École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
5e année
2013 - 2014

Rapport de projet de fin d'études

Prise en compte des reliquats dans la fabrication de chimiothérapies

Encadrants
Jean-Charles Billaut
jean-charles.billaut@univ-tours.fr
Université François-Rabelais, Tours, France

Étudiant
Thibault DREVON
thibault.drevon@etu.univ-tours.fr

DI5 2013 - 2014

Version du 13 mai 2014

Table des matières

Table des figures

Liste des tableaux

Introduction

Le présent document constitue le rapport de projet de fin d'études de l'étudiant Thibault Drevon, élève ingénieur de l'Ecole Polytechnique de l'Université de Tours spécialité informatique, promotion 2014. Il présente la problématique du projet ainsi que le travail effectué et contient tous les commentaires de l'étudiant concernant ces travaux. Le projet en lui-même porte sur la réalisation d'un ordonnancement efficace pour la prise en compte des reliquats dans l'ordonnancement de la production de produits de chimiothérapie. Ce sujet sera développé plus en détails dans les parties suivantes.

Hormis l'étudiant, les individus suivants sont également impliqués dans le projet :

- **Jean-Charles Billaut**, encadrant du projet de fin d'études, chargé de la maîtrise d'oeuvre.
- **Jean-François Tournamille**, pharmacien à l'hôpital Bretonneau de Tours, Pôle Santé Publique et Produits de Santé. En sa qualité de superviseur de la production de produits de chimiothérapie, il joue le rôle de responsable de la maîtrise d'ouvrage.
- **La société ETICSYS** et son représentant **François-Xavier Baillet**, potentiellement intéressés par l'implémentation au sein de leurs produits de l'algorithme trouvé. A la fin du projet, cependant, aucun contact n'a été établi entre l'étudiant et l'entreprise.

Remerciements

L'étudiant tient à remercier les personnes suivantes pour leur aide au cours ce long et périlleux ouvrage :

- **Jean-Charles Billaut**, bien sûr, pour son aide, son enthousiasme et sa patience.
- **Sébastien Aupetit**, professeur au département informatique de Polytech Tours, pour nous avoir appris à écrire proprement et efficacement un programme en C. Le projet n'aurait jamais pu fonctionner sans ses judicieux conseils.
- **Yannick Kergosien**, professeur au département informatique de Polytech Tours, pour m'avoir expliqué lors d'un précédent projet ce qu'était la programmation par contraintes et comment s'en servir. Cette solution a bien failli aboutir. Quel dommage.
- **L'anonyme relecteur**, il va en falloir du courage pour aborder un aussi ennuyeux document. J'admire la patience du lecteur qui arrivera à la dernière page.
- **Antoine Durand**, étudiant de la même promotion qui a une fois corrigé une erreur de ce projet et souhaité apparaitre dans cette section à l'occasion. Chose promise, chose due.

Première partie

Présentation du sujet

Le contexte du projet

Remarque Dans la plupart du document, les exemples cités se rapportant au fonctionnement d'une unité de production de chimiothérapies sont tirés de l'hôpital Bretonneau de Tours.

1.1 La fabrication de produits de chimiothérapie

1.1.1 Définitions

La chimiothérapie se définit comme le traitement du cancer à l'aide de cytotoxiques, c'est-à-dire des produits qui détruisent les cellules. Ces cytotoxiques sont chargés de détruire les cellules tumorales tout en limitant les dégâts causés sur les cellules saines.
La fabrication de produits de chimiothérapie est un processus complexe et coûteux. Les matières premières utilisées, dangereuses et rares, sont fabriquées par des laboratoires indépendants dans des conditions d'hygiène très strictes. Cette matière première est ensuite achetée par la sécurité sociale et acheminée vers les quelques centres de production habilités à fabriquer les poches de cytotoxiques. En France, une demi-douzaine de centres (tous des hôpitaux) ont l'autorisation et la charge de fabriquer ces produits.

1.1.2 Un double enjeu

La production de cytotoxiques est soumise à deux fortes contraintes qu'il est impossible d'ignorer. La difficulté réside dans le fait qu'elles sont le plus souvent d'objectifs contradictoires.

La satisfaction du patient

Lorsqu'un médecin spécialiste commande à l'un des centres la fabrication d'une poche sur-mesure, le patient est convié à se rendre au point d'administration du médicament dans des délais fixés à l'avance (1 ou 2 heures après le passage de la commande). La santé publique, dans son souci de satisfaire le citoyen, met alors tous les moyens en oeuvre pour que la poche de médicament commandée soit livrée dans les temps.
Ceci n'est pas toujours facile : réduction des ressources de production, augmentation de la demande et acheminement sont autant de difficultés qu'il n'est pas toujours possible de surmonter. Ainsi, en moyenne à Tours, plus de 20% des poches sont livrées avec 30 minutes de retard, certaines dépassant 1 heure.

Remarque L'acheminement est un problème à part entière (tournée de véhicule), et ne sera pas abordé dans le présent projet.

L'économie des ressources

Le second problème provient du fait que les molécules impliquées dans la fabrication des cytotoxiques ont des durées de vie très limitées. En temps normal, il n'est pas rare de devoir se débarrasser du reste d'un flacon de matière première parce que le composé est devenu instable ou a précipité. Cette mise au rebut implique parfois des pertes monétaires non négligeables. En effet, un flacon de matière première peut atteindre une valeur de 13 000 €, et les prix connaissent une inflation constante. Une commande (une poche) utilise en moyenne 3 flacons de matières première pour un coût total de 312 € environ. Sur une production annuelle évaluée à 12 000 000 €, c'est jusqu'à 1 000 000 € qui peut ainsi être inutilement dépensé par le

centre de fabrication. C'est pourquoi un ordonnancement rigoureux est extrêmement important.
Il existe déjà des algorithmes permettant la planification des tâches composant le processus de fabrication de ces produits [1] [2]. La problématique nouvelle qui se pose ici est la prise en compte des reliquats de matière première qui sont utilisées pour fabriquer les produits finaux.
La prise en compte de cette valeur dans l'ordonnancement, de façon à minimiser la quantité de matière ouverte puis jetée, pourrait permettre de réaliser des économies substantielles se comptant en centaines de milliers d'euros.

Le problème posé

2.1 Critères divergents

Comme on l'a vu dans le chapitre précédent, la problématique d'un ordonnancement correct de la production est double :

- L'ordonnancement choisi ne doit pas trop contrarier le patient. pour cela, des temps d'attentes raisonnables doivent être assurés. Ce critère encourage de fabriquer le plus vite possible, quel qu'en soit le prix.
- La fabrication doit prendre en compte le coûts des matières première et organiser la production de façon à minimiser la matière perdue (en réutilisant les matières premières avant péremption). Ce critère façonne un arrangement orienté sur une minimisation du prix au détriment de la qualité de service.

Il apparait d'ores et déjà que les deux objectifs, lorsqu'ils sont difficiles à satisfaire, sont contradictoires. Le problème posé s'avère complexe.

2.2 L'existant

2.2.1 ETICSYS

ETICSYS est une société de conseil indépendante spécialisée en innovation et ingénierie informatique auprès des acteurs de la santé. Dans le cadre de l'une de ses prestations auprès du centre hospitalier universitaire de Bretonneau, Tours, ETICSYS fournit un logiciel permettant entre autres d'ordonnancer la production de produits de chimiothérapie.

2.2.2 PLANIF

PLANIF est le logiciel mis en place par ETICSYS pour le CHU de Bretonneau pour ordonnancer la fabrication des cytotoxiques. Ce logiciel, qui remplit son rôle correctement, ne tient cependant compte que du critère de satisfaction du patient.
Toutes les 5 minutes, le logiciel est utilisé pour réarranger de façon dynamique la production en prenant en compte des nouvelles commandes passées depuis la dernière veille. Ainsi, l'attente du malade au point d'administration du médicament est minimisée, avec tout de même des retards résiduels conséquents (voir chapitre précédent).
La minimisation des pertes en matières premières est assurée "à vue" par le personnel de fabrication. Et si le centre de Tours fait office de brillant élève en la matière, ce n'est pas le cas de tous les autres lieux de production.

2.2.3 En collaboration avec ETICSYS

Pour améliorer sa prestation et la satisfaction de son client, ETICSYS souhaiterait intégrer une telle solution à son logiciel PLANIF .
Il s'avère nécessaire de mettre en place un suivi des flacons de cytotoxiques, une modélisation plus fine du problème de leur prise en compte pour la planification et de proposer de méthodes de résolution.

L'objectif du projet

3.1 Objectifs

L'objectif de ce projet est la mise en place d'une nouvelle solution d'ordonnancement prenant en compte les contraintes d'économies sur les reliquats. L'ordonnancement choisi devra minimiser les coûts liés au rejet de matière première périmée en encourageant la réutilisation des flacons entamés. La nouvelle solution devra cependant respecter les contraintes de délai de livraison des produits (ou au moins ne pas dégrader la qualité de service actuelle), de sorte que le patient n'ait pas à être impacté par la réorganisation de la production.
Le projet se veut avant tout théorique. il doit permettre, par des manipulations rapides, de tester l'impact de variations subtiles des données d'entrées (coûts des matières, temps de production...) sur la structure de l'ordonnancement construit et la qualité de la solution obtenue. De cette manière, il pourrait devenir un support pour effectuer une batterie de tests et mettre en évidence des schémas intéressants.

Initialement, selon l'avancement du projet et les priorités de l'UBCO et de la société ETICSYS, il était envisagé que l'algorithme du projet puisse éventuellement être intégré dans le logiciel PLANIF. Malheureusement, aucun contact n'ayant été établi dans le temps imparti, cette éventualité a été mise de côté. Toutefois, l'étudiant s'est assuré de laisser un maxmum de documentation et de rendre les sources facilement réutilisables.

3.2 La démarche de travail

La présente section décrit la démarche de travail adoptée pour mener à bien le projet.

3.2.1 Analyse du problème

En premier lieu, il est nécessaire de s'approprier la problématique dans sa globalité. La compréhension de chaque étape de la production et des contraintes qui y sont associées est indispensable. Des rencontres avec les intervenants du service de chimiothérapie du centre hospitalier de Bretonneau doivent avoir lieu pour faciliter cette assimilation. Après avoir intégré la problématique existante et la solution actuelle, il convient d'introduire la nouvelle contrainte économique.

3.2.2 Modélisation du problème

Après avoir assimilé de façon naturelle l'environnement et les éléments constitutifs du problème, il conviendra de réaliser un modèle formel mettant en relation les paramètres et les contraintes du problème. Ce modèle sera ensuite validé par un expert.

3.2.3 Modélisation de la solution

Une fois le problème clairement défini, il sera nécessaire de trouver une solution efficace pour résoudre ce problème. Si plusieurs techniques de résolution peuvent être mises en place, il convient soit :

– D'éliminer les solutions inefficaces et de choisir la plus efficace.

– D'établir un comparatif permettant de sélectionner une méthode plutôt qu'une autre en fonction de la situation.

3.3 Stratégie d'exploration

La complexité du problème mise en avant dès le début de l'exercice présuppose qu'il n'existe pas de méthode polynomiale pour trouver une solution optimale. Dans ces conditions, il convient d'explorer plusieurs méthodes de résolution différentes.
On commencera par explorer les méthodes exactes pour voir s'il n'est pas possible avec un peu d'astuce de construire un modèle permettant d'obtenir rapidement une solution de qualité supérieure. Puisque, en accord avec nos estimations, il s'avère que la construction d'une solution exacte est compromise, on aborde ensuite les méthodes approchées.
La partie suivante montre l'analyse formelle du problème.

Deuxième partie

Approche du problème

Formalisation du système

Remarque Les paramètres et données du problème sont tirées de l'exemple du CHU de Bretonneau à Tours, mais le problème doit être modélisé de telle sorte qu'il puisse être applicable à des environnements similaires jusqu'à un certain point, comme le CHU de Toulouse.

4.1 Représentation générale du système

Au CHRU de Bretonneau, Tours, comme dans d'autres villes de France, il existe une unité chargée de réaliser des produits de chimiothérapie (32000 poches par an environ). Pour y parvenir, l'unité fait intervenir des ressources différentes :

- des matières premières pour composer les produits.
- des machines pour fabriquer et contrôler les produits mis en jeu.
- des intervenants humains pour préparer, livrer puis administrer les produits.

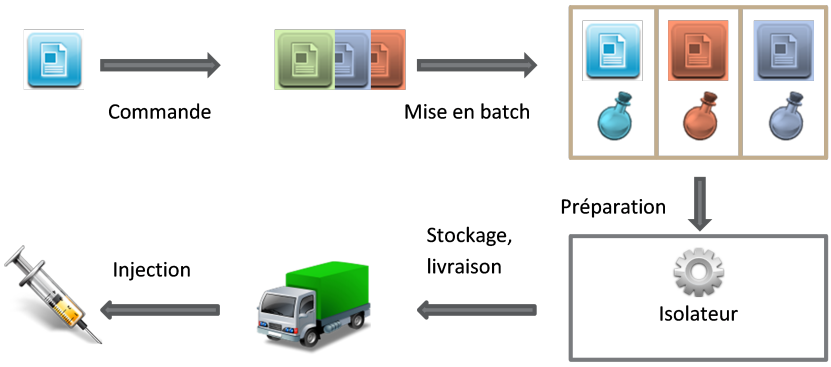

FIGURE 4.1 – Processus orienté commande

Le processus actuel de fabrication d'un produit se déroule ainsi :

1. Une commande est envoyée par un intervenant extérieur (médecin, pharmacien) pour demander la fabrication d'un produit. Cette demande spécifie :
   - Un type de produit à fabriquer qui implique une matière première, une quantité de cette matière première, une durée de fabrication, un délai de conservation une fois fabriqué.
   - Une date de livraison qui doit être respectée.
   - un temps de transport incompressible.

   La commande rejoint la file d'attente des commandes qui doivent être traitées.
2. Suivant la décision d'un préparateur ou celle d'un logiciel, la commande est à un instant donné ajoutée à un batch , c'est-à-dire un ensemble de commande réalisées simultanément (un paquet de commandes).
3. Les matières premières pour réaliser la commande sont réunies avec celles des autres commandes du batch et le tout est placé dans un isolateur. Après une période fixe de stérilisation , l'ensemble des commandes du batch sont fabriquées dans l'isolateur (en environnement stérile, donc) par des préparateurs qui restent à l'extérieur de la machine.

4. Chaque fois qu'un produit est fabriqué, il est sorti de la machine et est placé dans la file d'attente pour le contrôle , sans attendre la fin du batch.
5. Une fois le contrôle effectué (temps fixe), la commande est mise en attente pour son transfert vers le lieu d'administration du produit. L'attente peut être nulle ou d'une durée indéterminée. L'attente a lieu dans un réfrigérateur de capacité infinie.
6. Le produit est pris en charge par un livreur avec éventuellement d'autres commandes et le produit est livré sur le lieu de l'administration. Enfin, un personnel de soins se charge d'injecter le médicament au patient. Le produit ne doit pas arriver trop en retard ni périmer avant injection.

4.2 Autres points de vue

4.2.1 Point de vue consommation de ressources

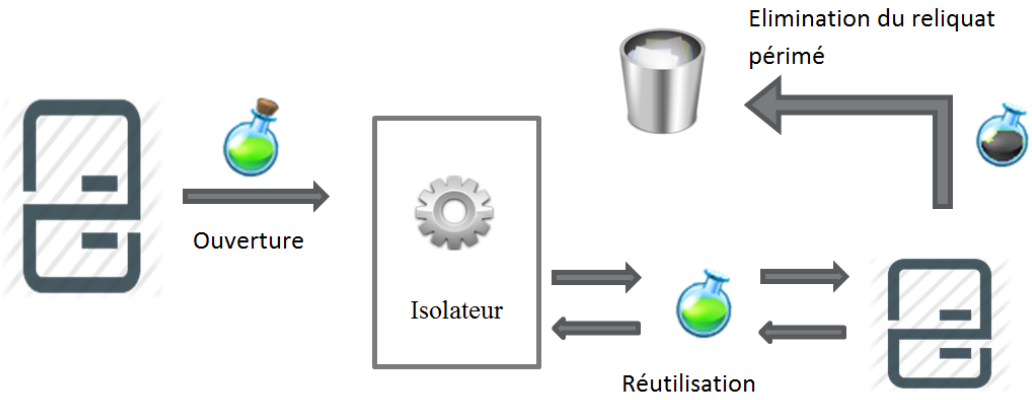

FIGURE 4.2 – Processus orienté ressource

Les matières premières sont stockées dans d'énormes réfrigérateurs. Tant qu'elles ne sont pas ouvertes, elles sont considérées impérissables et en quantités infinies. Une fois qu'un flacon est sorti du réfrigérateur, il est placé dans un panier pour participer à la réalisation d'un batch. Dans l'isolateur, le flacon est ouvert et partagé par toutes les commandes qui le nécessitent. Une fois ouvert, il acquiert une date de péremption qui dépend de sa nature. Une fois le batch terminé, s'il reste encore de la matière dans le flacon, le produit est remis au réfrigérateur jusqu'à sa réutilisation ou son élimination par dépassement de la date de péremption.

A une matière première (aussi appelée type de produit) sont associées :

- Des volumes de flacon spécifiques disponibles. Ces volumes sont spécifiques à chaque type de matière première. Les volumes ne sont pas tous égaux mais contenus dans un même ordre de grandeur.
- Un prix par unité de volume ou de matière (ml principalement).
- Une durée de conservation une fois ouvert.

Remarque A l'heure actuelle, il n'existe pas de relation définie entre le délai accordé pour réutiliser un flacon de matière et le temps imparti pour consommer un produit issu de cette même matière. Dans la pratique, cela signifie que lorsqu'on utilise une matière pour fabriquer une poche, peu importe le temps que cette matière a passé ouverte, on remet le compteur à zéro pour la durée de conservation du produit fini (même si la matière première était presque périmée).

4.2.2 Point de vue atelier

Deux types de machines sont employés dans le processus de fabrication :
- L'isolateur permet de réaliser le produit dans un environnement stérile.
- La contrôleuse vérifie que le produit est conforme.

Isolateurs

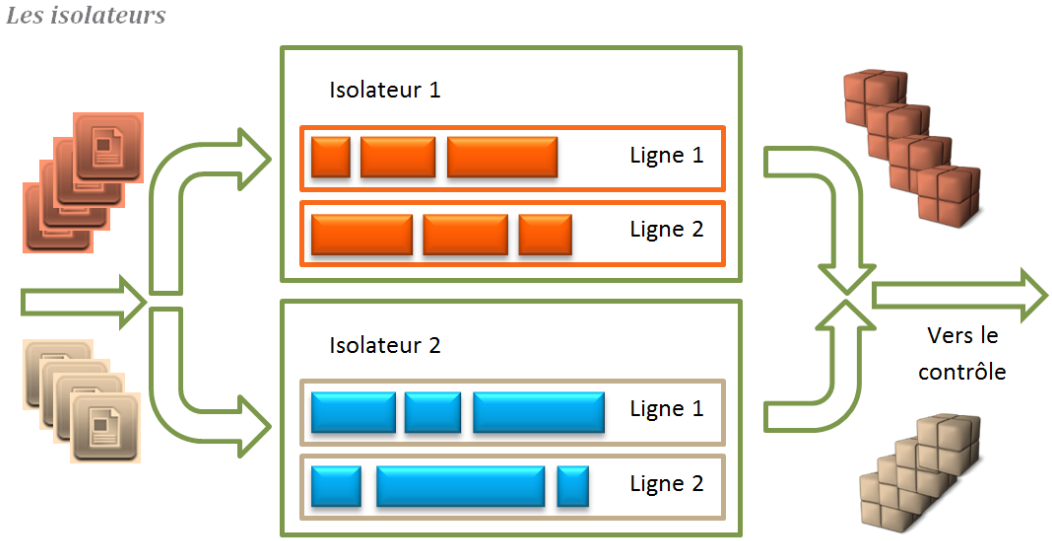

FIGURE 4.3 – Entrée/Sortie des isolateurs

Il n'existe pas qu'un unique isolateur permettant de fabriquer les produits. A Tours, par exemple, il en existe trois qui travaillent en parallèle. En tout état de cause, il pourrait y en avoir plus.

Le groupe d'isolateurs voit arriver des séries de batches à réaliser. Ces batches sont ordonnancés sur les isolateurs de façon à minimiser le temps de traitement de tous les batches. Pour commencer le processus de fabrication d'un batch sur un isolateur, il est nécessaire de passer par une phase incontournable de stérilisation. Cette stérilisation est constante et indépendante de la taille et de la composition du batch. Le fait de réutiliser des matières premières n'a aucune influence également.
Chaque isolateur dispose de deux lignes de fabrication. Encore une fois, il pourrait y en avoir plus (et c'est le cas dans d'autres hôpitaux). Les lignes de fabrication sont indépendantes et fonctionnent également en parallèle.

Il est impossible d'interrompre un batch et d'ouvrir l'isolateur (outre les conséquences sanitaires, cela nécessiterait une nouvelle stérilisation et donc une perte de temps). De plus, aucun batch ne peut être lancé sur un isolateur tant que les paniers (conteneurs physiques pour les matières) du batch précédent n'ont pas été retirés. Il est donc impossible par exemple d'anticiper la phase de stérilisation.

Contrôle

Tout produit terminé doit passer par la phase de contrôle. Il n'existe qu'une machine permettant de réaliser ce contrôle à Tours. Mais comme précédemment, on pourrait en retrouver plus. La phase de contrôle est constante et relativement plus courte que le processus de fabrication (ce qui explique qu'une machine suffise dans l'exemple tourangeau). On ne considère pas le cas où un produit pourrait être rejeté par le contrôle. Il n'y a aucun délai entre deux contrôles successifs.

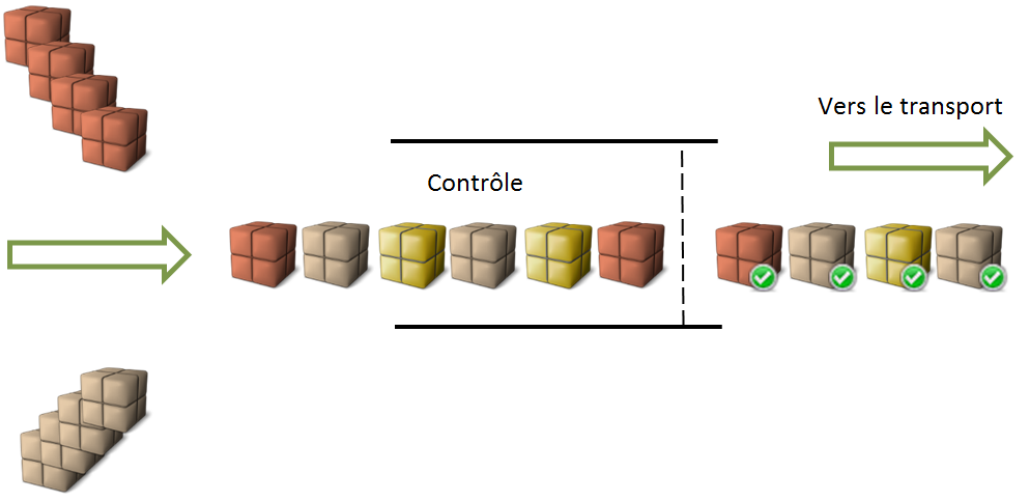

FIGURE 4.4 – Passage au contrôle

4.2.3 Les préparateurs

Les préparateurs sont chargés de fabriquer les produits sur les lignes de fabrication. Une ligne de fabrication nécessite un et un seul opérateur pour être opérationnelle. Le contrôle ne requiert pas de préparateur. Les préparateurs ne sont pas tous aussi expérimentés mais on considère grossièrement qu'ils travaillent tous à la même vitesse. Cependant, il faut considérer que les préparateurs ne sont pas disponibles en continu. Ils disposent de temps de repos (pause déjeuner, soir, nuit, matin) qui restent à définir.

4.3 Discernement de l'objectif

Une rapide analyse du système détaillé ci-dessus amène à la synthèse suivante de l'objectif de la solution à concevoir :

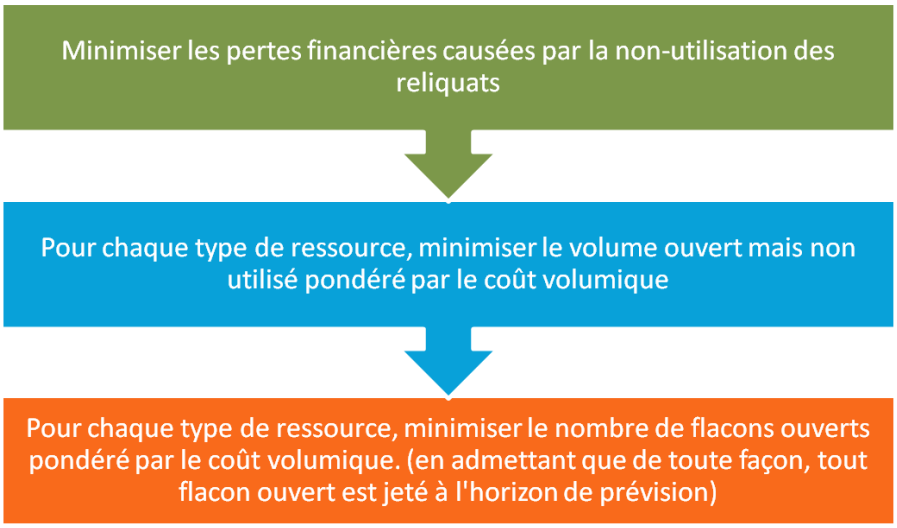

FIGURE 4.5 – Synthèse de la problématique

A ce stade, il est déjà possible de formaliser la plupart des paramètres relatifs au système évoqué.

Cependant, comme ceux-ci ont évolué et ont été adaptés avec chaque outil de résolution abordé, on préférera détailler l'ensemble des paramètres à chaque approche.

Modélisation sous forme linéaire

5.1 Justification de l'approche

5.1.1 Bref rappel

La modélisation sous forme linéaire d'un problème est, au sens de l'auteur et quand elle est possible, la plus simple et la meilleure façon d'aborder un problème dont la résolution optimale n'est pas polynomiale. Elle représente en tout cas une bonne entrée en matière.
Au travers de plusieurs cours et projets pédagogiques, il a été possible d'observer la puissance d'une telle approche, lorsqu'elle est associée à un solveur de qualité. Il en existe plusieurs qui sont pour la plupart payants. CPLEX, conçu par IBM, en est un bon exemple.
Pour ce projet, les tests effectués ont été réalisés avec GLPK, solveur gratuit disponible sous licence publique générale GNU. Il permet dans un premier temps d'observer à moindre coût l'efficacité du modèle conçu, laissant si besoin est l'opportunité d'investir ensuite dans un outil de meilleure qualité.

5.1.2 Les difficultés

Toute la difficulté de la programmation linéaire réside dans la bonne formulation des variables et des contraintes du modèle construit. Il n'existe pas vraiment, au sens de l'auteur, de règles strictes pour formaliser ces éléments. Tout juste peut-on reconnaître un ensemble de bonnes pratiques et des patterns basiques. Plus encore, il est des problèmes qui ne peuvent en aucun cas se formaliser sous forme linéaire. Mais bien sûr, la frontière est parfois floue entre manque d'ingéniosité et excès de complexité du problème.

Seule l'expérience et le talent permettent de construire un modèle efficace et en ce sens, la modélisation s'apparente en quelque sorte à un art. Tout ceci pour en arriver au fait que la conclusion sur l'utilisation de la programmation linéaire pour le projet qui nous occupe ne relève pas d'une vérité absolue mais des estimations de l'auteur du documents après plusieurs semaines de travail. Il appartient ensuite au lecteur de se forger sa propre opinion, en fonction du crédit qu'il apporte à l'auteur de ce document.

5.2 Le modèle mis au point

5.2.1 Préambule

La présente section montre le résultat de l'exploration de la programmation linéaire. Il est nécessaire d'appuyer le fait que cette version n'a pas abouti de part sa complexité. elle est ici exposée à titre d'exemple et éventuellement avec l'objectif de servir de base à une nouvelle recherche.
En aucun cas il n'est affirmé que cette version est aboutie ou exempte de toute erreur.

5.2.2 Présentation des paramètres du problème

Paramètres relatifs aux matières premières

- $nbMat$: nombre de matières première.
- $w_k, k \in [1..nbMat]$: poids de la perte d'une unité de produit k.
- $v_k, k \in [1..nbMat]$: volume d'un flacon de produit k.

– $f_k, k \in [1..nbMat]$: durée autorisée de conservation après ouverture d'un flacon de produit k.

Paramètres relatifs aux préparations (jobs)

– $nbJobs$: le nombre de jobs.
– $p_i, i \in [1..nbJobs]$: durée du job i.
– $s_i, i \in [1..nbJobs]$: durée de transport du job i à destination.
– $r_i, i \in [1..nbJobs]$: date de disponibilité du job i.
– $d_i, i \in [1..nbJobs]$: date due du job i.
– $g_i, i \in [1..nbJobs]$: durée de conservation du job i une fois fini.
– $b_{ki}, k \in [1..nbMat], i \in [1..nbJobs]$: quantité requise du produit k pour réaliser le job i.

Paramètres généraux

– $nbBatchs$: le nombre maximum de batchs (Dans la pratique, ce paramètre est égal au nombre de jobs, mais on pourrait forcer le regroupement avec une valeur plus petite).
– $maxPerBatch$: le nombre maximum de jobs par batch.
– p_{ste} : durée de la stérilisation.
– p_{ctrl} : durée de contrôle d'un produit fini.
– P_{inj} : durée d'injection d'un produit fini.
– H : tolérance au retard de livraison des jobs.
– HV : Grande valeur utilisée pour annuler certaines contraintes.

5.2.3 Présentation des variables du modèle

– $xb_{ib}, i \in [1..nbJobs], b \in [1..nbBatchs], binaire$: 1 si le job i appartient au batch b, 0 sinon.
– $xm_{im}, i \in [1..nbJobs], m \in [1..2], binaire$: 1 si le job i est sur la machine m, 0 sinon.
– $xj_{ij}, i \in [1..nbJobs], j \in [1..maxPerBatch], binaire$: 1 si le job i est à la position j sur sa machine, 0 sinon.
– $xc_{ij}, i \in [1..nbJobs], j \in [1..nbJobs], binaire$: 1 si le job i est en position j au contrôle, 0 sinon.
– $u_{kb}, k \in [1..nbMat], b \in [1..nbBatchs]$: nombre de flacons de produit k ouvert par le batch b.
– $reused_{kbl}, k \in [1..nbMat], b \in [1..nbBatchs]$: sous-quantité de produit de type k ouverte durant le batch b et réutilisée durant le batch l ultérieur a b (seulement si is_reused $= 1$).
– $is_reused_{kbl}, k \in [1..nbMat], b \in [1..nbBatchs]$: indique si on réutilise une sous quantité de produit k ouverte en b durant le batch l.
– $ifReusedThenQuantity_{kbl}, k \in [1..nbMat], b \in [1..nbBatchs]$: Variable déduite, juste pour faciliter la lecture (et je pense le calcul) : vaut $reused_{kbl}$ si $is\_reused_{kbl} = 1$, -HV sinon.
– $D_b, b \in [1..nbBatchs]$: date de début du batch b (avant stérilisation).
– $C_b, b \in [1..nbBatchs]$: date de fin du batch b.
– $Cxb_i, i \in [1..nbJobs]$: date de fin du job i dans son batch.
– $Cxc_i, i \in [1..nbJobs]$: date de fin du job i au contrôle.
– $Cxa_i, i \in [1..nbJobs]$: Variable déduite, juste pour faciliter la lecture (et je pense le calcul), date d'administration du job i.

5.2.4 Contraintes sur la formation des batchs

Tous les jobs sont assignés à un unique batch

$$\sum_{b}^{nbBatchs} xb_{ib} = 1, \forall i \in [1..nbJobs]$$

Tous les jobs sont assignés à une unique machine dans un batch.

$$\sum_{m}^{1..2} xm_{im} = 1, \forall i \in [1..nbJobs]$$

Tous les jobs sont assignés à position sur une machine dans un batch.

$$\sum_{j}^{maxPerBatch} xj_{ij} = 1, \forall i \in [1..nbJobs]$$

Les jobs du batch doivent être disponibles avant d'être lancés.

$$r_i \leq D_b + r_i \times (1 - xb_{ib}), \forall i \in [1..nbJobs], b \in [1..nbBatchs]$$

La taille d'un batch ne peut excéder la taille maximale autorisée.

$$\sum_{i}^{nbJobs} xb_{ib} \leq maxPerBatch, \forall b \in [1..nbBatchs]$$

La fin du batch b est égale à la plus grande fin de ses composantes.

$$C_b \geq Cxb_i - HV \times (1 - xb_{ib}), \forall i \in [1..nbJobs], b \in [1..nbBatchs]$$

Le premier job sur la machine 1 et la machine 2 à une date de fin (relative au début) égale à sa durée.

$$Cxb_i \geq D_b + p_i - HV \times (1 - xb_{ib}), \forall i \in [1..nbJobs], b \in [1..nbBatchs]$$

La tache en position j sur la machine m se termine après la tache en position j - 1 + sa propre durée.

$$\forall i \in [1..nbJobs - 1], b \in [1..nbBatchs], m \in [1..2], j \in [2..maxPerBatch], l \in [1..nbJobs]$$

$$Cxb_i \geq Cxb_l + p_i - HV \times (6 - xb_{ib} - xb_{lb} - xm_{im} - xm_{lm} - xj_{ij} - xj_{l(j-1)})$$

La date de début du batch b est supérieur ou égale à la date de fin du batch précédent sur la même machine + la durée de stérilisation.

Les isolateurs peuvent être considérés comme des machines parallèles. Les éléments d'index pair concernent le premier isolateur et les éléments d'index impair concernent le second isolateur.

$$D_b \geq C_{b-2}, \forall b \in [3..nbBatchs]$$

5.2.5 Contraintes sur le passage des jobs au contrôle

Tous les jobs passent au contrôle à une position différente.

$$\sum_{j}^{nbJobs} xc_{ij} = 1, \forall i \in [1..nbJobs]$$

Toutes les positions de contrôle sont occupées.

$$\sum_{i}^{nbJobs} xc_{ij} = 1, \forall j \in [1..nbJobs]$$

Un job passe après son prédécesseur sur le contrôle.

$$\forall (i,l) \in [1..nbJobs], j \in [1..nbJobs], l \neq i$$
$$Cxc_i \geq Cxc_l + p_{ctrl} - HV \times (2 - xc_{ij} - xc_{l(j-1)})$$

Un job termine le contrôle au moins après qu'il soit passé au stérilisateur et par la phase de contrôle.

$$Cxc_i \geq Cxb_i + p_{ctrl}, \forall i \in [1..nbJobs]$$

5.2.6 Contraintes sur l'administration des jobs

La date d'administration d'un job est supérieure ou égale à la fin de son contrôle plus le transport plus l'administration.

$$Cxa_i \geq Cxc_i + p_{inj} + s_i$$

La date d'administration d'un job est inférieure à la date de fin de fraicheur du produit.

$$Cxa_i \leq Cxb_i + g_i$$

La date d'administration d'un job est inférieure à la date de livraison plus le retard toléré.

$$Cxa_i \leq d_i + H$$

5.2.7 Contraintes sur les reliquats

Définition de la variable *ifReusedThenQuantity*.

Définit la quantité de produit k ouvert en b et réutilisé en j SI on choisit de le réutiliser à ce moment là. Sinon, vaut une valeur négative permettant d'annihiler les contraintes de date due. Ce faisant, il est possible que le solveur affecte à la variable $reused_{kbj}$ une valeur supérieure à zéro même si le produit n'est

pas réutilisé. Mais il n'y trouve pas d'intérêt car les contraintes vont le pousser à économiser ce qui peut l'être pour ne pas ouvrir de nouvelles bouteilles.

$$\forall b \in [1..nbBatchs-1], j \in [b+1..nbBatchs], k \in [1..nbMat]$$
$$ifReusedThenQuantity_{kbj} = reused_{kbj} - HV \times (1 - is\_reused_{kbj})$$

La quantité de produit k ouverte en b et réutilisée en j est positive seulement si le batch b est terminé et que le produit est utilisé.

$$\forall b \in [1..nbBatchs-1], j \in [b+1..nbBatchs], k \in [1..nbMat]$$
$$(D_j - C_b) \times v_k \geq ifReusedThenQuantity_{kbj}$$

La quantité de produit k ouverte en b et réutilisée en j est positive seulement si le produit est toujours frais et que le produit est utilisé.

$$\forall b \in [1..nbBatchs-1], j \in [b+1..nbBatchs], k \in [1..nbMat]$$
$$(D_b + p_{ste} + f_k - C_j) \times v_k \geq ifReusedThenQuantity_{kbj}$$

La somme des quantités de produit k ouverte au batch 1 et réutilisées ensuite est inférieure à la quantité épargnée à ce batch. L'épargne à ce batch est égale à la quantité ouverte - la quantité requise.

$$\sum_{b}^{2..nbBatchs} reused_{k1b} \leq u_{k1} \times v_k - \sum_{i}^{nbJobs} (b_{ki} \times xb_{i1}), \forall k \in [1..nbMat]$$

La somme des quantités de produit k ouverte en b et réutilisées ultérieurement est inférieure à la quantité épargnée en b. L'épargne à chaque batch est égale à la ouverte -(la quantité requise - la quantité réutilisée).

$$\sum_{c}^{b+1..nbBatchs} reused_{kbc} \leq u_{kb} \times v_k - \sum_{i}^{nbJobs} (b_{ki} \times xb_{ib}) - \sum_{l}^{1..b-1} reused_{klb}$$
$$\forall k \in [1..nbMat], b \in [2..nbBatch-1]$$

Le volume ouvert correspond au besoin - la quantité réutilisable.

$$u_{kb} \times v_k \geq \sum_{i}^{nbJobs} (b_{ki} \times xb_{ib}) - \sum_{l}^{1..b-1} reused_{klb}$$
$$\forall k \in [1..nbMat], b \in [1..nbBatchs]$$

5.2.8 Définition de l'objectif

On minimise la somme des volumes pondérés des flacons de matières ouverts.

$$minimize \sum_{b}^{nbBatchs} \sum_{k}^{nbMat} u_{kb} \times w_k$$

5.3 Limites et conclusion sur une solution linéaire

5.3.1 Observation des tests effectués

Le jugement est sans appel. Si le modèle conçu a pu être lancé, les résultats ne sont pas bons. Au bout de 3 minutes, le solveur commençait tout juste à trouver une première solution, qui s'avérait loin d'être bonne. Pour rappel, le logiciel PLANIF dispose de 5 minutes pour s'exécuter dans son intégralité. Les résultats ne sont donc pas satisfaisants.

5.3.2 Critique du modèle

Même si la représentation sous forme linéaire a pu être atteinte, le modèle souffre de plusieurs maux. Une analyse rapide permet de comprendre ceci. Les critiques se rassemblent en trois points :

1. La multiplicité d'utilisation de chaque contrainte, définie le plus souvent pour au moins deux paramètres, avec même une contrainte (ordre des tâches sur une machine) définie pour 5 paramètres différents.
2. L'utilisation massive d'une grande constante (HV) pour annuler les contraintes indésirables. Ce genre de pratique n'est pas très apprécié par le solveur et devrait rester marginal. Sans doute ne l'est-il pas suffisamment ici.
3. Dans les dernières contraintes évoquées, les sommes sont utilisées de façon récurrente. il est évident que la somme d'un produit est longue à calculer, surtout, si elle se répète un grand nombre de fois.

Modélisation par contraintes

6.1 Justification de l'approche

6.1.1 Les raisons de l'échec de la formalisation linéaire

On l'a vu dans le chapitre précédent, la formalisation sous l'aspect d'un modèle linéaire est difficilement abordable. Cela vient en partie du fait que nombre de contraintes exprimées ne sont pas linéaires et la tentative de les rendre linéaire aboutit à un modèle estropié qui ne mérite même pas une place handicapé sur un parking public.
Cependant, à ce stade, une compréhension affinée du modèle a permis à l'étudiant et son encadrant de souligner un fait important. Malgré la complexité du problème, il se trouve que l'ordonnancement est tout de même grandement contraint par les règles imposées (satisfaction du patient, durée de vie des produits, taille des batches...). Dans ces conditions, il se pourrait que, là où la programmation linéaire a échoué, la programmation par contrainte s'avère intéressante.

6.1.2 Bref rappel

La programmation par contraintes ne repose pas sur la méthode du simplexe comme c'est le cas pour la programmation linéaire. Au lieu de cela, pour chaque variable du problème, le programme conçoit un ensemble des valeurs possibles pour chaque variable et réduit cet ensemble au cours de l'exploration en appliquant les contraintes spécifiées. Si les contraintes sont suffisamment restrictives, et ce même si elles ne sont pas linéaires, il peut arriver que le solveur obtienne d'excellents résultats.
Le principal avantage est que cet outil permet l'utilisation d'opérateurs ensemblistes ainsi que quelques fonctions non linéaires bien utiles, comme le $\min$ et le $\max$. Dans ces conditions, la plupart des problèmes deviennent formalisables. Reste que comme pour la modélisation linéaire, l'efficacité du modèle repose avant tout sur l'ingéniosité du modélisateur.

6.2 Le modèle mis au point

6.2.1 Préambule

Encore une fois, le modèle suivant ne peut être considéré comme abouti. Cependant l'étudiant est moins catégorique quant à son inefficacité. Il reste persuadé qu'un peu plus d'intuition et d'expérience que celles dont il a fait preuve pourraient aboutir à des conclusions très prometteuses.

6.2.2 Présentation des paramètres du problème

Paramètres relatifs aux matières premières

- $nbMat$: nombre de matières première.
- $w_k, k \in [1..nbMat]$: poids de la perte d'une unité de produit k.
- $v_k, k \in [1..nbMat]$: volume d'un flacon de produit k.
- $f_k, k \in [1..nbMat]$: durée autorisée de conservation après ouverture d'un flacon de produit k.

Paramètres relatifs aux préparations (jobs)

- $nbJobs$: le nombre de jobs.
- $p_i, i \in [1..nbJobs]$: durée du job i.
- $s_i, i \in [1..nbJobs]$: durée de transport du job i à destination.
- $r_i, i \in [1..nbJobs]$: date de disponibilité du job i.
- $d_i, i \in [1..nbJobs]$: date due du job i.
- $g_i, i \in [1..nbJobs]$: durée de conservation du job i une fois fini.
- $b_{ki}, k \in [1..nbMat], i \in [1..nbJobs]$: quantité requise du produit k pour réaliser le job i.

Paramètres généraux

- MPB : le nombre maximum de jobs par batch.
- $pSte$: durée de la stérilisation.
- $pCtrl$: durée de contrôle d'un produit fini.
- $pInj$: durée d'injection d'un produit fini.
- H : tolérance au retard de livraison des jobs.

6.2.3 Présentation des variables du modèle

Variables portant sur les jobs

- $xb_i, i \in [1..nbJobs], D_f = [1, nbJobs]$: numéro du batch auquel appartient le job i.
- $xm_i, i \in [1..nbJobs], D_f = [1, 2]$: numéro de la machine sur laquelle est placé le job i au sein de son batch.
- $xj_i, i \in [1..nbJobs], D_f = [1, MPB]$: position du job i sur sa machine dans son batch.
- $xc_i, i \in [1..nbJobs], D_f = [1, nbJobs]$: position de passage du job i au contrôle parmi les autres jobs.
- $Ciso_i, i \in [1..nbJobs], D_f = [r_i + p_i, \infty]$: date de fin du job i dans l'isolateur
- $Cctrl_i, i \in [1..nbJobs], D_f = [r_i + p_i, \infty]$: date de fin du job i au contrôle.
- $Cinj_i, i \in [1..nbJobs], D_f = [r_i + p_i, \infty]$: date d'administration du job i.

Variables portant sur les batchs

- $S_b = \{i | xb_i = b\}, \forall b \in [1..nbJobs]$: ensemble déduit des jobs appartenant au batch. Réécriture pratique pour la définition de contraintes.
- $bb_{kb} = \sum_{i \in S_b} b_{ki}, \forall b \in [1..nbJobs], k \in [1..nbMat]$: Déduction de la quantité requise de matière k pour le batch b.
- $D_b, b \in [1..nbJobs], D_f = [0, \infty]$: date de début du batch b **avant stérilisation**.
- $Cb_b, b \in [1..nbJobs], D_f = [pSte, \infty]$: date de fin du batch b.
- $u_{bk}, b \in [1..nbJobs], k \in [1..nbMat], D_f = [0, \infty]$: nombre de flacons de matière k ouverts durant le batch b.
- $spared_{kb}, k \in [1..nbMat], b \in [1..nbJobs], D_f = [0, v_k]$: volume de matière première k épargnée (ouverte mais non consommée) durant le batch b.
- $reused_{klb}, k \in [1..nbMat], l \in [1..nbJobs], b \in [1..nbJobs], D_f = [0, v_k]$: volume de matière première k épargnée (ouverte mais non consommée) durant le batch l et réutilisée en b.

6.2.4 Contraintes sur la formation des batchs

La taille d'un batch ne peut excéder la taille maximale autorisée.

$$|S_b| \leq MPB, \forall b \in [1..nbJobs]$$

Les jobs du batch doivent être disponibles avant d'être lancés.

$$D_b \geq max(r_i), \forall b \in [1..nbJobs], i \in S_b$$

La fin du batch i est égale à la plus grande fin de ses composantes.

$$Cb_b = max(Ciso_i), \forall b \in [1..nbJobs], i \in S_b$$

Deux taches de même batch et de même machine ne peuvent être superposés.

$$\forall i \in [1..nbJobs - 1], j \in [i + 1..nbJobs]$$
$$(xb_i = xb_j) \wedge (xm_i = xm_j) \implies (Ciso_i \geq Ciso_j + p_i) \vee (Ciso_j \geq Ciso_i + p_j)$$

Un batch ne peut pas être lancé si un autre batch est en cours d'exécution.

$$D_b \geq Cb_{b-2}, \forall b \in [1..nbJobs]$$

6.2.5 Contraintes sur le passage des jobs au contrôle

Tous les jobs passent au contrôle à une position différente.

$$xc_i \neq xc_j, \forall (i, j) \in [1..nbJobs], i \neq j$$

Un job passe après son prédécesseur sur le contrôle.

$$xc_i \geq xc_j \Leftrightarrow Cctrl_i \geq Cctrl_j + pCtrl, \forall (i, j) \in [1..nbJobs], i \neq j$$

Un job termine le contrôle au moins après qu'il soit passé au stérilisateur et par la phase de contrôle.

$$Cctrl_i \geq Ciso_i + pCtrl, \forall i \in [1..nbJobs]$$

6.2.6 Contraintes sur l'administration des jobs

La date d'administration d'un job est supérieure ou égale à la fin de son contrôle plus le transport plus l'administration.

$$Cinj_i \geq Cctrl_i + pInj + s_i$$

La date d'administration d'un job est inférieure à la date de fin de fraicheur du produit.

$$Cinj_i \leq Ciso_i + g_i$$

La date d'administration d'un job est inférieure à la date de livraison plus le retard toléré.

$$Cinj_i \leq d_i + H$$

6.2.7 Contraintes sur les reliquats

Le volume ouvert correspond au besoin moins la quantité réutilisée.

$$u_{bk}v_k \geq bb_{bk} - \sum_{l=1}^{b-1} reused_{blk}, \forall b \in [1..nbJobs], k \in [1..nbMat]$$

La quantité épargnée à chaque batch est le volume ouvert moins la quantité utilisée (elle même étant le besoin moins la réutilisation).

$$spared_{kb} = u_{bk}v_k - (bb_{bk} - \sum_{l=1}^{b-1} reused_{blk}, \forall b \in [1..nbJobs], k \in [1..nbMat])$$

La somme des quantités de produit k ouverte en b et réutilisées ultérieurement est inférieure à la quantité épargnée en b.

$$spared_{kb} \geq \sum_{l=1}^{b-1} reused_{blk}, \forall b \in [1..nbJobs], k \in [1..nbMat]$$

Il est impossible de réutiliser en b de la matière k d'un batch l non terminé au début de b.

$$D_b \leq Cb_l \implies reused_{klb} = 0, \forall l \in [1..nbJobs], \forall b \in [l+1..nbJobs], k \in [1..nbMat]$$

Il est impossible de réutiliser en b de la matière k d'un batch l si cette matière périme avant la fin du batch b.

$$Cb_b \geq D_l + pSte + f_k \implies reused_{klb} = 0, \forall l \in [1..nbJobs], \forall b \in [l+1..nbJobs], k \in [1..nbMat]$$

6.2.8 Définition de l'objectif

On minimise la somme des volumes pondérés des flacons de matières ouverts.

$$minimize \sum_{b=1}^{nbJobs} \sum_{k=1}^{nbMat} u_{bk}w_k$$

6.3 Limites et conclusion sur une solution par contraintes

La conclusion tirée ici est très différente. Il s'avère que le modèle n'a jamais pu être exécuté convenablement. Probablement que les contraintes sont mal exprimées, ceci étant dû à l'inexpérience de l'étudiant en la matière. Impossible de dire donc si le modèle est réellement inefficace. La décision de l'abandonner a été motivée par une incapacité prolongée à l'améliorer. Toutefois, la piste mériterait d'être explorée de nouveau.
On peut tout de même exprimer une critique à l'issue de l'exercice. Cette critique porte sur le processus d'attribution des flacons aux jobs. Cette attribution peut se faire de deux façons différentes et aucune n'apparaît adaptée à la programmation par contrainte :

- Soit on laisse le solveur décider, en définissant une variable décisionnelle pour l'attribution de chaque flacon à un job ou non. Cette façon est mauvaise parce que le choix est en fait simple : Soit il y a des flacons ouverts et non péris que l'on peut réutiliser et auquel cas on utilise celui qui périme au plus tôt, soit il n'y en a pas et on ouvre un nouveau flacon (ils sont de tailles quasi équivalentes donc pas de choix).
  S'en remettre au solveur relève du gaspillage. le problème est déjà assez complexe pour qu'on sacrifie du temps à se consacrer de façon brute à un problème polynomial.
- Soit on essaie d'appliquer le principe évoqué dans le point précédent. C'est ce qui a été essayé dans le modèle ci-dessus. Mais le problème dans ce cas est le suivant : cette construction est itérative. C'est-à-dire qu'il faut créer les batchs dans l'ordre de leur exécution, sinon comment savoir quels sont les flacons ouverts à un instant t et ceux qui ne le sont pas ?
  Hors le solveur par contraintes ne procède pas du tout de façon temporelle. Cette notion lui est d'ailleurs inconnue. tous les problèmes ne sont pas temporels. Ceci explique peut-être pourquoi le modèle n'a jamais pu être exécuté convenablement. Il est possible néanmoins de donner au solveur une stratégie d'exploration. Malheureusement, cela dépasse les capacités de l'étudiant.

Troisième partie

Mise en place d'une solution adaptée

Tactique et outils sélectionnés

7.1 Conclusion sur les méthodes exactes

Conformément aux attentes, il s'avère extrêmement difficile de construire une solution optimale ou tout du moins très bonne avec une méthode exacte. Il était néanmoins important, par souci de rigueur et pour améliorer la compréhension du problème, de montrer que ces méthodes s'avéraient inefficaces et pourquoi. Ainsi, la recherche n'aura pas été inutile. Elle a permis de mettre en évidence une certaine dualité de l'exercice de construction, qui donne toute sa difficulté et son piment à la réalisation d'une solution :

- Plusieurs sous-parties de la construction peuvent être effectuées de façon polynomiale avec de très bon résultats. On a déjà cité l'attribution des flacons dans le chapitre précédent. On peut aussi parler de la répartition des jobs au sein d'un batch. Dès lors que l'on a choisi les jobs du batch, les matières sont engagées et la répartition des jobs se fait très bien en les triant par date due et en les assignant à la machine de l'isolateur qui termine au plus tôt.
- Cependant, le problème total reste trop complexe pour être traité entièrement de façon polynomiale. Les paramètres à prendre en compte sont si nombreux qu'il apparait difficile de les gérer dans leur intégralité.

7.2 Le choix tactique

En suivant les conseils de son encadrant, l'étudiant s'est porté sur une construction en deux étapes :
- Réalisation d'une solution initiale obtenue par un algorithme polynomial qui reste à définir.
- Application d'une méta-heuristique pour améliorer la solution initiale. L'encadrant ayant un fort penchant pour la méthode taboue, c'est celle-ci qui a été choisie.

7.3 Le choix opérationnel

Même s'il n'est pas question de présenter un fragment de code dans ce rapport, il n'est pas non plus question de faire l'impasse sur le choix du langage et de l'outil de programmation. Vient un moment où un choix doit être fait et ce choix doit reposer sur une argumentation convaincante.
Cette argumentation repose sur 3 points :
- La puissance, rapidité du langage.
- La complexité, le confort de programmation avec le langage et l'outil.
- La maîtrise du langage et de l'outil par l'étudiant et l'encadrant.

7.3.1 Argumentation

Pour un programme effectuant un calcul scientifique, la puissance est un argument de poids. Lorsque les fonctions principales d'un programme sont exécutées des centaines de millions de fois, il est nécessaire que ces fonctions soient optimisées au maximum.
Langage de bas niveau et complexité de programmation sont assez contradictoires. Forcément, ces langages, croisés avec un manque de rigueur, aboutissent à un plus grand nombre d'erreurs. En particulier, la difficulté de rattrapage d'une erreur est difficile : Si le programme utilise de l'aléatoire et que les erreurs surviennent

à la 32000ème itération de la fonction, le débuggage va être plus compliqué. A noter que dans ce cas l'IDE a également son rôle à jouer.
En vertu de ces arguments, on penche pour l'utilisation d'un langage de bas niveau dans lequel l'allocation mémoire est gérée par l'utilisateur. Les langages qui viennent immédiatement à l'esprit sont le C et le C++. Il en existe d'autre, mais on restera conventionnel, d'une part en raison des connaissances de l'étudiant, d'autre part dans un souci de faciliter la réutilisation du projet.
Suivant les préférences de l'étudiant et celles de son encadrant, il est convenu d'utiliser le C. L'étudiant s'appuiera sur Visual Studio pour le développement pour trois raisons principales :

- Forte maîtrise de l'outil.
- La documentation écrite s'affiche tout au long du développement pour aider l'écriture du code.
- Visual Studio comporte des outils de profilage qui permettront en fin de parcours d'identifier les fonctions critiques et d'orienter le processus d'optimisation du code.

Conception de l'algorithme

8.1 Principe

La construction de l'heuristique présentée dans ce document se fait donc par étapes :

1. Recherche d'un codage minimal de la représentation d'une solution.
2. Élaboration d'un algorithme permettant de reconstituer la solution complète à partir du codage.
3. Utilisation d'un algorithme glouton permettant d'obtenir une solution initiale réalisable en temps réduit.
4. Application d'une recherche locale (méta-heuristique tabou) pour améliorer cette solution initiale.

Les prochaines sections s'attardent à détailler chacune de ces étapes.

8.2 Codage de la solution

8.2.1 Pourquoi un codage ?

Lorsque l'on souhaite utiliser une heuristique chargée de générer des voisins à partir d'une solution initiale, il apparaît intéressant que cette solution puisse être représentée de façon minimaliste. Cette stratégie facilite les opérations de voisinages.
L'intérêt d'un codage est aussi la comparaison de l'égalité entre deux solutions. Si le codage choisi est simple, la comparaison est rapide. On pourrait ajouter que nombre de tests élémentaires de faisabilité peuvent être fait directement sur le codage, ce qui accélère le calcul (si le codage est simple, encore une fois).

8.2.2 Codage adopté

Conjointement avec l'encadrant, l'étudiant a choisi de définir le codage de la solution de la façon suivante. Une solution au problème posé se définit comme l'agrégation des codages des jobs qu'elle contient. Pour chaque job dans l'ordonnancement, on doit au moins préciser :

- M_i : la machine (et par déduction l'isolateur) sur laquelle est placé le job.
- B_i : numéro du batch du job sur l'isolateur.
- K_i : position du job dans son batch sur sa machine.
- C_i : position de passage du job au contrôle.
- L_i : tableau de taille fixe contenant les numéros des flacons de matière à utiliser (ou réutiliser) pour le job. Une valeur négative indiquant l'absence de flacon.

En supposant qu'on fixe le nombre de flacons maximum par job à m, le codage d'un job est donc :

$$[M_i, B_i, K_i, C_i, L_{i1}, L_{i2} \ldots, L_{im}]$$

Et le codage d'une solution :

$$[M_1, B_1, K_1, C_1, L_{11}, L_{12}, \ldots, L_{1m}] \ldots [M_n, B_n, K_n, C_n, L_{n1}, L_{n2}, \ldots, L_{nm}]$$

8.3 Reconstitution de la solution

Le codage décrit ci-dessus contient les informations minimales permettant de représenter sans ambiguïté une solution. Il ne fournit cependant aucune information dérivée comme les dates de début et de fin des jobs ou la consommation totale en matières premières.
Pour obtenir ces informations, il est nécessaire d'effectuer une reconstruction qui doit permettre :

- D'estimer la faisabilité de la solution.
- D'obtenir toutes les informations dérivées utiles en sortie.
- D'établir la qualité (*fitness*) de cette solution.

Cette reconstitution à lieu en deux grandes étapes :

1. On reconstruit l'ordonnancement total.
   (a) Des structures sont crées pour représenter les batchs par isolateur. Tous les codages de jobs sont parcourus et assignés à leur batch à la position adéquate.
   (b) Les batchs sont triés par position sur leur isolateur puis, à l'aide des paramètres sur les jobs et la disponibilité des opérateurs, les dates de débuts et de fins des jobs et de leur batchs sont calculées.
   (c) Grâce aux dates de fin des jobs et de leur ordre de passage au contrôle, on calcule les dates de fin de contrôle des jobs, on détermine le nombre de jobs en retard et le retard maximum.
2. On reconstruit la consommation totale.
   (a) Une structure contenant tous les flacons disponibles avec leur volumes est initialisée. Ce "frigo" gère les demandes en matières et conserve les dates de péremption dynamiques des produits ainsi que des verrous d'utilisation pour éviter les utilisations d'un même flacon par deux batchs concurrents.
   (b) Les batchs sont triés par date de début puis chaque job de chaque batch récupère dans le frigo, dans un ordre prédéfini, la quantité qui lui est nécessaire en fonction des flacons qui lui sont assignés. A la fin de cette étape, le frigo contient des informations sur les matières comme la quantité ouverte ou périmée.
   Il est nécessaire d'insister sur la délicatesse de cette étape. La chronologie de récupération dans le frigo est extrêmement importante et doit toujours être la même. Car la quantité que chaque job prélève dans un flacon n'est pas inscrite dans le codage de la solution. Autrement dit, si les jobs ne récupèrent pas toujours dans le même ordre les matières de leur flacon, il est possible d'arriver à des résultats dissemblables voire parfois infaisables.

8.3.1 La faisabilité

Une fois la solution reconstituée, il est possible d'évaluer si elle est faisable ou non. En pratique, le test de faisabilité est parfois effectué en parallèle et la reconstitution peut ne pas s'achever si elle est rendue impossible par des données incohérentes.
Les tests suivants sont effectués pour chaque job :

Tests élémentaires

Ces tests ne nécessitent pas la reconstruction de la solution.

- Vérification de l'index de batch (doit être positif et inférieur à une constante à définir).
- Vérification de l'index de position au contrôle (doit être positif et inférieur ou égal au nombre de jobs).
- Vérification de l'index de position sur la machine (doit être positif et inférieur ou égal au nombre maximum de jobs par batch).

- Vérification de l'index de machine (doit être positif et inférieur ou égal au nombre total de machines disponibles).
- Vérification de l'index des flacons attribués (doivent être positif et inférieur ou égal au nombre de flacons par matière).
- Vérification de l'unicité d'attribution d'une position. Le triplet (M_i, B_i, K_i) permet d'identifier uniquement une position particulière dans l'ordonnancement. Cette position ne doit être attribuée qu'une fois au plus.

Tests avancés

Ces tests ne peuvent être effectués qu'après reconstitution de la solution.

- Vérification de la taille des batchs (doit être inférieure au nombre maximum de jobs par batch).
- Vérification que la demande en matière de chaque job est satisfaite par les flacons qui lui sont assignés (les flacons doivent avoir un volume restant suffisant au moment du prélèvement).
- Vérification de l'utilisation de flacons non périmés au moment du prélèvement.
- Vérification de l'utilisation de flacons disponibles (non verrouillés par un batch concurrent) au moment du prélèvement.

A savoir que dépasser la tolérance maximum au retard n'entraine pas l'invalidité de la solution. Cela a néanmoins des conséquences sur le *fitness* de cette même solution.

8.3.2 Évaluation de la qualité

La qualité de la solution est assez difficile à évaluer car elle repose sur deux critères :

- La satisfaction de la contrainte temporelle (retard max, nombre de jobs en retard).
- La minimisation de la quantité de matière première perdue (ouverte mais non utilisée).

La difficulté d'estimation de tels critères est souvent délicate. Au-delà de refléter la seule qualité d'une solution, ils doivent permettre de "guider" un algorithme de recherche locale dans ses choix. C'est-à-dire qu'il doivent récompenser les opérations qui satisfont au plus les attentes de l'utilisateur final.

Le critère matériel

Ce critère est le plus simple à établir. On choisit de le définir comme le taux de matière gaspillée :

$$MF = 1 - \frac{requiredValue}{openedValue}$$

Avec $requiredValue$ La valeur totale minimale requise par les jobs et $openedValue$ la valeur réellement ouverte pour ces jobs. L'objectif est de minimiser ce critère.

Le critère d'ordonnancement

Ce critère plus compliqué. Il doit prendre en compte le plus grand retard ainsi que le nombre de jobs en retard. De plus, l'hôpital a indiqué qu'il était plus important de réduire le retard maximum que de réduire le nombre de retards.
Partant de ce principe, on a défini le critère comme étant :

$$SF = \alpha L_{max} + \beta \sum_{i=1}^{nbJobs} U_i$$

Avec L_{max} le retard max et $\sum U_i$ le nombre de jobs en retard. Encore une fois l'objectif est de minimiser ce critère.

Critère final

On évaluera globalement une solution par une combinaison linéaire de ces deux critères, sachant que la priorité est placée sur la satisfaction du critère ordonnancement :

$$F = SF + \gamma MF$$

Fixer les scalaires

Il ne reste plus qu'à définir la valeur des trois scalaires α, β, γ. Les valeurs ont été fixées de la façon suivante :

$$\alpha = 0.2, \beta = \alpha \frac{L^0_{max}}{\sum_{i=1}^{nbJobs} U^0_i}, \gamma = 1 - \beta$$

où L^0_{max} et $\sum_{i=1}^{nbJobs} U^0_i$ sont les valeurs de la solution initiale (solution 0).

Remarque

Le choix de ces fonctions d'évaluation est d'une grande importance. Leur modification peut entraîner un comportement radicalement différent de l'algorithme tabou.

8.4 L'algorithme glouton

L'objectif de cet algorithme est d'obtenir rapidement une solution faisable moyenne. Cette solution pourra ensuite être améliorée par la recherche locale.
L'algorithme se déroule globalement comme suit :

1. Construction des batchs :
   (a) On trie les jobs par date due. C'est dans cet ordre qu'on va progressivement les ajouter à la solution.
   (b) On prend l'isolateur disponible au plus tôt (en terme de fin de batch précédent et en terme de disponibilité des opérateurs). Si aucun isolateur n'est disponible à la date courante, on avance la date courante à la date de disponibilité au plus tôt du plus disponible des isolateurs.
   (c) S'il est possible de placer au moins un job avant la prochaine indisponibilité de la l'isolateur, on ajoute un batch sur cet isolateur puis dans ce batch on ajoute progressivement les jobs (disponibles selon r_i) de la liste triée tant que cela est possible. Si le batch est plein ou qu'aucune machine de l'isolateur ne dispose plus d'opérateur pour traiter le job à ajouter, on arrête l'insertion.
   (d) On calcule les dates de début et de fin des jobs du batch et celles du batch lui-même et on met à jour les dates courantes de l'isolateur. On reprend à l'étape (b) tant qu'il reste des jobs à placer.
2. Définition du passage au contrôle :
   (a) A l'aide des informations sur les jobs calculées précédemment, on effectue un tri des jobs sur leur date de fin à la sortie de leur batch respectif.
   (b) On assigne les positions de contrôle dans ce même ordre.
3. Attribution des flacons aux jobs :
   (a) On trie les batchs par date de début.
   (b) On trie les flacons du plus grand au plus petit pour chaque matière.

(c) Pour chaque job de chaque batch, on attribue les flacons non périmés et non réservés par un batch concurrent qui lui permettent de satisfaire son besoin. Dans un batch, les jobs sont évalués dans l'ordre de leur id. Il est extrêmement important que les jobs soient traités dans le même ordre à la construction et la vérification de la solution car on ne dispose pas d'informations sur les quantités que chaque job prélève dans ses flacons. C'est uniquement l'ordre de prélèvement qui définit cette valeur. Il est donc essentiel que cet ordre soit conservé, sous peine de réactions en chaîne qui pourraient invalider une solution normalement faisable.

(d) Une fois les besoins de tous les jobs satisfaits, on remplace dans le codage des jobs les flacons ouverts mais non consommés totalement par des flacons de moindre contenance satisfaisant également le besoin (diminuant ainsi la matière perdue).

La solution primaire obtenue passe par une reconstruction et un test de faisabilité évoqué précédemment pour vérifier sa validité. Sa qualité est moyenne mais sa logique globale de construction lui fournit un bon potentiel d'amélioration, notamment parce que les jobs sont insérés en suivant *EDD*.

8.5 La recherche locale Tabou

8.5.1 Principe général

Le principe de la recherche Tabou est simple et éprouvé :

1. A partir d'une solution donnée et d'un opérateur de voisinage, on va construire un ensemble de solutions voisines (parfois toutes) dont on évalue la qualité (*fitness*).
2. La nouvelle solution courante est choisie parmi ces voisins comme étant celle ayant la meilleure qualité. L'ancienne solution est ensuite ajoutée à une file de taille fixe appelée "liste taboue".
3. On recommence les étapes précédentes mais en aucun cas on ne saurait choisir comme nouvelle solution courante une solution contenue dans la liste tabou. De cette façon, on se protège contre le bouclage dans des minimas locaux.

Les critères d'arrêts sont multiples et à définir (temps limite de recherche, nombre maximum d'itération avant amélioration de la meilleure solution...)

8.5.2 Adaptation

Pour les besoins du problème, il est nécessaire de fixer les paramètres globaux de l'algorithme.

Nombres de voisins explorés

Du fait du trop grand nombre de voisins d'une solution, on se propose de n'en construire que 2000 [1] à chaque itération. Cependant, pour cette même raison, on conservera 3 solutions dans la liste tabou seulement (peu de chance qu'on revienne sur une exploration similaire avec autant de voisins).
Il va de soi que le nombre de voisins pouvant être explorés en temps raisonnable à chaque itération est fortement corrélé à la taille d'une solution. C'est-à-dire que si l'horizon de planification change et passe de deux jours à 1 mois, il sera nécessaire de revoir à la baisse ce paramètre.

Conditions d'arrêt

On arrêtera la recherche au bout de 5 minutes ou si 200 itérations infructueuses (sans amélioration de la meilleure solution) sont effectuées. Les 5 minutes semblent convenables pour une exploration de sorte que l'amélioration nous a semblé très faible au delà de ce délai dans les tests effectués. Les 200 itérations permettent de sortir des minimas locaux pour explorer de meilleures solutions.

1. Ceci était vrai pour les premiers tests avec un horizon de 2 jours. Pour un horizon de 30 jours, on se restreint à 200 voisins.

Opérateur de voisinage

L'opérateur de voisinage choisi est un SWAP aléatoire. Il consiste en la permutation des positions (M_i, B_i, K_i, C_i) de deux jobs choisis aléatoirement. Ce choix est fait après élimination d'un SWAP dirigé qui n'apportait pas de meilleure solution et de tests sur l'opérateur INSERT.
L'opérateur INSERT a été exclu pour plusieurs raisons :

- Il nécessite plus de temps car une insertion implique de décaler les jobs dans leur batch à l'endroit où le job est retiré et à celui où il est inséré. Ces calculs supplémentaires ralentissent fortement l'algorithme.
- Dans le cas d'une insertion dans un batch autre que celui du job inséré, il faut que ledit batch ne soit pas plein. Cela implique donc de constituer une solution initiale avec des jobs en sous-effectif et donc d'ajouter des batchs supplémentaires. Hors, avec les coûts de stérilisation, la solution initiale est de bien moins bonne qualité et l'algorithme tabou ne parvient pas même à rattraper une solution initiale avec des jobs complets.
- Dans le cas d'une insertion au sein d'un même batch mais pas sur la même machine, le tabou ne fait que déstabiliser la solution initiale dans laquelle les temps totaux par machines étaient équilibrés. Et la probabilité que ce déséquilibre soit compensé plus tard au cours de l'algorithme est très faible avec un INSERT aléatoire.
- Enfin, avec un INSERT sur la même machine du même batch, on ne trouve pas d'intérêt car dans la solution initiale, les jobs d'un batch sont triés par date due croissante sur une machine. et de toute façon les flacons sont ouverts au démarrage du batch et non en fonction de la position du job dans son batch.

Evaluation d'un voisin

Le meilleur voisin est choisi sur le critère de fitness global F.

Remarque Avant comparaison du meilleur voisin avec la meilleure solution trouvée jusque-là, on remplace dans le codage des jobs de ce voisin les flacons ouverts mais non totalement consommés par des flacons de moindre contenance satisfaisant également le besoin (diminuant ainsi la matière perdue).

Remarque Aucune recherche locale n'est effectuée directement sur la consommation en matière première. Cela provient du fait que l'échange des flacons assignés est un processus extrêmement périlleux qui aboutit rarement à une solution faisable. En effet, modifier ne serait-ce qu'un flacon attribué peut avoir d'importantes répercutions si ce flacon est partagé par plusieurs jobs, car alors les matières prélevées chronologiquement par chaque job ne sont plus les mêmes (voir la partie reconstitution de la solution) , et c'est toute l'assignation des flacons qui peut devenir invalide.
Pour cette raison, et puisque une assignation polynomiale des flacons fournit de bons résultats, on préférera échanger uniquement les positions et réassigner pour chaque nouveau voisin les flacons depuis le début.

Le calcul des bornes inférieures

Le fitness d'une solution du problème est déterminé à partir de deux critères :
- MF, Le critère de consommation de matières, qui encourage l'économie des matières premières ouvertes pour la fabrication des produits de chimiothérapie.
- SF, Le critère d'ordonnancement, qui tend à minimiser les retards des jobs par rapport à leur date due.

Hypothèse Le fitness est une combinaison linéaire des deux critères, de la forme :

$$F = \alpha MF + \beta SF$$

Il est toujours intéressant de disposer de bornes inférieures lors de la résolution d'un problème, parce que celles-ci sont des critères d'arrêt naturels (on ne peut pas faire mieux que la borne inférieure).
Dans cette section, on va donc s'atteler à déterminer les bornes inférieures de chaque critère.

9.1 Borne sur le critère de consommation de matières

9.1.1 Définition

Trouver la borne inférieure du critère de consommation revient à trouver pour chaque type de matière première l'ensemble des flacons à utiliser qui satisfont la demande des jobs et minimisent la quantité perdue. C'est à dire :

$$\forall k \in [1 \dots nbMat], L_k^* \subset L \backslash \forall L_k^\alpha \subset L, B_k \leq \sum_{l \in L_k^*} v_{kl} \leq \sum_{l \in L_k^\alpha} v_{kl}$$

Avec L_k^* l'ensemble optimal, L, l'ensemble des flacons disponibles pour k et B_k la demande totale en produit k.
On a donc k sous problèmes à résoudre.

9.1.2 Remarque sur la faisabilité de la borne inférieure

Il est important de remarquer que la définition ci-dessus ne prend pas en compte les contraintes de faisabilité suivantes :
- Non dépassement de la péremption des flacons après ouverture.
- Impossibilité de partager un flacons sur deux batchs concurrents.

C'est pourquoi il est peu probable qu'une solution faisable puisse tout à fait correspondre à la borne inférieure.

9.1.3 Formalisation et complexité de détermination

Dans cette section et dans un souci de simplification, on omettra l'indice k, puisqu'on traitera uniquement d'un seul type de matière première à la fois.
Pour chaque type de matière première, les volumes des flacons sont définis par une moyenne et un écart-type et ne sont pas constants. On peut donc formaliser formaliser le problème ainsi :

- v_l volume d'un flacon de matière.
- x_l variable booléenne qui vaut 1 si l on inclut le flacon l dans l'ensemble et 0 sinon.
- w_l L'intérêt d'insérer le flacon dans l'ensemble optimal. En fait, puisque le prix de la matière première est associé au volume contenu et non au flacon lui-même, on considère que tous les flacons sont d'intérêt égal toutes proportions gardées et alors $w_l = v_l \forall l$.

Minimiser la quantité perdue, cela revient à maximiser la quantité économisée, sans pour autant dépasser la limite spécifiée par B. On peut donc dire que trouver la borne inférieure de consommation revient à :

$$\text{maximiser} \sum_{l=1}^{L}(1-x_l)v_l \text{ t.q. } \sum_{l=1}^{L}(1-x_l)v_l \leq \sum_{l=1}^{L} v_l - B, x \in \{0,1\}$$

La fonction définie ci-dessus est bien connue. Il s'agit d'un cas spécial du **problème du sac-à-dos** avec poids égaux dit **problème de la somme des sous-ensembles**.
Ce problème est reconnu comme étant NP-difficile. Il n'est donc pas possible d'obtenir facilement une borne inférieure fiable pour chaque k et à fortiori pour le problème complet, bien que des algorithmes gloutons simples puissent fournir une bonne valeur approchée.

9.2 Borne sur le critère d'ordonnancement

9.2.1 Définition

Le critère d'ordonnancement est plus difficile à définir simplement car il repose sur deux sous-critères aux objectifs différents :

- Minimiser le retard maximum des jobs (L_{max})
- Minimiser le nombre de jobs en retards ($\sum_{i=1}^{nbJobs} U_i$).

Une approche intuitive suffit à montrer que ces objectifs sont incompatibles : Pour minimiser le nombre de jobs en retard, on aura tendance à négliger L_{max} voire à l'augmenter de façon à obtenir de la marge permettant de satisfaire les dates dues d'autres jobs dont le retard peut être évité. Alors que pour diminuer L_{max} on essaiera de diminuer l'écart-type sur le retard moyen, ce qui entrainera possiblement une augmentation du nombre de jobs en retard.

Hypothèse Le critère sur l'ordonnancement est une combinaison linéaire des deux sous-critères, de la forme :

$$SF = \alpha L_{max} + \beta \sum_{i=1}^{nbJobs} U_i$$

Avec α et β des scalaires.
On va tenter de trouver une borne inférieure à chaque sous-critère.

9.2.2 Borne inférieure de L_{max}

Dans le cas d'un ordonnancement à une seule machine sans contraintes particulières, la minimisation du retard max se résout de façon optimale en appliquant un tri de type EDD (*Earliest due date*) et en plaçant les jobs sur la machine dans cet ordre.
On peut tenter d'étendre cet algorithme à notre problème, puisque les isolateurs sont indépendants les un des autres et que chaque machine au sein d'un isolateur l'est également (seule la contrainte de taille maximum d'un batch doit être satisfaite).
On construit alors itérativement la solution d'ordonnancement en plaçant à chaque étape le job (disponible selon r_i) de plus petite date due sur la machine disponible au plus tôt.

Il faut faire attention à la date de disponibilité des machines, qui doit prendre en compte les périodes de stérilisation de batch ainsi que les indisponibilités d'opérateur.
L'algorithme est présenté sur la page suivante.

Data:
J = {jobs}
M = tableau de 1 à nbMachines entiers : dates de disponibilité au plus tôt des machines. Initialisé à TEMPS_STERILISATION.
I = tableau de 1 à nbIsolateurs entiers : nombre de jobs dans le batch courant. Initialisé à 0.
while $J \neq \emptyset$ **do**
    j = 0 ;
    m = 0 ;
    minDateM = 0 ;
    minDate = INFINI ;
    for *i = 1 à nbMachines* **do**
        j = premierDisponibleEn(J, M[j]) ;
        if *machinePasEnPauseSurPlage(i, M[i], M[i] + p_j)* **then**
            minDateM = M[i] ;
        else
            minDateM = prochainePlageDisponibilité(i, M[i] + p_j) ;
        end
        if *minDateM < minDate* **then**
            m = i ;
            minDate = minDateM ;
        end
    end
    C_j = minDate + p_j ;
    M[i] = minDate + p_j ;
    retirer(J, j) ; iso = isolateur(i) ;
    I[iso] = I[iso] + 1 ;
    if *I[iso] == MAX_PAR_BATCH* **then**
        I[iso] = 0 ;
        for *i2 = 1 à nbMachines* **do**
            if *isolateur(i2) == iso* **then**
                M[i2] = M[i] + TEMPS_STERILISATION ;
            end
        end
    end
end

Cet algorithme, qui pourrait paraître optimal ne l'est pas dans un cas au moins : lorsque 2 machines ont une date de disponibilité égale, et parce qu'il y a des dates d'indisponibilité des opérateurs propres à chaque machine, il est possible qu'un choix ne soit pas optimal alors que l'autre oui.
L'exemple qui suit illustre ce propos. Dans cet exemple, la ligne en pointillés indique un début d'indisponibilité pour la machine 2. La machine 1 est tout le temps disponible. Les jobs sont triés selon EDD.

Il n'existe donc pas d'algorithme polynomial facile pour déterminer la borne inférieure du L_{max}. A partir du moment où l'on doit effectuer un choix, on peut même se demander s'il en existe un ou pas du tout.

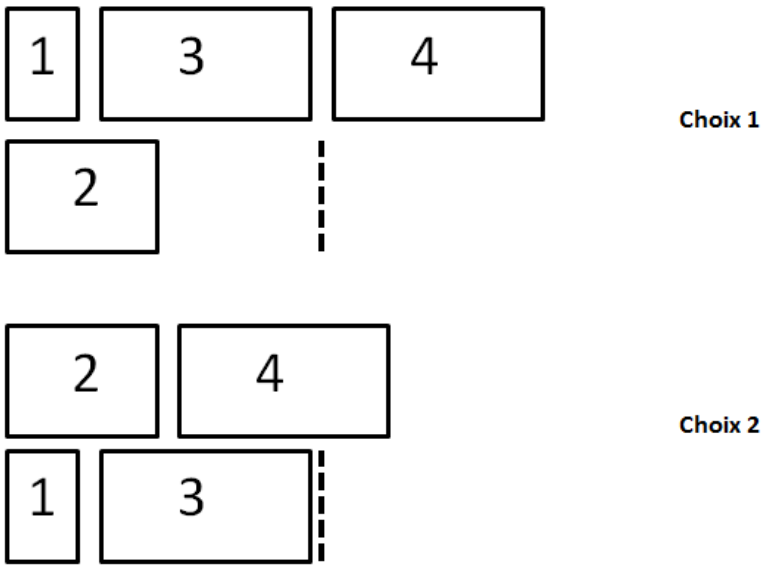

FIGURE 9.1 – Le choix 1 ne conduit pas à l'optimal

9.2.3 Borne inférieure de $\sum_{i=1}^{nbJobs} U_i$

Ici la conclusion est beaucoup plus rapide. En effet, Eugène L. Lawler à apporté la preuve en 1983 dans son article *Scheduling a Single Machine to Minimize the Number of Late Jobs*, que le problème de minimisation de $\sum_{i=1}^{nbJobs} U_i$ est NP-complet pour une unique machine lorsque les tâches sont définies par des dates dues et des dates de release. Dans notre problème, nous avons plus de machines, des dates d'indisponibilités par machine et des durées de stérilisation qui ne peuvent que complexifier le problème. Le calcul de la borne apparait donc comme étant NP-complet également.

9.2.4 Conclusion sur la borne inférieure du critère d'ordonnancement

S'il subsiste un doute sur la complexité du calcul de la borne inférieure sur le seul sous-critère L_{max}, le second sous-critère est clairement NP-complet. Par lien de cause à effet, le calcul de la borne inférieure du critère d'ordonnancement, obtenu par interpolation linéaire des deux sous-critères est forcément NP-complet.

9.3 Conclusion sur la détermination d'une borne inférieure

Dans chaque cas, le calcul du critère est qualifié de NP-Complet. On en déduit que la détermination d'une borne inférieure du problème relève également de la NP-Complétude.

La génération de jeux de données adéquats

10.1 Difficultés

La génération de jeux de données pour le problème est une problématique à part entière :
- Les paramètres sont nombreux et parfois corrélés (un job nécessite en moyenne 3 flacons. Donc son besoin en volume est proportionnel à la taille des flacons de la matière qu'il utilise).
- De façon générale, la distribution de chaque paramètre ne peut se résumer à une distribution aléatoire ou normale. Il faut être plus fin.
- Pour fournir des résultats cohérents et exploitables, ces paramètres doivent représenter assez fidèlement la réalité.

Un programme personnalisé de génération de paramètres a été mis en place afin de répondre à ce besoin.

Remarque Le programme est implémenté en java et n'a pas vocation à être exécuté directement. Il se lance seulement depuis un IDE. Se reporter à l'annexe *Guide d'utilisation des sources* pour plus de détails.

10.2 Distribution des paramètres

Remarque Les distributions décrites dans cette partie montrent la capacité de précision de l'outil de génération de paramètres. Mais les valeurs des paramètres ont évolués depuis, surtout pour les tests finaux.

Les distributions choisies dans l'exemple suivant doivent permettre de refléter au mieux la réalité à laquelle sont confrontés les préparateurs de chimiothérapies dans le cadre de leur travail. Toutefois, l'horizon de planification a été réduit. Celui-ci est fixé à deux jours. La journée commence à 8 heures (date 0). L'unité de temps est fixée à 5 minutes.
On émet donc l'hypothèse d'une certaine uniformité dans la charge de travail quotidienne.
Les paramètres généraux d'abord sont fixés et invariables :

P_STE	3
P_CTRL	1
P_INJ	1
H	6
NB_JOBS	300
NB_MAT	20
NB_BOTTLES	60

FIGURE 10.1 – Paramètres simples

300 jobs sur deux jours soit 150/jour représente une bonne moyenne. Le nombre de matières ($NbMat$) est réduit par rapport à la réalité en supposant que toutes les matières (100 dans la réalité) ne sont pas utilisées sur deux jours.
Il est important de préciser que l'augmentation de lk à tendance à améliorer la solution finale, car il est alors

plus facile de trouver des flacons de moindre contenance adaptée pour l'échange final de flacons. La valeur choisie ici est assez restreinte. La plupart des flacons d'une matière seront utilisés dans la solution finale.

Pour les paramètres d'ordonnancement et de consommation, des "classes" ont été créées. Chaque classe possède des caractéristiques propres en termes de paramètres. De cette façon, on obtient un paramétrage plus fin qu'avec une seule distribution globale.
Chaque valeur générée se situe autour de la moyenne évoquée dans un alentour spécifié par la précision.
Les paramètres d'ordonnancement des jobs sont les suivants :

	R		D		P		S	
	MAX	PRECISION	MEAN	PRECISION	MEAN	PRECISION	MEAN	PRECISION
classe 1 (25%)	0	+-00%	40	+-3	3	+-25%	3	+-0
classe 2 (09%)	24	+-10%	60	+-3	3	+-25%	3	+-0
classe 3 (08%)	24	+-10%	84	+-3	3	+-25%	3	+-0
classe 4 (08%)	24	+-10%	108	+-3	3	+-25%	3	+-0
classe 5 (25%)	288	+-00%	328	+-3	3	+-25%	3	+-0
classe 6 (09%)	312	+-10%	348	+-3	3	+-25%	3	+-0
classe 7 (08%)	312	+-10%	372	+-3	3	+-25%	3	+-0
classe 8 (08%)	312	+-10%	396	+-3	3	+-25%	3	+-0

FIGURE 10.2 – Paramètres d'ordonnancement

Les jobs peuvent être regroupés en deux blocs similaires pour chaque jour. Le premier bloc est formé de jobs (classes 1 à 4) disponibles et dus le premier jour. L'autre bloc est son pendant pour le deuxième jour. Pour chaque jour, 50% des jobs sont disponibles dès le départ, les autres arrivant progressivement.
On remarque qu'il s'agit bien d'une valeur max qui est exprimée pour r. Cela permet de refléter la capacité réelle de l'atelier à anticiper des commandes non encore enregistrées.
Le paramètre de délai de livraison, s, est constant. Dans la pratique, le problème de livraison est plus complexe (tournée de véhicule) mais il n'est pas étudié ici.
Les délais sont courts, quelques heures au plus après l'arrivée de la commande, ce qui va forcer l'existence d'un retard (assez représentatif de la réalité). L'idée d'un tel jeu est aussi de permettre d'évaluer la capacité de l'algorithme tabou à améliorer la qualité d'ordonnancement de la solution initiale, ce qui serait impossible si dès la solution initiale aucun job n'était en retard.
Enfin, les paramètres propres à la consommation de matières premières sont :

	G		W		V		F		B	
	MEAN	PRECISION	MEAN	PRECISION	MEAN	PRECISION	MEAN	PRECISION	MEAN	PRECISION
classe 1 (25%)	96	+-20%	10	+-20%	100	+-20%	288	+-8%	20	+-50%
classe 2 (50%)	48	+-20%	100	+-20%	50	+-20%	108	+-8%	200	+-50%
classe 3 (25%)	12	+-20%	500	+-20%	10	+-20%	36	+-8%	1000	+-50%

FIGURE 10.3 – Paramètres de consommation

On distingue dans l'exemple trois grandes catégories de matières :
- Les matières coûteuses (classe 3) périssent vite et sont disponibles en flacons de petite quantités.
- Les matières peu chères (classe 1) ont une longue durée de vie et sont disponibles dans de grands contenants.
- La dernière classe se situe entre les deux.

On peut remarquer aussi que les besoins en volume d'un job sont dépendant de la nature du produit. Ceci est important car dans le codage de notre solution le nombre de flacons utilisables est limité, il faut donc qu'il y ait corrélation (approximative) entre le volume des flacons disponibles et la demande d'un job.

Résultats obtenus

11.1 Premiers résultats théoriques

Ces premiers résultats sont obtenus à partir des distributions détaillées ci-dessus. A partir de ces mêmes distributions, plusieurs instanciations de paramètres ont permis d'obtenir 5 jeu de données différents.
Ces jeux de données doivent refléter approximativement la réalité. Globalement, on peut déjà apercevoir l'efficacité du tabou appliqué sur la solution initiale.
S'il est difficile de savoir si la qualité de l'ordonnancement est suffisante (par manque d'informations sur les conditions réelles), la qualité sur la consommation est prometteuse, avec presque toujours moins de 4% de matière perdue, là où le service de Bretonneau en perd 8% en moyenne.
Ces résultats poussent à mener plus avant les expérimentations autour de cette méthode.

Set	Caractéristique	Glouton	Glouton + local
set 1	Volume perdu	1718	906
	Valeur perdue	170472	86911
	Fitness consommation	**0,0721635**	**0,0381399**
	Retard max	8	7
	Nombre de retards	24	17
	Fitness ordonnancement	**0,47314**	**0,4421**
set 2	Volume perdu	1779	944
	Valeur perdue	161377	88467
	Fitness consommation	**0,0688943**	**0,0389813**
	Retard max	10	9
	Nombre de retards	32	18
	Fitness ordonnancement	**0,52133**	**0,492**
set 3	Volume perdu	1939	911
	Valeur perdue	181362	85626
	Fitness consommation	**0,0713459**	**0,0350026**
	Retard max	7	6
	Nombre de retards	32	15
	Fitness ordonnancement	**0,4521**	**0,41**
set 4	Volume perdu	1945	1057
	Valeur perdue	198861	103503
	Fitness consommation	**0,0808418**	**0,0437734**
	Retard max	10	9
	Nombre de retards	27	17
	Fitness ordonnancement	**0,518**	**0,049133**
set 5	Volume perdu	1802	1077
	Valeur perdue	193122	102486
	Fitness consommation	**0,0726761**	**0,0399297**
	Retard max	9	9
	Nombre de retards	34	20
	Fitness ordonnancement	**0,50267**	**0,49333**

FIGURE 11.1 – Premiers résultats théoriques.

11.2 Simulation à plus grande échelle

L'ordonnancement sur deux jours proposant de bons résultats, nous avons ensuite souhaité étendre l'horizon de planification. Pour cela, nous avons dupliqué les paramètres (de façon logique) pour les étendre à 14 puis 30 jours. Pour ces jeux de données, le nombre de voisins explorés à été ramené à 1000 (14 jours) et 200 (30 jours).

Remarque Une optimisation de l'algorithme a permis entre temps d'améliorer les résultats de l'algorithme sur deux jours. Dans les résultats affichés, l'échelle du critère d'ordonnancement a également changé sans conséquence.

Horizon	Caractéristique	Glouton	Glouton + local
2 jours	Volume perdu	1802	670
	Valeur perdue	193122	51964
	Fitness consommation	**0,0726761**	**0,0206523**
	Retard max	9	6
	Nombre de retards	34	11
	Fitness ordonnancement	**3,6**	**1,78235**
14 jours	Volume perdu	10204	4227
	Valeur perdue	1309634	517988
	Fitness consommation	**0,0737331**	**0,0304086**
	Retard max	12	12
	Nombre de retards	218	207
	Fitness ordonnancement	**4,8**	**4,6789**
30 jours	Volume perdu	92354	87277
	Valeur perdue	8012495	7547094
	Fitness consommation	**0,1835766**	**0,174688**
	Retard max	15	11
	Nombre de retards	469	469
	Fitness ordonnancement	**6**	**6**

FIGURE 11.2 – Influence de l'horizon sur les résultats.

11.2.1 Observations

A 14 jours, l'algorithme tabou améliore toujours de façon significative le critère de consommation. Cependant on observe déjà une moins bonne amélioration du critère ordonnancement que dans la version sur deux jours.
Le jeu à 30 jours est plus radical. Dans ce dernier, le critère ordonnancement n'a pas été amélioré du tout et le critère de consommation n'a que peu diminué.

11.2.2 Interprétation

Pourquoi si peu d'amélioration ? Outre l'évidente difficulté croissante de construire et d'évaluer des solutions de grande taille, on peut s'interroger sur la légitimité de tels jeux.
L'algorithme tabou effectue des échanges de jobs choisis aléatoirement. L'échange n'est intéressant que s'il n'entraine pas une augmentation considérable du retard d'un des deux jobs échangés. Sur un petit horizon, cela est possible ; intervertir deux jobs sur une plage de quelques heures peut améliorer les critères de sélection sans dégrader le retard d'un des jobs échangés.
Mais sur un horizon de 14 ou 30 jours, il est peu probable que cela arrive. Les échanges effectués sont le plus souvent mauvais car la marge de retard pour chaque job ne dépasse que rarement les quelques heures. Une interversion de deux jobs dont les dates de fins sont distantes de plus d'une journée invalide

certainement la solution (d'un point de vue ordonnancement, cela s'entend).
Pour palier à ce problème, on pourrait peut-être découper un horizon trop grand en sous-problèmes d'horizon faible et lancer l'algorithme sur ces mêmes sous-problèmes. Les échanges de jobs auraient alors plus de probabilité d'améliorer la solution et la concaténation des solutions serait sans doute meilleure que les résultats obtenus ici.

11.3 Simulation de la continuité : flacons pré-ouverts

Pour améliorer le réalisme des résultats trouvés, nous avons ensuite créé des jeux de données dans lesquels on définit pour chaque type de matière première des flacons pré-ouverts. Ces flacons ont une quantité ainsi qu'une durée de vie amoindries d'un même facteur aléatoirement. 10 flacons de chaque matière remplissent ces conditions.
Les jeux sans flacons pré-ouverts présentés en parallèle ne doivent pas être comparés directement car ils n'ont pas les mêmes paramètres (puisqu'ils n'ont pas de flacons pré-ouverts) mais ils ont les mêmes méta-paramètres, c'est-à-dire les mêmes paramètres de génération de jeux de données.

		Sans frigo ouvert		Avec frigo ouvert	
Horizon	Caractéristique	Glouton	Glouton + local	Glouton	Glouton + local
2 jours	Volume perdu	1802	670	1980	808
	Valeur perdue	193122	51964	201304	82328
	Fitness consommation	**0,0726761**	**0,0206523**	**0,0769141**	**0,0329538**
	Retard max	9	6	13	10
	Nombre de retards	34	11	30	12
	Fitness ordonnancement	**3,6**	**1,78235**	**5,2**	**3,04**
14 jours	Volume perdu	10204	4227	35398	32125
	Valeur perdue	1309634	517988	3102008	2661797
	Fitness consommation	**0,0737331**	**0,0304086**	**0,1582541**	**0,1402699**
	Retard max	12	12	14	13
	Nombre de retards	218	207	227	216
	Fitness ordonnancement	**4,8**	**4,6789**	**5,6**	**5,26432**

FIGURE 11.3 – Simulation avec des flacons pré-ouverts.

11.3.1 Observations

Sur deux jours, l'algorithme est un peu impacté par la mesure mais reste efficace. Le jeu de données sur 14 jours n'est cependant pratiquement pas amélioré par le tabou.

11.3.2 Interprétation

L'observation et l'interprétation des résultats est quelque peu difficile. Plusieurs facteurs peuvent entraver le travail de l'algorithme sans que la faute ne lui en incombe directement. En particulier :

- Si les flacons pré-ouverts aléatoirement correspondent à une matière première qui n'est pas ou peu utilisée (les tirages des matières attribuées aux jobs sont aléatoires), alors l'algorithme ne pourra de toute façon pas empêcher la perte des produits entamés.
- Si les flacons pré-ouverts aléatoirement correspondent à une matière qui n'est nécessaire que tardivement (un job commandé pour une semaine après le début de l'ordonnancement par exemple), alors il est possible que le critère ordonnancement l'emporte sur le critère de consommation et que l'algorithme laisse périmer la matière première plutôt que de mettre en retard plus de jobs.

La combinaison de ces deux points explique peut-être les résultats médiocres de l'algorithme tabou sur le jeu de données à 14 jours.
Cependant il faut insister sur le côté théorique de ces résultats. Dans la pratique, les flacons ne sont pas ouverts aléatoirement et correspondent sans doute aux produits les plus utilisés (en termes de fréquence

et/ou de quantité) et donc les points précédents devraient avoir une importance moindre.
Dans de bonnes conditions avec le test sur deux jours par exemple, on peut voir que l'algorithme résiste bien.

11.4 Observation de la tendance au regroupement

Nous avons ensuite souhaité observé la structure de la solution proposée en se focalisant sur la consommation en matière première. Existe t-il un schéma observable sur la répartition des jobs dans la solution ? En particulier, est-ce que les jobs qui consomment la même matière première sont regroupés dans le même batch. Si tel était le cas, cela permettrait d'orienter la recherche de solution et peut être d'améliorer les résultats.
Pour tenter d'observer un phénomène, on effectue les adaptations suivantes sur les jeux de données :

- On se place sur un horizon moyen (14 jours) puis long (30 jours). L'objectif est d'avoir un échantillon de jobs suffisamment grand pour éliminer le facteur aléatoire de la répartition et pouvoir en tirer des conclusions.
- A causes des perturbations évoquées, on ne pré-ouvrira aucun flacon.
- Pour se focaliser sur la consommation, on ignorera arbitrairement le paramètre d'ordonnancement (réduit à 0 pour toute solution).

Horizon	Caractéristique	Glouton	Glouton + local
14 jours	Volume perdu	9910	3093
	Valeur perdue	1296819	433522
	Fitness consommation	**0,0726573**	**0,0255861**
	Moyenne de la matière la plus utilisée par batch	**2,32967**	**2,29121**
	Max de la matière la plus utilisée par batch	**4**	**4**
30 jours	Volume perdu	21196	6907
	Valeur perdue	2660010	1060316
	Fitness consommation	**0,0702279**	**0,0298812**
	Moyenne de la matière la plus utilisée par batch	**2,35128**	**2,32051**
	Max de la matière la plus utilisée par batch	**5**	**5**

FIGURE 11.4 – Observation des regroupements.

11.4.1 Observations

Contrairement à nos espérances, la figure 9 ne montre pas un regroupement particulier des jobs en fonction de la matière première qu'ils consomment. Au contraire, la tendance est légèrement à la baisse.
Sur 30 jours, en moyenne, la matière la plus utilisée dans un batch est partagée par 2,35 jobs dans la solution initiale et 2,32 dans la solution avec application de la recherche tabou.
Le batch pour lequel le partage est maximum ne contient que 5 jobs Mais la plupart ne contiennent qu'un ou deux jobs au maximum.
Le regroupement par matière ne semble donc pas présenter d'intérêt particulier. Cependant, le retour de bons résultats sur la diminution du fitness de consommation pour de grands horizons tend à souligner l'hypothèse évoquée plus haut dans la sous-section *Simulation à grande échelle*, à savoir que sur de grands horizons, les échanges aléatoires sont mauvais à cause du critère d'ordonnancement. Si on supprime ce critère, l'algorithme est toujours aussi efficace pour l'amélioration de la consommation de matière.

11.5 Utilisation d'un SWAP intelligent

Les résultats observés en **11.2** pour de grands jeux de données étaient médiocres. Toutefois, nous avions évoqué la possibilité que cette médiocrité soit due à des opérations d'échange aléatoire sur de trop grands intervalles. On mettait en exergue le fait que des échanges de deux jobs séparés de plus d'une journée n'avaient que peu de chance d'améliorer la solution.
Aussi avons-nous voulu essayer d'appliquer un autre opérateur. Nous avons donc mis en place un SWAP aléatoire mais qui respecte comme contrainte que les jobs échangés ne soient pas distants dans la solution de plus de 12 heures.
Un tel opérateur est beaucoup plus lourd à mettre en place qu'un SWAP totalement aléatoire. La vérification de l'intervalle de temps entre les deux jobs nécessite une reconstruction et donc un temps de calcul supplémentaire non négligeable. Il faut donc que le gain observé soit important pour contrebalancer la lenteur en sus. Sur un jeu de données de 14 jours (2100 jobs) et 30 jours (4500 jobs), sans flacon pré-ouvert, en prenant en compte les dates dues, on observe les résultats présentés sur la figure suivante.

Horizon	Caractéristique	Glouton	Glouton + local
14 jours	Volume perdu	10615	2506
	Valeur perdue	1378059	287635
	Fitness consommation	**0,075306**	**0,0169953**
	Retard max	132	132
	Nombre de retards	834	752
	Fitness ordonnancement	**52,8**	**50,20432**
30 jours	Volume perdu	24262	6397
	Valeur perdue	3093208	861243
	Fitness consommation	**0,0794823**	**0,0234421**
	Retard max	133	132
	Nombre de retards	1833	1801
	Fitness ordonnancement	**53,2**	**52,53562**

FIGURE 11.5 – Application d'un swap aléatoire à portée réduite.

11.5.1 Observations

Les résultats sont extrêmement prometteurs. Bien qu'à l'exécution du programme, on ait pu observer que l'exploration des voisins était plus lente, cet opérateur (qu'on appellera SWAP Range) fournit d'excellents résultats.
Le critère de consommation est optimisé aussi bien (voire mieux) qu'avec un swap totalement aléatoire. Le critère d'ordonnancement n'est que peu affecté. Cela est en accord avec la priorité placée sur le critère de consommation dans la fonction objectif.

11.5.2 Conclusion

L'opérateur SWAP Range, même s'il est plus lent que le swap purement aléatoire, est à privilégier dès que le jeu de données dépasse plusieurs jours d'horizon. Il faut aussi prendre en compte dans le calcul la répartition des dates dues et des releases des jobs.

11.6 Tests finaux

Très peu de temps avant la fin du projet, nous avons eu l'occasion de retrouver Jean-François Tournamille pour discuter de nos résultats et lui poser quelques questions supplémentaires. Au sortir de cette

réunion, nous avons établi un tout dernier ensemble de méta-paramètres sensés refléter avec beaucoup plus de précision la réalité de la situation.
Il s'est avéré que beaucoup de paramètres que nous pensions liés (comme la quantité d'un flacon en fonction de sa valeur au ml) ne l'étaient. D'une façon générale, il y avait beaucoup plus de disparités et moins de règles que nous le pensions.
Nous avons augmenté le nombre de classes de matières et de jobs et resserré les contraintes de temps. Malheureusement, le temps manque à l'heure de l'écriture de ce rapport pour tout détailler mais il est probable de toute façon qu'après 50 pages, personne n'y prête grande attention.
A l'aide de ces méta-paramètres, nous avons créé 10 jeux de données supplémentaires et calculé la moyenne des résultats obtenus. Ces résultats sont affichés dans la figure finale.

Notes	Caractéristique	Glouton	Glouton + local
moyenne 10 sets	Valeur consommée	2062688,2	1970263,1
- 30 jours	Volume perdu	39541,5	7224,8
- opérateur range (+-12h)	Valeur perdue	111950,7	19525,6
- sans flacon pré-ouvert	**Fitness consommation**	**5,433066**	**0,98723**
	Retard max	288,1	288,1
	Retard moyen	42,7804	59,5821
	Nombre de retards	3802,5	2974,9
	Fitness ordonnancement	**115,24**	**102,698392**

FIGURE 11.6 – Résultats finaux plus représentatifs.

11.6.1 Observations

Les paramètres de chaque set ont été générés à l'aide des mêmes méta-paramètres. Ils sont donc similaires. De même, l'écart-type entre les résultats de chaque set est très faible.
On voit que même dans ces conditions, l'amélioration du critère consommation est excellent. Sur 30 jours, on a donc une perte en matière première de moins d'un pour-cent. Le fitness ordonnancement est peu amélioré, en accord avec le poids accordé à ce critère, mais n'est pas dégradé conformément aux contraintes posées sur notre algorithme.

Quatrième partie

Conclusion et annexes

Retour sur la gestion de projet

A tous les étudiants, il a été transmit le message qu'il fallait que ce rapport contienne une part de gestion de projet. Aussi me sens-je obligé d'en parler, ne serait-ce que pour n'en rien dire.

Car je reste, en tant que futur ingénieur, fidèle à mes convictions lorsque j'estime celles-ci justifiées. Le projet que j'ai conduit n'a pas été guidé par une quelconque démarche de gestion de projet avec un planning et des objectifs, des pourcentages de tâche accomplies à saisir ou des rapports d'incident à gérer. Pourquoi mettre en place une plateforme de reporting de l'avancement du projet si personne ne s'en préoccupe ? Je travaille seul et mon encadrant ne serait probablement pas allé y jeter un oeil. Je l'ai néanmoins tenu au courant toutes les semaines si ce n'est plus de l'avancement du projet.

La gestion de projet doit être un outil. Elle est le plus souvent essentielle. Mais elle ne doit jamais entraver inutilement le travail qu'elle est sensée diriger. Et si pour contrer cet argument on m'expose que l'objectif est à vue éducative, je réponds qu'à ce stade, nous avons eu suffisamment de projets pour en avoir une bonne idée.
De toute façon, point de tâche clairement définie il n'y avait au démarrage : il s'agit d'un projet de recherche, avec un objectif vaguement défini qui permet de justifier une exploration et des essais intéressants. Mon travail s'est redéfini au cours des semaines, et plus j'avançais plus on me donnait du travail, tant et si bien qu'en suivant cette démarche, selon toute vraisemblance, je n'aurai mathématiquement pu qu'être en retard une fois la fin des projets en vue.

Certes, j'ai sauvegardé régulièrement mes sources et pris en notes nos réunions. J'ai dialogué avec mon encadrant. J'ai rigoureusement documenté chacun des travaux que j'ai été amené à réaliser. Mais plus aurait été trop. J'ai vu plusieurs étudiants perdre du temps sur cette fameuse gestion de projet et finalement en manquer pour réaliser leur objectif principal.

J'ai à coeur de penser qu'à Polytech, on nous a enseigné le bon sens et à savoir le défendre lorsque cela s'avère nécessaire.

Conclusion

13.1 Sur le projet

On peut considérer que le projet a abouti. L'algorithme fonctionne, et permet de tester des jeux de données variables à volonté. Les résultats obtenus sont bons. En 5 minutes, l'algorithme est capable d'ordonnancer la production sur 30 jours. C'est assez plaisant, lorsqu'on sait que PLANIF gère dans ce même temps seulement une journée de prévision.
J'espère que mes travaux ne tomberont pas totalement dans le néant même si cela est très probable. j'ai pris à coeur de documenter mon code et à ma connaissance, il ne présente pas de défaillance majeure.

13.2 Personnelle

Ce fut un PFE résolument instructif. Il avait l'intérêt d'être concret tout en étant un projet de recherche. D'autre part, quel plaisir de se dire que mes travaux pourraient peut-être servir à améliorer la qualité de service de nos hôpitaux (même si ce ne fut pas le cas, sans nouvelles d'ETICSYS) !
Ce projet aura été l'occasion d'aborder de bout en bout un beau projet de recherche, chose que je souhaitais faire avant de me diriger définitivement vers le milieu privé. Et pourtant, il a aussi été l'occasion de concevoir des algorithmes compliqués et d'améliorer mes techniques de développement.
Les rencontres avec notre correspondant à l'hôpital, Jean-François Tournamille, ont été l'occasion d'une belle rencontre humaine, mais aussi d'un bon aperçu de ce qu'est la récolte d'information en amont du démarrage d'un projet : une chose parfois malaisée lorsque les interlocuteurs ne parlent pas le "même langage"...

Annexe 1 : Les sources : mode d'emploi

A.1 Guide d'utilisation des sources

L'élaboration de l'algorithme d'optimisation de l'ordonnancement de la production à conduit à la réalisation de plusieurs modules et programmes. Cette partie a pour objectif d'expliquer comment se servir correctement des sources pour faire fonctionner l'algorithme avec succès.

A.1.1 Résumé de la configuration

Avant toute chose, il est nécessaire de rappeler que le programme développé relève avant tout d'un objectif de recherche. Si son exploitation n'est pas la plus simple, l'auteur présente donc ses excuses mais affirme que l'objectif n'était pas d'avoir une "jolie interface", mais un programme effectif.

La configuration se passe donc en quatre étapes :

1. La création d'un jeu de données via le projet Java **ChimioGenerator**. Le produit de cette étape est un fichier de paramètres *params.txt* qui sera lu par l'algorithme.
2. La définition dans le fichier *availabilities.txt* des disponibilités des opérateurs.
3. La définition dans le fichier source config.h des grandes constantes du programme.

Les sections suivantes s'attachent à détailler chacune de ces parties.

A.1.2 Création de jeux de données

Le projet **ChimioGenerator** a été créé pour répondre aux exigences de détails quant au paramétrage du problème. Il permet de définir assez précisément les données en entrée du problème.
En particulier, pour chaque paramètre, il permet de préciser une moyenne et un écart en pourcentage autour de cette moyenne pour les valeurs générées aléatoirement. Il permet également de créer des classes de jobs et de matières premières pour lesquelles les paramètres corrélés sont définis différemment.
Pour modifier les paramètres de génération, il faut se rendre dans le fichier source *Config3.java*. Dans ce fichier des commentaires vous aideront à définir les paramètres à votre convenance. Une fois les paramètres correctement définis. Il faut recompiler le programme et l'exécuter (le point d'entrée étant bien sûr dans la classe Main.java). Le plus simple est de démarrer le projet java avec l'IDE Eclipse et de lancer l'exécution. Vous pouvez également tenter une compilation et une exécution à la main en ligne de commandes.
Le fichier produit par cette exécution contient, au format GLPK, les paramètres du problème. Ce fichier doit être renommé *params.txt* et doit être placé dans le répertoire d'exécution de l'algorithme.

A.1.3 Définition des disponibilités des opérateurs

Dans le répertoire d'exécution de l'algorithme doit se trouver un fichier *availabilities.txt* qui définit les **disponibilités**, et non pas les pauses, des opérateurs pour chaque machine du problème. Ce fichier contient en commentaire toutes les instructions nécessaires à la saisie des disponibilités.
Si le fichier était absent, une copie de démonstration existe dans un répertoire de niveau supérieur. La copie est appelée *availabilities_format.txt*.
Si le format du fichier n'était pas respecté, il est fort probable que l'exécutable affiche une erreur indiquant le problème.

A.1.4 Modification de la configuration globale

Pour des raisons d'optimisation, certaines constantes du problème ont été déportées dans le fichier source *config.h*. Elles sont peu nombreuses.

Les constantes dépendantes de la taille de l'ordonnancement sont :
- **JOBS_MAX** : Le nombre maximum de jobs à ordonnancer. Il est possible de laisser cette constante à une grande valeur. Mais plus cette constante sera proche du nombre réel, meilleures seront les performances.
- **MAX_BOTTLES_BY_MATERIAL** : Le nombre maximum de flacons disponibles par matière première. Dépend du nombre de jobs à ordonnancer. Il est possible de laisser cette constante à une grande valeur. Mais plus cette constante sera proche du nombre réel, meilleures seront les performances.

Les constantes de contexte d'atelier sont :
- **NB_MACHINES** : Le nombre total de machines.
- **NB_ISOLATORS** : Le nombre d'isolateurs.

- **MAX_PER_BATCH** : Le nombre maximum de jobs pouvant être placés dans le même batch.
- **MAX_PUT_PER_BATCH** : Le nombre maximum de jobs à placer par batch dans la solution initiale (inférieur à MAX_PER_BATCH). Cette constante est utilisée en mode INSERT par le tabou (voir plus bas).

Enfin, les constantes propres au fonctionnement du tabou sont :
- **TABOU_TIMEOUT** : Le temps de recherche maximum du tabou, en minutes.
- **TABOU_TABOUS_MAX** : Le nombre de solutions taboues conservées.
- **TABOU_ITER_MAX** : Le nombre maximum d'itérations infructueuses du tabou avant l'arrêt de l'algorithme.
- **TABOU_MAX_NEIGHBORS** : Le nombre de voisins explorés à chaque itérations.

Il existe aussi des symboles qui permettent d'activer ou de désactiver des comportements de l'algorithme. Il suffit de marquer un symbole avec *#define* pour qu'il soit activé :
- **TABOU_INSERT_MODE** : Utilise l'opérateur d'insertion pour le tabou. Les jobs sont retirés d'une position puis insérés dans une autre. A noter que cet opérateur s'est révélé inefficace.
- **TABOU_RANGE_SWAP_MODE** : Utilise un opérateur de swap intelligent. Les jobs échangés ne sont pas distants de plus de 12h. Cependant, cet opérateur est plus lourd. Il n'est pas utile pour de petits jeux de données ($<$ 3 jours).
- **IGNORE_DUE_DATES** : Indique que les dates dues peuvent être ignorées. L'algorithme ne tient plus compte du critère ordonnancement et ne mesure son efficacité qu'en terme de réduction de la consommation de matière.
- **FRIDGE_PREOPENED** : Indique que le frigo doit être pré-ouvert. Pour augmenter le réalisme de la simulation, FRIDGE_PREOPENED_RATE flacons vont être préouvert avant l'exécution de l'algorithme. Ces flacons vont voir leur quantité ainsi que leur durée de vie réduites d'un même pourcentage aléatoire.

A.1.5 Exécution

Une fois l'intégralité de ces constantes bien définies, il est possible de recompiler le programme et de lancer l'algorithme. Le programme produit en sortie deux fichiers : *output_a.txt* et *output_b.txt* qui contiennent respectivement la solution initiale trouvée et la solution améliorée par l'heuristique tabou. En plus des données brutes, ils contiennent chacun quelques informations complémentaires sur le fitness de la solution.

Bibliographie

[1] J.C. Billaut. A mixed packing-sequencing problem in the delivery of chemotherapy drugs. *Springer Science*, page 10, 2009.

[2] J.C. Billaut. A mixed packing-sequencing problem in the delivery of chemotherapy drugs. *ETFA*, page 7, 2011.

Index

Prise en compte des reliquats dans la fabrication de chimiothérapies

Département Informatique
5e année
2013 - 2014

Rapport de projet de fin d'études

Résumé : L'ordonnancement de la fabrication de chimiothérapie présente un double objectif qui rend sa réalisation complexe. Le critère de minimisation du retard étant déjà pris en compte dans les algorithmes existants, on s'attèle dans ce projet à intégrer en sus la gestion de la minimisation de la perte en reliquat (matière périmée non utilisée). Ce travail est réalisé en partenariat avec l'Hôpital Bretonneau de Tours.

Mots clefs : Ordonnancement, multi-critère, reliquat, consommation, retard, tabou, Bretonneau, chimiothérapie

Abstract: Chemotherapy production scheduling includes two separated objectives which turn it into a complex issue. Lateness minimization is already handled in many existing algorithms. In this paper we focus on integrating wasting minimization (that is bought but unused matter). This work results from a partnership with Hôpital Bretonneau of Tours.

Keywords: Scheduling, multi-criterias, relics, consumption, lateness, tabou, Bretonneau, chemotherapy

---

Encadrants
Jean-Charles Billaut
jean-charles.billaut@univ-tours.fr
Université François-Rabelais, Tours, France

Étudiant
Thibault DREVON
thibault.drevon@etu.univ-tours.fr

DI5 2013 - 2014