau ci-dessous , on trouve les informations associes a ce processus :

Machines	Dosages	Capacités
A02	500 mg	126 000 boites par jour
A02	850 mg	48 000 boites par jour
A02	1000 mg	42 000 boites par jour
A05	500 mg	42 000 boites par jour
A05	850 mg	45 000 boites par jour
A05	1000 mg	21 000 boites par jour

(c) L'emballage :

Dans ce processus, les articles de conditionnement primaire en aluminium et en PVC sont utilisés pour protéger les médicaments contre l'humidité, la lumière et l'oxygène, assurant ainsi leur stabilité. Le PVC, en particulier, offre une flexibilité permettant de former différents formats pour répondre aux besoins spécifiques des médicaments, tandis que l'aluminium est utilisé pour imprimer des informations essentielles telles que le nom et la posologie du médicament.

Les articles de conditionnement extérieur, tels que les étuis et les notices, fournissent des informations cruciales pour garantir une utilisation sûre et efficace du médicament. Ils facilitent la communication entre les fabricants de médicaments, les professionnels de la santé et les patients, jouant ainsi un rôle essentiel dans la sécurité et la compréhension des traitements médicaux.

Dans le tableau ci-dessous , on trouve les informations associes a ce processus :

Machines	Dosages	Capacités
A21	500 mg	30 528 boites par jour
A21	850 mg	31 528 boites par jour
A21	1000 mg	25 440 boites par jour
A22	500 mg	40 704 boites par jour
A22	850 mg	48 844 boites par jour

4.4 Contraintes d'approvisionnement et demande du marché :

4.4.1 Contraintes d'approvisionnement :

Toute industrie pharmaceutique se base sur l'approvisionnement qui est une opération complexe en raison des exigences strictes en matière de qualité, de réglementation et de sécurité.

(a) Réglementaires et légales :

Conformité aux réglementations locales et internationales :

Les entreprises doivent se conformer aux normes et réglementations telles que les Bonnes Pratiques de Fabrication, les Bonnes Pratique de Distribution,...

Traçabilité et sécurité des médicaments :

Obligation de garantir la traçabilité des matières premières et des produits finis tout au long de la chaine d'approvisionnement.

(b) Qualité :

Certification des fournisseurs :

Les fournisseurs de matières premières doivent être certifié et approuvés, ce qui inclut des audits réguliers pour s'assurer de la qualité et de la conformité.

Contrôle de qualité :

Les matières premières doivent subir des tests rigoureux pour garantir leur conformité aux spécifications.

(c) Sécurité et stabilité :

Stabilité des produits :

Les matières premières et produits finis nécessitent des conditions spécifiques de stockage et de transport (température contrôlée, humidité, etc.)

(d) Logistiques et transport :

Transports spécialisés :

Besoin de moyens de transport adaptés pour les produits sensibles (chaine du froid, transport sécurisé pour les substances contrôlées)

Délais de livraison :

Importance de délais de livraison pour éviter les ruptures de stock, ce qui peut impacter la production et la disponibilité des médicaments.

(e) Gestion des stocks :

Stock de sécurité :

Maintien d'un stock de sécurité pour pallier les imprévus (retards, ruptures chez les fournisseurs).

Péremption :

Gestion des ateliers de péremption des matières et des produis finis.

Ces contraintes nécessitent une coordination étroite être les différents de l'entreprise, ainsi qu'une collaboration efficace avec les fournisseurs, les transporteurs et les régulateurs.

4.4.2 Contraintes de Demande du Marché :

Comme son nom l'indique c'est la satisfaction de la demande du marché dans un délai prédéterminé.

• Après cette analyse , effectivement un problème est identifié .Il consiste en la non-optimalité des choix de machines sur lesquels doit-on produire et l'enchaînement de chacun des produits sur ces machines, ce qui entraîne un retard dans la finalisation du projet, et qui décale ainsi la date de livraison prévue.

Pour répondre a cette problématique , une procédure a suivre est établie comme suit :

- Formuler notre problématique, en la modélisant mathématiquement, en prenenant en compte de toutes les contraintes dont on se dispose.Ensuite, estimer sa complexité .

- Proposer une méthode de résolution, en expliquant l'algorithme à suivre.

4.5 Modélisation du problème :

Pour une période T donnée de la chaine de production, on veut minimiser le temps des différents produits effective sur les différentes machines.
Ainsi, si à chaque période ce temps est minimisé, en sommant ces durées gagnées pendant chaque périodes on aurait définit de nouvelles périodes de production.

4.5.1 Variables de Décision

Soit x_{ij} variable binaire indiquant si la tâche i est exécutée sur la machine j.

$$x_{ij} = \begin{cases} 1 & \text{si la tâche i est effectuée sur la machine j} \\ 0 & \text{sinon} \end{cases}$$

4.5.2 Objectif

Minimiser la durée de production totale :
$Z = \min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot t_{ij}$

où t_{ij} est le temps de production du produit i sur la machine j.
Tel que :
n est le nombre de tâches et m est le nombre de machines

4.5.3 Contraintes

Contrainte de capacité :

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in \{1, 2, \ldots, n\}$$

Ce qui signifie que chaque tâche ne peut être traitée que sur une seule machine pendant une période donnée t_{ij}.

Contraintes spécifiques :

$$\sum_{i=1}^{n} x_{ij} \leq 1 \quad \forall j \in \{1, 2, \ldots, m\}$$

Ce qui signifie que chaque machine ne peut traiter qu'une seule tâche à la fois pendant une periode donnee $t$ij et qu'il y peut avoir des arrêts-machines planifiés et non planifiés .

$x_{ij} \in \{0, 1\}$ pour toutes les tâches i et les machine j.
Le Modèle mathématique associé pendant une période donnée :

$$\begin{cases} Z = \min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot t_{ij} \\ \sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in \{1, 2, \ldots, n\} \\ \sum_{i=1}^{n} x_{ij} \leq 1 \quad \forall j \in \{1, 2, \ldots, m\} \\ x_{ij} \in \{0, 1\} \end{cases}$$

4.5.4 La taille du modèle : :

Pour évaluer la taille de ce modèle, nous devons considérer deux aspects principaux : la taille de la fonction objectif et la taille des contraintes.

La taille de la fonction objectif :

La fonction objectif comporte une double somme sur n et m, ce qui donne une taille de $O(nm)$.

Taille des contraintes :

- La première contrainte $\sum_{i=1}^{m} x_{ij} = 1$ est évaluée pour chaque $i \in \{1, 2, \ldots, n\}$, donc elle a une complexité de $O(n)$.
- La deuxième contrainte $\sum_{i=1}^{n} x_{ij} \leq 1$ est évaluée pour chaque $j \in \{1, 2, \ldots, m\}$, donc elle a également une taille de $O(m)$.

En général, la taille du modèle est donnée par la taille de la fonction objectif car elle comporte une double somme.
Ainsi, la taille du modèle est de $O(n \times m)$.

Alors l'évaluation de ce problème nous montre que c'est un problème de programmation linéaire en nombre entier (PLNE). Car, un PLNE est une classe de problème d'optimisation où la fonction objectif ainsi que toutes les contraintes sont linéaire, mais certaines ou toutes les variables doivent prendre des valeurs entières.
Dans notre cas particulier, la fonction objectif ainsi que toutes les contraintes sont linéaires, mais les variables x_{ij} doivent prendre des valeurs binaires (0 ou1), ce qui correspond à un PLNE binaire.
La résolution de ce problème nécessitent l'utilisation d'algorithmes spécialisés pour les PLNE, tels que la programmation dynamiques, la programmation en nombre entier mixte (PNE), ou des techniques de branch and bound.

4.6 Méthode de résolution :

L'algorithme de branch and bound

(a) Présentation :

L'algorithme de Branch and Bound noté (B&B), est un ensemble de techniques fondamentales en optimisation combinatoire pour résoudre des problèmes d'optimisation en examinant systématiquement l'ensemble des solutions possibles de manière à identifier la meilleure solution ou à déterminer si une solution optimale existe.

L'idée centrale derrière les algorithmes de B&B [1] est de diviser l'espace de recherche en une hiérarchie arborescente, appelée arbre de recherche. Cette approche permet d'identifier la meilleure solution ou de déterminer si une solution optimale existe, en minimisant le nombre de solution explorée.

Définissons alors le processus de Branch and Bound qui se déroule comme suit :

(b) Fonctionnement des algorithmes de Branch and Bound :

— Initialisation :

–Commencer par initialiser une solution partielle (souvent vide ou une solution triviale) et définissez une borne supérieure initiale (meilleure solution actuelle) à une valeur très grande ou infinie.

–Créer une liste (arbre de recherche) des solutions partielles à explorer.

— Branche :

–A chaque itération, sélectionner une solution partielle de la liste.

–Générer des extension de cette solution en ajoutant une nouvelle décision à chaque fois, ce qui crée de nouvelles solutions partielles.

–Les solutions partielles générées doivent être valides selon les contraintes du problème.

— Évaluation et élagage (Bounding) :

–Évaluer chaque solution partielle générée pour estimer si elle pourrait mener à une solution meilleure que la meilleure solution actuelle. Si ce n'est pas le cas, éliminer cette solution (élagage).

–Si une solution partielle est complète (tous les éléments sont inclus ou toutes les décisions sont prises), comparer la à la meilleure solution actuelle et mettre à jour la meilleure solution si nécessaire.

— Bound :

–Utiliser les informations sur les solutions partielles évaluées pour déterminer une borne inférieure pour les solutions restantes à explorer.Cette borne est généralement basée sur les valeurs des solutions dèjà évaluées.

— Arrêt :

–L'algorithme s'arrête lorsque toutes les solutions partielles ont été évaluées, ou lorsque la borne inférieure pour les solutions restantes dépasse la valeur de la meilleure solution actuelle.

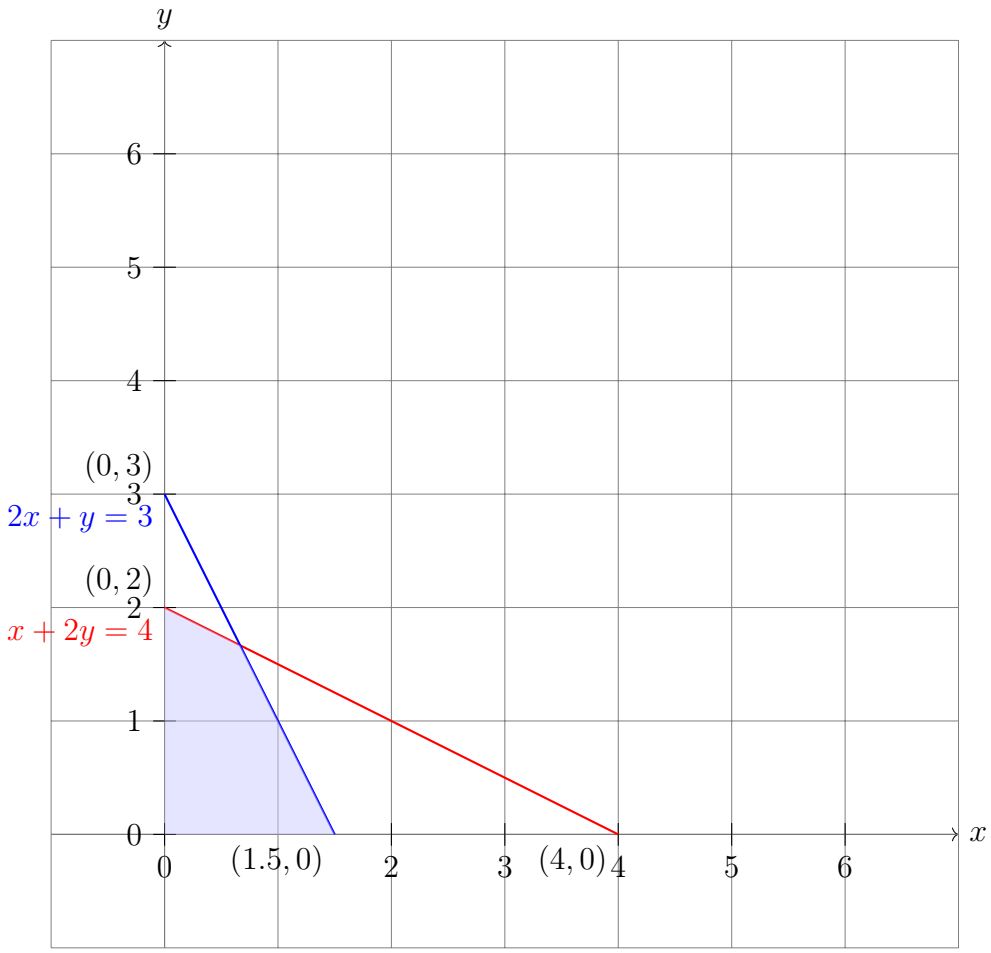
— Optimisation :

–Pour améliorer l'efficacité, utiliser des techniques telles que l'élagage basé sur les bornes supérieures (pruning), l'optimisation de l'ordre de sélection des solutions partielles, ou encore la parallélisation de certaines étapes de l'algorithme.

1. B&B :Branch and Bound

Exemple :

$$\begin{cases} Z = \max 3x + 2y \\ x + 2y \leq 4 \\ 2x + y \leq 3 \\ \text{x,y}\in\text{N}. \end{cases}$$



Nous allons appliquer l'algorithme de branch and bound pour trouver la solution optimale.

(1)Initialisation : Nous commençons par résoudre le problème relaxé en ignorant les contraintes d'entier. La solution optimale du problème relaxé est (x^*, y^*)

Branche 1 : $x \leq 0$ sa solution optimale est (x, y)=$(0, 2)$ avec $Z = 4$ (infeasible).

Branche 2 : $x \geq 1$ sa solution optimale est (x, y)=$(1, 3/2)$ avec $Z = 6$.

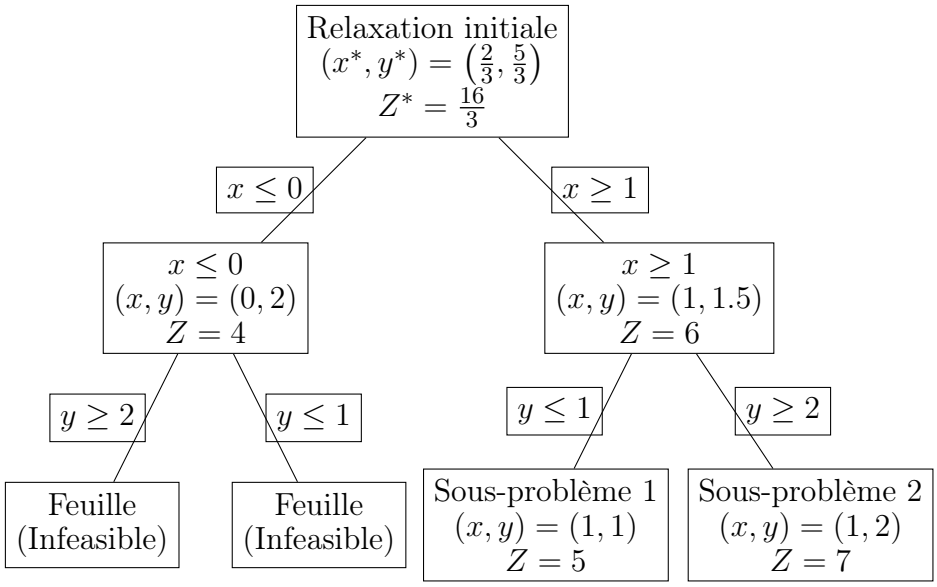
(a) Solution fractionnaire $(x, y) = (1, 1.5)$, $Z = 6$

(b) Sous problème (1) : $y \leq 1$: Solution entière obtenue : $(x, y) = (1, 1)$, $Z = 5$.

(c) Sous problème (2) : $y \geq 2$: Solution fractionnaire :$(x, y) = (1, 2)$, $Z = 7$.

Conclusion :

La solution entière optimale obtenue $(x, y) = (1, 1)$ avec $Z = 5$.



4.7 Conclusion :

Nous nous sommes concentrés dans ce chapitre sur la problématique au seine du site LMTO, tout en explorant les différents processus de production et leurs interactions sur les ateliers (machines).Et comment respecter les contraintes d'approvisionnement et demande du marché. Ensuite on entamer l'étape de modéliser et formuler le problème, et faire une étude sur la taille du modèle à l'aide de la taille de la fonction objectif et les contraintes associées à ce modèle, ce qui nous a permit de savoir notre espace de travaille est de l'ordre de $O(n \times m)$.

Après cette étape vient l'étape d'évaluation du modèle mathématique, dont on constate que c'est PLNEB [2], et la méthode la plus efficace pour résoudre ce type de problème c'est Branch and Bound (B&B).

2. PLNEB :Un problème de programmation linéaire en nombre entier binaire

Chapitre 5

Simulations , résultats et implémentation

5.1 Introduction

Dans un monde où la productivité est devenue une priorité absolue pour les entreprises, l'automatisation des tâches répétitives s'impose comme un levier de gain de productivité important. En effet, les tâches répétitives, fastidieuses et peu valorisantes peuvent représenter un frein à l'efficacité des équipes et à la rentabilité des entreprises. C'est pourquoi, il est de plus en plus crucial de comprendre comment cette automatisation peut aider à améliorer la productivité au travail. [18]
Dans ce chapitre, nous explorerons en profondeur le développement d'outil d'automatisation. Nous examinerons comment cette transition vers l'automatisation a été rendue possible et comment elle façonne notre façon de travailler et de vivre au quotidien

5.2 Méthodologie

5.2.1 Processus de Développement :

Le développement de notre application a suivi une approche méthodique et itérative. Voici un aperçu du processus que nous avons entrepris :

Le langage de programmation utilisé :

Parmi tous les langages de programmation disponibles, nous avons choisi d'opter pour Python en raison de ses nombreux avantages.
Python a été créé en 1989 par Guido van Rossum. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991.Il s'inspire de plusieurs langages : ABC, Modula, C, Smalltalk, langages de script. [18]
Ce langage de programmation présente de nombreuses caractéristiques intéressantes :

(a) Il est orienté objet. C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel avec un certain nombre de règles de fonctionnement et d'interactions. [18]
(b) Il dispose de bibliothèques standard et est riche de modules et frameworks, rendant le développement plus rapide et plus efficient. [18]

(c) C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++. [18]
(d) Python est connu pour sa rapidité de développement, ce qui signifie qu'il est possible de créer des applications rapidement et efficacement. [18]
(e) Enfin, il est très utilisé dans divers domaines tels que le développement web, l'analyse de données, l'intelligence artificielle et bien plus encore. [18]

En conclusion, en optant pour Python, nous investissons dans un langage qui non seulement répond à nos exigences actuelles, mais qui est aussi prêt à évoluer avec nous dans l'avenir technologique.
Dans la section suivante, nous allons détailler les différentes bibliothèques que nous avons utilisées et expliquer comment nous les avons intégrées pour construire notre application Python.

Outils et Technologies :

Librairies utilisées :
En langage Python, une librairie c'est un ensemble de fonctions, de classes d'objets et de constantes qui permettent de travailler sur un thème particulier. Il existe de très nombreuses bibliothèques Python qui jouent un role tres important . [**?**]
Avec l'import d'une bibliothèque spécifique par exemple, un développeur ou un analyste de données n'a plus besoin de coder les fonctions dont ils ont besoin de bout en bout. Bien au contraire, il leur suffit de faire appel à la fonction voulue. Parce qu'ils n'ont pas besoin de tout réinventer, il s'en suit un gain de temps notable. [**?**]
Avant de commencer par utiliser une bibliothèque Python, il faut procéder à son installation. Elle peut être faite avec pip qui est un utilitaire fourni avec Python et qui aide à installer les bibliothèques déposées sur Pypi. La ligne de code est celle qu'il faut taper dans la fenêtre pour l'installer.

```
pip install <nomdelabibliotheque>
```

Les bibliothèques Python qui suivent sont très pratiques pour la conception de notre application :
Numpy :
Son nom signifie Numerical Python. C'est une librairie incontournable en Python. Sa spécificité ? Elle permet de faire du calcul numérique et permet la gestion des tableaux de données et matrices. Elle possède la plupart des fonctions usuelles telles que l'exponentielle, le logarithme ou encore arctan. De plus, elle est optimisée pour les calculs et va permettre de paralléliser les opérations, c'est-à-dire utiliser tous les processeurs de l'ordinateur pour aller plus vite dans les calculs. Pour des formats de type données, comme un CSV par exemple, numpy ne va pas permettre d'étiqueter les données. Grâce à cette librairie, vous pouvez intégrer directement du code en C, C++ ou Fortran.
Pandas :
La librairie Pandas est basée sur Numpy et permet de manipuler facilement des données structurées :

(a) Créer des nouvelles colonnes.
(b) Gérer les données manquantes.
(c) Filtrer des données.
(d) Agréger des informations selon des colonnes.

(e) Calculer des métriques telles que la moyenne, la médiane ou des sommes. [13]

Elle est basée sur 2 types d'objets : les séries, très similaires aux listes en termes de fonctionnement et les data frames qui sont donc des tableaux à plusieurs colonnes. On peut aussi inclure un type d'objet appelé Panels qui permet de manipuler des objets en 3 ou 4 dimensions.

De plus, elle facilite la lecture de données provenant de différentes sources : CSV, SQL ou encore texte. Bref, c'est l'outil incontournable pour manipuler des données sur Python. Cette librairie permet aussi d'avoir une meilleure vue d'ensemble sur les données.

Openpyxl :

`openpyxl` est une bibliothèque Python open source qui permet de travailler avec des fichiers Excel au format `xlsx`. Elle offre des fonctionnalités pour créer, modifier, manipuler et analyser des fichiers Excel en utilisant des scripts Python. `openpyxl` est une bibliothèque puissante pour travailler avec des fichiers Excel en utilisant Python. Elle est largement utilisée dans le domaine de l'automatisation des tâches, de la manipulation de données et de la génération de rapports automatisés. Voici une description plus détaillée de cette bibliothèque :

- **Lecture et écriture de fichiers Excel** : charger un fichier Excel existant, lire les données, créer de nouveaux fichiers Excel, etc. [14]
- **Travail avec les feuilles de calcul** : renommer, supprimer, créer de nouvelles feuilles, etc. [14]
- **Manipulation des cellules** : définir des valeurs, des formats, des styles, des couleurs de fond des cellules, etc. [14]
- **Itération sur les rangées et colonnes** : `openpyxl` propose des méthodes pour itérer sur les rangées et les colonnes de feuilles de calcul. [14]
- **Formules** : ajouter et gérer des formules dans les cellules. [14]
- **Styles et formats** : appliquer des styles et des formats aux cellules, modification de la police, de la couleur, du bord, etc. [14]
- **Gestion des images et graphiques** : ajouter des images et des graphiques dans les fichiers Excel, ainsi que les personnaliser. [14]
- **Protection des feuilles et cellules** : ajouter des mots de passe pour protéger les feuilles ou certaines cellules, restreignant ainsi l'accès. [14]
- **Manipulation avancée** : en plus des fonctionnalités de base, `openpyxl` permet également des opérations plus avancées telles que la gestion des filtres, la création de tableaux, la gestion des commentaires, etc. [14]

Datatime :

Le module datetime fournit des classes pour manipuler de façon simple ou plus complexe des dates et des heures. Bien que les calculs de date et d'heure sont gérés, l'implémentation est essentiellement tournée vers l'efficacité pour extraire des attributs pour les manipuler et les formater pour l'affichage. Pour d'autres fonctionnalités associées, voir aussi les modules time et calendar. [15]

Interface graphiques pour utilisateur :

Les interfaces graphiques servent à rendre les programmes plus conviviaux. Elles sont pratiques à utiliser mais elles demandent un peu de temps pour les concevoir. Un programme sans interface exécute des instructions les unes à la suite des autres, le programme a un début - un point d'entrée - et une fin. Avec une interface, le programme fonctionne de manière différente. Il n'exécute plus successivement les instructions mais attend un événement - pression d'une touche du clavier, clic de souris - pour exécuter une fonction. C'est comme si le programme avait une multitude de points d'entrée.

Il existe plusieurs modules permettant d'exploiter les interfaces graphiques. Le plus simple est le module tkinter présent lors de l'installation du langage Python.

Tkinter :

`Tkinter` (Tk interface) est un module intégré à la bibliothèque standard de Python, permettant de créer des interfaces graphiques :

— des fenêtres,
— des widgets (boutons, zones de texte, cases à cocher, etc.),
— des évènements (clavier, souris, etc.).

`Tkinter` est disponible sur Windows et la plupart des systèmes Unix, ce qui rend les interfaces créées avec `Tkinter` portables. [16]

Les fonctionnalités de `Tkinter` sont disponibles dans le module `tkinter`.

La conception de l'interface :

La conception d'une interface graphique se déroule généralement selon deux étapes. La première consiste à dessiner l'interface, c'est-à-dire choisir une position pour les objets de la fenêtre (boutons, zone de saisie, liste déroulante, ...). La seconde étape définit le fonctionnement de la fenêtre, c'est-à-dire associer à chaque objet des fonctions qui seront exécutées si un tel événement se réalise (pression d'un bouton, pression d'une touche, ...). [17]

Les objets :

Les interfaces graphiques sont composées d"objets ou widgets ou contrôles. Comme ils reçoivent des événements, en un sens, ce sont ces objets qui pilotent un programme ou qui le contrôlent.

Zone de texte :

Une zone de texte sert à insérer dans une fenêtre graphique une légende indiquant ce qu'il faut insérer dans une zone de saisie voisine comme le montre la figure qui suit. Une zone de texte correspond à la classe Label.

Bouton :

Un bouton a pour but de faire le lien entre une fonction et un clic de souris. Un bouton correspond à la classe Button.

Zone de saisie :

Une zone de saisie a pour but de recevoir une information entrée par l'utilisateur. Une zone de saisie correspond à la classe Entry .

Zone de saisie à plusieurs lignes :

Une zone de saisie à plusieurs lignes est identique à la précédente à ceci près qu'elle autorise la saisie d'un texte sur plusieurs lignes. Cette zone correspond à la classe Text.

Case à cocher :

Une case à cocher correspond à la classe Checkbutton. Pour créer une case à cocher avec `Tkinter`, vous pouvez utiliser le widget `Checkbutton`.

Liste :

Un objet liste contient une liste d'intitulés qu'il est possible de sélectionner. Une liste correspond à la classe ListBox.

7. **Sources de Données :**
   Détailler les données d'entrée utilisées pour le développement et les tests de l'outil.

5.2.2 Mise en Œuvre

Dans cette section, nous allons présenter l'interface utilisateur conçue pour être accessible aux différents utilisateurs, accompagnée d'un guide d'utilisation.Par la suite, nous allons vous présenter les principaux algorithmes qui réalisent les différentes opérations automatisées effectuées par cet outil Python. Enfin, nous allons examiner les résultats obtenus et les comparer à ceux que nous obtiendrons habituellement manuellement.

Partie I : Ordonnancement :

(a) **Présentation de l'interface Tkinter :**
Les champs vides seront remplis avec les demandes mensuelles de chaque dosage que nous avons.

Monthly Production Plan | Shift Pattern | Performance per Machine | Blacklist

Product	January	February	March	April	May	June	July	August	September	October	November	December
NF500mg												
NF850mg												
NF1000mg												

FIGURE 5.1 – Demandes mensuelles data entry

Une fois la tâche terminée, nous passons au deuxième onglet.
Dans ce deuxieme onglet, l'utilisateur devrait sélectionner(cocher sur) le patern de shift c'est à dire le mode de travail adapté à cette machine.
Quand l'utilisateur fait son choix du mode de travail dans l'onglet précédent, la durée machine s'affiche automatiquement dans cet onglet . Ce qui reste est de remplir les autres champs . Ici , il y'a une particularité qui est des qu'on remplit les 3 premiers champs , le calcul du rendement se fait automatiquement.

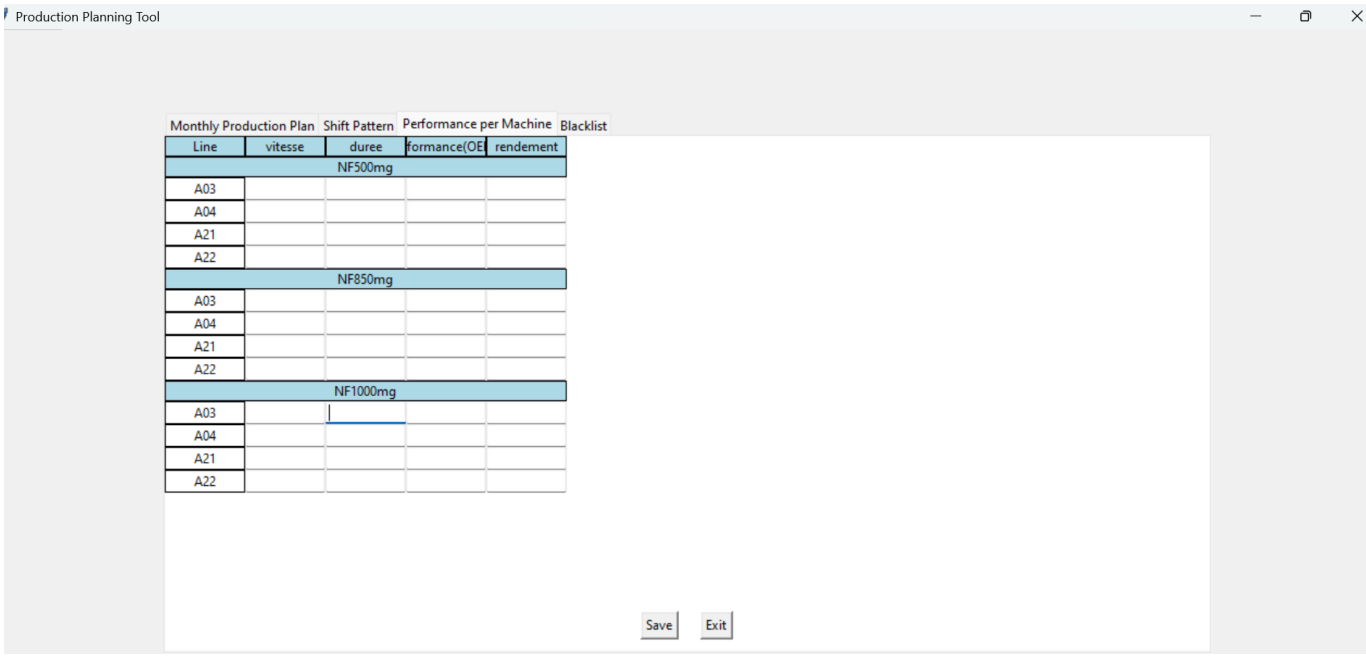


FIGURE 5.2 – Perfromance data entry

Dans le dernier onglet, nous avons trois listes à compléter. La première liste concerne les jours fériés de toute l'année. La deuxième option est réservée aux weekends (si l'on ne travaille que 5 jours par semaine). La dernière correspond aux vacances.

(b) **Procédures et instructions :**

Le nombre de jours de production requis pour chaque processus varie pour le Bulk (compression, enrobage) et le conditionnement en fonction des demandes mensuelles. Ce calcul doit prendre en compte l'efficacité des machines dans chaque machine de production pour chaque produit, ainsi que les heures de travail quotidiennes, les week-ends, les jours fériés et les vacances.

Dans un premier temps, les dates de début et de fin de conditionnement sont établies pour chaque produit. Nous délibérons ensuite sur quelle ligne les produire, en considérant en priorité le NF1000mg sur la ligne 1 car comme on a dit précédemment que le dosage NF1000mg se produit seulement sur la ligne 1. Après une planification journalière, il devient crucial de vérifier s'il nous reste encore suffisamment de jours pour programmer le NF500mg. Si c'est le cas, nous le plaçons juste après les mois d'achèvement pour NF1000mg ; sinon, nous passons à la ligne 2.

Dans la ligne 2, notre première priorité est de planifier ce qui reste de la dose de NF500 mg avant de passer au dosage NF850 mg. Chaque étape suivante doit être séquencée en conséquence pour un processus de planification et de production efficace sur les deux lignes.

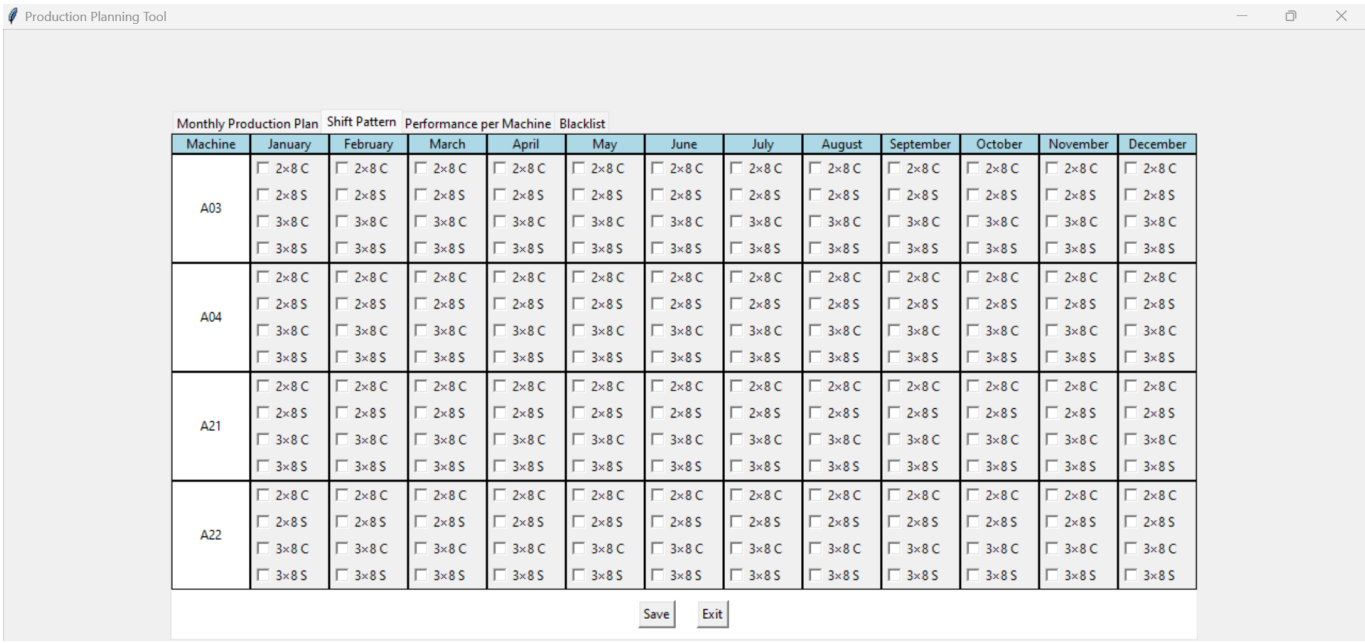


FIGURE 5.3 – Mode de travail data entry

Après avoir terminé tout cela, il est nécessaire de déterminer les dates de fin de tableting en revenant en arrière 15 jours de la date de fin du conditionnement. Après avoir déterminé cette date, nous passons à la prochaine étape : calculer la date de début du tableting en se basant sur le calcul de la date de fin du tableting, en diminuant le nombre de jours nécessaires pour le tableting, pour chaque produit et pour chaque demande. Ensuite, l'ordre de fabrication des deux dosages NF500mg et NF1000mg est déterminé sur la machine A05 car les deux dosages se produisent seulement cette machine. Il est nécessaire de réaliser la fabrication de la NF850mg sur la machine A02.

La dernière étape consiste à établir le délai de libération, ce qui nécessite d'ajouter 30 jours calendriers à la date de fin du conditionnement pour chaque dosage et chaque requête.Après avoir établi ces dates, on établit le calendrier quotidien de libération.

(c) **Extraits des algorithmes :**

Algorithme : Vérification de Jours Ouvrables et de Jours Fériés

L'algorithme suivant permet de vérifier si une date donnée est un jour ouvrable ou un jour férié.

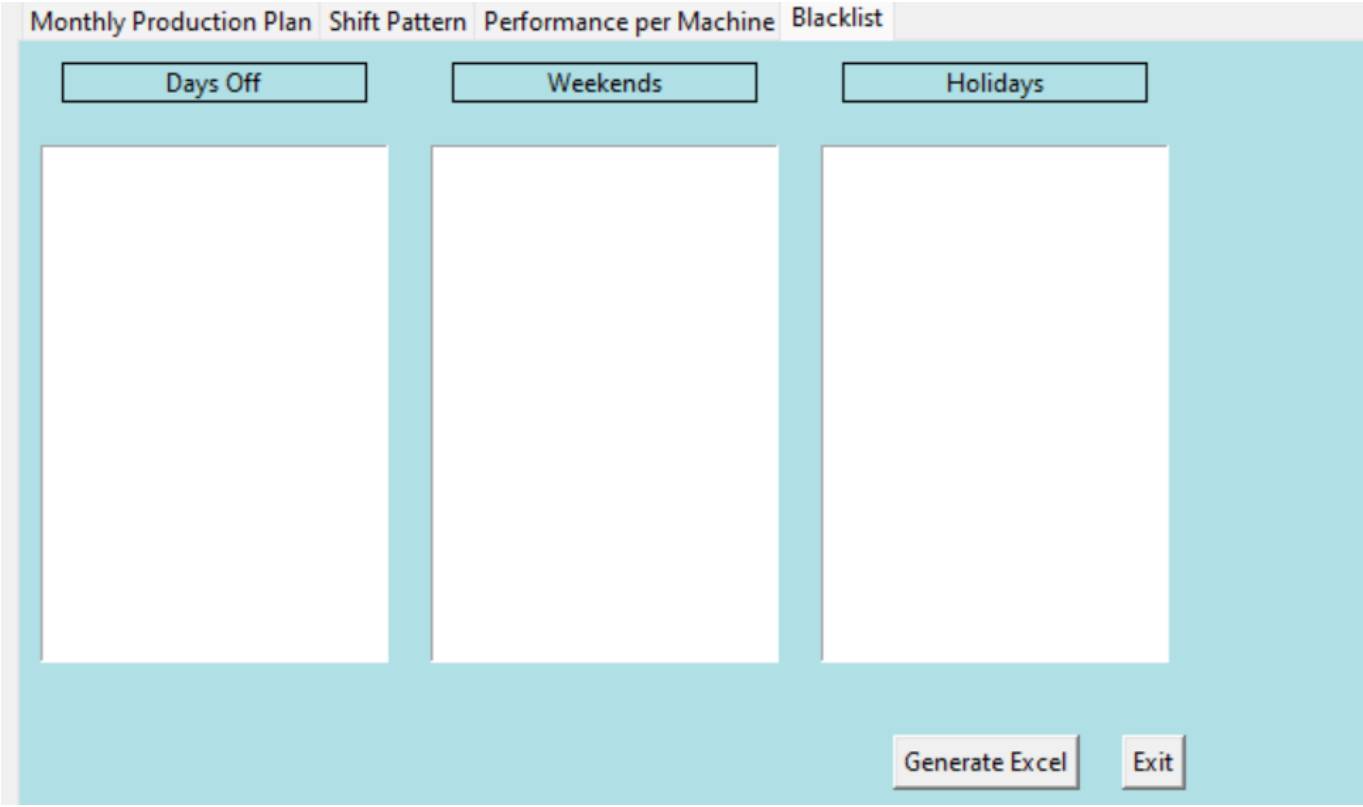


FIGURE 5.4 – Liste des jours offs data entry

Algorithm 1 – Vérification de Jours Ouvrables et de Jours Fériés

Data: date_to_check : date à vérifier, holidays : liste des jours fériés
Result: True si la date est un jour ouvrable, False sinon

```
Function IsBusinessDay(date_to_check, holidays) :
    weekday ← date_to_check.dayOfWeek() if weekday = Friday ∨ weekday = Saturday then
        return False
    end
    foreach holiday ∈ holidays do
        if date_to_check = holiday then
            return False
        end
    end
    return True
```

Algorithme : Calcul des Jours de Production et des Dates

L'algorithme suivant permet de calculer le nombre de jours de production nécessaires pour satisfaire la demande mensuelle, ainsi que les dates de début et de fin de production, en tenant compte des jours non ouvrables.

Algorithm 2 – Calcul des Jours de Production et des Dates

Data: demande_mensuelle : demande de production pour le mois, capacite_quotidienne : capacité de production par jour, jours_non_ouvrables : liste des jours non ouvrables, date_debut : date de début du mois
Result: Nombre de jours de production nécessaires, date de fin de production

Function `CalculerJoursEtDatesProduction`(*demande_mensuelle, capacite_quotidienne, jours_non_ouvrables, date_debut*) **:**
    jours_necessaires ← 0 production_accumulee ← 0 date_courante ← date_debut
    while *production_accumulee* < *demande_mensuelle* **do**
        if *date_courante* ∉ *jours_non_ouvrables* **then**
            production_accumulee ← production_accumulee + capacite_quotidienne jours_necessaires ← jours_necessaires + 1
        end
        date_courante ← date_courante + 1 jour
    end
    date_fin ← date_courante − 1 jour **return** (jours_necessaires, date_fin)

Algorithme : Génération du Plan de Production pour la NF1000mg

L'algorithme suivant permet de générer un plan de production pour la NF1000mg en fonction des jours de production, des dates de début et de fin, et des jours non ouvrables.

Algorithm 3 – Génération du Plan de Production

```
Data: quantite_totale : quantité totale à produire, capacite_quotidienne : capacité
      de production par jour, jours_non_ouvrables : liste des jours non ouvrables,
      date_debut : date de début, date_fin : date de fin
Result: Plan de production détaillé avec dates et quantités
Function GenererPlanProduction(quantite_totale, capacite_quotidienne,
 jours_non_ouvrables, date_debut, date_fin) :
    production_restante ← quantite_totale date_courante ← date_debut
     plan_production ← tableau vide
    while date_courante ≤ date_fin & production_restante > 0 do
        if date_courante ∉ jours_non_ouvrables then
            if production_restante ≥ capacite_quotidienne then
                quantite_du_jour ← capacite_quotidienne
            else
                quantite_du_jour ← production_restante
            end
            production_restante ← production_restante − quantite_du_jour
             ajouter (date_courante, quantite_du_jour) au plan_production
        end
        date_courante ← date_courante + 1 jour
    end
    return plan_production
```

Algorithme : Calcul des Jours Restants et Possibilité de Production de la NF500mg

L'algorithme suivant permet de calculer les jours restants de chaque mois après la production de la NF1000mg et de vérifier si la machine peut également produire la NF500mg.

Algorithm 4 – Calcul des Jours Restants et Possibilité de Production

Data: capacite_quotidienne : capacité de production par jour, plan_production_NF1000 : plan de production de la NF1000mg, plan_production_NF500 : plan de production de la NF500mg
Result: Jours restants de production pour chaque mois et possibilité de production de la NF500mg

```
34 Function CalculerJoursRestantsEtPossibilite(capacite_quotidienne, plan_production_NF1000, plan_production_NF500) :
35     jours_restants ← dictionnaire vide
36     foreach mois ∈ plan_production_NF1000 do
37         jours_restants[mois] ← 0  foreach jour ∈ mois do
38             capacite_restante ← capacite_quotidienne
39             if jour ∈ plan_production_NF1000[mois] then
40                 capacite_restante ← capacite_restante − plan_production_NF1000[mois][jour]
41             end
42             if jour ∈ plan_production_NF500[mois] then
43                 capacite_restante ← capacite_restante − plan_production_NF500[mois][jour]
44                 if capacite_restante ≥ 0 then
45                     jours_restants[mois] ← jours_restants[mois] + 1
46                 end
47             end
48         end
49     end
50     return jours_restants
```

Algorithme : Génération du Plan de Production de la NF500mg et NF850mg

L'algorithme suivant permet de générer le plan de production de la NF500mg avec priorité sur la machine A22 (ligne 2) si elle ne peut pas être produite sur la ligne 1, puis de générer le plan de production de la NF850mg.

Algorithm 5 – Génération du Plan de Production

Data: capacite_quotidienne : capacité de production par jour, plan_production_NF500_L1 : plan de production de la NF500mg sur la ligne 1, plan_production_NF500_L2 : plan de production de la NF500mg sur la ligne 2, plan_production_NF850 : plan de production de la NF850mg

Result: Plan de production de la NF500mg sur la ligne 1 ou ligne 2, et plan de production de la NF850mg

```
Function GenererPlanProduction(capacite_quotidienne, plan_production_NF500_L1, plan_production_NF500_L2, plan_production_NF850) :
    if plan_production_NF500_L1 = vide then
        plan_production_NF500 ← plan_production_NF500_L2
    end
    else
        plan_production_NF500 ← plan_production_NF500_L1
    end
    plan_production_NF850 ← plan_production_NF850
    return (plan_production_NF500, plan_production_NF850)
```

Algorithme : Calcul de la Date de Fin de Production en Bulk et des Jours de Production

L'algorithme suivant permet de calculer la date de fin de la production en bulk et de déterminer le nombre de jours nécessaires pour la production, en se basant sur les demandes mensuelles et les capacités des machines.

Algorithm 6 – Calcul de la Date de Fin de Production en Bulk et des Jours de Production

Data: demandes_mensuelles : demandes mensuelles de production, capacites_machines : capacités des machines par mois et jour, date_fin_packaging : date de fin du packaging

Result: Date de fin du bulk, Nombre de jours nécessaires pour la production

```
Function CalculerDateFinBulkEtNombreJoursProduction(demandes_mensuelles, capacites_machines, date_fin_packaging) :
    date_fin_bulk ← date_fin_packaging − 15 jours
    nombre_jours_production ← 0
    foreach mois ∈ demandes_mensuelles do
        foreach jour ∈ demandes_mensuelles[mois] do
            demande ← demandes_mensuelles[mois][jour] capacite ← capacites_machines[mois][jour] if demande > capacite then
                nombre_jours_production ← nombre_jours_production + 1
            end
        end
    end
    return date_fin_bulk, nombre_jours_production
```

Algorithme : Génération du Plan de Production Bulk pour la NF850mg sur la Machine A03

L'algorithme suivant permet de générer le plan de production bulk pour la NF850mg sur la machine A03, en se basant sur les demandes mensuelles et les capacités de la machine.

Algorithm 7 – Génération du Plan de Production Bulk pour la NF850mg

Data: demandes_mensuelles : demandes mensuelles de production, capacites_machine : capacités de la machine A03 par jour, date_debut_production : date de début de la production
Result: Plan de production détaillé avec dates et quantités

```
Function GenererPlanProductionBulk(demandes_mensuelles, capacites_machine, date_debut_production) :
    date_courante ← date_debut_production  plan_production ← liste vide
    foreach mois ∈ demandes_mensuelles do
        foreach jour ∈ demandes_mensuelles[mois] do
            demande ← demandes_mensuelles[mois][jour]  while demande > 0 do
                capacite ← capacites_machine[date_courante]  if capacite > demande then
                    capacite_utilisee ← demande
                end
                else
                    capacite_utilisee ← capacite
                end
                demande ← demande − capacite_utilisee  plan_production ← plan_production ∪ {date_courante : capacite_utilisee}
                date_courante ← date_courante + 1 jour
            end
        end
    end
    return plan_production
```

Algorithme : Génération du Plan de Production Bulk pour la NF500mg et la NF1000mg

L'algorithme suivant permet de générer le plan de production bulk pour la NF500mg et la NF1000mg sur la machine A04, en respectant l'ordre d'antériorité.

Algorithm 8 – Génération du Plan de Production Bulk pour la NF500mg et la NF1000mg

Data: demandes_mensuelles_500mg : demandes mensuelles pour la NF500mg, demandes_mensuelles_1000mg : demandes mensuelles pour la NF1000mg, capacites_machine : capacités de la machine A04 par jour, date_debut_production : date de début de la production

Result: Plan de production détaillé avec dates et quantités pour chaque produit

```
Function GenererPlanProductionBulk(demandes_mensuelles_500mg, demandes_mensuelles_1000mg, capacites_machine, date_debut_production) :
    date_courante ← date_debut_production  plan_production ← liste vide
    // Produire la NF500mg en premier
    foreach mois ∈ demandes_mensuelles_500mg do
        foreach jour ∈ demandes_mensuelles_500mg[mois] do
            demande ← demandes_mensuelles_500mg[mois][jour]  while demande > 0 do
                capacite ← capacites_machine[date_courante]  if capacite > demande then
                    capacite_utilisee ← demande
                end
                else
                    capacite_utilisee ← capacite
                end
                demande ← demande − capacite_utilisee  plan_production ← plan_production ∪ {NF500mg, date_courante : capacite_utilisee}  date_courante ← date_courante + 1 jour
            end
        end
    end
    // Une fois la NF500mg produite, produire la NF1000mg
    foreach mois ∈ demandes_mensuelles_1000mg do
        foreach jour ∈ demandes_mensuelles_1000mg[mois] do
            demande ← demandes_mensuelles_1000mg[mois][jour]  while demande > 0 do
                capacite ← capacites_machine[date_courante]  if capacite > demande then
                    capacite_utilisee ← demande
                end
                else
                    capacite_utilisee ← capacite
                end
                demande ← demande − capacite_utilisee  plan_production ← plan_production ∪ {NF1000mg, date_courante : capacite_utilisee}  date_courante ← date_courante + 1 jour
            end
        end
    end
    return plan_production
```

Une fois que tout les plans de production sont établis , on passe au coté gestion des stocks .

Partie II : Gestion des stocks :

(a) **Présentation de l'interface :**

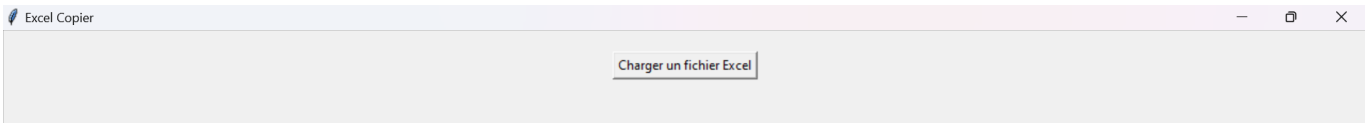


FIGURE 5.5 – Import data

Sur cette interface , l'utilisateur doit importer les données associes : BOM (recette) , Scrap(taux de déchets) et les stocks initiaux de chaque matière première.

(b) **Procédures et instructions :**
Pour ce faire, nous commencerons par calculer les consommations journalières, eu égard des quantités reprises sur le plan de production, en utilisant le BOM de référence. Ensuite on leur ajoute les taux de déchets pour obtenir la consommation réelle. Par la suite, nous ferons un suivi des stocks pour chaque matière première. Lorsque le stock actuel est inférieur à la moitié de la différence entre le stock maximum et le stock minimum, nous lancerons une commande de réapprovisionnement.

(c) **Extraits des algorithmes :**

Algorithme : Calcul des Consommations Journalières d'après le Plan de Production

L'algorithme suivant permet de calculer les consommations journalières des matériaux nécessaires pour la production d'après le plan de production sur chaque machine, en utilisant les BOM.

Algorithm 9 – Calcul des Consommations Journalières

Data: plan_production : plan de production avec dates, quantités, et machines, bom : bill of materials pour chaque produit
Result: Consommations journalières pour chaque matériau sur chaque machine

```
Function CalculerConsommationsJournalieres(plan_production, bom) :
    consommations_journalieres ← dictionnaire vide
    foreach jour ∈ plan_production do
        foreach machine ∈ plan_production[jour] do
            foreach produit, quantite ∈ plan_production[jour][machine] do
                foreach materiau, quantite_par_unite ∈ bom[produit] do
                    quantite_necessaire ← quantite × quantite_par_unite
                    if jour ∉ consommations_journalieres then
                        consommations_journalieres[jour] ← dictionnaire vide
                    end
                    if machine ∉ consommations_journalieres[jour] then
                        consommations_journalieres[jour][machine] ← dictionnaire vide
                    end
                    if materiau ∈ consommations_journalieres[jour][machine] then
                        consommations_journalieres[jour][machine][materiau] ←
                          consommations_journalieres[jour][machine][materiau] +
                          quantite_necessaire
                    end
                    else
                        consommations_journalieres[jour][machine][materiau] ←
                          quantite_necessaire
                    end
                end
            end
        end
    end
    return consommations_journalieres
```

Algorithme : Calcul des Consommations Réelles avec Taux de Scrap

L'algorithme suivant permet de calculer les consommations réelles des matériaux nécessaires pour la production d'après le plan de production sur chaque machine, en utilisant les BOM et les taux de scrap.

Algorithm 10 – Calcul des Consommations Réelles

Data: plan_production : plan de production avec dates, quantités, et machines, bom : bill of materials pour chaque produit, taux_scrap : taux de scrap pour chaque produit
Result: Consommations réelles pour chaque matériau sur chaque machine

```
Function CalculerConsommationsReelles(plan_production, bom, taux_scrap) :
    consommations_reelles ← dictionnaire vide
    foreach jour ∈ plan_production do
        foreach machine ∈ plan_production[jour] do
            foreach produit, quantite ∈ plan_production[jour][machine] do
                quantite_ajustee ← quantite × (1 + taux_scrap[produit])
                foreach materiau, quantite_par_unite ∈ bom[produit] do
                    quantite_necessaire ← quantite_ajustee × quantite_par_unite
                    if jour ∉ consommations_reelles then
                        consommations_reelles[jour] ← dictionnaire vide
                    end
                    if machine ∉ consommations_reelles[jour] then
                        consommations_reelles[jour][machine] ← dictionnaire vide
                    end
                    if materiau ∈ consommations_reelles[jour][machine] then
                        consommations_reelles[jour][machine][materiau] ←
                          consommations_reelles[jour][machine][materiau] +
                          quantite_necessaire
                    end
                    else
                        consommations_reelles[jour][machine][materiau] ←
                          quantite_necessaire
                    end
                end
            end
        end
    end
    return consommations_reelles
```

Algorithme : Suivi des Stocks et Génération de Commandes

Cet algorithme suit les niveaux de stock et génère des commandes lorsque le stock est inférieur à un seuil déterminé.

Algorithm 11 – Suivi des Stocks et Commandes

Data: stock_actuel : dictionnaire des niveaux de stock actuels pour chaque produit, stock_max : dictionnaire des niveaux de stock maximum pour chaque produit, stock_min : dictionnaire des niveaux de stock minimum pour chaque produit

Result: Génère des commandes pour les produits dont le stock est inférieur au seuil

```
Function SuiviStocks(stock_actuel, stock_max, stock_min) :
    commandes ← liste vide
    foreach produit ∈ stock_actuel do
        seuil ← (stock_max[produit] − stock_min[produit])/2 if stock_actuel[produit] < seuil then
            quantite_a_commander ← stock_max[produit] − stock_actuel[produit]
            commandes.append((produit, quantite_a_commander))
        end
    end
    return commandes
```

Gestion des exceptions :

La gestion des exceptions consiste a fournir des instructions que l'utilisateur n'a pas le droit de faire .

Parmi ces exceptions :

Sur l'interface :

L'utilisation n'a pas le droit de remplir aucun champ avec des données type chaine de caractère .S'il le fait , une boite de dialogue va s'afficher automatiquement et va le notifier que y'a une erreur.

Aussi , l'utilisateur n'a pas le droit d'importer n'importe quel fichier a part un fichier sous-forme xlsx , autrement dit un fichier excel.

5.3 Résultats et Analyse

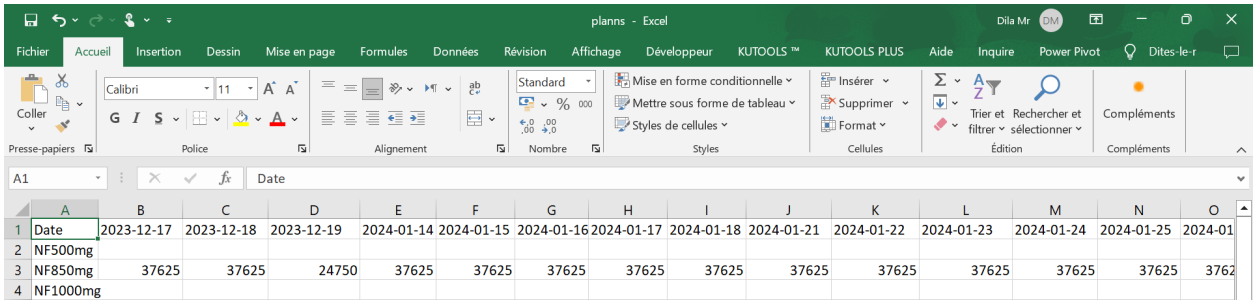
Les résultats obtenus avec notre outil basé sur notre modèle mathématique, sont développés selon des algorithmes mathématiques, des lois et surtout une logique mathématique pour résoudre notre problèmes qui consiste en l'optimisation des processus de production.

Sur la base des connaissances acquises grâce à l'ordonnancement, nous avons pu créer un plan de production pour chaque processus et en même temps calculer les dates de début et de fin de production pour chacun des trois processus pour chaque produit fini.

De plus, on a utilisé nos compétences en gestion des stocks pour surveiller et suivre les stock des produits finis et des matières premières.

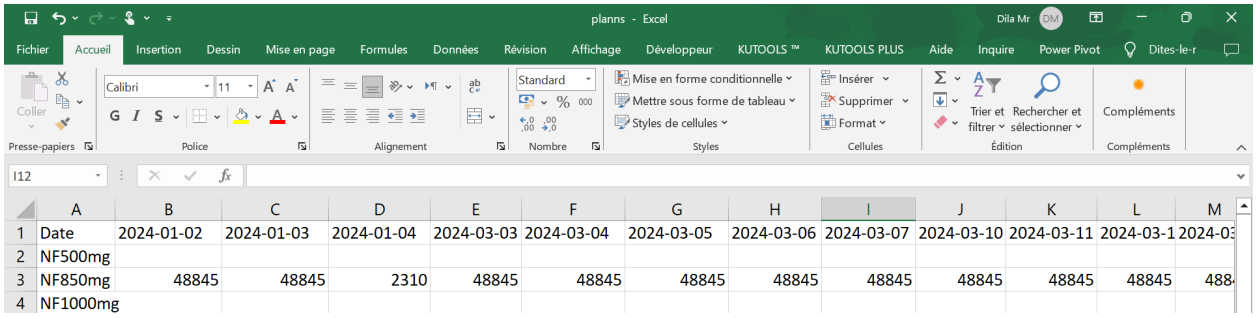
5.3.1 Part I : Ordonnancement :

Dans la figure ci-dessous, l'outil établit un plan pour un des 3 processus de production , tout en prenant en compte les contraintes et exigences de notre modèle mathématique .Aussi , il a pris en considération que tous nos résultats doivent être fournis en 2024 pour le conditionnement.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Date	2023-12-17	2023-12-18	2023-12-19	2024-01-14	2024-01-15	2024-01-16	2024-01-17	2024-01-18	2024-01-21	2024-01-22	2024-01-23	2024-01-24	2024-01-25	2024-01
2	NF500mg														
3	NF850mg	37625	37625	24750	37625	37625	37625	37625	37625	37625	37625	37625	37625	37625	3762
4	NF1000mg														

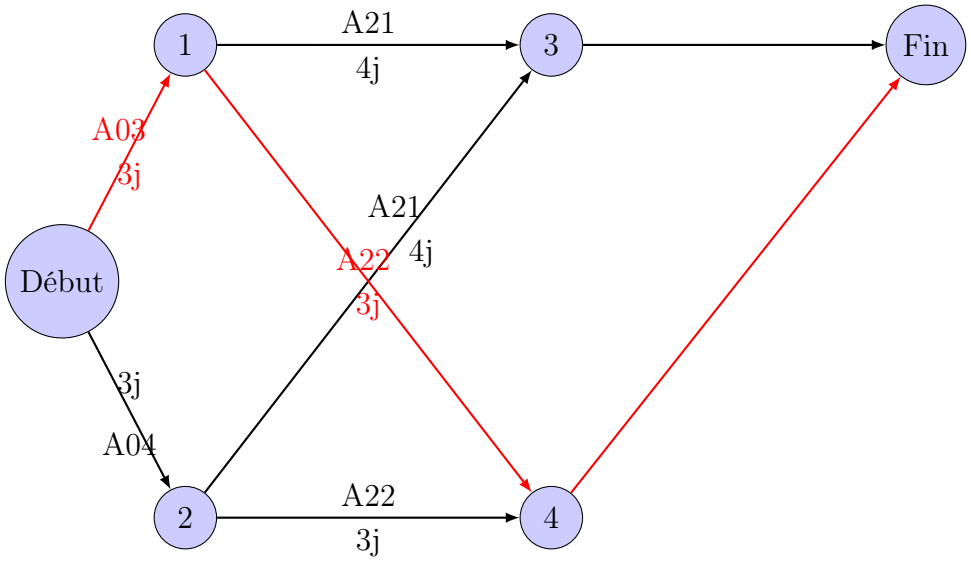
FIGURE 5.6 – Résultats du Bulk



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	2024-01-02	2024-01-03	2024-01-04	2024-03-03	2024-03-04	2024-03-05	2024-03-06	2024-03-07	2024-03-10	2024-03-11	2024-03-1	2024-03
2	NF500mg												
3	NF850mg	48845	48845	2310	48845	48845	48845	48845	48845	48845	48845	48845	488
4	NF1000mg												

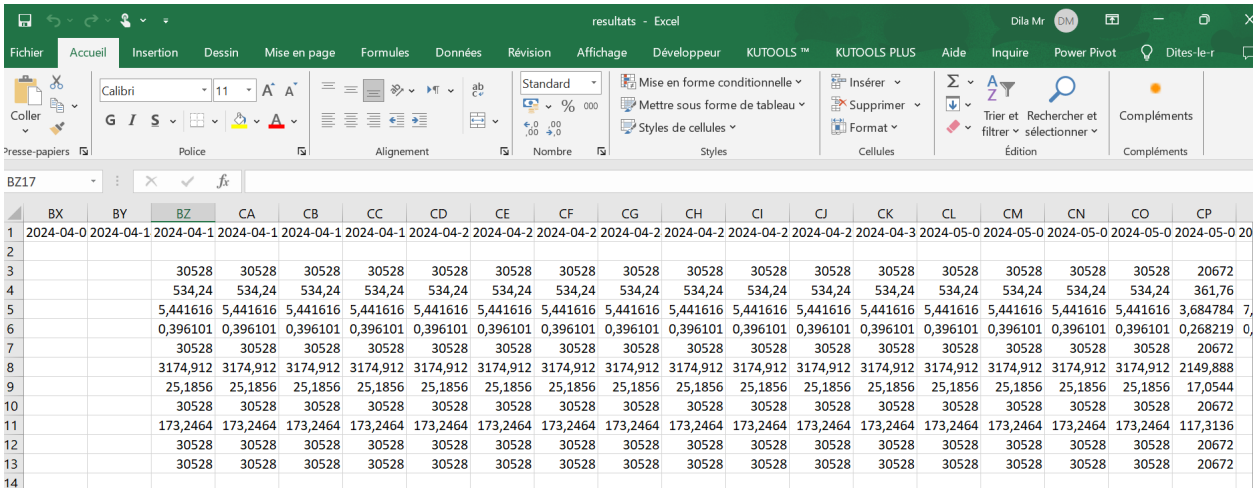
FIGURE 5.7 – Résulta du packaging

Voici une représentation des résultats obtenus :



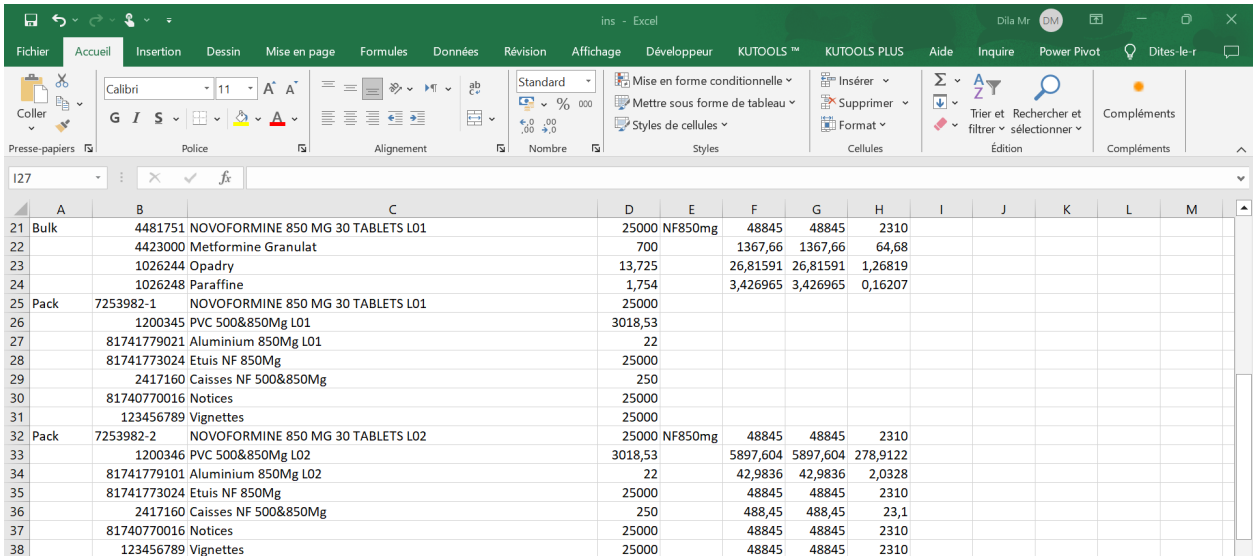
- Notons que les arcs entre les sommets représentent les durées de production du produit NF850mg sur chaque machine durant chaque processus .
- Le chemin optimal est mis en évidence en rouge dans le diagramme.

5.3.2 Part II : Gestion des stocks :



	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG	CH	CI	CJ	CK	CL	CM	CN	CO	CP
1	2024-04-0	2024-04-1	2024-04-1	2024-04-1	2024-04-1	2024-04-1	2024-04-2	2024-04-2	2024-04-2	2024-04-2	2024-04-2	2024-04-2	2024-04-2	2024-04-3	2024-05-0	2024-05-0	2024-05-0	2024-05-0	2024-05-0
2																			
3			30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	20672
4			534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	534,24	361,76
5			5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	5,441616	3,684784
6			0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,396101	0,268219
7			30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	20672
8			3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	3174,912	2149,888
9			25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	25,1856	17,0544
10			30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	20672
11			173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	173,2464	117,3136
12			30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	20672
13			30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	30528	20672
14																			

FIGURE 5.8 – Consommations du produit fini NF500mg



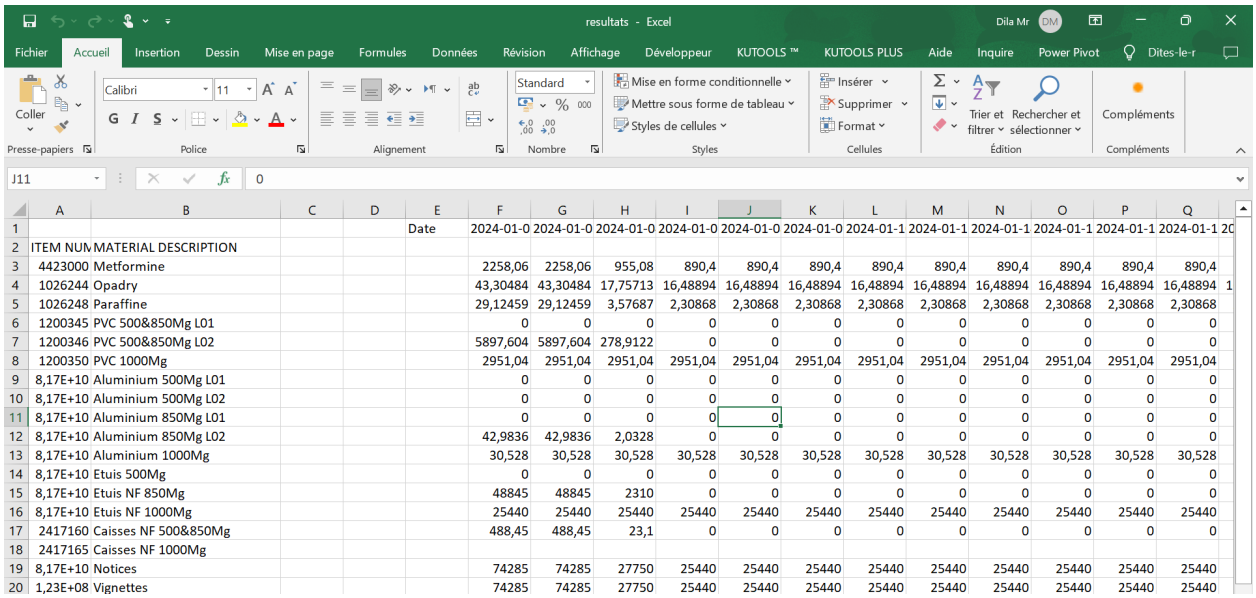
	A	B	C	D	E	F	G	H
21	Bulk	4481751	NOVOFORMINE 850 MG 30 TABLETS L01	25000	NF850mg	48845	48845	2310
22		4423000	Metformine Granulat	700		1367,66	1367,66	64,68
23		1026244	Opadry	13,725		26,81591	26,81591	1,26819
24		1026248	Paraffine	1,754		3,426965	3,426965	0,16207
25	Pack	7253982-1	NOVOFORMINE 850 MG 30 TABLETS L01	25000				
26		1200345	PVC 500&850Mg L01	3018,53				
27		81741779021	Aluminium 850Mg L01	22				
28		81741773024	Etuis NF 850Mg	25000				
29		2417160	Caisses NF 500&850Mg	250				
30		81740770016	Notices	25000				
31		123456789	Vignettes	25000				
32	Pack	7253982-2	NOVOFORMINE 850 MG 30 TABLETS L02	25000	NF850mg	48845	48845	2310
33		1200346	PVC 500&850Mg L02	3018,53		5897,604	5897,604	278,9122
34		81741779101	Aluminium 850Mg L02	22		42,9836	42,9836	2,0328
35		81741773024	Etuis NF 850Mg	25000		48845	48845	2310
36		2417160	Caisses NF 500&850Mg	250		488,45	488,45	23,1
37		81740770016	Notices	25000		48845	48845	2310
38		123456789	Vignettes	25000		48845	48845	2310

FIGURE 5.9 – Consommations du produit fini NF850mg

- Pour calculer les consommations de chacun des produits finis , on utilise la triple règle.
Prenons par exemple , le cas du produit NF850mg , durant le premier jour on a produit 48445 unités. on veut savoir combien consomme-t-on des matières premières ,tout en prenant en considération les données du Bom .on applique cette règle :

$$\text{Nouvelle quantité de metformine} = \frac{\text{Quantité à produire} \times \text{Quantité BOM}}{\text{Quantité produite du BOM}}$$

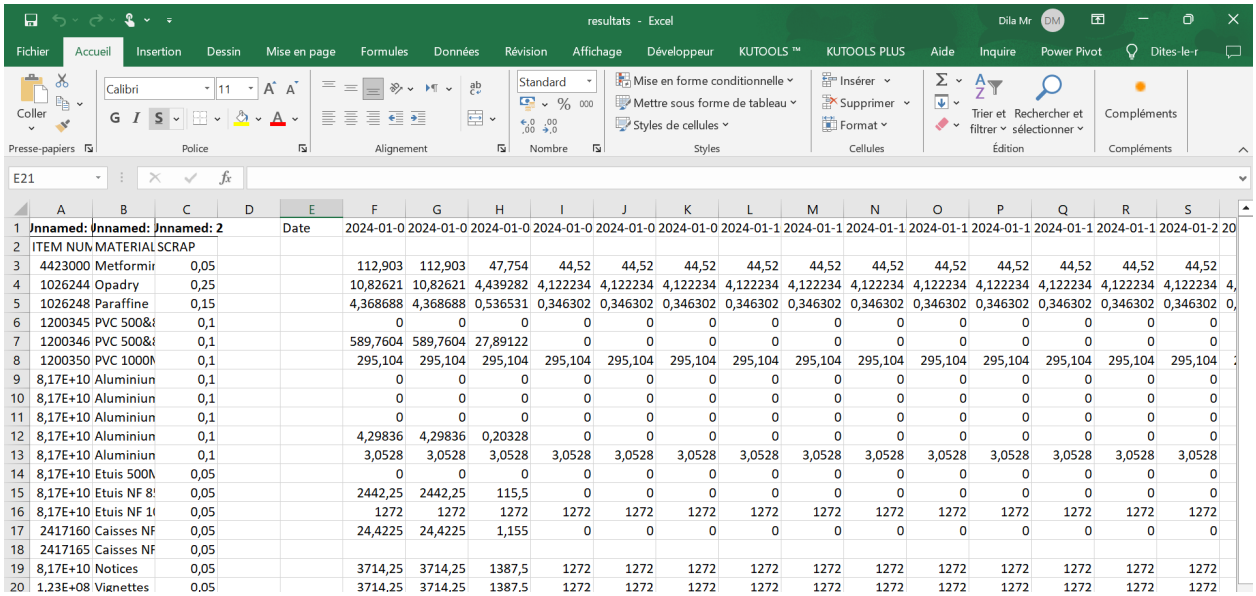
On obtient $1367,66kg$ exactement comme le résultat obtenu avec l'outil.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1					Date	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-1
2	ITEM NUM	MATERIAL DESCRIPTION															
3	4423000	Metformine				2258,06	2258,06	955,08	890,4	890,4	890,4	890,4	890,4	890,4	890,4	890,4	890,4
4	1026244	Opadry				43,30484	43,30484	17,75713	16,48894	16,48894	16,48894	16,48894	16,48894	16,48894	16,48894	16,48894	16,48894
5	1026248	Paraffine				29,12459	29,12459	3,57687	2,30868	2,30868	2,30868	2,30868	2,30868	2,30868	2,30868	2,30868	2,30868
6	1200345	PVC 500&850Mg L01				0	0	0	0	0	0	0	0	0	0	0	0
7	1200346	PVC 500&850Mg L02				5897,604	5897,604	278,9122	0	0	0	0	0	0	0	0	0
8	1200350	PVC 1000Mg				2951,04	2951,04	2951,04	2951,04	2951,04	2951,04	2951,04	2951,04	2951,04	2951,04	2951,04	2951,04
9	8,17E+10	Aluminium 500Mg L01				0	0	0	0	0	0	0	0	0	0	0	0
10	8,17E+10	Aluminium 500Mg L02				0	0	0	0	0	0	0	0	0	0	0	0
11	8,17E+10	Aluminium 850Mg L01				0	0	0	0	0	0	0	0	0	0	0	0
12	8,17E+10	Aluminium 850Mg L02				42,9836	42,9836	2,0328	0	0	0	0	0	0	0	0	0
13	8,17E+10	Aluminium 1000Mg				30,528	30,528	30,528	30,528	30,528	30,528	30,528	30,528	30,528	30,528	30,528	30,528
14	8,17E+10	Etuis 500Mg				0	0	0	0	0	0	0	0	0	0	0	0
15	8,17E+10	Etuis NF 850Mg				48845	48845	2310	0	0	0	0	0	0	0	0	0
16	8,17E+10	Etuis NF 1000Mg				25440	25440	25440	25440	25440	25440	25440	25440	25440	25440	25440	25440
17	2417160	Caisses NF 500&850Mg				488,45	488,45	23,1	0	0	0	0	0	0	0	0	0
18	2417165	Caisses NF 1000Mg															
19	8,17E+10	Notices				74285	74285	27750	25440	25440	25440	25440	25440	25440	25440	25440	25440
20	1,23E+08	Vignettes				74285	74285	27750	25440	25440	25440	25440	25440	25440	25440	25440	25440

FIGURE 5.10 – Consommations par matière première

- Pour calculer les consommations par matière première qui sont aussi appelées les consommations théoriques , on utilise la somme des consommations par produit fini pour chaque matière première et pour chaque jour .

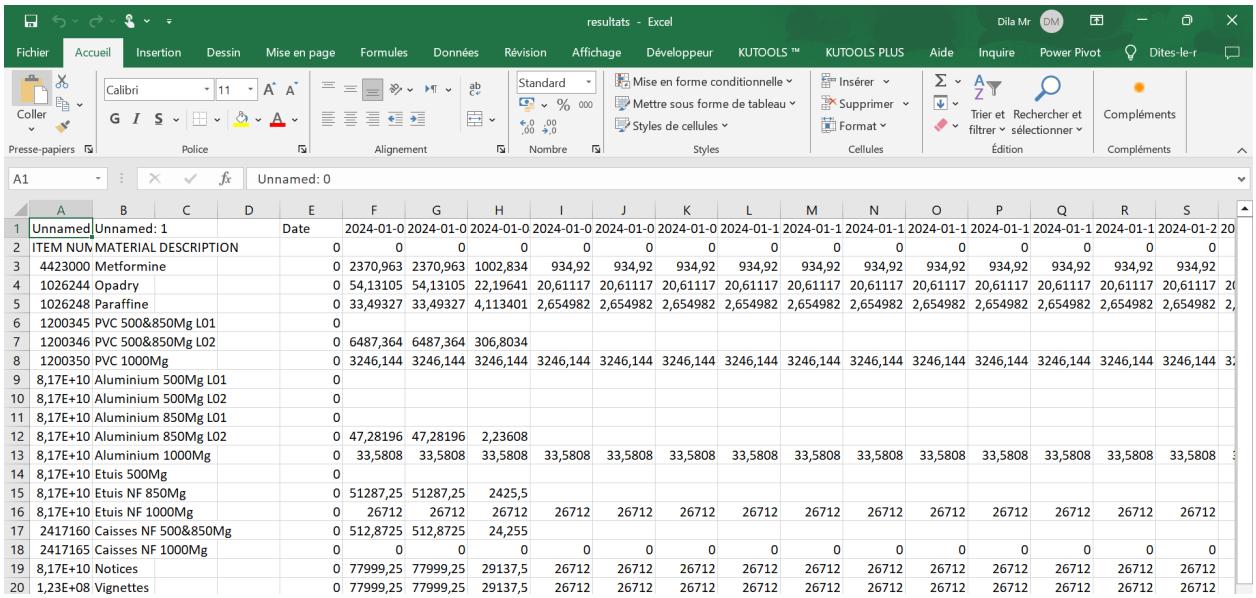


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Unnamed:	Unnamed:	Unnamed: 2		Date	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-2
2	ITEM NUM	MATERIAL	SCRAP																
3	4423000	Metformir	0,05			112,903	112,903	47,754	44,52	44,52	44,52	44,52	44,52	44,52	44,52	44,52	44,52	44,52	44,52
4	1026244	Opadry	0,25			10,82621	10,82621	4,439282	4,122234	4,122234	4,122234	4,122234	4,122234	4,122234	4,122234	4,122234	4,122234	4,122234	4,122234
5	1026248	Paraffine	0,15			4,368688	4,368688	0,536531	0,346302	0,346302	0,346302	0,346302	0,346302	0,346302	0,346302	0,346302	0,346302	0,346302	0,346302
6	1200345	PVC 500&	0,1			0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1200346	PVC 500&	0,1			589,7604	589,7604	27,89122	0	0	0	0	0	0	0	0	0	0	0
8	1200350	PVC 1000M	0,1			295,104	295,104	295,104	295,104	295,104	295,104	295,104	295,104	295,104	295,104	295,104	295,104	295,104	295,104
9	8,17E+10	Aluminiun	0,1			0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	8,17E+10	Aluminiun	0,1			0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	8,17E+10	Aluminiun	0,1			0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	8,17E+10	Aluminiun	0,1			4,29836	4,29836	0,20328	0	0	0	0	0	0	0	0	0	0	0
13	8,17E+10	Aluminiun	0,1			3,0528	3,0528	3,0528	3,0528	3,0528	3,0528	3,0528	3,0528	3,0528	3,0528	3,0528	3,0528	3,0528	3,0528
14	8,17E+10	Etuis 500M	0,05			0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	8,17E+10	Etuis NF 85	0,05			2442,25	2442,25	115,5	0	0	0	0	0	0	0	0	0	0	0
16	8,17E+10	Etuis NF 1(	0,05			1272	1272	1272	1272	1272	1272	1272	1272	1272	1272	1272	1272	1272	1272
17	2417160	Caisses NF	0,05			24,4225	24,4225	1,155	0	0	0	0	0	0	0	0	0	0	0
18	2417165	Caisses NF	0,05																
19	8,17E+10	Notices	0,05			3714,25	3714,25	1387,5	1272	1272	1272	1272	1272	1272	1272	1272	1272	1272	1272
20	1,23E+08	Vignettes	0,05			3714,25	3714,25	1387,5	1272	1272	1272	1272	1272	1272	1272	1272	1272	1272	1272

FIGURE 5.11 – Taux de Scrap pour chaque matière première

8. Les taux de scrap sont obtenus en appliquant cette règle :

nouveau taux de scrap = Taux de scrap du Bom ×consommation théorique

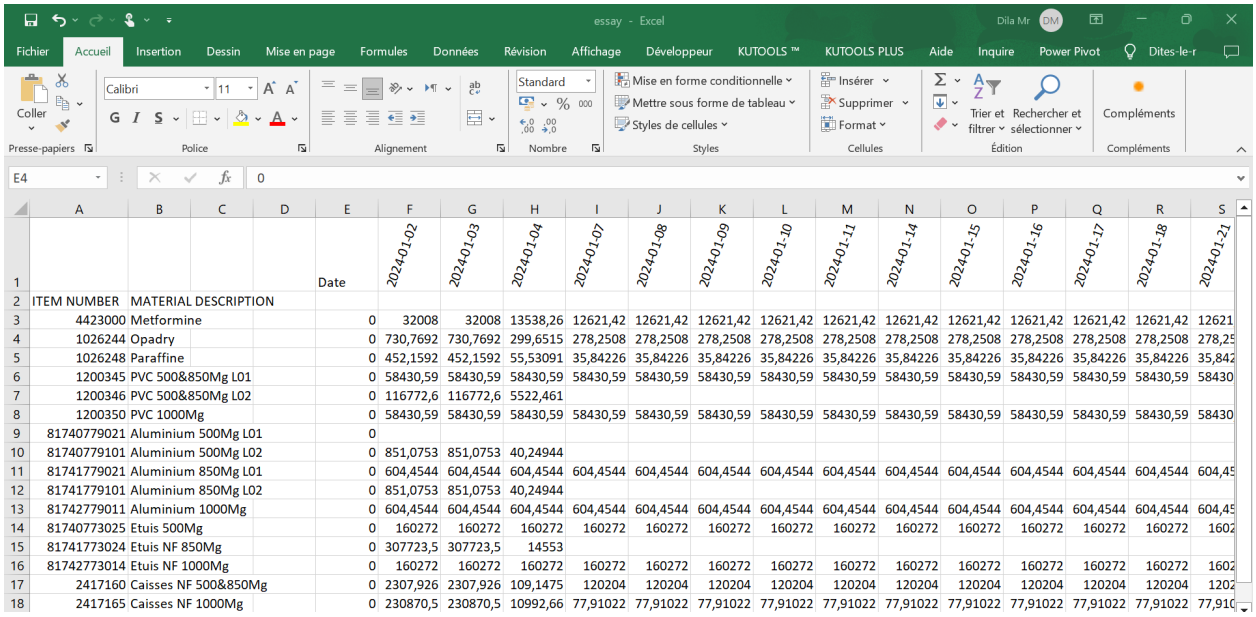


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Unnamed	Unnamed: 1			Date	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-0	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-1	2024-01-2
2	ITEM NUM	MATERIAL DESCRIPTION			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	4423000	Metformine			0	2370,963	2370,963	1002,834	934,92	934,92	934,92	934,92	934,92	934,92	934,92	934,92	934,92	934,92	934,92
4	1026244	Opadry			0	54,13105	54,13105	22,19641	20,61117	20,61117	20,61117	20,61117	20,61117	20,61117	20,61117	20,61117	20,61117	20,61117	20,61117
5	1026248	Paraffine			0	33,49327	33,49327	4,113401	2,654982	2,654982	2,654982	2,654982	2,654982	2,654982	2,654982	2,654982	2,654982	2,654982	2,654982
6	1200345	PVC 500&850Mg L01			0														
7	1200346	PVC 500&850Mg L02			0	6487,364	6487,364	306,8034											
8	1200350	PVC 1000Mg			0	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144	3246,144
9	8,17E+10	Aluminium 500Mg L01			0														
10	8,17E+10	Aluminium 500Mg L02			0														
11	8,17E+10	Aluminium 850Mg L01			0														
12	8,17E+10	Aluminium 850Mg L02			0	47,28196	47,28196	2,23608											
13	8,17E+10	Aluminium 1000Mg			0	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808	33,5808
14	8,17E+10	Etuis 500Mg			0														
15	8,17E+10	Etuis NF 850Mg			0	51287,25	51287,25	2425,5											
16	8,17E+10	Etuis NF 1000Mg			0	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712
17	2417160	Caisses NF 500&850Mg			0	512,8725	512,8725	24,255											
18	2417165	Caisses NF 1000Mg			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	8,17E+10	Notices			0	77999,25	77999,25	29137,5	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712
20	1,23E+08	Vignettes			0	77999,25	77999,25	29137,5	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712	26712

FIGURE 5.12 – Consommations réelles

- Les consommations réelles par matière première sont obtenues par cette loi :
Consommation réelle = Taux de scrap + Consommation théorique

- Ensuite , les stocks de sécurité sont calculés à l'aide de la loi mentionnée au 3ème chapitre.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1					Date	2024-01-02	2024-01-03	2024-01-04	2024-01-07	2024-01-08	2024-01-09	2024-01-10	2024-01-11	2024-01-14	2024-01-15	2024-01-16	2024-01-17	2024-01-18	2024-01-21
2	ITEM NUMBER	MATERIAL DESCRIPTION																	
3	4423000	Metformine			0	32008	32008	13538,26	12621,42	12621,42	12621,42	12621,42	12621,42	12621,42	12621,42	12621,42	12621,42	12621,42	12621
4	1026244	Opadry			0	730,7692	730,7692	299,6515	278,2508	278,2508	278,2508	278,2508	278,2508	278,2508	278,2508	278,2508	278,2508	278,2508	278,25
5	1026248	Paraffine			0	452,1592	452,1592	55,53091	35,84226	35,84226	35,84226	35,84226	35,84226	35,84226	35,84226	35,84226	35,84226	35,84226	35,842
6	1200345	PVC 500&850Mg L01			0	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430
7	1200346	PVC 500&850Mg L02			0	116772,6	116772,6	5522,461											
8	1200350	PVC 1000Mg			0	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430,59	58430
9	81740779021	Aluminium 500Mg L01			0														
10	81740779101	Aluminium 500Mg L02			0	851,0753	851,0753	40,24944											
11	81741779021	Aluminium 850Mg L01			0	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,45
12	81741779101	Aluminium 850Mg L02			0	851,0753	851,0753	40,24944											
13	81742779011	Aluminium 1000Mg			0	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,4544	604,45
14	81740773025	Etuis 500Mg			0	160272	160272	160272	160272	160272	160272	160272	160272	160272	160272	160272	160272	160272	1602
15	81741773024	Etuis NF 850Mg			0	307723,5	307723,5	14553											
16	81742773014	Etuis NF 1000Mg			0	160272	160272	160272	160272	160272	160272	160272	160272	160272	160272	160272	160272	160272	1602
17	2417160	Caisses NF 500&850Mg			0	2307,926	2307,926	109,1475	120204	120204	120204	120204	120204	120204	120204	120204	120204	120204	1202
18	2417165	Caisses NF 1000Mg			0	230870,5	230870,5	10992,66	77,91022	77,91022	77,91022	77,91022	77,91022	77,91022	77,91022	77,91022	77,91022	77,91022	77,910

FIGURE 5.13 – Stock de sécurité

- Le suivi de stock ou la stratégie de stock que l'entreprise utilise est obtenu en appliquant d'abord les lois mentionnés dans le 3ème chapitre .

Exemple de calcul du stock minimum :

Stock Min = 23508.04 $\times 45$ = 1057891.8

ou :

- Consommation moyenne quotidienne =23508.04
- Nombre de jours = 45

Exemple de calcul du stock maximum :

Stock Max = 23508.04 $\times 120 + 32008$

Stock Max =2820964.8 + 32008 = $\approx$ 2852972.8

ou :

- Consommation moyenne quotidienne =23508.04
- Délai d'approvisionnement = 120
- Stock de sécurité = 32008

En appliquant le dernier algorithme , on calcule le seuil pour la matière première Metformine Granulat :

(a) **Développement du Stock :**

- Jour 0 (Initial) : Stock Initial = 252349.1 unités
- Jour 1 :

$$\text{Stock Jour 1} = 252349.1 - 2370.963 = 250978.137 \text{ unités}$$

- Jour 2 (si nécessaire) :

$$\text{Stock Jour 2} = 250978.137 - 2370.963 = 248607.174 \text{ unités}$$

(b) **Calcul du Seuil :**

$$\text{Seuil} = \frac{\text{Stock Max} - \text{Stock Min}}{2}$$

tel que :

$$\text{Stock Max} = 2852972.8\,\text{unités}$$
$$\text{Stock Min} = 1057891.8\,\text{unités}$$

Calculer :

$$\text{Seuil} = \frac{2852972.8 - 1057891.8}{2} = \frac{1795081.0}{2} = 897540.5\,\text{unités}$$

(c) Vérifier si Stock Actuel < Seuil :

$$248607.174\,\text{unités} < 897540.5\,\text{unités} \quad \text{vrai ?}$$

(d) Décision : Comme le Stock Actuel est inférieur au Seuil, une commande doit être lancée.

5.3.3 Performance :

Notre outil Python se distingue par sa capacité à analyser de vastes quantités de données en temps réel, à identifier les problèmes et à proposer des solutions optimisées pour une production fluide et efficace. Grâce à une interface Communicative et à des fonctionnalités puissantes, il permet aux équipes de production et aux gestionnaires de prendre des décisions éclairées, basées sur des données concrètes et des analyses approfondies.

5.4 Conclusion

Ce chapitre donne un aperçu de l'outil développé en Python qui vise à automatiser les tâches répétitives et à optimiser les processus de production.
Nous avons d'abord décrit l'approche utilisée pour développer l'outil, puis présenté les résultats obtenus lors de l'analyse et enfin commenté la performance et la validité de ces résultats.

5.5 Conclusion générale :

Notre travail avait pour intitulé : «l'optimisation des flux au sein du site de LMTO (Local Manufacturing Tizi-Ouzou)». Le but ; poursuivi était de montrer comment l'entreprise peut produire à l'optimum compte tenu des contraintes aux quelles elle est confrontée.

Comme mentionné précédemment, l'industrie pharmaceutique est un secteur qui joue un rôle fondamental dans la santé publique en développant des médicaments sûrs et efficaces. Pour répondre aux exigences de qualité, de conformité réglementaire et de demande croissante, les entreprises pharmaceutiques doivent constamment rechercher des moyens d'améliorer leur efficacité opérationnelle toutes en optimisant leurs flux. Dans ce contexte, l'adoption de la démarche de l'optimisation des flux s'avère être une approche stratégique pertinente pour optimiser les processus tout en maintenant un haut niveau de qualité.

L'exploitation de la gestion de production au sein de l'entreprise NOVO NORDISK nous a permit d'étudier les processus de production ainsi que leurs organisation. L'entreprise est confrontée à une situation où elle a des difficultés de décision sur les lignes de production dues au problème de qualification des machines c'est à dire comme par exemple le dosage 1000 mg peut seulement se produire sur la ligne 1. C'est pour cela qu'elle a fait appel aux méthodes de recherche opérationnelle pour obtenir une amélioration du planning de production tout en évitant les longs durées dans la production. Donc, nous avons constaté que c'est un problème à qui on doit faire face à l'aide d'un bon ordonnancement des lignes de production en utilisant la méthode PERT. En suite, grâce à la description du système étudié on a élaboré un modèle mathématique qui maximise la production, sous forme d'un problème de programmation linéaire en nombre entier pour lequel on a choisi la méthode de Branch and Bound (B&B) qui est une méthode motivée par le fait qu'elle évoque des solutions optimale, vue que notre instances n'est pas assez grande. D'un autre côté, nous avons utilisé la gestion des stocks avec ses différents types et coûts, nécessitant l'utilisation du modèle de Wilson pour effectuer les calculs nécessaires et les inclure dans notre outil qui est notre objectif final pour minimiser le temps de planification et d'obtenir une devisions optimale sur les lignes de productions.

Références

[1] Embarek Melissa et Mahiddine thinhinane. "L'approvisionnement et la gestion des stocks, Cas : matière Granulat au sein de la société Local Manifavturing Oued - Aissi Tizi-Ouzou". Mémoire de fin de formation. Institut National Spécialisé de Formation Professionnelle, IMERZOUKENE Mohammed Arezki, 2023.

[2] Hameg Leticia et Hamdane Lynda et Hamadache leiticia. "Analyse de l'efficacité de la chaine d'approvisionnement, Cas : NOVO NORDISK". Rapport de stage en vue de l'obtention du diplôme de licence en Sciences Commerciales. Université Mouloud Mammeri de Tizi Ouzou, Département des Sciences Commerciales, 2023.

[3] Michel Sakarovitch (2009). *Optimisation combinatoire : Graphes et programmation linéaire*. Éditeur Hermann, 1984.

[4] GONDRAN Michel, MINOUX Michel. *Graphes et algorithmes (4e ed.)*. Éditeur Lavoisier, 2009.

[5] Claude Berge. (2010). *Théorie des graphes et ses applications*. Éditeur Dunod, 1967.

[6] DJARANE DYHIA et AIT AMMAR SARA. "Chaînes de Markov pour la modélisation et la résolution du problème de la gestion des stocks et du planning de production". Mémoire fin d'étude master 2. Université Mouloud Mammeri de Tizi Ouzou Faculté des Sciences, Département de Mathématiques, Laboratoire LAROMAD, 2022.

[7] Garey and D.S.Jhonson. *Computers and Intractability : A Guide to the Theory of Np-Completeness*. Éditeur Freemen 1979.

[8] Taieb Rayel. "Résolution d'un problème de planification de production : cas de l'entreprise MORTERO SPA, 2022.

Les sites Web :

[9] https://prospareblog.wordpress.com/2015/09/03/les-differents-types-de-stocks/

[10] https://pharmnet-dz.com

[11] https://python.sdv.univ-paris-diderot.fr/cours-python.pdf

[12] https://www.data-bird.co/blog/bibliotheque-python

[13] https://www.jedha.co/formation-python/librairies-bibliotheques

[14] https://www.tresfacile.net/la-bibliotheque-openpyxl-python/

[15] https://docs.python.org/fr/3.6/library/datetime.html

[16] https://docs.python.org/fr/3/library/tk.html

[17] http://www.xavierdupre.fr/app/teachpyx/helpsphinx/c_gui/tkinter.html

[18] https://blog.ig-conseils.com/automatiser-taches-repetitives-gagner-en-productivite/

Annexe A

Dessin d'un employé qui résume La stratégie de l'entreprise

FIGURE A.1 – Dessin d'un employé qui résume La stratégie de l'entreprise