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

Comparaison de modèles mathématiques et implémentation d'une métaheuristique pour le Master Surgical Scheduling Problem

Encadrant
Yannick Kergosien
yannick.kergosien@univ-tours.fr

Étudiant
Antoine GIRET
antoine.giret@etu.univ-tours.fr

École polytechnique de l'université de Tours,
Tours

DI5 2013 - 2014

Version du 4 mai 2014

Table des matières

Introduction

Ce document présente le travail effectué lors de mon projet de fin d'études sur le "Master Surgical Scheduling Problem". Ce projet était principalement un projet de recherche, réalisé en relation avec mon encadrant Mr Y. Kergosien, ainsi qu'avec Mr P. Soriano, membre du CIRRELT (Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport) de Montréal. Le "Master Surgical Scheduling Problem", aussi appelé MSSP, est un problème qui traite de la programmation d'interventions dans des salles opératoires québécoises (dans notre cas). Il n'existe pas dans la littérature d'étude de ce problème possédant l'intégralité des contraintes que nous devons prendre en compte.

L'objectif du projet était dans un premier temps de comprendre puis comparer deux modèles mathématiques livrés au début de celui-ci. Dans un second temps, l'objectif était de réaliser un travail de recherche afin d'améliorer les modèles, puis de comparer les résultats obtenus. La dernière phase du projet visait à implémenter une métaheuristique permettant la résolution du problème et à trouver les paramètres permettant d'obtenir les meilleurs résultats possible.

Nous commencerons par présenter le problème, son contexte, la problématique qui lui est associée ainsi que les différentes données nécessaires à sa compréhension. Nous verrons ensuite la méthode de génération de données nécessaires pour la résolution du problème, dont nous verrons les deux modélisations en question, ainsi que les résultats obtenus lors de la comparaison. Nous présenterons ensuite le travail de recherche effectué, tout d'abord au travers de différentes coupes, puis en présentant la métaheuristique mise en place. Enfin, nous conclurons sur ce projet et sur les résultats obtenus.

Présentation du problème

Le "Master Surgical Scheduling Problem" est un problème d'ordonnancement relatif aux blocs opératoires des hôpitaux Québécois, ces derniers ayant constaté une augmentation de leurs listes d'attentes. Cette augmentation est due à un certain nombre de facteurs et entraine des pertes économiques (notamment dues à la formation onéreuse des chirurgiens qui ne se voit pas rentabilisée) dont les hôpitaux cherchent à s'acquitter.

2.1 Contexte des hôpitaux Québécois

Un des facteurs de l'augmentation des listes d'attente est une sous-programmation de certaines spécialités opératoires. Cette sous-programmation est due au fait que l'affectation des salles opératoires aux spécialités ne se fait en général pas en fonction des besoins mais en fonction de l'ancienneté des chirurgiens.

Le fait que l'affectation ne se fasse pas en fonction des besoins entraine une sous-utilisation des salles opératoires, alors que certaines spécialités manquent de temps d'intervention. Des normes gouvernementales ont pourtant été mises en place afin d'éviter ce phénomène et obliger la programmation de certaines spécialités à intervalles réguliers, voire même à obliger la programmation d'une certaine spécialité un jour donné. Ces normes ne sont que très peu respectées, du fait de la difficulté à mettre en place de tels plannings.

Un autre facteur entrainant l'augmentation des listes d'attente est la difficulté à gérer les ressources, humaines comme matérielles. Afin de réaliser une intervention, il est nécessaire d'avoir à disposition un certain nombre de ressources, mais ces ressources peuvent ne pas être disponibles au moment de l'intervention. Réaliser une intervention entraine également une hospitalisation du patient. L'intervention peut donc ne pas être réalisée dans le cas où l'hôpital ne possède plus de lits disponibles. Une mauvaise planification ne tenant pas compte des ressources entraine donc inévitablement un retard dans les interventions prévues et une sous-utilisation des salles opératoires.

Des études ont montré que le nombre d'arrivées pour chaque intervention ne change pas beaucoup au cours de l'année. On a constaté qu'il y avait une modification des besoins lors des changements de saison. La planification est à l'heure actuelle périodique mais ne tient pas compte des saisons.

Le contexte dans les hôpitaux québécois montre que la planification des spécialités opératoires doit tenir compte d'un grand nombre de variables et de contraintes, et qu'il est très difficile de réaliser cette planification sans pour autant augmenter les listes d'attentes.

2.2 Problématique du "Master Surgical Scheduling Problem"

La problématique du projet est de planifier au niveau tactique des spécialités et leurs interventions pour chaque journée, dans un macro-planning cyclique. En d'autres termes, nous n'allons pas chercher à planifier un patient à un horaire précis dans une salle précise, mais des interventions pour chaque jour et chaque salle.

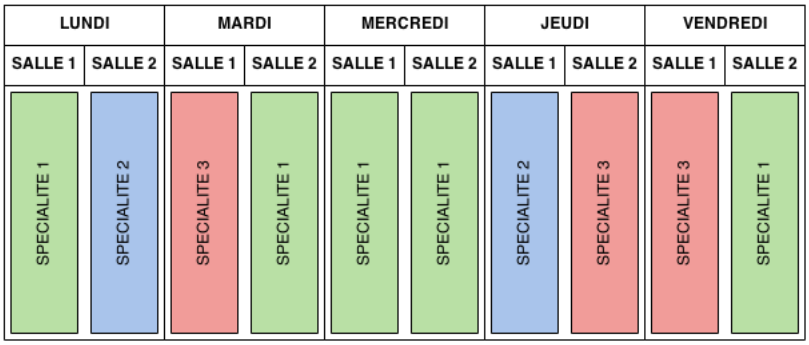

FIGURE 2.1 – Exemple d'un macro-planning

Ce macro-planning doit respecter un ensemble de contraintes :
- Respect des listes d'attente.
- Respect des durées d'ouverture des salles.
- Respect des normes ministérielles.
- Respect des ressources disponibles.
- Respect des lits disponibles.

Le premier travail effectué lors du projet a été de lister l'ensemble des données nécessaires à la résolution du problème en tenant compte de l'ensemble des contraintes citées précédemment.

2.3 Données du problème

2.3.1 Données générales

Soient :
- $|J|$ la durée de la période, soit la fréquence à laquelle va se répéter notre macro-planning.
- b_j le nombre de lits disponibles pour le jour j. Soit b le nombre que nous considèrerons, tel que $b_j = b \forall j$
- $|O|$ le nombre de salles opératoires.
- h_{oj} la durée d'ouverture de la salle o le jour j.
- U_c l'ensemble de ressources communes. Une ressource commune peut être utilisée par toutes les spécialités.
- c^c_{ju} le nombre de ressources communes de type u disponibles pour le jour j.

2.3.2 Données sur les spécialités

Soient :
- $|S|$ le nombre de spécialités.
- $Fmax_s$ le nombre maximum de jours séparant deux programmations de la spécialité s.
- $c_{min_{sj}}$ le nombre minimum de programmations de la spécialité s pour le jour j.
- U_s l'ensemble des ressources propres à la spécialité s. Une ressource propre à une spécialité ne peut en aucun cas être utilisée par une autre spécialité.
- c^s_{ju} le nombre de ressources propres à la spécialité s de type u disponibles pour le jour j.

2.3.3 Données sur les interventions types

Chaque spécialité opératoire regroupe un certain nombre d'interventions. Certaines de ces interventions se ressemblent compte tenu, par exemple, de leur durée d'intervention ou des ressources qu'elles utilisent. Elles peuvent alors regroupées en intervention dites "types".

Soient :
- $|K_s|$ le nombre d'interventions types de la spécialité s.
- τ_k le nombre d'arrivées pour l'intervention type k sur l'ensemble de la période.
- p_k la durée d'intervention de l'intervention type k.
- l_k la durée d'hospitalisation de l'intervention type k.
- m^c_{ku} le besoin en ressource commune de type u pour l'intervention type k.
- m^s_{ku} le besoin en ressource propre de type u pour l'intervention type k.

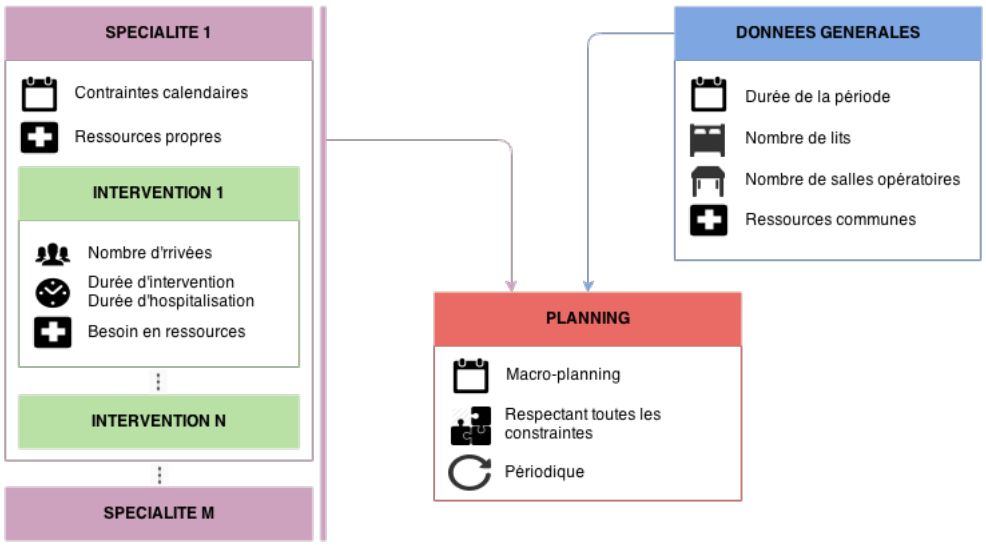

FIGURE 2.2 – Schéma récapitulatif des différentes données du problème

Génération de données

Il n'existe pas de jeux de données réels pour le "Master Surgical Scheduling Problem". Il a donc fallu, après avoir recherché des bornes réalistes sur les différentes données, concevoir un programme permettant la génération de données. Cette génération est basée sur un certain nombre de variables, définies par l'utilisateur en amont de la génération. Certaines données sont ensuite générées aléatoirement, afin d'obtenir des jeux de données différents à chaque exécution du programme.

3.1 Variables paramétrables

Comme nous l'avons vu ci-dessus, l'utilisateur a la possibilité de paramétrer certaines variables, afin d'obtenir un jeu de données relativement proche du problème souhaité.

- La durée de la période (en nombre de jours ouvrés).
- Le nombre de spécialités.
- Le nombre maximal d'interventions par spécialité.
- Le nombre maximal de jours séparant deux programmations d'une spécialité.
- La durée minimale d'intervention (en nombre de minutes).
- La durée maximale d'intervention (en nombre de minutes).
- Le nombre minimal d'arrivées pour une intervention dans la période.
- Le nombre maximal d'arrivées pour une intervention dans la période.
- La durée minimale d'ouverture d'une salle (en nombre d'heures).
- La durée maximale d'ouverture d'une salle (en nombre d'heures).
- Le nombre maximal de salles à affecter à une spécialité pour un jour au minimum.
- Le nombre de types de ressources communes.
- Le nombre maximal de types de ressources propres à une spécialité.
- Le besoin maximal d'un type de ressource pour une intervention.
- Le coefficient de flexibilité pour le calcul du nombre de salles.
- Le coefficient de flexibilité pour le calcul du nombre de lits.
- Le coefficient de flexibilité pour le respect des listes d'attente.

Toutes ces variables sont définies dans un fichier batch, puis passées en paramètre lors de l'appel à l'exécutable. Cela permet à l'utilisateur de modifier les valeurs des différentes constantes sans toucher au code source du programme, de manière à se rapprocher au plus des données de son problème. Les données générées aléatoirement en fonction de ces variables sont générées de telle manière à ce que le problème ne soit ni trop simple, ni impossible à résoudre. Elles sont liées aux différentes variables, comme nous allons pouvoir le constater ci-dessous.

Trois des variables sont des coefficients. Ces coefficients permettent d'obtenir un jeu de données ayant une solution réalisable, notamment le coefficient de flexibilité pour le respect des listes d'attentes. Les coefficients de flexibilité pour le calcul du nombre de salles et de lits permettent également d'obtenir un jeu de données ayant une solution plus ou moins contrainte en nombre de lits ou de salles.

3.2 Données aléatoires

3.2.1 Données sur les spécialités

Le nombre de jours maximal séparant deux programmations d'une même spécialité est utilisé afin de générer aléatoirement, pour chaque spécialité, un nombre maximal de jours séparant deux programmations compris entre zéro et le nombre maximal (inclus).

Le nombre minimal de salles à affecter à une spécialité pour un jour au maximum est utilisé afin de générer aléatoirement, pour chaque spécialité et chaque jour, un nombre minimal de salles à affecter à cette dernière compris entre zéro et le nombre maximal (inclus).

```
SPECIALITE 1
        Nombre d'interventions: x
        Nombre maximum de jours séparant deux programmations: y
        NOMBRE MINIMUM DE SALLES A AFFECTER
        ------------------------------------------------------------
        Jour    J1      J2      J3      J4      J5
        Nb/Jour a       b       c       d       e
```

FIGURE 3.1 – Données générées en rapport avec une spécialité

3.2.2 Données sur les interventions

Le nombre maximal d'interventions par spécialité est défini par l'utilisateur. Pour chaque spécialité, le programme génère un nombre aléatoire d'interventions compris entre un et ce nombre maximal (inclus). Ce nombre généré, les durées minimales et maximales d'interventions et d'hospitalisation, ainsi que le nombre maximal et minimal d'arrivées, sont utilisés afin de générer aléatoirement les données ci-dessous.

Durées d'intervention

La durée d'intervention de chaque intervention type pour chaque spécialité est générée aléatoirement entre la durée minimale et la durée maximale d'intervention (inclues).

Durées d'hospitalisation

La durée d'intervention de chaque intervention type pour chaque spécialité est générée aléatoirement entre un jour et le double de la durée de notre période.

Nombre d'arrivées

Le nombre d'arrivées pour chaque intervention type est généré aléatoirement entre le nombre minimal et le nombre maximal d'arrivées (inclus).

```
INTERVENTION 1
        Durée de l'intervention: xmn
        Durée de l'hospitalisation: yj
        Nombre d'arrivées: z
```

FIGURE 3.2 – Données générées en rapport avec une intervention type

3.2.3 Données sur les salles

Le nombre de salles et le nombre d'arrivées doivent être lié. En effet, si notre problème a un trop grand nombre d'arrivées pour le nombre de salles disponibles, aucune solution ne sera trouvée. A l'inverse, on ne veut pas que notre problème soit trop facile à résoudre. Il nous faut donc une cohérence dans nos données.

Voici la démarche suivie afin d'obtenir le nombre de salles ainsi que les durées d'ouverture pour chaque jour nécessaires, en fonction des heures d'ouvertures minimales et maximales et du nombre d'arrivées :

- On calcule tout d'abord, pour notre période, la durée totale en heures de toutes les interventions types mises bout à bout, en fonction du nombre d'arrivée de chacune d'entre elles.
- Tant que l'on a pas généré suffisamment d'heures d'ouvertures de salles (leur somme est inférieure à la durée calculée précédemment), on ajoute une salle. On génère aléatoirement les durées d'ouvertures pour chaque jour de cette dernière, comprises entre la durée d'ouverture minimale et la durée d'ouverture maximale (inclues).

Algorithm 1 $GENERER\_DONNEES\_SALLES()$

$P \leftarrow 0$
for all $s \in S$ **do**
  for all $k \in k_S$ **do**
    $P \leftarrow D + p_k[s][k] * \tau_k[s][k]$
  end for
end for
$N \leftarrow 0$
$H \leftarrow 0$
while $H < (P * \alpha)$ **do**
  $N \leftarrow N + 1$
  for all $j \in J$ **do**
    $h_oj[N][j] \leftarrow RAND(h_{min}, h_{max})$
    $H \leftarrow H + h_oj[N][j]$
  end for
end while

```
Nombre de salles: x
Salle   S1      ...     Sx
Jour_1  a       ...     b

Jour_y  c       ...     d
```

FIGURE 3.3 – Données générées sur les salles

3.2.4 Données sur les ressources

Comme nous l'avons vu dans la présentation du problème, il existe plusieurs types de ressources pour notre problème, des ressources communes à toutes les spécialités, ainsi que des ressources propres à chacune d'entre elles. Chaque intervention a besoin d'un certain nombre de ressources communes et un certain nombre de ressources propres.

Ressources communes

Les besoins en un type de ressources communes pour l'ensemble des interventions types et la quantité de ce type de ressource doivent être liés. Le principe de calcul de la quantité nécessaire pour chaque type ressources communes est le suivant :

- Pour un type de ressource donné, on calcule le besoin maximal de ce dernier, toutes interventions types confondues.
- On multiplie ce besoin maximal par le nombre de salles, afin d'obtenir le nombre maximal de ressources du type pouvant être mises à disposition simultanément. La quantité est volontairement générée large de manière à obtenir des problèmes plus contraints en nombre de lits et de salles.

```
NOMBRE DE RESSOURCES COMMUNES
-------------------------------------------------------------
Nombre de types: x
Type    T1      ...     Tx
Nb/type a       ...     b
Jour_1  c       ...     d

Jour_y  e       ...     f
```

FIGURE 3.4 – Données générées sur les ressources communes

```
BESOINS EN RESSOURCES PROPRES
-------------------------------------------------------------
Type    T1      ...     Tx
Besoins a       ...     b
BESOINS EN RESSOURCES COMMUNES
-------------------------------------------------------------
Type    T1      ...     Ty
Besoins c       ...     d
```

FIGURE 3.5 – Données générées en rapport avec le besoin d'une intervention type

Ressources propres

Sur le même principe, le besoin en un type de ressources propres des différentes interventions types d'une même spécialité et la quantité de ce type de ressources doivent être liés. Le principe de calcul de la quantité nécessaire pour chaque type de ressources propres est le suivant :

- Pour une spécialité et un type de ressource donné, on calcule le besoin maximal de ce dernier, toutes interventions de la spécialité en question confondues.
- On multiplie ce besoin maximal par le nombre de salles, afin d'obtenir le nombre maximal de ressources du type pouvant être mises à disposition simultanément.

3.2.5 Données sur les lits

Les durées d'hospitalisation des différentes interventions types, le nombre d'arrivées et la durée de notre période doivent être liés. Le principe de calcul du nombre de lits est le suivant :

- On fait la somme des durées d'hospitalisation de chaque intervention type, et on la multiplie par le nombre d'arrivées de l'intervention en question.
- On divise cette somme par la durée de notre période, puis on multiplie le résultat par le coefficient pour le calcul du nombre de lits défini par l'utilisateur, afin d'obtenir notre nombre de lits.

Modélisations mathématiques

Le projet s'inscrit dans un travail de recherche réalisé en collaboration avec le CIRRELT de Montréal. Des recherches avaient d'ors et déjà été effectuées en amont du projet, sur les différentes modélisations mathématiques qu'il était possible de mettre en place pour la résolution du "Master Surgical Scheduling Problem".

Deux modèles m'ont donc été livrés au début du projet. Mon premier travail, avant la génération de données, a été de comprendre la modélisation de chacun, dont l'approche est très différente, et d'en corriger les éventuelles erreurs.

4.1 Modèle "classique"

Le premier modèle est un simple PLNE (Problème Linéaire en Nombres Entiers) visant à affecter pour chaque salle et chaque jour une spécialité en particulier et un certain nombre d'interventions types lui appartenant. Ces différentes affectations sont représentées par l'ensemble de variables définies ci-dessous.

4.1.1 Variables

Variables décisionnelles

- x_{ojk} un entier égal au nombre d'interventions de type k programmées dans la salle o le jour j.
- y_{ojk} un binaire égal à 1 si au moins une intervention de type k est programmée dans la salle o le jour j, et à 0 sinon.
- z_{ojs} un binaire égal à 1 si la spécialité s est programmée dans la salle o le jour j, et à 0 sinon.

Variables intermédiaires

- v_{ojus} un entier égal au besoin en ressources de type u propre à la spécialité s pour le jour j dans la salle o.
- w_{ojuc} un entier égal au besoin en ressources communes de type u pour le jour j dans la salle o.

4.1.2 Contraintes

Liaison des x_{ojk} et des y_{ojk}

Nous verrons par la suite que les variables x_{ojk} et y_{ojk} sont nécessaires à la modélisation de notre problème. Le but de la contrainte suivante est de lier ces dernières afin de forcer la variable binaire y_{ojk} à 1 dès lors que x_{ojk} est supérieur à 0.

$$\forall j, \forall o, \forall k, \frac{x_{ojk}}{\frac{max(h_{oj})}{p_k}} \leq y_{ojk} \leq x_{ojk} \tag{4.1}$$

Liaison des x_{ojk} et des z_{ojs}

Le but de la contrainte suivante est de s'assurer que la spécialité s est programmée dans la salle o le jour j dès lors qu'au moins une des interventions type de cette spécialité y est programmée.

$$\forall o, \forall j, \forall s, \frac{\sum_{k \in K_s} x_{ojk}}{\lceil \frac{max(h_{oj})}{min_{k \in K_s}(p_k)} \rceil} \leq z_{ojs} \leq \sum_{k \in K_s} x_{ojk} \tag{4.2}$$

Respect de l'affectation des spécialités aux salles et aux journées

Comme nous l'avons vu dans la problématique du "Master Surgical Scheduling Problem", notre but est de planifier au niveau tactique les spécialités, autrement dit affecter pour chaque jour j et chaque salle o une et une seule spécialité au maximum.

$$\forall j, \forall o, \sum_{s} z_{ojs} \leq 1 \tag{4.3}$$

Respect des durées d'ouverture des salles

Une salle opératoire n'a pas forcément la même disponibilité chaque jour (due à des contraintes extérieures, comme le nettoyage par exemple). Le but de la contrainte suivante est de s'assurer que l'on ne programme pas plus d'interventions que possible dans la salle o le jour j, connaissant les différentes durées opératoires et les différentes durées d'ouverture pour chaque salle et chaque jour.

$$\forall o, \forall j, \sum_{k} x_{ojk} \times p_k \leq h_{oj} \tag{4.4}$$

Respect des ressources communes pour une journée

Comme nous l'avons vu dans les données du problème, il existe deux types de ressources, des ressources communes à toutes les spécialités et des ressources propres à chaque spécialité. Le but de la contrainte suivante est de s'assurer que l'on utilise pas plus de ressources communes de type u qu'il y en a de disponibles pour le jour j. Afin d'avoir un PLNE, il est nécessaire de passer par la variable intermédiaire w_{ojuc} pour modéliser cette contrainte.

$$\forall o, \forall j, \forall u \in U_c, \forall k, y_{ojk} \times m_{ku} \leq w_{ojuc} \tag{4.5}$$

$$\forall j, \forall u \in U_c, \sum_{o} w_{ojuc} \leq c^c_{ju} \tag{4.6}$$

Respect des ressources propres pour une journée

Sur le même principe que la contrainte précédente, le but de la contrainte suivante est de s'assurer que l'on utilise pas plus de ressources de type u propres à la spécialité s qu'il y en a de disponibles pour le jour j.

$$\forall o, \forall j, \forall u \in U_s, \forall k, y_{ojk} \times m_{ku} \leq v_{ojus} \tag{4.7}$$

$$\forall j, \forall u \in U_s, \sum_{o} v_{ojus} \leq c^s_{ju} \tag{4.8}$$

Respect du nombre minimum de salles à attribuer à une spécialité un jour donné

Comme nous l'avons vu dans la présentation du problème, il existe des normes gouvernementales au Québec visant à stopper l'augmentation des listes d'attente. La première norme oblige à programmer un nombre défini de salles pour la spécialité s le jour j. Le but de la contrainte suivante est de s'assurer du respect de cette norme.

$$\forall j, \forall s, \sum_{o} z_{ojs} \geq c_{min_{sj}} \tag{4.9}$$

Respect de l'intervalle minimum entre deux programmations d'une spécialité

La seconde norme oblige à programmer la spécialité s de telle manière à ce que l'intervalle séparant deux programmations ne dépasse pas un nombre maximum g de jours défini. Le but de la contrainte suivante est de s'assurer du respect de cette norme.

$$\forall g \in J, \forall s, \quad \sum_{j=g}^{g+Fmax_s} \sum_{o} z_{(j \bmod J)s} \geq 1 \tag{4.10}$$

Respect du nombre d'arrivées

Le raison principale de la modélisation du "Master Surgical Scheduling Problem" est l'augmentation des listes d'attente. Cette augmentation est due au fait que lors d'une période, on programme moins d'interventions qu'il n'y a d'arrivées. Le but de la contrainte suivante est d'assurer la conservation du nombre de personnes dans les listes d'attente. Nous devons donc traiter autant (voire plus si possible) d'interventions types k qu'il n'y a d'arrivées pour cette dernière (à un certain coefficient près).

$$\forall k, \sum_{o} \sum_{j} x_{ojk} \geq \tau_k \times \text{coefficient} \tag{4.11}$$

Respect du nombre de lits

La plus grosse difficulté de notre problème est due à la cyclicité de notre programmation. En effet, une durée d'hospitalisation varie beaucoup suivant les interventions, et il peut arriver que cette durée soit supérieure à celle de notre période. Il faut tenir compte de cette spécificité dans notre programmation puisque nous avons un nombre de lits fixé. Le but de la contrainte suivante est de s'assurer que le nombre de personnes occupant des lits le jour j respecte le nombre de lits disponibles pour ce jour, en tenant compte de la périodicité.

$$\beta = \begin{cases} 1 \text{ si } g > j \\ 0 \text{ sinon} \end{cases}$$

$$\forall j, \sum_{o} \sum_{g=0}^{|J|-1} \sum_{k} x_{ogk} \times (\lceil \frac{l_k + (g-j)}{|J|} \rceil - \beta) \leq b_j \tag{4.12}$$

Le schéma 4.1 montre la difficulté liée à la cyclicité pour l'affectation des lits, due au fait qu'une intervention type puisse avoir une durée d'hospitalisation supérieure à la durée de notre période. On constate qu'à partir de la troisième semaine sur notre schéma, on obtient un besoin en nombre de lits stable. C'est ce besoin que la contrainte détermine.

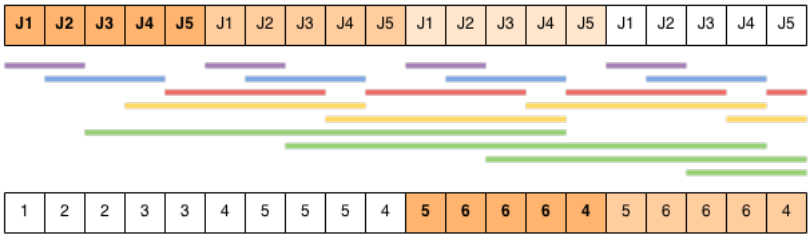

FIGURE 4.1 – Exemple montrant la difficulté de programmation due à la cyclicité pour l'affectation des lits

4.1.3 Fonction objectif

Le "Master Surgical Scheduling Problem" est un problème très contraint. L'objectif premier est d'obtenir une solution réalisable. Le but principal étant de respecter les listes d'attentes, nous avons fixé comme objectif une simple maximisation pondérée du nombre d'interventions à programmer, de manière à prévoir une marge dans le cas où le nombre d'arrivées pour une certaine intervention serait supérieur à la moyenne.

$$\sum_{k} \text{ponderations}_k \sum_{o} \sum_{j} x_{ojk} \tag{4.13}$$

4.2 Modèle avec journées types

Le second modèle est relativement différent dans l'approche. Il est basé sur la génération de colonnes. Nous n'allons pas chercher à programmer une spécialité et un ensemble d'interventions types de celle-ci pour un jour et une salle donnée, mais une journée type.

4.2.1 Principe

Une journée type est une journée composée d'un certain nombre d'interventions appartement à une même spécialité. Le principe de construction est simple. Il s'agit d'une simple énumération de toutes les combinaisons possibles, en sachant que la somme des durées opératoires de l'ensemble des interventions composant la journée type ne doit pas dépasser la durée d'ouverture maximale des différentes salles sur toute la période.

Afin de trouver l'ensemble des journées types, j'ai écris l'algorithme récursif ci-dessous. La fonction est appelée pour chacune des spécialités, afin de trouver l'ensemble des journées types qui lui sont associées. N est un tableau comprenant le nombre de chaque intervention type de la spécialité. Le premier appel à la fonction est réalisé avec un tableau vide.

Algorithm 2 $TROUVER\_JT\_SPE(s, N)$

```
D ← 0
for all k ∈ K_S do
    D ← D + N_k * p_k
end for
Ajouter N dans les journées types testées
if D ≤ h_max then
    Ajouter N dans les journées types de la spécialité
    for all k ∈ K_S do
        if (N_k + 1) * p_k ≤ h_max then
            N_k ← N_k + 1
            if N n'existe pas dans les testés then
                TROUVER_JT_SPE(s, N)
            end if
        end if
    end for
end if
```

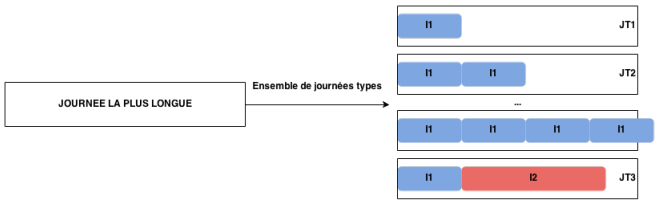

FIGURE 4.2 – Exemple illustrant l'algorithme permettant de construire l'ensemble de journées types

4.2.2 Données sur les journées types

De nouvelles données liées aux journées types sont nécessaires à la résolution de ce second modèle.
- Ω_s l'ensemble des journées types de la spécialité s
- Δ_t la durée de la journée type t
- α_k^t le nombre d'interventions types k pour la journée type t
- γ_{tu}^c le besoin en ressources communes de type u pour la journée type t
- γ_{tu}^s le besoin en ressources de type u propre à la spécialité s pour la journée type t

4.2.3 Variables

- x_{ojt} un binaire égal à 1 si la journée type t est programmée dans la salle o le jour j, et à 0 sinon.
- w_{js} un binaire égal à 1 si la spécialité s est programmée le jour j, et à 0 sinon.

4.2.4 Contraintes

Liaison des x_{ojj} et des w_{js}

Le but de la contrainte suivante est de s'assurer que la spécialité s est programmée dans la salle o le jour j dès lors qu'au moins une des interventions type de cette spécialité y est programmée.

$$\forall s, \frac{\sum_o \sum_{t \in \Omega_S} x_{ojt}}{|O|} \leq w_{js} \leq \sum_o \sum_{t \in \Omega_S} x_{ojt} \tag{4.14}$$

Respect des durées d'ouverture des salles

Comme nous l'avons vu dans le principe de construction des journées types, celles-ci respectent une durée égale à la durée d'ouverture maximale de notre période (h_{max}). Le but de la contrainte suivante est de supprimer toutes les possibilités de programmation d'une journée type t à la salle o le jour j si cette dernière ne respecte pas les horaires d'ouverture.

$$\forall j, \forall o, x_{ojt} = 0 \text{ si } h_{oj} < \sum_k \alpha_k^t \times p_k \tag{4.15}$$

Respect de l'affectation des journées types aux salles et aux journées

Le but de la contrainte suivante est d'affecter au maximum une journée type à la salle o et le jour j. Il se peut que les autres contraintes fassent qu'il est impossible de placer une journée type dans une salle un jour donné, quelle que soit la spécialité associée. La salle sera donc vide dans ce cas là, mais c'est évidemment un cas que nous souhaitons éviter.

$$\forall j, \forall o, \sum_t x_{ojt} < 1 \tag{4.16}$$

Respect des ressources communes pour une journée

Le problème étant le même pour les deux modèles, nous allons avoir un certain nombre de contraintes équivalentes. Seule leur formulation sera différente, due au fait que les variables ne sont pas les mêmes. Le but de la contrainte suivante est de s'assurer que l'on utilise pas plus de ressources communes de type u qu'il y en a de disponibles pour le jour j.

$$\forall j, \forall u \in U_c, \sum_o \sum_t x_{ojt} \times max(m_{ku}) \leq c_{ju}^c \tag{4.17}$$

Respect des ressources propres pour une journée

Le but de la contrainte suivante est de s'assurer que l'on utilise pas plus de ressources de type u propres à la spécialité s qu'il y en a de disponibles pour le jour j.

$$\forall j, \forall u \in U_s, \sum_{o} \sum_{t \in \Omega_S} x_{ojt} \times max(m_{ku}) \leq c^s_{ju} \tag{4.18}$$

Respect du nombre minimum de salles à attribuer à une spécialité un jour donné

Le but de la contrainte suivante est de s'assurer que le nombre de salles attribué à la spécialité s le jour j respecte le nombre minimum fixé par les normes québécoises.

$$\forall j, \forall s, \sum_{o} \sum_{t \in \Omega_S} x_{ojt} \geq c_{min_{sj}} \tag{4.19}$$

Respect de l'intervalle minimum entre deux programmations d'une spécialité

Le but de la contrainte suivante est de programmer la spécialité s de telle manière à ce que l'intervalle séparant deux programmations ne dépasse pas le nombre maximum g de jours fixés par les normes québécoises.

$$\forall g \in J, \forall s, \sum_{j=g}^{g+Fmax_s} w_{(j \bmod J)s} \geq 1 \tag{4.20}$$

Respect du nombre d'arrivées

Le but de la contrainte suivante est d'assurer la conservation du nombre de personnes dans les listes d'attente. Nous devons donc traiter autant (voire plus si possible) d'interventions types k qu'il n'y a d'arrivées pour cette dernière (à un certain coefficient près).

$$\forall k, \sum_{o} \sum_{j} \sum_{t} x_{ojt} \times \alpha^t_k \geq \tau_k \times \text{coefficient} \tag{4.21}$$

Respect du nombre de lits

Le but de la contrainte suivante est de s'assurer que le nombre de personnes présentes à l'hôpital le jour j respecte le nombre de lits disponibles pour ce jour, en tenant compte de la périodicité.

$$\beta = \begin{cases} 1 \text{ si } g > j \\ 0 \text{ sinon} \end{cases}$$

$$\forall j, \sum_{o} \sum_{g=0}^{|J|-1} \sum_{t} x_{ogt} \sum_{k} (\alpha^t_k \times (\lceil \frac{l_k + (g-j)}{|J|} \rceil - \beta) \leq b_j \tag{4.22}$$

4.2.5 Fonction objectif

De manière à comparer les deux modèles, la fonction objectif doit être la même, soit une maximisation pondérée du nombre d'interventions.

$$max(\sum_{k} ponderations_k \sum_{o} \sum_{j} \sum_{t} x_{ojt} \times \alpha^t_k) \tag{4.23}$$

4.3 Comparaison des résultats

Le travail suivant la compréhension et la correction des modèles a été l'implémentation de ceux-ci en C++ sous Visual Studio 2012, en utilisant la librairie Cplex pour la résolution des modèles mathématiques. J'ai généré plusieurs types d'instances, de manière à comparer les deux modèles dans des cas très différents de "Master Surgical Scheduling Problem".

4.3.1 Types d'instances générées

Comme nous l'avons vu dans la partie sur la génération de données, l'utilisateur a la possibilité de paramétrer un certain nombre de variables. Les bornes de certaines de ces variables ont été fixées pour l'ensemble de nos types d'instances, ces variables influençant moins la résolution du problème que celles que nous avons choisies de faire varier, et n'ayant pas le temps de tester toutes les combinaisons.

- Durées d'intervention : entre 30 minutes et 4 heures.
- Durées d'hospitalisation : entre 1 jour et deux fois la durée de notre période.
- Nombre d'arrivées : entre 10 et 20 personnes.
- Horaires d'ouverture : entre 8 heures et 10 heures.
- Nombre de ressources communes : 5 types.
- Nombre maximal de ressources propres pour une spécialité : 5 types.
- Besoin maximal par type de ressources : 5 unités.
- Coefficient pour le calcul du nombre de lits : 1,1.
- Coefficient pour le respect des listes d'attente : 0,9.

Les variables ci-dessous varient donc suivant les types d'instances. Le choix des différentes valeurs de ces variables a été réalisé suite à des tests (notamment pour les valeurs des coefficients), et suivant ce qui peut se pratiquer dans les hôpitaux québécois.

- Durée de la période : 5 ou 10 jours.
- Nombre de spécialités : 3 ou 5.
- Nombre maximal d'interventions par spécialité : 3 ou 5.
- Coefficient pour le calcul du nombre de salles : 1,1 ou 1,5. La valeur de ce coefficient va beaucoup influencer le type de problème que nous allons avoir. Pour une valeur faible le problème va être contraint en nombre de salles, alors que pour une valeur élevée le problème va être contraint en nombre de lits.

La durée maximale de recherche de la solution optimale par Cplex a été fixée à 30 minutes, de manière à pouvoir comparer un grand nombre d'instances.

4.3.2 Instances contraintes en nombre de lits

Comme nous venons de le voir, un des paramètres que nous avons fait varier est le coefficient pour le calcul du nombre de salles. Nous avons commencé par comparer les instances contraintes en nombre de lits, soit avec un coefficient égal à 1,5. Pour chaque modèle, nous avons choisi de comparer dans un premier temps la faisabilité, et dans un second temps les temps d'exécution.

Statuts des instances

Nb Jours	**5**				**10**				
Nb Spé.	**3**		**5**		**3**		**5**		
Nb Inter. max	**3**	**5**	**3**	**5**	**3**	**5**	**3**	**5**	
O	31	17	8	8	14	13	5	3	25%
R	36	23	8	10	21	15	16	4	33%
OOM	8	25	34	31	8	19	17	16	40%
NR	6	2	8	9	21	16	17	30	27%

TABLE 4.1 – Statuts des instances contraintes en nombre de lits pour le modèle classique

Légende :
- **O** : instances résolues à l'optimalité.
- **R** : instances réalisables.
- **OOM** : instances ayant généré des problèmes de mémoire.
- **NR** : instances non réalisables.

Nb Jours	**5**				**10**				
Nb Spé.	**3**		**5**		**3**		**5**		
Nb Inter. max	**3**	**5**	**3**	**5**	**3**	**5**	**3**	**5**	
O	44	45	42	35	29	30	27	17	67%
R	44	47	42	37	29	30	33	18	70%
OOM	0	1	0	4	0	4	0	2	3%
NR	6	2	8	9	21	16	17	30	27%

TABLE 4.2 – Statuts des instances contraintes en nombre de lits pour le modèle avec journées types

Comme nous pouvons le constater dans les tableaux ci-dessus et les graphiques 4.3 et 4.4, une grande part des instances du modèle classique ont engendré un problème de mémoire, dû à la trop grande complexité de celui-ci. La plupart de ces instances sont en revanche résolues à l'optimalité dans le second modèle. On aurait pu s'attendre à avoir le même problème, du au nombre exponentiel de variables, étant donné le nombre important de journées types.

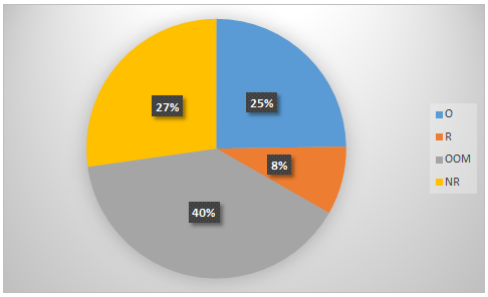

FIGURE 4.3 – Modèle classique

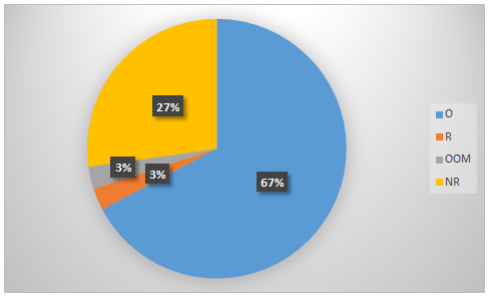

FIGURE 4.4 – Modèle avec journées types

Temps d'exécution des instances

Afin de comparer les deux modèles de manière équitable, soit sans prendre en compte les problèmes de mémoire, la comparaison des temps d'exécution est réalisée sur les instances ayant obtenu des résultats optimaux pour les deux modèles.

Nb Jours	**5**				**10**			
Nb Spé.	**3**		**5**		**3**		**5**	
Nb Inter. max	**3**	**5**	**3**	**5**	**3**	**5**	**3**	**5**
Moy	18,2	80,9	52,7	281,6	14,1	18,3	259,2	12,0
Min	0,1	0,8	2,5	23,7	0,4	1,5	36,6	12,0
Max	1256,4	603,7	255,8	1138,1	85,8	53,8	548,9	12,0

TABLE 4.3 – Temps d'exécution des instances contraintes en nombre de lits pour le modèle classique (en s)

Nb Jours	**5**				**10**			
Nb Spé.	**3**		**5**		**3**		**5**	
Nb Inter. max	**3**	**5**	**3**	**5**	**3**	**5**	**3**	**5**
Moy	6,0	30,5	1,8	257,1	8,0	7,8	32,6	13,3
Min	0,05	0,2	0,4	3,0	0,1	0,3	3,8	13,3
Max	124,6	285,4	6,1	749,5	62,4	27,3	56,4	13,3

TABLE 4.4 – Temps d'exécution des instances contraintes en nombre de lits pour le modèle avec journées types (en s)

Les tableaux ci-dessus montrent que le modèle avec journées types est en moyenne plus rapide que le modèle classique, hormis pour le dernier type d'instance. Ce dernier n'est cependant pas comparable du fait qu'il n'y ait qu'une seule instance ayant obtenu un résultat optimal pour les deux modèles.

4.3.3 Instances contraintes en nombre de salles

Après les instances contraintes en nombre de lits, nous avons comparé les instances contraintes en nombre de salles, soit avec un coefficient égal à 1,1. Sur le même principe que précédemment, nous comparerons dans un premier temps la faisabilité, et dans un second temps les temps d'exécution des deux modèles.

Statut des instances

Nb Jours	5				10				
Nb Spé.	3		5		3		5		
Nb Inter. max	3	5	3	5	3	5	3	5	
O	26	1	0	0	10	18	1	1	14%
R	34	10	9	1	15	20	4	4	24%
OOM	5	36	31	43	10	9	20	25	45%
NR	11	4	10	6	25	21	26	21	31%

TABLE 4.5 – Statuts des instances contraintes en nombre de salles pour le modèle classique

Nb Jours	5				10				
Nb Spé.	3		5		3		5		
Nb Inter. max	3	5	3	5	3	5	3	5	
O	39	46	37	43	25	29	24	25	67%
R	39	46	40	44	25	29	24	27	68%
OOM	0	0	0	0	0	0	0	2	1%
NR	11	4	10	6	25	21	26	21	31%

TABLE 4.6 – Statuts des instances contraintes en nombre de salles pour le modèle avec journées types

Comme pour les instances contraintes en nombre de lits, une grande part des instances du modèle classique a engendré un problème de mémoire, et la plupart de ces instances sont résolues à l'optimalité dans le second modèle.

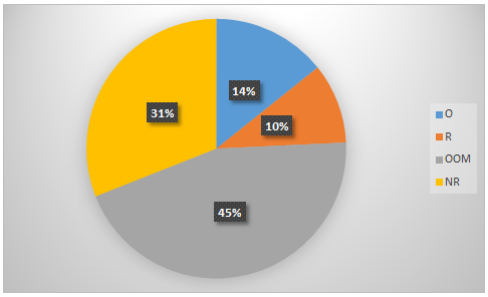

FIGURE 4.5 – Modèle classique

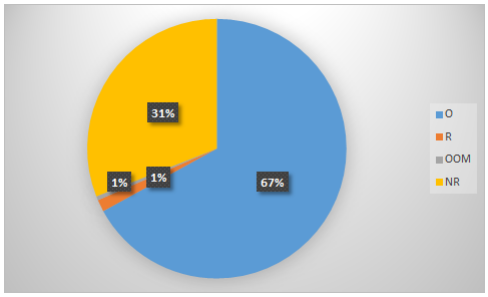

FIGURE 4.6 – Modèle avec journées types

Temps d'exécution des instances

Ici encore, la comparaison des temps d'exécution est réalisée sur les instances ayant obtenu des résultats optimaux pour les deux modèles. Comme nous allons le constater, certains types d'instances n'ont qu'une voire aucune instance dans ce cas, du fait du nombre important d'instances ayant engendré des problèmes de mémoire pour le modèle classique.

Nb Jours	5				10			
Nb Spé.	3		5		3		5	
Nb Inter. max	3	5	3	5	3	5	3	5
Moy	87020,4	49205	x	x	124077	30414	1771905	204765
Min	507	49205	x	x	858	172	1771905	204765
Max	1148195	49205	x	x	517802	208058	1771905	204765

TABLE 4.7 – Temps d'exécution des instances contraintes en nombre de salles pour le modèle classique (en s)

Nb Jours	5				10			
Nb Spé.	3		5		3		5	
Nb Inter. max	3	5	3	5	3	5	3	5
Moy	695,2	209	x	x	4447,5	847,5	440	14342
Min	46	209	x	x	140	44	440	14342
Max	4252	209	x	x	31543	7937	440	14342

TABLE 4.8 – Temps d'exécution des instances contraintes en nombre de salles pour le modèle avec journées types (en s)

4.3.4 Conclusion de la comparaison

Nous aurions pu nous attendre à avoir de meilleurs résultats avec le modèle classique du fait du nombre élevé de variables que le modèle avec journées types peut engendrer (notamment pour les types d'instances ayant un nombre de jours, de spécialités et d'intervention important).

Cependant, après avoir implémenté les deux modèles et les avoir testé sur des types d'instances très différents et dans les mêmes conditions, nous nous sommes aperçu que le modèle classique avait beaucoup de mal à résoudre le problème, engendrant des problèmes de mémoires dus au trop grand nombre d'opérations pour trouver la solution optimale. Le modèle avec journées types, quand à lui, arrive dans plus de 95% des cas (sur les instances réalisables, contraintes en nombre de lits ou de salles) à trouver une solution optimale, et plus rapidement que le modèle classique en règle générale.

Nous avons, au vu des résultats, décidé de nous concentrer dans un premier temps sur l'amélioration du modèle avec journées types avec des coupes, et dans un second temps sur l'implémentation d'une métaheuristique basée sur les journées types.

Tests de coupes pour le modèle avec journées types

L'étape suivant la comparaison des modèles a été de réfléchir à des coupes pouvant permettre l'amélioration des résultats du modèle avec journées types. Ces coupes ont pour but de réduire l'ensemble de solutions, et ainsi de diminuer le temps de calcul de Cplex. Nous avons ainsi trouvé trois coupes ayant la possibilité d'engendrer de meilleurs résultats. Mon travail a été d'implémenter et de comparer les résultats avec et sans coupes (coupe par coupe).

5.1 Coupe 1

5.1.1 Objectif

Nous avons constaté la présence de solutions symétriques dans notre ensemble de journées types. L'objectif de cette première coupe est de supprimer ces solutions symétriques.

5.1.2 Modélisation

$$\forall j, \forall o, h_{oj} \nearrow, h_{oj} = h_{(o+1)j}, \sum_t t \times x_{ojt} \leq \sum_t t \times x_{(o+1)jt} \tag{5.1}$$

5.1.3 Résultats

Type d'instance	**5_3_3**		**5_5_5**	
Coupe	**Non**	**Oui**	**Non**	**Oui**
Moy	3633	38482	145596	351439
Min	36	48	850	1514
Max	38773	1038815	1396598	1753560

TABLE 5.1 – Comparaison des temps d'exécution avec et sans la première coupe pour le modèle avec journées types et les instances ayant obtenu un résultat optimal (en s)

5.2 Coupe 2

5.2.1 Objectif

Cette seconde coupe a pour objectif de réaliser un préprocessing. En effet, dès lors que les normes gouvernementales obligent la programmation de la spécialité s le jour j, le binaire w_{js} doit être égal à 1.

5.2.2 Modélisation

$$\forall j, \forall s, w_{js} = 1 \text{ si } c_{min_{sj}} > 0 \tag{5.2}$$

5.2.3 Résultats

Type d'instance	**5_3_3**		**5_5_5**	
Coupe	**Non**	**Oui**	**Non**	**Oui**
Moy	2036	2046	77542	77591
Min	47	47	1311	1263
Max	21397	21358	563618	564012

TABLE 5.2 – Comparaison des temps d'exécution avec et sans la seconde coupe pour le modèle avec journées types et les instances ayant obtenu un résultat optimal (en s)

Comme nous pouvons le constater dans le tableau, les temps d'exécution sont quasiment équivalents en ajoutant la coupe au modèle avec journées types. On peut en conclure que Cplex réalise ce préprocessing de lui même, et donc que la coupe n'améliore pas les résultats.

5.3 Coupe 3

5.3.1 Objectif

Là encore, cette coupe a pour objectif de réaliser un préprocessing. Une salle ne peut contenir qu'une seule spécialité, mais une spécialité peut être programmée dans plusieurs salles lors de la même journée. On en déduit que pour une journée donnée, le nombre de spécialités programmées ne dépasse pas le nombre de salles.

5.3.2 Modélisation

$$\forall j, \sum_{s} w_{js} \leq |O| \tag{5.3}$$

5.3.3 Résultats

Type d'instance	5_3_3		5_5_5	
Coupe	**Non**	**Oui**	**Non**	**Oui**
Moy	20113	20298	0	0
Min	140	138	0	0
Max	387194	398891	0	0

TABLE 5.3 – Comparaison des temps d'exécution avec et sans la troisième coupe pour le modèle avec journées types et les instances ayant obtenu un résultat optimal (en s)

Comme nous pouvons une nouvelle fois le constater dans le tableau, les temps d'exécution sont quasiment équivalents en ajoutant la coupe au modèle avec journées types. Cplex réalise là encore ce préprocessing de lui même.

5.4 Conclusion sur les coupes

Les coupes trouvées n'améliorent pas le modèle mathématique avec journées types. Les préprocessings auxquels nous avions pensés sont réalisés par Cplex en amont de la recherche de la solution optimale. Nous pensions avoir de bons résultats en supprimant les symétries, mais cette coupe entraine une difficulté supplémentaire à la résolution du problème.

Nous en avons conclu que le modèle ne pouvait pas être amélioré d'un point de vue de temps d'exécution. Cette amélioration pourrait cependant être réalisée à l'aide d'une métaheuristique.

Implémentation d'une métaheuristique

Un des travaux de recherche les plus importants de ce projet a été la mise au point d'une métaheuristique traitant le "Master Surgical Scheduling Problem". Le principe est le même que celui du second modèle mathématique vu précédemment. Nous allons chercher à affecter une journée type pour chaque jour et chaque salle.

6.1 Principe

Notre métaheuristique est composée de trois modules principaux, qui seront articulées afin de réaliser l'affectation d'une journée type pour chaque jour et chaque salle :

- Un module permettant de choisir quelle spécialité sera affectée pour chaque salle et chaque jour. Ce module est composé d'une fonction réalisant cette affectation pseudo-aléatoirement, et une autre tenant compte des résultats d'ors et déjà trouvés.
- Un module permettant de choisir un ensemble de journées types suivant des critères de sélection. Là encore, deux fonctions seront implémentées, une première aléatoire et une seconde plus "intelligente".
- Un module contenant un PLNE qui va, compte tenu de la programmation des différentes spécialités et de l'ensemble de journées types, chercher l'ordonnancement qui respecte l'ensemble des contraintes et qui maximise le nombre d'interventions programmées de manière pondérée.

Le but recherché est de diminuer le nombre de contraintes du PLNE ainsi que l'espace de solutions afin qu'il trouve une solution plus rapidement.

6.2 Affectation des spécialités

6.2.1 Objectif

L'objectif de cette phase est de déterminer quelle spécialité sera programmée dans une salle donnée un jour donné. Les contraintes importantes à respecter lors de cette phase sont les suivantes :

- Il faut choisir les spécialités de telle manière à satisfaire la contrainte du nombre maximum de jours entre deux programmations.
- Il faut également les choisir en tenant compte du nombre minimum de salles à affecter à chaque spécialité pour chaque jour.
- Enfin, il faut les choisir de telle manière à ce qu'on ne prenne pas le risque d'avoir un besoin en ressources supérieur à ce qui est disponible.

6.2.2 Choix aléatoire

La première version pour la programmation des spécialités consiste en une programmation pseudo-aléatoire des salles (tout en respectant les contraintes définies ci-dessus).

Étape 1

La première étape consiste à programmer la spécialité s le jour j dans $c_{min_{sj}}$ salles choisies aléatoirement parmi les salles disponibles ce jour-ci. Cette étape permet de supprimer la contrainte suivante du modèle mathématique :

$$\forall j, \forall s, \sum_{o} \sum_{t \in \Omega_S} x_{ojt} \geq c_{min_{sj}}$$

Étape 2

La seconde étape consiste à programmer les spécialités triées par $Fmax$ chaque $Fmax_s$ jour à partir d'un jour de la période choisi aléatoirement (dans la mesure où la solution est réalisable) dans les salles disponibles. Cette étape permet de supprimer la contrainte suivante du modèle mathématique :

$$\forall g \in J, \forall s, \sum_{j=g}^{g+Fmax_s} w_{(j \bmod J)s} \geq 1$$

Il est nécessaire de s'assurer que le problème sera réalisable lorsque nous fournirons la solution de l'affectation des spécialités aux salles au PLNE. Notons $c_{max_{sj}}$ le nombre maximum d'affectations de la spécialité s le jour j, calculé en fonction des besoins de ses interventions types.

$$c_{max_{sj}} = \max_{k} \left(\min_{u} \left(\lfloor \frac{c_{ju}}{m_{ku}} \rfloor \right) \right)$$

Étape 3

La troisième étape consiste à calculer pour chaque spécialité le nombre minimum de salles à lui affecter pour la période, en fonction des durées opératoires de ses interventions et des arrivées. Notons $c^J_{min_s}$ ce nombre pour la spécialité s. L'algorithme ci-dessous permet de calculer ce nombre pour chaque spécialité :

Algorithm 3 $CHERCHER\_CMIN\_PERIODE(s)$

$D \leftarrow \sum_{k \in S} p_k \times \tau_k \times \text{coefficient}$
$c^J_{min_s} \leftarrow 0$
$H \leftarrow \emptyset$
$i \leftarrow 1$
for all $o \in O$ **do**
  for all $j \in J$ **do**
    $H_i \leftarrow h_{oj}$
    $i \leftarrow i + 1$
  end for
end for
$SORT(H)$
for all $h \in H$ **do**
  if $D > 0$ **then**
    $D \leftarrow D - h$
    $c^J_{min_s} \leftarrow c^J_{min_s} + 1$
  else
    $BREAK$
  end if
end for

Une fois les calculs effectués pour chaque spécialité, les spécialités n'ayant pas été programmées suffisamment sont programmées aléatoirement dans les salles restantes, toujours en respectant pour chaque spécialité et chaque jour le $c_{max_{sj}}$.

Étape 4

La dernière étape consiste à programmer aléatoirement des spécialités dans les salles restantes, là encore en respectant pour chaque spécialité et chaque jour le $c_{max_{sj}}$.

6.2.3 Choix par roulette

La seconde version pour la programmation des spécialités consiste en une programmation des spécialités par roulette. Soit $Sbest$ la meilleure solution trouvée (ayant programmé un maximum d'interventions), respectant toutes les contraintes.

Étape 1

La première étape est basée sur la version aléatoire. On réitère les trois premières étapes de telle manière à obtenir un début de solution S.

Étape 2

La seconde étape consiste à calculer pour S ainsi que pour $Sbest$ les taux d'occupation pour chaque spécialité ainsi que leur temps de programmation respectifs (basé sur les horaires d'ouverture des salles dans le cas de S), respectivement notés τ_s, t_s, τ_s^{best} et t_s^{best}.

Une fois les calculs effectués, on choisit par roulette une spécialité, en tenant compte de la probabilité que cette dernière a d'être prise. Cette probabilité est calculée pour chaque spécialité en fonction de la différence $\tau_s^{best} \times t_s^{best} - \tau_s \times t_s$. Une fois la spécialité choisie, elle est affectée aléatoirement à une salle,

à condition que le $c_{max_{sj}}$ soit respecté. On réitère ainsi l'étape 2, jusqu'à ce que toutes les salles soient remplies.

6.3 Choix du sous ensemble de journées types

6.3.1 Notations

Soient :
- Ω l'ensemble des journées types
- Ω' le sous ensemble à considérer
- Ω_s l'ensemble des journées types de la planification S

6.3.2 Objectif

L'objectif de cette phase est de construire un sous ensemble Ω' de Ω composé d'un nombre X de journées types. Les contraintes importantes à respecter lors de cette phase sont les suivantes :
- Il faut choisir les journées types de telle manière à satisfaire au mieux le nombre d'arrivées pour chacune des interventions.
- Il faut également les choisir en tenant compte des horaires d'ouverture, afin de remplir au mieux nos journées.
- Enfin, il faut les choisir de telle manière à ce qu'on ne prenne pas le risque de dépasser la capacité en nombre de lits.

6.3.3 Choix aléatoire

La première version pour le choix de journées types pour notre sous ensemble est de les choisir au hasard parmi $\Omega \setminus \Omega_s$ un sous ensemble Ω' composé de X journées types. Cette version nous servira de base de comparaison pour l'autre version que nous allons voir ci-dessous.

6.3.4 Choix des X meilleurs scores par tournoi

La seconde version pour le choix de journées types est réalisée sous forme de tournoi. Afin d'obtenir une solution réalisable et d'essayer d'améliorer notre solution, des journées types vides sont sélectionnées pour chacune des spécialités, ainsi que les journées types de la meilleure solution S.

Clustering

Avant de procéder au choix des journées types pour notre sous ensemble, nous avons choisi de regrouper les différentes horaires d'ouverture existantes en trois paquets, en suivant la procédure suivante :
- On liste les différentes horaires d'ouverture des salles lors de notre période.
- On regroupe ensuite les deux horaires les plus proches en un seul (c'est à dire que si nous avions 5h, 7h, 8h et 10h comme différents horaires, nous avons désormais 5h, 7h30 et 10h). On réitère cette étape jusqu'à n'avoir plus que trois paquets.

Une fois les paquets regroupés, on calcule le pourcentage d'ouvertures de salles correspondant à chacun d'entre eux, avec α, β et γ ces pourcentages respectifs.

Scoring

Pour chacun des paquets, on sélectionne les journées types ayant comme durée Δ_t un temps plus faible que la durée d'ouverture maximum du paquet. Ensuite, on affecte un score à chacune de ces journées types sur ce principe :

- Si le nombre d'interventions de type k sur la période ne respecte pas le nombre d'arrivées, on ajoute un point à la journée type t pour chaque intervention de type k qu'elle possède.
- On ajoute $\frac{\alpha_k^t}{max(\alpha_k^t)}$ points à la journée type t afin de favoriser les journées types ayant beaucoup d'interventions.
- Si dans notre planification S on a un nombre de lits occupés d'au moins 90% tous les jours, alors on retire $\frac{l_k}{max(l_k)}$ points à la journée type t.

Une fois le clustering et le scoring effectués, on procède à la sélection par tournoi, en suivant le principe suivant :

- On tire au hasard quatre journées types correspondant au paquet pour lequel nous souhaitons choisir des journées types.
- On compare les scores de chacune d'entre elles et on garde pour notre sous ensemble la meilleure.
- On réitère ce procédé jusqu'à avoir sélectionné αX, βX ou γX journées types suivant le paquet.

6.4 PLNE

6.4.1 Objectif

L'objectif de cette phase est de trouver la meilleure solution, autrement dit la solution respectant l'ensemble des contraintes et maximisant le nombre d'interventions, compte tenu de l'affectation préalable des spécialités aux salles et l'ensemble réduit des journées types disponibles.

6.4.2 Variables

- x_{ojt} un binaire égal à 1 si la journée type t est programmée dans la salle o le jour j, et à 0 sinon.
- y_k un entier égal au nombre d'interventions types k manquantes dans la programmation.

6.4.3 Préprocessing

Le but de ce préprocessing est de forcer le x_{ojt} à 0 dès lors que la spécialité ayant été programmée en amont dans la salle o le jour j ne correspond pas à celle de la journée type t.

Algorithm 4 $SUPPRIMER\_JT\_IMPOSSIBLES()$

```
for all o ∈ O do
  for all j ∈ J do
    for all t ∈ Ω' do
      if AFFECTATION[o][j] ≠ SPECIALITE[t] then
        x_ojt ← 0
      end if
    end for
  end for
end for
```

6.4.4 Contraintes

Respect de l'affectation des journées types aux salles et aux journées

Le but de la contrainte suivante est d'affecter au maximum une journée type à la salle o et le jour j.

$$\forall j, \forall o, \sum_{t} x_{ojt} < 1 \tag{6.1}$$

Respect des durées d'ouverture des salles

Le but de la contrainte suivante est de supprimer toutes les possibilités de programmation d'une journée type t à la salle o le jour j si cette dernière ne respecte pas les horaires d'ouverture.

$$\forall j, \forall o, x_{ojt} = 0 \text{ si } h_{oj} < \sum_{k} \alpha_k^t \times p_k \tag{6.2}$$

Respect des ressources communes pour une journée

Le but de la contrainte suivante est de s'assurer que l'on utilise pas plus de ressources communes de type u qu'il y en a de disponibles pour le jour j.

$$\forall j, \forall u \in U_c, \sum_{o} \sum_{t} x_{ojt} \times max(m_{ku}) \leq c_{ju}^c \tag{6.3}$$

Respect des ressources propres pour une journée

Le but de la contrainte suivante est de s'assurer que l'on utilise pas plus de ressources de type u propres à la spécialité s qu'il y en a de disponibles pour le jour j.

$$\forall j, \forall u \in U_s, \sum_{o} \sum_{t \in \Omega_S} x_{ojt} \times max(m_{ku}) \leq c_{ju}^s \tag{6.4}$$

Respect du nombre d'arrivées

La modélisation de cette contrainte diffère du modèle avec journées types que nous avons vu dans la comparaison des deux modèles. Nous allons en effet passer par la variable intermédiaire y_k. Le but des contraintes suivantes est d'assurer la conservation du nombre de personnes dans les listes d'attente.

$$\forall s, \forall k, y_k = max\left(\tau_k - \sum_{o} \sum_{j} \sum_{t \in \Omega'} \begin{cases} x_{ojt} * \alpha_k^t \text{ si SPECIALITE[t]} = \text{s} \\ 0 \text{ sinon} \end{cases}, 0\right) \tag{6.5}$$

$$\forall s, \forall k, \sum_{o} \sum_{j} \sum_{t \in \Omega'} \begin{cases} x_{ojt} * \alpha_k^t \text{ si SPECIALITE[t]} = \text{s} \\ 0 \text{ sinon} \end{cases} \geq (\tau_k * \text{coefficient}) - y_k \tag{6.6}$$

Respect du nombre de lits

Le but de la contrainte suivante est de s'assurer que le nombre de personnes présentes à l'hôpital le jour j respecte le nombre de lits disponibles pour ce jour, en tenant compte de la périodicité.

$$\beta = \begin{cases} 1 \text{ si } g > j \\ 0 \text{ sinon} \end{cases}$$

$$\forall j, \sum_{o} \sum_{g=0}^{|J|-1} \sum_{t} x_{ogt} \sum_{k} \alpha_k^t \times \left(\lceil \frac{l_k + (g-j)}{|J|} \rceil - \beta \right) \leq b_j \tag{6.7}$$

6.4.5 Fonction objectif

Nous avons relâché la contrainte sur le respect du nombre d'arrivées en passant par la variable intermédiaire y_k. La fonction objectif doit avoir pour buts de minimiser cet y_k (étant donné qu'il s'agit du nombre d'interventions de type k manquantes), et de maximiser le nombre d'interventions réalisées sur la période (afin de s'approcher de la solution optimale). Nous avons donc la possibilité de trouver un résultat violant la contrainte du nombre d'arrivées, s'il n'est pas possible d'avoir tous les y_k égaux à 0.

$$max \left(\sum_{o} \sum_{j} \sum_{t} x_{ojt} * \alpha_k^t - \sum_{k} 10000 * y_k \right) \tag{6.8}$$

6.5 Algorithme et paramètres

Comme nous l'avons dit dans le principe de la métaheuristique, nous allons articuler les différents modules entre eux à l'aide de paramètres afin de rechercher pour un problème la meilleure solution possible. L'algorithme ci-dessous présente le principe de recherche de la solution.

Algorithm 5 $METAHEURISTIQUE()$

```
for i de 1 à NB_MAX_IT_SALLES do
  AFFECTATION_POSSIBLE ← false
  for j de 1 à NB_MAX_ECHECS et !AFFECTATION_POSSIBLE do
    if i = 1 then
      AFFECTER_SPECIALITE_ALEATOIREMENT()
    else
      AFFECTER_SPECIALITE_PAR_ROULETTE()
    end if
    if AFFECTATION_REALISEE then
      AFFECTATION_POSSIBLE ← true
    end if
  end for
  if AFFECTATION_POSSIBLE then
    CHOISIR_JOURNEES_TYPES_ALEATOIREMENT()
    RESOUDRE_PLNE()
    NUM_IT_SANS_AMELIORATION ← 0
    while NUM_IT_SANS_AMELIORATION < NB_IT_SANS_AMELIORATION
    do
      NUM_IT_SANS_AMELIORATION ← NUM_IT_SANS_AMELIORATION + 1
      CHOISIR_JOURNEES_TYPES_PAR_TOURNOI()
      RESOUDRE_PLNE()
      if MEILLEURE_SOLUTION then
        NUM_IT_SANS_AMELIORATION ← 0
      end if
    end while
  end if
end for
```

A chaque appel de la fonction de résolution du PLNE, on compare la solution venant d'être trouvée à la meilleure solution trouvée jusque là. Si la solution venant d'être trouvée a un meilleur ratio de respect du nombre d'arrivées, alors on remplace l'ancienne meilleure solution par celle-ci. On la remplace également si la solution a le même ratio et un nombre d'interventions plus important.

Les variables écrites en rouge dans l'algorithme ci-dessus représentent les différents paramètres en relation avec l'articulation de nos modules. Suite à des tests, nous avons fixé le nombre maximum d'échecs dans l'affectation des salles à 3. Nous avons donc quatre paramètres à faire varier de manière à trouver la meilleure combinaison :

- N, le nombre maximum d'itérations pour l'affectation des spécialités aux salles.
- M, le nombre maximum d'itérations sans amélioration de la solution.
- X, le coefficient pour le calcul de journées types. Si X est égal à 2, alors nous sélectionnerons un total de deux fois le nombre de jours multiplié au nombre de salles, journées types de la meilleure solution et journées types vides comprises.
- Y, le nombre de concurrents pour le tournoi.

Conclusion

En conclusion de ce rapport, je pourrais dire que les objectifs fixés au début de ce projet ont été atteints. Je me suis approprié les deux modèles et j'ai pu les comparer avant de développer une métaheuristique, de la coder et de la tester avec différents jeux de données.

Les résultats trouvés lors de la comparaison des deux modèles ont mis en avant une approche basée sur la génération de colonnes, pour laquelle on n'attendait pas de bons résultats du fait de la possibilité d'avoir un grand nombre de variables. Nous aurions en revanche peut-être pu trouver de meilleurs résultats pour le modèle classique avec une machine dotée de plus de mémoire. La tentative d'amélioration n'a quand à elle pas apporté de meilleurs résultats, Cplex réalisant déjà le travail de préprocessing auquel nous avions pensé. Les résultats obtenus par la métaheuristique sont quand à eux encourageants. Nous trouvons en effet des résultats comparables au modèle avec journées types du point de vue de la faisabilité, mais le nombre d'interventions types programmées n'est quand à lui pas à l'optimalité dans la plupart des cas.

Le projet n'ayant pas été axé que sur la métaheuristique, toutes les combinaisons de paramètres n'ont pas pu être testées. Il serait donc sans doute possible d'améliorer les résultats, en trouvant le jeu de paramètres le plus efficace et en réalisant une affectation des spécialités aux salles et une sélection des journées types plus intelligente.

Le fait que ce projet puisse permettre aux hôpitaux de réaliser une meilleure programmation des interventions, en tenant compte de l'ensemble des contraintes, et afin de diminuer les listes d'attentes, a rendu ce projet très intéressant. Le travail de recherche m'a permis de découvrir un domaine dans lequel je pourrais travailler après mes études. J'ai également eu l'opportunité de présenter mon projet à la ROADEF, ce qui a pu m'apporter une expérience très enrichissante.

Comparaison de modèles mathématiques et implémentation d'une métaheuristique pour le Master Surgical Scheduling Problem

Département Informatique
5e année
2013 - 2014

Rapport de Projet de Fin d'Études

Résumé : Le MSSP est un problème clé pour la gestion des blocs opératoires des hôpitaux québécois. Ce problème intervient au niveau stratégique, il s'agit d'affecter pour chaque jour et chaque salle une spécialité sur un horizon de plusieurs mois. L'idée est de trouver un macro planning périodique. L'objectif du projet était dans un premier temps de comprendre puis comparer deux modèles mathématiques livrés au début de celui-ci. Dans un second temps, l'objectif était de réaliser un travail de recherche afin d'améliorer les modèles, puis de comparer les résultats obtenus. La dernière phase du projet visait à implémenter une métaheuristique permettant la résolution du problème et à trouver les paramètres permettant d'obtenir les meilleurs résultats possible.

Mots clefs : MSSP, ordonnancement, planning, blocs opératoires, chirurgie, modélisation, métaheuristique

Abstract: The MSSP is a key issue for the management of operating rooms in Quebec hospitals. This problem occurs at the strategic level to assign to every day and every room, a specialty over a period of several months. The idea is to find a macro periodic schedule. The project was initially to understand and compare two mathematical models delivered at the beginning of it. In a second step, the goal was to create a research work to improve the models, and then compare the results. The last phase of the project was to implement a metaheuristic to solve the problem and find the settings to get the best possible results.

Keywords: MSSP, scheduling, schedule, operating rooms, surgery, modeling, metaheuristic

---

Encadrant
Yannick Kergosien
yannick.kergosien@univ-tours.fr

Étudiant
Antoine GIRET
antoine.giret@etu.univ-tours.fr

École polytechnique de l'université de Tours,
Tours

DI5 2013 - 2014