Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Projet Recherche & Développement
2015-2016

Gestion et Optimisation des Parcours Patients

Tuteurs académiques
Yannick Kergosien

Étudiants
Jean Coquelet (DI5)

26 septembre 2016

Liste des intervenants

Nom	Mail	Qualité
Jean Coquelet	jean.coquelet@etu.univ-tours.fr	Étudiant DI5
Yannick Kergosien	yannick.kergosien@univ-tours.fr	Tuteur académique, Département infomatique

Avertissement

Ce document a été rédigé par Jean Coquelet susnommé l'auteur.

L'école polytechnique de l'université François Rabelais de Tours est représentée par Yannick Kergosien susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnait assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnait qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document :
Jean Coquelet, *Gestion et Optimisation des Parcours Patients*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2015-2016.

```
@mastersthesis{
    author={Coquelet, Jean},
    title={Gestion et Optimisation des Parcours Patients},
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2015-2016}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V1.0 (2016/02/01)

Table des matières

Table des figures

B Diagramme de Gantt prévisionnel

Liste des tableaux

Liste des codes sources

Introduction

Ce projet s'inscrit dans le cadre de mon projet de recherche et développement (PRD), au sein de l'école Polytech Tours, lors de la 5ème année d'étude au département informatique.

Le projet de gestion et d'optimisation de parcours patients est un ensemble de projets au sein de Polytech Tours. C'est un projet en deux aspects : la gestion et l'optimisation. La partie gestion a été développée dans le cadre du projet de SI de 5ème année, la partie d'optimisation est l'aspect principal de ce PRD.

Les deux projets sont sous l'égide de Yannick Kergosien, maître de conférences au département informatique de l'Ecole Polytechnique de Tours, et sous la tutelle de Lucie Roussel, ingénieur en ordonnancement à l'Assistance Publique - Hôpitaux de Paris (AP-HP) et maître d'ouvrage (MOA).

Le projet de gestion consiste au développement d'un système d'information comportant une interface, une base de données et une application mettant en oeuvre les deux éléments précédents. Celle-ci a été développée au cours du semestre 9 par six étudiants de 5ème année, dont je fus le chef de projet. Cette application permet d'ajouter des patients, des ressources médicales, du personnel médical, de gérer les parcours et les activités médicales dont le projet d'optimisation aura besoin par la suite. C'est depuis cette application que l'utilisateur pourra optimiser les planifications et avoir une visibilité accrue sur le fonctionnement général du système.

Un projet de 4ème année est également en cours de développement, réalisé par Chenghao Wang et Fu Dejin. Ils ont contibué conjointement à la modélisation du problème, ainsi qu'au développement d'un valdiateur de solutions et d'une heuristique permettant de calculer une solution d'ordonnancement.

Première partie

Partie I - Recherche

1 Présentation du problème

Le chapitre qui suit correspond en grande partie au cahier des charges qui a été formulé lors de la période de recherche.

1 Contexte & définition

La question des dépenses de santé dans le monde est un sujet prédominant. D'un côté, les Etats cherchent à réduire les charges de fonctionnement. D'autre part, ils essayent de maintenir une qualité de vie supérieure. Le vieillissement démographique accru - dans les pays occidentaux principalement - accentue un besoin nécessaire et croissant du monde médical. Dans les hôpitaux, la problématique est identique : il est nécessaire de réduire le gaspillage budgétaire, tout en cherchant à maximiser la satisfaction des patients, avec des moyens – au mieux – stagnants. Afin d'améliorer l'organisation des prises de rendez-vous, des outils numériques intégrant des techniques de la Recherche Opérationnelle peuvent être mis en place. Ces derniers permettent de gérer de meilleure façon l'utilisation des ressources médicales. Ils prennent en compte également les imprévus et aléas du système. L'usage de ces outils conduisent à des temps d'attente moins longs pour les patients, qui en tireront satisfaction.

Pour désengorger les services hospitaliers écrasés sous la charge des patients venant pour des besoins différents avec des priorités différentes, les centres de soins mettent en place de plus en plus des services dédiés à l'accueil de patients pour une matinée, un après-midi ou même durant une journée complète. Ces services, principalement accessibles par rendez-vous, vont également permettre au patient de ne pas rester hospitalisé sur plusieurs jours, ce qui impliquerait pour lui de rester une ou plusieurs nuits sans que cela s'avère nécessaire pour lui, et consommera presque inutilement des ressources (chambre, lits, personnel de nuit de surveillance, etc.). Lorsqu'un établissement, ou partie de cet établissement, propose un tel service diurne, celui-ci est nommé « hôpital de jour ». Dans la littérature anglo-saxonne, on retrouve le terme de « outpatient clinic ».

Les parcours cliniques (en anglais Clinical pathways, CP) sont des plans de soins pluridisciplinaires structurés exposant en détail les étapes essentielles que doivent suivre les patients présentant un problème clinique spécifique (Kinsman et al. (2010)). Ceux-ci se décomposent en plusieurs activités prédéfinies, certaines devant être effectuées avant d'autres (contraintes de précédence), nécessitant des ressources multiples et diverses au sein d'un même service ou de plusieurs (contraintes de ressources), dans des délais et créneaux horaires journaliers (contraintes de délais, contraintes temporelles). Il n'existe que peu d'exemples où les parcours patients (tels qu'on les a défini ci-dessus) sont mis en oeuvre, étant donné la rigidité certaine entre les services d'un même établissement, et la difficulté d'avoir un outil central permettant le bon fonctionnement de cette communication pluridisciplinaire. Or, ce type d'organisation devrait permettre une meilleure utilisation des ressources matérielles, médicales et humaines. De plus, l'utilisation de parcours cliniques est susceptible d'avoir un effet favorable sur les résultats des patients, la durée d'hospitalisation, les coûts hospitaliers et les pratiques professionnelles (Kinsman et al. (2010)).

L'Assistance Publique - Hôpitaux de Paris (AP-HP), dans son plan stratégique 2015 - 2019, a lancé une réflexion sur la mise en place d'une clinique ambulatoire. Avec la mutualisation de certains services au sein de différents hôpitaux de la région, le site de l'hôpital Béclère à Clamart obtiendra un gain de place suffisant pour proposer une telle solution. Il regroupera tous les hôpitaux de jour existants du site, avec un management centralisé et transversal à l'aide de parcours patients comme décrit ci-dessus. Les équipes médicales et administratives ont pris le soin de définir d'ors et déjà une vingtaine de parcours patients en adéquation avec les ressources déjà en place, et présenté une répartition type des prises en charge hebdomadaire. Celle-ci devrait en théorie permettre d'accueillir 179 patients chaque semaine, et à terme environ 1000 par mois. On comprend donc aisément que la centralisation organisationnelle complexe va nécessiter la conception d'un système d'aide à la décision pour la prise de rendez-vous, prenant en compte les disponibilités des patients, des ressources et du personnel médical.

Le nouveau procédé de prise de rendez-vous et d'organisation de rendez-vous devrait s'articuler en trois étapes distinctes. La première étape est l'affectation des patients à une journée définie, en essayant de répartir au mieux les consommations en ressources et de respecter les journées types (définies lors des travaux préparatoires, réalisés dans le cadre du plan stratégique de l'AP-HP). Puis, environ un mois avant ce jour J, l'infirmière de coordination déclenchera le processus d'ordonnancement, permettant de définir à quelle heure chaque patient devra arriver le jour considéré pour leur première activité médicale. Cette première solution d'ordonnancement est dite hors-ligne (offline). Dès lors que la solution que nous proposons est validée, les patients seront appelés afin de confirmer leur horaire de première activité. S'ensuit la deuxième phase de réordonnancement, ou ordonnancement en-ligne (online). Pour des raisons d'urgence principalement, il est possible que certains patients se rajoutent pour la date considérée, bien que le premier ordonnancement ait été effectué et que certaines activités soient considérées comme immuables dans le temps. La phase temps-réel, troisième et dernière étape, s'effectuera le jour-même via une interface de visualisation des ressources, où l'agent chargé de la coordination pourra dynamiquement réajuster la planification.

2 Identification du besoin

Le but de ce projet est de concevoir un ensemble de programmes permettant l'ordonnancement et la planification de rendez-vous. Ces programmes devront être capable de fournir une solution acceptable d'ordonnancement, respectant les contraintes du problème d'ordonnancement général. Il n'est donc pas nécessaire de chercher la solution optimale.

Chaque programme correspondra à l'un des trois problèmes d'ordonnancement, chacun ayant ses spécificités quant à la réponse à apporter. Le programme chargé du premier ordonnancement sera un programme dit hors-ligne, avec toutes les données d'ordonnancement connues à l'avance. Les programmes de réordonnancement, eux, devront se baser sur les activités déjà planifiées ainsi que sur celles à rajouter ou à retirer.

L'utilisateur interagira avec ce logiciel via l'interface du système d'information. Il déclenchera le processus d'ordonnancement (ou de réordonnancement selon le cas) à l'aide d'un bouton de l'interface qui appellera le logiciel. Cette interaction unique permettra de maintenir une indépendance entre le programme d'ordonnancement et le système d'information associé.

Le programme acceptera en entrée un fichier de données correspondant à toutes les informations nécessaires (ressources, tâches à ordonnancer, etc.) dans un format déterminé. Il créera un fichier contenant la solution de l'ordonnancement dans un format déterminé également. Cette solution sera par la suite traitée afin d'être insérée dans la base de données du système d'information. Étant donné l'aspect multi-critères des problèmes sous-jacents, des paramètres de pondération seront aussi pris en entrée.

3 Définition des objectifs

On va distinguer plusieurs objectifs dans le cadre de ce projet, ceux-ci correspondant aux différentes phases de l'ordonnancement.

3.1 Phase 1 : Première solution d'ordonnancement

On considère lors de cette phase que toutes les données nécessaires pour l'ordonnancement sont connues et déterministes.

Le logiciel devra être capable de fournir une première solution d'ordonnancement. Cette solution doit être réalisable, c'est-à-dire qu'elle correspondra aux contraintes définies dans notre modélisation du problème (contraintes de précédence, de durée, etc.). On n'apporte aucune garantie d'optimalité de cette solution, en revanche on cherchera à minimiser le temps d'attente moyen des patients, ainsi qu'à minimiser le plus grand temps d'attente de notre ensemble de patients.

Le programme devra prendre en entrée un fichier contenant toutes les données nécessaires pour planifier l'ordonnancement, à savoir :

- La liste des patients prévus pour une date déterminée et leurs disponibilités
- La liste des ressources et leurs disponibilités pour cette date
- La liste des parcours patients, des activités médicales et leurs propriétés

Ce programme utilisera ces données dans sa méthode de résolution, et cherchera à trouver un ordonnancement viable. Celui-ci minimisera le temps d'attente moyen de tous les patients ainsi que le pire temps d'attente de l'ensemble des patients, et s'exécutera dans un temps relativement court. Ce temps d'exécution sera fourni en entrée du programme (NB : ce temps ne devrait pas excéder les 5 minutes).

Lorsqu'une solution d'ordonnancement est trouvée et validée (i.e respecte l'ensemble des contraintes du problème), elle sera enregistrée dans un fichier en sortie.

3.2 Phase 2 : Réordonnancement

Lorsque la première solution est déterminée, un ordonnancement est mis en place pour un ensemble de patients fini. Des lors, l'équipe chargée des rendez-vous pourra insérer de nouveaux patients dans le système. Cela impliquera un éventuel réordonnancement des tâches prévues, suite à l'insertion de nouvelles activités médicales.

Le programme chargé du réordonnancement récupérera la planification prévisionnelle, ainsi que le nouvel ensemble d'activités à insérer. Il se chargera d'ajouter ces dernières, en respectant les contraintes du problème général. Il faudra également veiller à ne pas modifier les dates prévisionnelles des tâches déjà planifiées pour lesquelles une modification est formellement proscrite (on considère par exemple que la première activité d'une succession d'activités médicales doit être immuable dans le temps).

En entrée du programme, on retrouve les différentes données des patients et ressources, ainsi que l'ordonnancement prévisionnel. De plus, on aura les données du nouveau patient à incorporer dans l'ordonnancement. En sortie, un nouveau fichier d'ordonnancement sera créé. Lorsque le réordonnancement est effectué et validé, il écrasera le précédent, considéré comme obsolète.

3.3 Phase 3 : Réordonnancement en temps-réel

La troisième phase correspond au jour même de planification. Lorsqu'un incident survient, le système doit être en mesure de gérer la résolution de cet incident sur la planification. Ces incidents peuvent conduire à décaler les activités au sein du planning prévisionnel et accentuer les temps de retards du système. Pour palier à ce genre de situation, l'agent chargé de la coordination doit avoir alors la possibilité de déclencher un réordonnancement, via l'interface du SI.

Au premier regard, les incidents qui peuvent survenir le plus fréquemment lors du fonctionnement du système sont les retards ou les absences des patients. En cas d'absence du patient, il faut retirer toutes les activités qui étaient planifiées pour lui, puis réordonnancer les activités médicales restantes (toujours dans le respect des contraintes). Lorsqu'un patient est en retard pour un rendez-vous médical, le problème est bien plus complexe. On ne sait pas encore a priori si le patient va venir d'une minute à l'autre, considérant que certains patients ne prendront pas la peine de prévenir le personnel médical de leur imprévu. Le risque est donc que ses activités entraînent un décalage de toutes les autres activités, même celles des autres patients. L'idée serait donc de retirer le patient du planning tant que sa présence n'a pas été attestée, puis le réinsérer dans le planning à compter du moment où celui-ci se sera manifesté. A noter que l'interface du système d'information sera en mesure d'avertir l'utilisateur si un incident est détecté.

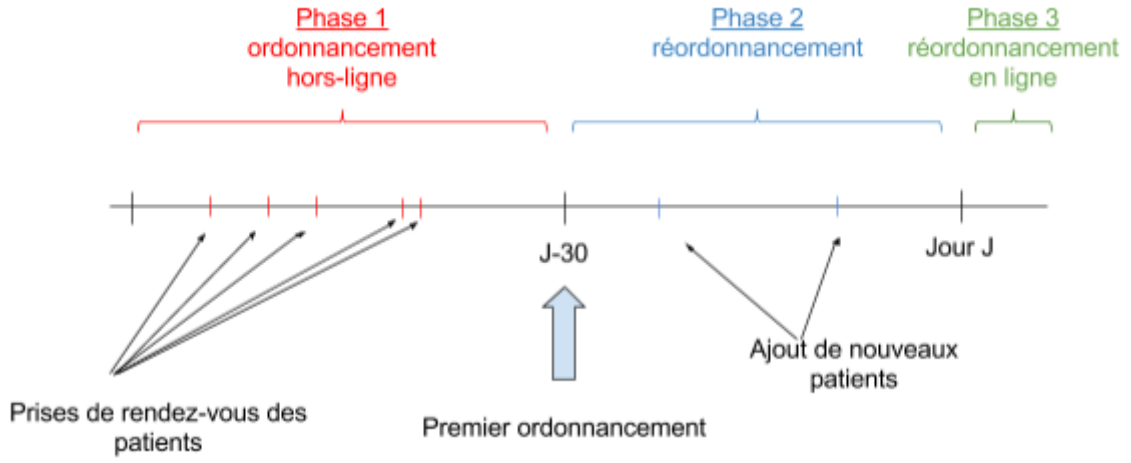


Figure 1 – *Timeline d'exécution de la prise de rendez-vous*

4 Description fonctionnelle & contraintes

On considère que toutes les informations nécessaires au bon fonctionnement du programme sont disponibles avant son exécution. L'agent chargé de la prise de rendez-vous aura dûment rempli les formulaires d'ajout du patient dans le système d'information. La plupart des descriptions fonctionnelles sont communes à tous les livrables.

4.1 Les contraintes du problème

Les problèmes d'ordonnancement sont soumis à des contraintes qui sont listées ci-dessous :

1. On ne modifie pas une activité considérée comme fixée dans le temps.

Lorsque le premier ordonnancement est validé, chaque patient se voit attribuer une heure de rendez-vous, pour commencer son parcours. Or, avec les réordonnancements possibles, il ne faudrait pas que ces premières activités soient déplacées. On va donc considérer que la première activité de chaque patient est fixée dans le temps. Néanmoins, il faut souligner que dans ce cas présent, c'est l'heure de rendez-vous qui est fixée dans le temps, et non pas l'activité en elle-même. Il est possible que la première activité soit permutée avec une autre activité (dans le respect des contraintes de précédence), mais l'activité permutée en première place devra commencer à l'heure qui a été initialement affectée. Certaines activités dont le réordonnancement est rigide (imagerie médicale, consultations externes, etc.) seront considérées comme fixes.

2. On ne planifie pas une activité avant que toutes ses activités précédentes soient terminées (Contrainte de précédence)
3. On ne peut pas planifier deux activités nécessitant le même patient en même temps
4. On ne peut pas planifier deux activités nécessitant la même ressource en même temps

Le personnel et les patients ne peuvent pas faire preuve d'ubiquité. Ces contraintes vont garantir aux patients et aux ressources de ne participer qu'à au plus une activité à tout instant t.

5. Une activité médicale est non-préemptive, c'est à dire que l'on ne peut pas stopper son exécution
6. On ne peut pas planifier une activité pour un patient dans la période de temps précédent sa fenêtre de disponibilité
7. On ne peut pas planifier une activité pour une ressource dans la période de temps précédent sa fenêtre de disponibilité

On se doit de respecter les disponibilités des patients et ressources. A noter que l'on peut dépasser la date limite de fin de disponibilité.

8. Le minimum de temps entre deux activités doit être supérieur ou égal au délai minimum défini entre ces deux activités
9. Le maximum de temps entre deux activités doit être inférieur ou égal au délai maximum défini entre ces deux activités

Certaines activités médicales sont sujettes à des contraintes temporelles entre elles. Il est impératif de les respecter à la lettre.

4.2 Communication avec la base de données & parseur

Le programme devra être capable d'interagir avec la base de données du système d'information, afin de récupérer toutes les données nécessaires pour l'ordonnancement. Cette interaction s'effectuera au travers de fichiers intermédiaires. Le format de ces fichiers est décrit en annexe.

Les fichiers d'entrée seront créés par le système d'information, dès l'appel à l'une des fonctions d'ordonnancement/réordonanncement sur l'interface. Ils seront ensuite transmis au programme qui en extraira les données et qui initialisera les objets en conséquence à l'aide d'un parseur. Lors du réordonnancement, la solution déjà calculée ainsi que les données du nouveau patient seront fournis au programme dans deux fichiers séparés.

Lorsque la solution a été modélisée, le programme exportera dans un fichier l'ordonnancement. Ce dernier sera ensuite traité par le système d'informations afin d'incorporer les modifications à sa base de données.

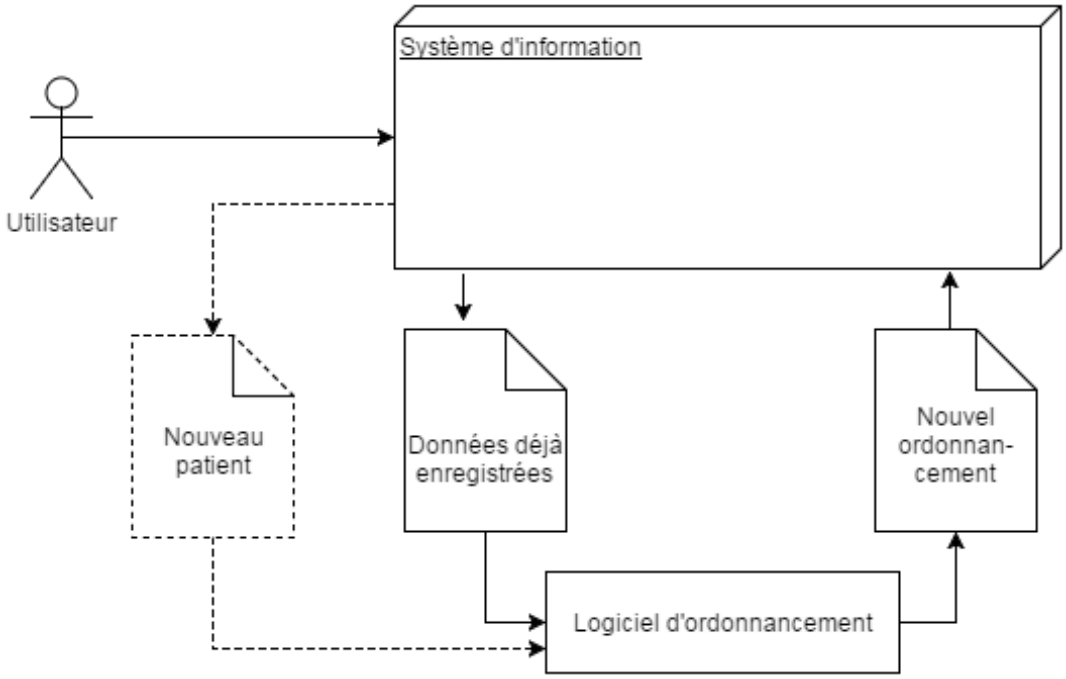


Figure 2 – *Diagramme indiquant les phases de communication entre les objets*

Sur la Figure 2 ci-contre, on peut observer que le logiciel d'ordonnancement sera apte à gérer à la fois le premier ordonnancement (phase 1), mais également tous les réordonnancements (phase 2) en fournissant également les données du (ou des) nouveau(x) patient(s) à insérer dans la planification.

4.3 Méthode de résolution

Une fois le chargement des données et leur initialisation achevés, l'ordonnancement peut débuter. Une ou plusieurs méthodes de résolution seront implémentées. Le principe est élémentaire : à partir des données

et des contraintes, ces méthodes devront proposer une solution d'ordonnancement qui minimisera deux critères :

— le temps d'attente moyen de l'ensemble des patients
— le pire temps d'attente relevé dans l'ensemble des patients

Ces deux critères sont pondérés par une variable d'ajustement. La relaxation des critères d'heures supplémentaires comme vus ci-dessus est elle aussi soumise à une variable d'ajustement. Celle-ci est arbitrairement grande, afin de garantir l'éjection de ces solutions si une solution respectant les fenêtres de temps est trouvée.

L'espace des solutions réalisables peut s'avérer être beaucoup trop restreint selon si les jeux de données fournis sont chargés ou non. Pour palier à cela, on va inclure des solutions ne respectant pas les contraintes des fenêtres de temps des ressources et des patients. Cette surcharge en heures (*overtime*) ne doit pas être privilégiée, mais peut être considéré comme acceptable (relaxation de contrainte).

On peut considérer que les phases 1 et 2 de l'ordonnancement utiliseront la même méthode de résolution (avec des entrées différentes).

4.4 Suite et fin du projet de SI

Ce PRD est développé conjointement avec un projet de SI lors de la 5ème année du département informatique de Polytech Tours dont j'ai été chef de projet. Cela a permis à mon équipe et à moi d'avoir une vue d'ensemble sur les deux aspects distincts, à la fois le système d'information, et également le système logiciel pour le calcul des ordonnancements.

La partie développement du PRD n'ayant pas encore été initialisée, les interactions entre la partie d'ordonnancement et la partie SI n'ont pas été configurées. Il est donc important de souligner que, d'un point de vue fonctionnel, certains aspects du système d'information seront à développer dans le cadre du PRD. On peut notamment citer la création des fichiers intermédiaires entre le SI et le logiciel d'ordonnancement.

5 Contraintes liées au projet

Le projet devra être terminé pour la dernière semaine de mars 2016. Le déroulement du développement est indiqué en détail dans le diagramme de Gantt dans le Chapitre 4

Le langage de programmation utilisé sera le C++/11. Ce langage permet de concevoir des applications robustes en termes d'utilisation de ressources matérielles ainsi qu'en temps d'exécution.

2 Etat de l'art

1 Recherche bibliographique

La conception d'un système d'ordonnancement de rendez-vous (Appointment Scheduling) a été maintes fois traitée dans la littérature en Recherche Opérationnelle. Les travaux de Bailey [1] et Lindley [12] ont ouvert la voie, mettant en avant les problèmes des temps d'attente dans les hôpitaux de jour. Ces deux articles sont considérés comment étant les pionniers du domaine. Cayirli et Veral [5] propose une revue de littérature, rassemblant une grande quantité d'articles publiés avant 2003, traitant des systèmes de rendez-vous pour les patients externes dans les hôpitaux de jour. Ces articles sont triés en fonction de leur définition de leur problème (le nombre de services, le nombre de serveurs, les surcharges de rendez-vous (overbooking), le processus d'arrivée des patients, etc.), de leur fonction objective, ou encore de leur règle d'ordonnancement pour les processus d'arrivée. Cardoën, Demeulemeester et Beliën [3] propose une revue d'articles plus récents, portés sur l'ordonnancement de rendez-vous pour la planification des blocs opératoires au sein des établissements de santé.

1.1 Service unique

On retrouve dans la littérature principalement des articles ne prenant en compte qu'un seul praticien, ou tout du moins qu'un seul service au sein d'un établissement médical composé de plusieurs médecins effectuant le même travail en parallèle. En d'autres termes la consultation du patient ne s'effectue qu'en une seule étape classique, comme chez un médecin généraliste - c'est ce que l'on définit par service unique. C'est le cas de Zacharias et Pinedo [21] et Kaandorp et Koole [11], qui ont un horizon de planification se limitant à la journée considérée. Dans les deux cas, la journée est divisée en intervalles de temps (time slots) qui doivent être attribués aux patients. Zacharias et Pinedo considère, en plus, deux cas de figure. Dans le premier cas de figure, on considère un ensemble de patients homogène, c'est à dire avec des caractéristiques identiques (temps de traitement, probabilité d'absence, etc.). Dans le second cas de figure, on divise l'ensemble des patients en deux sous-ensembles, l'un où les patients ont une probabilité d'absence nulle, l'autre où la probabilité d'absence est variable mais non nulle. Cela permet de générer une population hétérogène. Cet article autorise les médecins à effectuer des heures supplémentaires. Kaandorp et Koole prend en compte des temps de service stochastiques. Toujours en se plaçant dans le cas d'un serveur unique, Feldman, Liu, Topaloglu et Ziya [7] étend son horizon de planification à un ensemble de T jours, en commençant par le jour de prise de rendez-vous par le patient. Il prend en compte notamment les préférences des patients pour les choix des dates de rendez-vous, aspect souvent délaissé dans la littérature. Liu, Ziya et Kulkarni [13] modélise également le problème d'ordonnancement de rendez-vous avec un horizon de planification de T jours, en considérant qu'un rendez-vous demandé à la date t peut être planifié pour ce même jour t (*meeting today's demand today*). En revanche, cet article

Table 1 – *Tableau récapitulatif des différents articles cités*

Auteurs	PP[1]	Méth.[2]	Mod.[3]	Hor.[4]	Qté.[5]	Type[6]	O[7]	N[8]
Kaandorp & Koole	Non	Rech. locale	Stoch.	Jour	1	1	N	O
Liu, Ziya & Kulkarni	Non	Heuristique	MDP	Ens. de j.	1	1	N	O
Zacharias & Pinedo	Non	Heuristique	Stoch.	Jour	1	1	O	O
Feldman et al.	Non	Heuristique	Stoch.	Ens. de j.	*NC*	*NC*	N	O
Zeng et al.	Non	Rech. locale	Stoch.	Jour	1	1	O	O
Turkcan et al.	Oui	Heuristique	ILP	Ens. de j.	1	1	N	N
Swisher et al.	Oui	*NC*	Simulation	Jour	1	N	N	N
Vermeulen et al.	Oui	Rech. locale	Empirique	Jour	*NC*	*NC*	N	N
Schimmelpfeng et al.	Non	CPLEX	ILP	Ens. de j.	N	N	N	N
Braaksma et al.	Oui	CPLEX	ILP	Semaine	1	N	N	N
Gartner & Kolisch	Oui	CPLEX	Graphes	Semaine	1	N	N	N
Gonçalves et al.	Oui	GA	ILP	*NC*	N	N	N	N
Du et al.	Oui	GA	ILP	Jour	N	N	N	N
Hsu et al.	Non	Rech. Tabou	ILP	Semaine	1	N	N	N
Castro & Petrovic	Oui	CPLEX	ILP	Semaine	N	N	N	N
Parizi & Ghate	Non	Relax. Lag.	MDP	Ens. de j.	N	N	O	O
Van der Velde et al	Oui	CPLEX	ILP	Jour	1	N	N	N
Wolf	Oui	*NC*	ILP	Jour	N	N	N	N

ne prend pas en compte les préférences du patient, où l'on considère que le patient accepte la première date de rendez-vous proposée par l'équipe médicale.

1.2 Multi-disciplinarité et parcours patients

Fréquemment, les patients peuvent être amenés à devoir faire plusieurs activités médicales au cours d'un traitement. Les établissements médicaux, regroupant plusieurs disciplines différentes, pourraient donc travailler de concert afin d'accueillir des patients qui suivraient des traitements en naviguant

1. Présence de parcours patients
2. Méthode de résolution
3. Modélisation
4. Horizon de planification
5. Quantité de ressources pouvant être du même type par activités
6. Types de ressources différents dans une même activité
7. Overbooking (Oui/Non)
8. No-shows (Oui/Non)

entre les différents services. La difficulté de mise en place réside dans la rigidité des services, ayant chacun leur propre système d'ordonnancement intrinsèque, conduisant à un manque d'échanges entre les services. A l'heure actuelle, la grande majorité des services au sein d'un hôpital ont des systèmes de rendez-vous indépendants entre les services. On se retrouve alors avec des systèmes d'ordonnancement de rendez-vous décentralisés, qui fonctionnent quasiment selon le même principe que ce qui a été décrit dans les articles du paragraphe précédent.

Matta et Patterson [14] observe la difficulté d'une mise en place d'un tel service de prise de rendez-vous dans une clinique ambulatoire multi-services, en prenant l'exemple du service d'oncologie du CHU de Duke en Caroline du Nord (USA). Ce domaine médical a une sensibilité accrue dans la collaboration entre les services médicaux et est un excellent point d'accroche pour présenter les possibilités de mutualisation des ressources entre les services, en concluant que "malgré l'interdépendance et le fait que les patients sont partagés entre les services, les systèmes de soins ambulatoires ne sont jamais considérés comme un sous-système coordonné d'un centre hospitalier". C'est avec cette approche que Wolf [20] propose une modélisation mathématique des parcours patients, basée sur des contraintes issues de problèmes que l'on retrouve dans le domaine d'ordonnancement d'ateliers dans l'industrie. Les tâches correspondent aux activités médicales, qui vont utiliser des ressources (par analogie, des médecins, infirmières, consommables, etc.) qui peuvent être exclusives (une ressource effectue une activité au plus à un instant t) ou cumulatives (un ensemble d'infirmières peuvent être affectées à différentes tâches en même temps), alternatives (i.e un assortiment de ressources), et qui auront entre elles des éventuelles contraintes de précédence. Cet article est une base solide pour modéliser un parcours patient, et pour son implémentation dans le cadre d'une programmation par contraintes.

Dans la littérature, certains articles se sont penchés sur le cas des activités pluridisciplinaires. Vermeulen et al. [19] décrit un ensemble de départements médicaux, avec pour chaque département un agent de coordination qui va se charger de proposer aux patients des créneaux adaptés à leurs besoins. C'est une approche décentralisée, plus adaptée à la logistique existante dans le domaine médical. Mais cet article ne prend pas en compte les parcours cliniques comme nous l'avons défini, puisqu'il s'agit d'affecter sans réelle précédence des activités successives pour chaque patient, avec pour objectif qu'elles soient planifiées au cours de la même journée dans la mesure du possible.

Schimmelpfeng, Helber et Kasper [16] s'intéresse à un problème d'ordonnancement dans une clinique de réadaptation. Ce genre de structure met en jeu différentes spécialités utilisant diverses ressources communes. On y retrouve le distinguo entre les praticiens et les ressources médicales génériques (infirmières, masseurs, etc.), avec des types de ressources qui diffèrent. Les rendez-vous correspondent à des cheminements d'activités que les patients doivent suivre, et où des ressources médicales de différents types sont affectées. Ces parcours patients ne sont pas clairement définis *stricto sensu*, dans cet article, mais plutôt à la carte, s'adaptant aux besoins du patient. L'ordonnancement est de type hors-ligne. Une clinique de réadaptation est aussi étudiée dans Braaksma et al. [2]. Mais à la différence qu'ici on s'intéresse au problème en-ligne. Les parcours sont également peu définis, on parle plutôt de plan de traitement, qui consiste en une succession d'activités assez flexibles, où l'objectif est d'en assurer une grande partie. L'horizon de planification est la semaine. Les activités doivent être programmées tout au long d'une semaine, avec la possibilité d'en mettre dans les semaines adjacentes, ou même d'en supprimer certaines dans une moindre mesure. Gartner et Kolisch [8] s'intéresse à l'ordonnancement de patients au sein d'une journée. Celui-ci, multidisciplinaire, est un ordonnancement hors-ligne. Cet article prend en compte la notion de parcours patients, avec des regroupements de patients en fonction de la similarité de leur diagnostic (diagnosis-related group, DRG). Ces parcours sont modélisés à l'aide de graphes, où les noeuds représentent les activités médicales, les arcs leur enchaînement et la valuation des arcs le délai minimal entre deux activités.

1.3 No-shows, walk-ins et overbooking

Un problème récurrent dans l'ordonnancement de rendez-vous de patients dans le milieu hospitalier est la gestion des imprévus. Cela se traduit par la possibilité de retards par rapport à l'heure de rendez-vous initial (problème de ponctualité), la présence ou non de patients sans rendez-vous (walk-ins) ou même assez souvent on peut avoir des patients qui ne se présentent pas à leur rendez-vous (no-shows) - cette pratique est monnaie courante dans certaines spécialités, notamment en oncologie, où le traitement ne peut s'effectuer que si le patient est dans de bonnes dispositions. Si celui-ci ne peut pas effectuer son traitement, en particulier lors de chimiothérapies, il faudra donc annuler les réservations et les reporter à une date ultérieure.

N'étant pas à l'abri de ce genre d'imprévus dans la pratique, la plupart des modèles énoncés dans la littérature les prennent en compte. C'est le cas de Kaandorp et Koole [11], qui introduit un facteur de probabilité de non-présence de certains patients. Liu, Ziya et Kulkarni [13] considère, en plus du no-show, la possibilité de retard, à l'aide du modélisation par un processus de décision markovien (MDP). Zacharias et Pinedo [21] ainsi que Zeng, Turkcan, Lin, and Lawley [22] s'intéressent en plus du facteur de no-show à l'overbooking. Cette pratique consiste à assigner un nombre de rendez-vous supérieur aux capacités d'accueil de l'établissement hospitalier. On retrouve cette stratégie notamment dans les réservations de billets dans le transport aérien. Le principe mise sur le fait que certains patients se décommanderont à la dernière minute, ou ne se présenteront tout simplement pas. Le risque de cette pratique, c'est d'avoir des prévisions en deça du taux de présence effectif. Il y aura alors une accumulation de retards, ainsi que des heures supplémentaires (overtime) du côté des ressources médicales.

1.4 Ordonnancement en-ligne/hors-ligne, prise de rendez- vous multi-étapes

La plupart des méthodes de résolution énoncées dans les articles suggèrent des méthodes de prise de rendez-vous, pour contourner l'utilisation de processus stochastiques afin de gérer le flot d'arrivée de demandes des patients. On peut retrouver deux paradigmes pas nécessairement exclusifs : hors-ligne (offline) et en-ligne (online). On définit l'ordonnancement hors-ligne comme étant un ordonnancement où toutes les données du système (tâches, machines, ressources) sont connues à l'avance, déterministes. L'ordonnancement en-ligne, quant à lui, est un ordonnancement où les tâches arrivent au fur et à mesure, et où la solution est réordonnancée en fonction des nouvelles arrivées. Zacharias et Pinedo [21] propose ces deux différents modèles. Kaandorp et Koole [11] ne se concentre que sur l'ordonnancement hors-ligne, tout comme Schimmelpfeng, Helber et Kasper [16]. En revanche, Braaksma et al. [2] traite le problème en-ligne. Turkcan, Zeng et Lawley [17] décrit un système d'ordonnancement de rendez-vous offline en deux étapes, dans un service d'oncologie. Dans un premier temps, on affecte des patients à des jours de consultation, puis une fois l'affectation réalisée, la seconde étape consiste à l'ordonnancement des différentes activités au sein de la journée considérée. Ces activités sont dépendantes de contraintes de précédence, étant donné la rigidité des protocoles dans les traitements chimiothérapiques. Ici aussi, seules les ressources matérielles sont considérées, et non pas les ressources médicales humaines.

1.5 Méthodes & évaluations

Les articles de la littérature proposant des solutions mettent en œuvre des méthodes de résolution différentes, approchées ou optimales. On retrouve différentes heuristiques. Gonçalves, Mendes et Resende [9] suggère l'implémentation d'un algorithme génétique pour la résolution des ordonnancements de projets, Du, Jiang, Yao et Diao [6] propose aussi un algorithme génétique, mais hybride (combiné avec une optimisation par essaims particulaires), permettant de simplifier les traitements pour l'opérateur de mutation lors de l'exécution de l'algorithme génétique. Kaandorp et al. [11] utilise une recherche locale simple. Hsu, Matta et Lee [10] ont développé une méthode de recherche tabou. Liu, Ziya et Kulkarni [13] propose ses propres heuristiques ; Il en reste que l'utilisation de l'Integer Linear Programming (ILP) reste prédominante (par exemple chez Gartner et al. [8], Schimmelpfang et al. [16], Castro et Petrovic [4], Turkcan et al. [17]). Les fonctions d'évaluation vont se distinguer selon le critère que l'on va chercher à minimiser. L'objectif essentiel dans l'immense partie des travaux de recherche à ce sujet est d'améliorer la qualité de service des établissements. On aura donc comme priorité le besoin d'optimiser les temps de traitements des patients, et donc de minimiser leur temps d'attente global dans le système. C'est le cas pour Kaandorp et al. [11], où le critère est le temps d'attente. Feldman et al. [7] met en place un système différent, basé sur des coûts et des revenus, afin de maximiser le "profit". Parizi et Ghate [15] inclut la relaxation lagrangienne dans sa fonction objectif. Turkcan et al. [17] propose comme objectif de minimiser les temps d'accès, les délais de traitement ainsi que les heures supplémentaires du personnel médical.

1.6 Articles pertinents pour l'étude

Parizi et Ghate [15], dans leur article, posent le problème d'ordonnancement d'interventions chirurgicales dans un hôpital. Les rendez-vous arrivent de façon stochastique et dynamique, et un opérateur peut soit

les accepter et les fixer à une certaine date, soit simplement les refuser. L'horizon de planification est établie sur un ensemble de jours T. Les opérations médicales, modélisées par un processus de décision Markovien, sont exécutées par un ensemble de ressources de types différents. Par exemple, une activité chirurgicale peut nécessiter un bloc opératoire pendant deux heures, un chirurgien, deux assistants, une infirmière et un certain équipement médical. Cet article, en revanche, ne fait pas été de succession d'activités médicales. Les ressources sont toujours disponibles, il n'y a pas de fenêtre temporelle de disponibilités. Il n'y a pas de parcours patient, ni de contraintes de précédence entre activités.

Du, Jiang, Yao et Diao [6] aborde un problème très similaire à notre problématique. Celui-ci propose une modélisation mathématique des patients et des ressources. Chaque agent appartient à un ensemble homogène correspondant à son type (médecin, infirmière, infirmière assistante) et possède un identifiant unique lui correspondant. Ici il n'y a que trois types de ressources différentes, mais il est trivial d'en incorporer d'autres dans la modélisation. Ces ressources sont disponibles dès le début de la planification, et n'ont pas de fenêtre de temps de disponibilité. L'article propose un ensemble de contraintes, permettant de s'assurer que la contrainte de précédence des activités est bien respectée, et que le nombre de ressources utilisées est viable. Le cas d'étude est un service de gynécologie, où les patientes sont accueillies sur plusieurs jours. Chaque journée correspond à un parcours patient. L'horizon de planification est donc fixé à la journée. Les successions de tâches sont considérées ici comme étant déterministes et connues à l'avance, définies selon un modèle statique, à la différence de notre problématique, où les enchaînements peuvent varier, certaines activités pouvant se faire avant ou après d'autres sans réelle importance pour le bon cours du traitement du patient. Les contraintes de précédence se chargent d'assurer une cohérence dans la succession des activités. Notre problème va être un peu différent, ici, chaque activité médicale est assurée par au plus une ressource médicale, tandis que nous considérons la possibilité qu'une activité médicale puisse être dispensée par plusieurs ressources en même temps, de types qui divergent. La recherche de solutions s'effectue à l'aide d'un algorithme génétique hybride, qui combine un algorithme génétique avec une optimisation par essaims particulaires.

Dans un autre domaine, Gonçalves et al. [9] cherche à répondre au problème d'ordonnancement avec des contraintes de précédence et de ressources, mais cette fois-ci dans l'organisation de plusieurs projets en parallèle. Cela revient à avoir un ensemble de M parcours, chacun composé de N_M activités. Une activité n'apparaît pas dans deux parcours différents. Cette analogie avec notre problème est intéressante. On retrouve ici l'idée de succession d'activités, avec contraintes de précédence, dans le domaine de l'ordonnancement et de management de projets. Chaque activité j nécessite $R_{j,k}$ ressources de type k, comme pour une activité médicale. Les personnes spécialisées naviguent donc d'un projet à l'autre, selon la nécessité de chaque projet, et ne possèdent pas de fenêtre de disponibilité temporelle. En revanche, l'objectif est de minimiser la date de fin des projets, dans le but de les livrer au plus tôt. Cela ne correspond pas directement à nos attentes dans notre cas présent. La majeure différence avec notre problématique, c'est que chaque ressource n'est pas affectée à une activité du début à la fin de celle-ci. Une activité peut se dérouler tant qu'elle a le nombre suffisant de ressources. Dans notre problème, chaque ressource est affectée à une activité, et sera considérée comme vacante uniquement dès la fin de cette activité. Cet article utilise également un algorithme génétique pour sa résolution. L'horizon de planification est la durée totale de l'ensemble des projets.

On va retrouver une forte similarité entre notre problème et celui décrit dans Van der Velde, Kortbeek et Livak [18]. Cet article, basé sur les travaux d'organisation des planifications de rendez-vous de patients au Centre Médical Académique (AMC) d'Amsterdam aux Pays-Bas, où un centre de diagnostic et de traitement des maladies neuromusculaires pour enfants (CMCA) a ouvert ses portes. Ce genre de service propose des activités médicales avec différentes spécialités du domaine, des neurologues, des physiothérapeutes, du personnel d'imagerie médicale, des psychologues, du personnel de soin, et beaucoup d'autres. L'objectif de ce nouveau système de rendez-vous mis en place est de ramener le nombre de passages au centre pour un patient de 6 visites annuelles à une seule tous les ans. Cela se traduit par une modélisation de parcours patients, chaque parcours nécessitant des ressources diverses, avec des contraintes de précédence. Les rendez-vous sont connus bien à l'avance, et l'ordonnancement des activités se fait un mois à l'avance. L'horizon de planification est de l'ordre de la journée. La résolution de ce problème est réalisée à l'aide d'une ILP, permettant de fournir une solution correspondant à l'enchaînement des activités pour chaque patient et pour chaque praticien de l'établissement, en tenant compte des heures de disponibilités pour chaque ressource. Leur fenêtre de disponibilité est connue à l'avance, et varie d'une ressource à l'autre. Cet article diffère légèrement de notre problème, puisque qu'il considère que certaines activités peuvent être considérées comme optionnelles, et instaure donc un système de priorité des activités. Cela conduit donc à des visites qualifiées de partielles, où le patient n'effectuera pas toutes les activités qu'il désirait (activités qui s'avèrent être considérées comme non

nécessaires). On peut aussi avoir des activités de groupe, aspect que nous ne considérerons pas dans notre problème d'ordonnancement. L'ordonnancement est effectué uniquement en hors-ligne, il n'est pas possible de faire de réordonnancement au cours du temps, étant donné les conditions draconiennes d'éligibilité du patient au programme de diagnostic.

2 Positionnement du projet

A l'heure actuelle, il n'existe pas à proprement parler de logiciel dans le commerce permettant de répondre à notre problème. Au sein des services hospitaliers, la gestion des prises de rendez-vous est très différente selon les établissements. Par manque de moyens matériels, et par la difficulté de la mise en place d'un système d'aide à la décision, les secrétaires médicales chargées de la prise de rendez-vous se retrouvent généralement dans une situation où il est impossible d'optimiser la prise de rendez-vous. En règle générale, chaque service aura son propre planning, et les patients devront systématiquement prendre rendez-vous auprès de chacun des différents services de façon totalement décentralisée. Ces rendez-vous seront inscrits soit dans un registre papier, soit dans un tableur Excel ou un agenda en ligne de type Google Agenda.

Dans le domaine médical, il n'existe que peu de logiciels d'ordonnancement. Ce sont des logiciels qui - au mieux - permettent un suivi des patients au sein d'un établissement. En revanche, on peut être amené à retrouver des logiciels permettant l'ordonnancement de rendez-vous, mais qui s'apparentent davantage à des agendas en ligne, comme TeamAgenda de Teamsoft.

3 Modélisation du problème

On distingue plusieurs entités à modéliser.

1 Les patients

Soit P l'ensemble des n patients à planifier. Un patient $i \in \{1 \dots n\}$ est caractérisé par :

- $[\mathrm{D}_i, \mathrm{F}_i]$ la plage horaire où le patient i est disponible avec $0 \leq \mathrm{D}_i < \mathrm{F}_i \leq 24 \times 60$
- $i\mathrm{P}p_i$ l'indice de parcours du patient i

Un patient peut être considéré comme une ressource dans le cas d'un problème d'ordonnancement d'ateliers. Cela permet de garantir qu'il est présent pendant au plus une activité médicale.

2 Les parcours & activités

Soit PP l'ensemble des m parcours patient. Chaque parcours $j \in \{1 \dots m\}$ a un nombre d'activités nb_j non-préemptives avec pour chaque activité $l \in \{1 \dots nb_j\}$:

- une durée p_j^l
- r_j^l l'ensemble des ressources nécessaires pour réaliser l'activité l. Cet ensemble est représenté par $nb\mathrm{R}_j^l$ couples (n_k, Ω_k), avec $k \in [1 \dots nb\mathrm{R}_j^l]$. (n_k, Ω_k) signifie qu'il faut n_k ressources parmi Ω_k, avec Ω_k la liste des indices des ressources disponibles pour réaliser l'activité.
- $Pred_j^l$ la liste des prédecesseurs de PP_j
- $Succ_j^l$ la liste des successeurs de PP_j
- Délai min $\delta_{min}^{l,l'}$, délai max $\delta_{max}^{l,l'}$ entre deux activités l et l'

On peut considérer qu'une activité est équivalente à une tâche dans un problème d'ordonnancement d'ateliers. A ce titre, cette tâche est non-préemptive, c'est à dire qu'on ne peut l'interrompre en cours d'exécution.

Ω_k est une liste d'indice de ressources d'un même type. Ces ressources doivent être disponibles pour effectuer l'activité. La modélisation diffère légèrement de celle de la base de données du système d'informations ici, puisque celle-ci ne permet pas de fournir une liste de ressources pour chaque activité, mais peut néanmoins proposer la liste des ressources et leurs indiponibilités.

3 Les ressources médicales

Soit R l'ensemble de nR ressources dans l'hôpital de jour. Chaque ressource $r \in [1 \dots n\mathrm{R}]$ est caractérisée par $[\mathrm{O}_r, \mathrm{T}_r]$ une fenêtre de temps pendant laquelle la ressource r est disponible.

Chaque ressource est unique, on les différencie à l'aide d'un identifiant. A l'image d'une machine dans un problème d'ordonnancement d'ateliers, les ressources médicales ont une fenêtre de disponibilité qui leur est propre. Dans les contraintes de notre problème, on peut considérer que la borne supérieure de cette fenêtre peut être transgressée lors de l'ordonnancement.

4 Solution

Une solution pour ce problème est définie par un ordonnancement d'activité médicales pour chaque ressource. Elle est donc représentée par la liste des ressources R, avec pour chaque ressource la liste des activités affectées à cette ressource (i.e paire patient/activité dans son parcours), chaque paire est caractérisée par $t_{p,r}$ la date de début et $c_{p,r}$ la date de fin de l'activité.

On évaluera une solution selon la somme pondérée des durées d'attente.

4 Planning prévisionnel du développement

		Nom	Durée	Début	Fin	Prédécesseurs
1		Livrable 1 - Ordonnancement	30j?	04/01/2016	12/02/2016	
2		Interaction avec la base de données	5j?	04/01/2016	08/01/2016	
3		Parseur	10j?	11/01/2016	22/01/2016	2
4		Méthode de résolution 1	20j?	11/01/2016	05/02/2016	
5		Tests - Benchmark	5j?	08/02/2016	12/02/2016	3,4
6		Livrable 2 - Réordonnancement	15j?	15/02/2016	04/03/2016	1
7		Méthode de résolution réordo	10j?	15/02/2016	26/02/2016	5
8		Tests	5j?	29/02/2016	04/03/2016	7
9		Livrable 3 - Réordonnancement en-ligne	10j?	07/03/2016	18/03/2016	6
10		Méthode de résolution réordo en ligne	7j?	07/03/2016	15/03/2016	8
11		Tests	3j?	16/03/2016	18/03/2016	10
12		Rédaction rapport	55j?	04/01/2016	18/03/2016	
13		Préparation soutenance	5j?	21/03/2016	25/03/2016	

Figure 1 – *Tableau listant les tâches du projet*

La phase de développement est à terminer pour la dernière semaine du mois de mars. On va conserver la distinction entre les trois phases de l'ordonnancement, qui seront analogues à nos trois livrables au cours du semestre.

Ci-dessous sont listés les trois livrables, accompagnés des tâches à effectuer.

1 Livrable 1

Date de fin estimée : 12/02/2016 (6 semaines)

Le premier livrable sera le logiciel d'ordonnancement de phase 1. Le développement sera achevé lors de la première quinzaine de février. Cette période est plus grande que les deux suivantes, car il faut tenir compte du développement des interactions avec la base de données au sein du système d'information via les fichiers intermédiaires, leur parsing, et le développement du modèle.

1.1 Interaction avec la base de données

L'une des premières tâches générales est la conception d'un module du SI permettant la génération des données de la base, dans un fichier dont le format est décrit en annexe. Ce module sera écrit en PHP à

l'aide du framework CodeIgniter. Le déclenchement se fera à l'aide d'un bouton à rajouter dans l'interface du SI.

Il faudra également générer le fichier de sortie de l'ordonnancement.

1.2 Parseur

Lorsque l'interaction avec la base de données sera opérationnel, il faudra concevoir un parseur, en C++, qui prendra en entrée un fichier et qui initialisera les objets du logiciel nécessaires à l'ordonnancement.

1.3 Méthode de résolution

La partie la plus importante du livrable est l'élaboration, la conception et le développement d'une méthode de résolution. Celle-ci utilisera les objets créés par le parseur, et générera une solution réalisable d'ordonnancement. Si le temps le permet, une seconde méthode de résolution pourra être implémentée.

1.4 Tests - Benchmark

Lorsque les tâches précédentes auront été effectuées, des batteries de tests seront mis au point, permettant d'assurer le bon fonctionnement du livrable. Les tests du premier livrable sont de loin les plus importants, étant donné que les autres livrables se baseront sur les résultats de celui-ci.

Si une seconde méthode de résolution a été implémentée, un benchmark sera mis en place, permettant de comparer les deux méthodes, et de déterminer laquelle est la meilleure et dans quelles circonstances.

2 Livrable 2

Date de fin estimée : 04/03/2016 (3 semaines)

Le deuxième livrable sera une légère mue du premier livrable, dans lequel on pourra incorporer de nouveaux patients dans l'ordonnancement. Les outils de parsing et le modèle auront déjà été faits.

2.1 Méthode de résolution

La méthode de réordonnancement sera vraisemblablement du même accabit que celle(s) du premier livrable. A la différence que celle-ci prendra en compte un ordonnancement déjà validé, avec des activités médicales considérées comme fixes dans le temps. Cette méthode fournira en sortie un nouvel ordonnancement comportant les activités des nouveaux patients incorporés à la planification.

2.2 Tests

Ce livrable sera lui aussi soumis à des tests, permettant de garantir encore une fois son bon fonctionnement, mais aussi d'évaluer ses performances en termes de coût (mémoire, ressources CPU, etc.) et de célérité.

3 Livrable 3

Date de fin estimée : 18/03/2016 (2 semaines)

Le troisième et dernier livrable correspond à la phase de réordonnancement en temps-réel.

3.1 Méthode de résolution

La méthode de résolution sera différente de celles citées précédemment, puisque son comportement sera soumis aux différents scénarios qui peuvent survenir lors d'une journée dans un hôpital. Il devra être apte à réordonnancer les tâches suite à la suppression d'un patient du planning, et à son éventuel ajout si jamais celui-ci se présente avec beaucoup de retard. La méthode de résolution devra être très pragmatique.

3.2 Tests

Les tests en temps-réel seront d'une importance capitale, compte-tenu de la sensibilité de cette phase d'ordonnancement. Il faudra que la méthode de résolution soit prompte, apte à gérer tous les imprévus, et à répondre aux besoins de l'agent de coordination de l'établissement hospitalier.

Conclusion

Le projet d'envergure qu'est celui-ci pourrait faire sauter les verrous des systèmes d'ordonnancement de rendez-vous tels qu'on les connaît aujourd'hui. La mise en place de parcours médicaux pluridisciplinaires est un sujet maintes fois avancé mais sans réelle solution aboutie. Ce projet pourrait permettre, à l'échelle d'un établissement médical, le passage de patients dans plusieurs services différents et autonomes durant une seule et unique journée d'auscultation et de traitements.

Les problèmes d'ordonnancements multi-critères multi-ressources sont des problèmes difficiles à mettre en pratique. La modélisation que nous avons apporté, combinée avec des méthodes de résolutions heuristiques, semblent permettre de les résoudre dans un temps acceptable. C'est là un véritable challenge de recherche opérationnelle.

Ce projet est complet, puisqu'il touche à plusieurs domaines de l'informatique, à savoir le développement de logiciels en langage orienté objet, ainsi qu'à la mise en place de features additionnelles au sein d'un système d'informations conjointement développé. C'est un challenge technique.

Enfin, ce projet est transversal pédagogiquement parlant, et met en œuvre différents acteurs au sein de l'école (étudiants, professeurs) et à l'Assistance Publique - Hôpitaux de Paris, chose qui n'est pas aisée à concorder. C'est un véritable challenge humain.

Deuxième partie

Partie II - Développement

5 Méthodologie de suivi et de gestion de projet

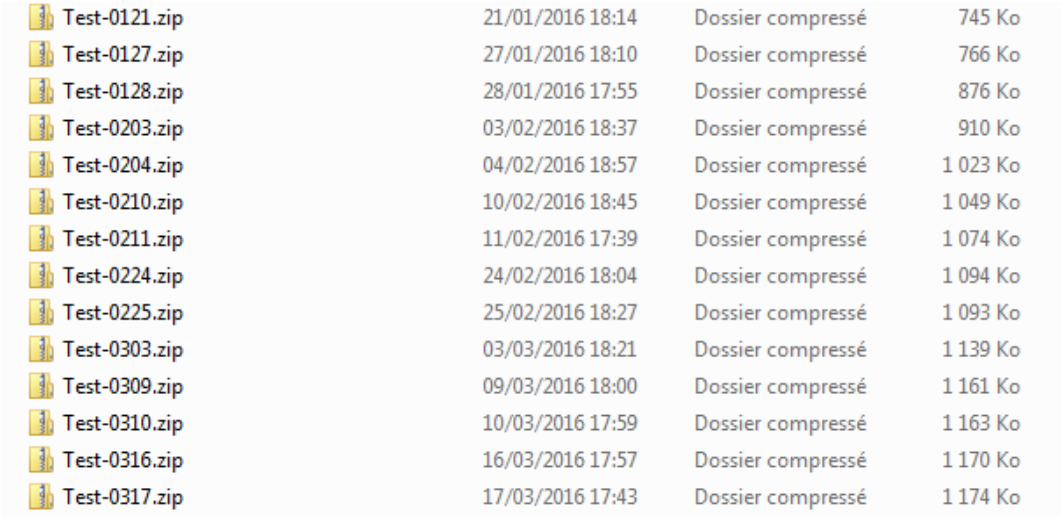
1 Suivi par l'encadrant

Concernant la méthodologie de suivi de projet, nous avons mis en place avec mon encadrant des rendez-vous assez fréquemment, afin de discuter de l'avancement du projet ainsi que des méthodes à utiliser tout au long de la mise en oeuvre.

Au cours du semestre, une mise au point a été effectuée concernant les méthodes d'évaluation de la solution, qui s'est traduite par un échange de mails avec à l'appui une documentation de la méthode.

2 Versionning et gestion de projet

Au cours de ce projet, aucun outil de versionning à proprement parler a été utilisé. Le projet s'est construit suivant une méthode agile, ce qui a donné lieu à une sauvegarde du projet et un archivage de celui-ci sous forme de zip datés à la fin de chaque journée de projet.



Test-0121.zip	21/01/2016 18:14	Dossier compressé	745 Ko
Test-0127.zip	27/01/2016 18:10	Dossier compressé	766 Ko
Test-0128.zip	28/01/2016 17:55	Dossier compressé	876 Ko
Test-0203.zip	03/02/2016 18:37	Dossier compressé	910 Ko
Test-0204.zip	04/02/2016 18:57	Dossier compressé	1 023 Ko
Test-0210.zip	10/02/2016 18:45	Dossier compressé	1 049 Ko
Test-0211.zip	11/02/2016 17:39	Dossier compressé	1 074 Ko
Test-0224.zip	24/02/2016 18:04	Dossier compressé	1 094 Ko
Test-0225.zip	25/02/2016 18:27	Dossier compressé	1 093 Ko
Test-0303.zip	03/03/2016 18:21	Dossier compressé	1 139 Ko
Test-0309.zip	09/03/2016 18:00	Dossier compressé	1 161 Ko
Test-0310.zip	10/03/2016 17:59	Dossier compressé	1 163 Ko
Test-0316.zip	16/03/2016 17:57	Dossier compressé	1 170 Ko
Test-0317.zip	17/03/2016 17:43	Dossier compressé	1 174 Ko

Figure 1 – *Liste des différentes sauvegardes du projet en archivage*

6 Modélisation des données

Dans la première partie de ce rapport, nous avons fixé une modélisation mathématique du problème. Au cours du développement, cette modélisation a été légèrement altérée afin de pouvoir s'adapter aux contraintes de développement.

1 Les objets

Tout au long du cycle de vie du programme, les données du problème vont être utilisées par les algorithmes afin d'en déduire un ordonnancement réalisable, respectant les contraintes strictes à la lettre et les contraintes assouplies au mieux.

1.1 Les patients

Le premier objet que l'on va manipuler va être un tableau permettant de recenser toutes les informations nécessaires au sujet des patients. On va tout d'abord définir une classe permettant de manipuler au mieux ces informations.

Cette classe possède quatre attributs :

- id, son identifiant unique
- idParcours, l'identifiant du parcours que le patient doit effectuer
- startDate, date de disponibilité au plus tôt du patient
- endDate, date de fin de disponibilité au plus tard du patient

1.2 Les ressources

Les ressources seront définies dans une classe également. Dans notre programme, nous allons créer un tableau de ressources permettant d'y accéder dans n'importe quelle fonction de celui-ci.

Cette classe possède cinq attributs :

- id, son identifiant unique
- idType, l'identifiant du type de la ressource
- startDate, date de disponibilité au plus tôt de la ressource
- endDate, date de fin de disponibilité au plus tard de la ressource
- indispo, tableau qui recense les indisponibilités de la ressource au cours de la journée.

1.3 Les activités

Pour définir les activités, nous n'avons besoin que de trois attributs :

- id, son identifiant unique
- duree, valeur entière correspondant à la durée estimée de l'activité en minutes
- besoins, tableau contenant les besoins en ressources pour effectuer l'activité. Ce tableau contient des paires (TypeRessource ; Quantité) qui nous seront utiles lors de l'affectation de l'activité aux ressources et aux patients

1.4 Les parcours

Le parcours est l'objet le plus complexe du programme. Il correspond à un ensemble d'activités, liées par des contraintes de précédences et de délais. Il contient quatre attributs définis comme tels :

- id, son identifiant unique
- duree, la somme des durées de toutes ses activités
- precedences, le tableau contenant les contraintes de précédences de ses activités

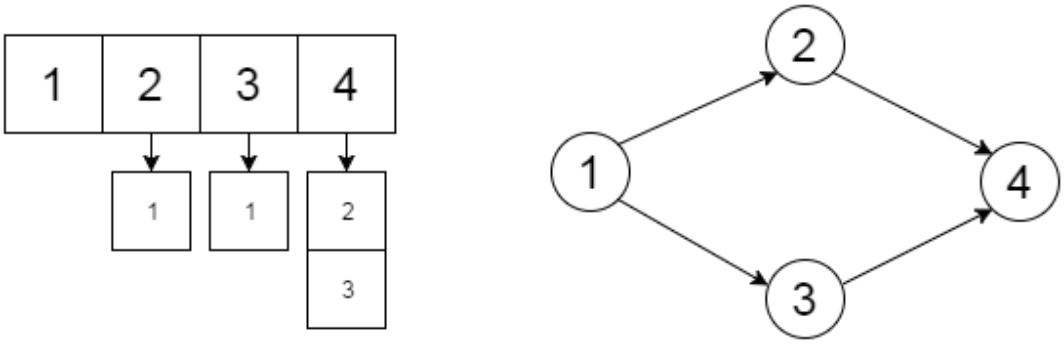


Figure 1 – *Tableau des contraintes de précédences du parcours joint*

- successeurs, le tableau contenant les contraintes de successions de ses activités

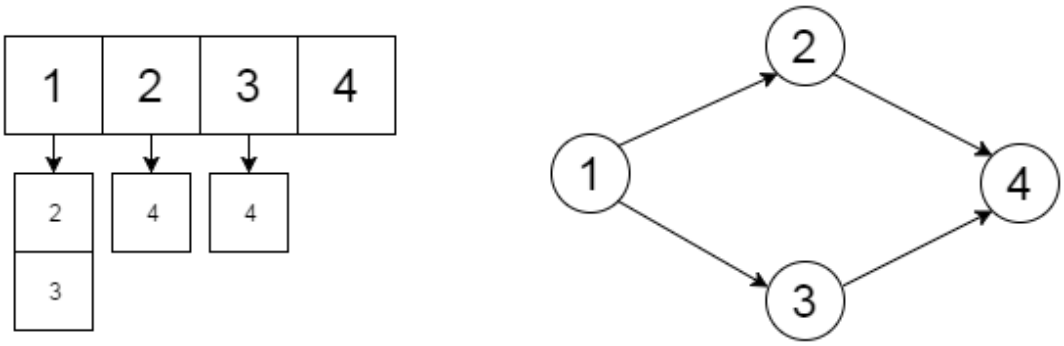


Figure 2 – *Tableau des contraintes de successions du parcours joint*

Les tableaux correspondent aux activités du parcours, modélisées par des noeuds dans les graphes joints. Chaque case du tableau aura une liste associée, contenant les identifiants des successeurs (ou prédécesseurs).

Il est important d'avoir les deux tableaux des contraintes, cela va permettre de simplifier les algorithmes pour le parcours de voisinage. Cela sera traité dans de plus amples détails dans la Section 4 (Chapitre 7).

2 Les paramètres d'entrée

Dans notre programme, trois fonctions nécessitent des entrées utilisateurs, permettant de paramétrer leurs travaux. Ils sont transmis sous forme de chaînes de caractères à la fonction main, fonction principale

et point d'entrée de notre programme. Ces paramètres, classés ici dans l'ordre dans lesquels ils doivent apparaître, sont décrits ci-dessous.

— Le fichier d'entrée, contenant les différentes informations permettant de construire les objets sus-cités. Sa structure est stricte, et elle est définie en annexe. Ce paramètre sera utilisé dans la fonction de parsage.
— ALPHA et BETA, les paramètres permettant de calculer la fonction objectif, en d'autres termes, d'évaluer le coût d'une solution. ALPHA permet d'ajuster la pondération de la somme des retards et du retard maximal relevé d'une solution. BETA quant à lui va être le coefficient qui permettra de discriminer les solutions dépassant les fenêtres de disponibilités.
— N, une valeur entière permettant la création de la solution initiale, correspondant à un contingent de patients. L'utilité de ce paramètre sera décrite plus en détail dans la Section 2 (Chapitre 7).

3 Le parsing

Le fichier d'entrée suit des contraintes fortes dans sa conception, il en va de même pour la fonction de parsage. On va traiter chaque bloc du fichier afin de l'insérer dans nos objets créés. L'ordre a son importance, il est nécessaire d'avoir les patients, les ressources et les activités en premier lieu, puis la composition des parcours, les besoins de chaque activité (dans le bloc Necessiter), et enfin les indisponibilités des ressources.

4 La solution

La modélisation de la solution va s'effectuer en deux objets distincts. Ces deux objets seront d'une part un tableau de blocs de solutions, d'autre part un tableau contenant des listes de blocs de solutions affectés à chacune des ressources.

4.1 Bloc de solution

Pour modéliser les deux objets de solution, il est nécessaire de définir des briques élémentaires de solutions, que l'on appellera dans la suite de ce rapport des blocs de solution, définies dans la classe BlocSolution. Celles-ci contiennent quatre attributs :

— idPatient, l'identifiant unique du patient
— idActivité, l'identifiant unique de l'activité à réaliser
— startDate, la date de début de l'activité planifiée
— endDate, la date de fin de l'activité planifiée

4.2 La solution en codage indirect

Pour pouvoir aisément appliquer nos algorithmes de voisinage, on va créer un objet sous la forme d'un tableau contenant des blocs de solution.

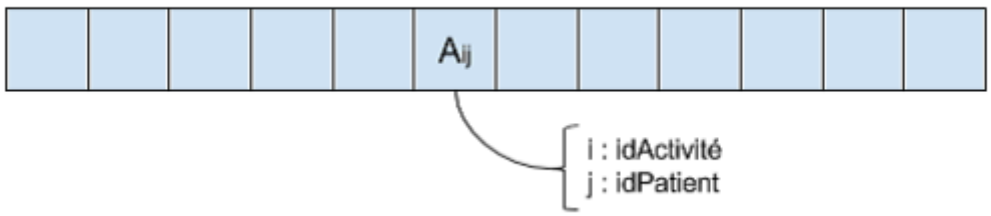


Figure 3 – *Schéma de solution en codage indirect*

L'ordre des blocs a une importance cruciale. Lors de l'évaluation de la solution en codage indirect, nous allons récupérer les blocs un par un en commençant par le début. Ceux-ci seront affectés suivant une heuristique décrite dans la Section 3 (Chapitre 7). Cela veut dire qu'une telle solution en codage indirect peut amener à des solutions différentes, en dépendance directe avec la fonction d'évaluation.

Lors de la construction de cette solution, seuls les identifiants du patient et de l'activité correspondants seront nécessaires. Les dates de début et de fin seront affectées uniquement lors de l'évaluation.

Il faut bien retenir que cette solution n'est pas utilisable en soi, elle permet de donner l'ordre de passage des activités pour leur affectation dans la solution définitive.

4.3 La solution affectée finale

L'objectif de ce programme est de fournir, pour chaque ressource médicale distincte, une liste de paires patient/activité, avec une date de début et une date de fin de l'activité à réaliser.

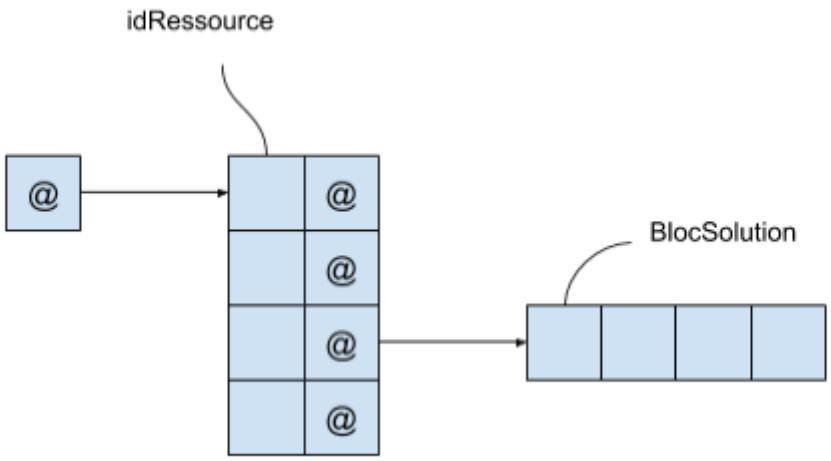


Figure 4 – *Schéma de la solution affectée finale*

Dans ce programme, on va créer un tableau de paires. Celles-ci contiennent comme premier membre l'identifiant d'une ressource, et en second membre la liste des blocs de solutions affectés à cette ressource. L'ordre de cette liste n'a aucune importance. Cette structure de données a le mérite de simplifier la gestion des blocs de solution lors de leur ajout durant l'évaluation, ainsi que lors de l'export de cette solution dans un fichier de sortie.

7 Mise en oeuvre

1 Fonctionnement général

Considérant un ensemble de patients connu, devant effectuer des parcours déterministes, le principe est de générer une solution d'affectation des patients et activités à un ensemble de ressources médicales typées. Cette solution, en codage indirect, sera évaluée à l'aide d'une heuristique de placement d'activités, qui permettra d'affecter à chaque paire patient/activité une date de début et une date de fin.

Cette solution sera par la suite améliorée (si possibilité d'amélioration il y a) à l'aide d'une recherche locale hybride, alliant permutations et insertions d'activités. Ces deux heuristiques utiliseront des opérateurs de voisinages définis, qui permettront de respecter les contraintes de précédences.

2 Conception de la solution initiale

La solution initiale se présente tout d'abord comme une solution en codage indirect. Pour chaque patient, on va donc ajouter chacune de ses activités, selon son parcours. Les activités sont ajoutées au fur et à mesure, en respectant les contraintes de précédences.

Etant donné que la construction de la solution définitive va s'effectuer dans l'ordre des blocs de solution de la solution en codage indirect, les premiers patients ajoutés à celle-ci vont donc être les premiers affectés à la solution finale. Il semble donc judicieux de trier les patients par ordre de disponibilité au plus tôt. Cela va permettre de maximiser les chances d'avoir un ordonnancement respectant les fenêtres de disponibilité dès le début de l'affectation.

Une autre astuce pour la solution initiale est d'affecter les patients par contingents. Compte tenu que ceux-ci ne feront pas nécessairement les mêmes parcours, ils n'utiliseront donc pas les mêmes ressources. On pourra par conséquent prendre les patients par groupes de N, pour densifier l'affectation aux différentes ressources, mais également pour permettre un mélange qui sera amélioré par la suite par l'heuristique hybride.

3 Focus : Evaluation d'une solution

La solution initiale en codage indirect est désormais créée, nous pouvons passer à l'évaluation de celle-ci. Nous allons utiliser une heuristique d'affectation des paires patient/activité aux différentes ressources disponibles de notre système.

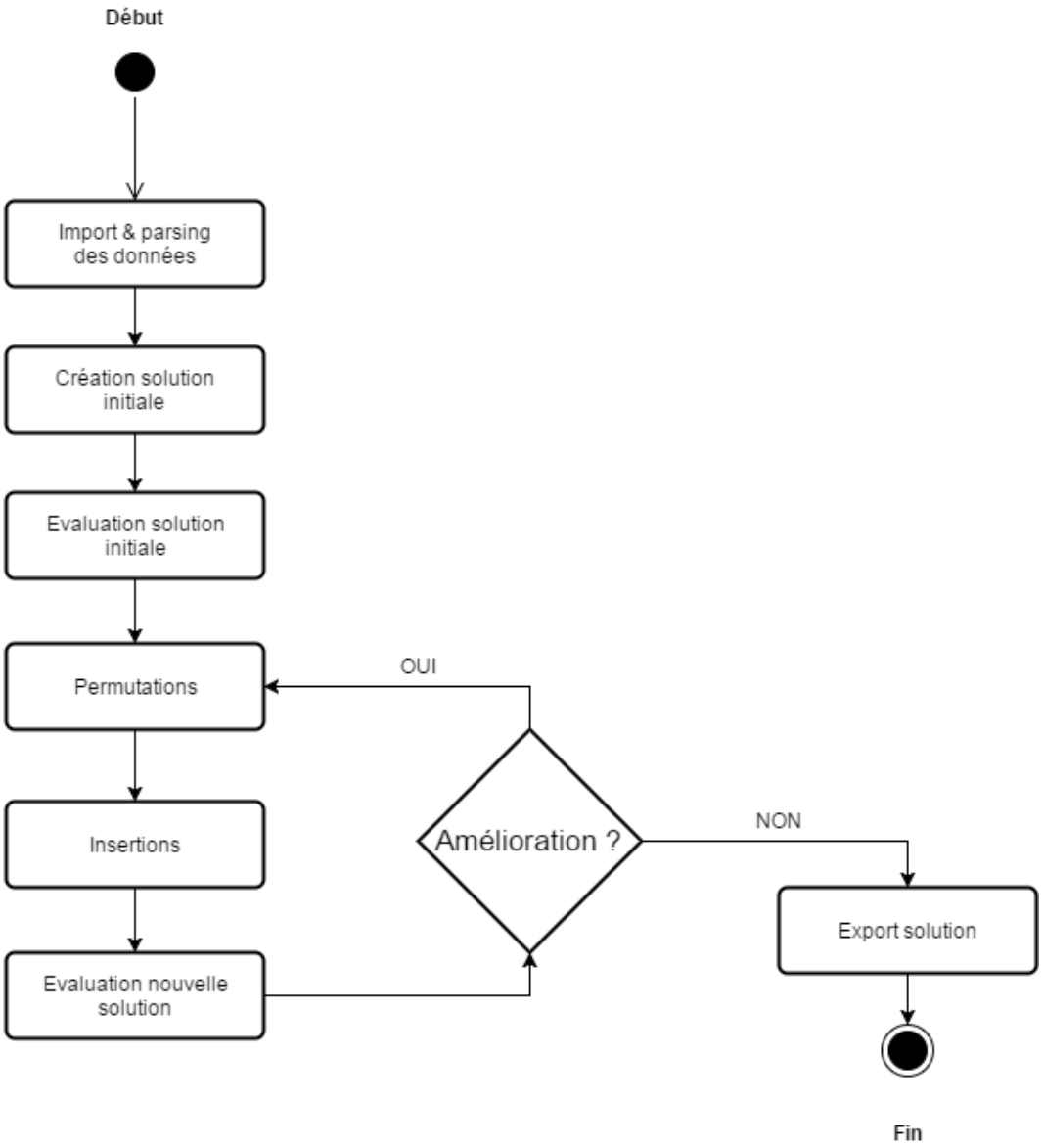


Figure 1 – *Diagramme d'activité du programme*

3.1 Première hypothèse d'évaluation

Dans un premier temps, l'évaluation s'effectuait de la façon suivante.

Pour chaque activité, on récupérait la date de première disponibilité du patient concerné, c'est à dire :

- sa date de disponibilité au plus tôt si ce patient n'a encore réalisa aucune activité
- sa date de fin de sa dernière activité planifiée sinon

et pour chaque type de ressources dans les besoins de l'activité, on récupérait la n-ième meilleure disponibilité de l'ensemble. Le maximum de toutes ces dates était considérée comme la date de démarrage au plus tôt de l'activité. Cela permettait de garantir d'une part que le patient est disponible, et d'autre

part que chaque ressource de chaque type était également disponible.

```
Entrées : Solution en codage indirect
Sorties : Coût de la solution
pour chaque bloc de la solution en codage indirect faire
    dispoP ← 0;
    dispoR ← 0;
    quantité ← nombre de ressources de type k nécessaires;
    pour chaque type k de ressource nécessaire faire
        dispo_k ← tableau des disponibilités des ressources de type k;
        trier dispo_k par date de disponibilité croissantes;
        dispoR ← max(dispoR ; dispo_k(quantité));
    fin
    dateDebut ← max(dispoP, dispoR);
    Ajouter à la solution;
fin
somme_attente ← 0;
attente_max ← 0;
depassement ← 0;
pour chaque patient planifié faire
    fin ← date de fin de la dernière activité;
    debut ← date de début de la première activité;
    attente ← fin - debut - duree_parcours;
    somme_attente ← somme_attente + attente;
    attente_max ← max(attente_max, attente);
    depassement ← depassement + max(fin - endDate, 0)
fin
retourner α × somme_attente + (1 − α) × attente_max + β × depassement
```

Algorithme 1 : Algorithme de la première version d'évaluation

L'inconvénient majeur de cette stratégie d'ordonnancement est qu'elle permettait bon nombre de trous dans le planning des patients, ce qui conduisait à des solution pouvant avoir des coûts de la fonction objectif relativement modérés. Il y a donc moyen de concevoir une heuristique proposant des solutions de meilleure qualité, c'est cette seconde heuristique qui a été choisie pour ce projet.

3.2 Seconde hypothèse d'évaluation

La seconde méthode d'évaluation est celle qui est utilisée au sein du programme livré. On considère un bloc de solution à ordonnancer. On sait que le patient concerné ne peut pas commencer son activité plus tôt que la fin de la dernière qu'il a effectué (ou à une date antérieure à sa fenêtre de disponibilité dans le cas de sa première activité). On va donc essayer de chercher un créneau disponible au sein de l'ordonnancement qui permettrait de réaliser cette activité, quand bien même certaines ressources aient déjà des activités planifiées ultérieurement. En d'autres termes, il s'agit là de meubler les trous de

l'ordonnancement dans la mesure du possible.

```
Entrées : Solution en codage indirect
Sorties : Coût de la solution
pour chaque bloc de la solution en codage indirect faire
    t ← date de dispo au plus tôt du patient;
    [A :];
    EnsembleRessourceNécessaire <- ensemble vide;
    pour tous les types k de ressources nécessaires faire
        x←0;
        y←0;
        h←infini;
        while (x < quantité de ressource nécessaire de type k) ET (y < nb ressources de type de k) do
            d ←CHERCHER( t , id_Ressource y , durée activité à planifier);
            si t=d alors
                //on peut placer l'activité du patient à cette date t sur la ressource y;
                x←x+1;
                Ajouter y à EnsembleRessourceNécessaire;
            fin
            h ← min (h ;d) //on stocke la plus petite date à décaler pour trouver une autre
              ressource;
            y←y+1;
        end
        si x !=quantité de ressource nécessaire de type k alors
            // aucune ressource dispo à t, il faut chercher à une date plus tard donc tout relancer
              t←h;
            Retour en [A];
        fin
    fin
    Fixer EnsembleRessourceNécessaire à l'activité du patient;
fin
somme_attente ← 0;
attente_max ← 0;
depassement ← 0;
pour chaque patient planifié faire
    fin ← date de fin de la dernière activité;
    debut ← date de début de la première activité;
    attente ← fin - debut - duree_parcours;
    somme_attente ← somme_attente + attente;
    attente_max ← max(attente_max, attente);
    depassement ← depassement + max(fin - endDate, 0)
fin
retourner α × somme_attente + (1 − α) × attente_max + β × depassement
```

Algorithme 2 : Algorithme de la seconde version d'évaluation

La méthode Chercher va permettre de combler les trous que créait la première méthode d'évaluation. Cette méthode prend en paramètres une date, un identifiant de ressource et la durée de l'activité à planifier. Elle va retourner la date au plus tôt à laquelle la ressource en paramètre sera disponible, à partir d'une certaine date. Lors de la première recherche de disponibilité, la date fournie est la date de disponibilité du patient dont l'activité est en cours d'ordonnancement.

Pour chaque bloc de la solution en codage indirect on va donc essayer de constituer un ensemble de ressources qui pourront être affectées dans ce créneau là. On va donc parcourir notre ensemble de ressources, et récupérer celles disponibles. Si jamais on ne parvient pas à constituer un contingent suffisant permettant d'effectuer l'activité, on repart à partir d'une nouvelle date de début. Cela signifiera par ailleurs que le patient aura un délai d'attente entre sa dernière activité et celle-ci.

Sur la Figure 2, quatre cas d'utilisation de la méthode Chercher ont été illustrés. Le début des lignes de temps est considéré comme t = 0. L'activité à planifier (en rouge), dure 25 minutes. On va donc appeler la fonction Chercher avec ces 2 paramètres.

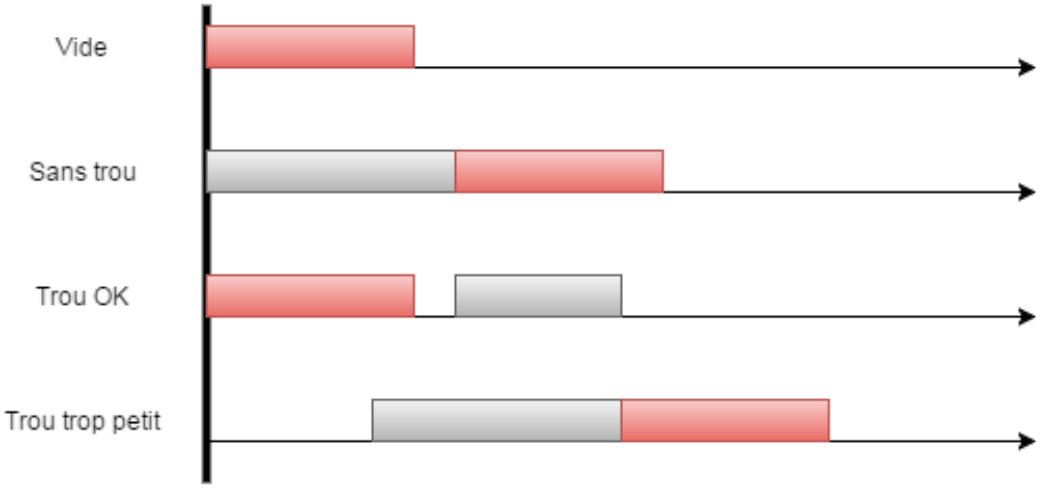


Figure 2 – *Illustration de la méthode Chercher*

— Dans le premier cas, la ressource n'a aucune activité de planifiée, on retourne t = 0.
— Dans le deuxième cas, il y a une activité de planifiée à t=0 et qui dure 30 minutes. L'activité rouge commencera à t = 0.
— Dans le troisième cas, une activité est planifiée à t = 30. On peut donc retourner la valeur 0 comme date de disponibilité, puisque notre activité peut s'effectuer sans empiéter sur la suivante.
— dans le quatrième cas, une activité est planifiée à t = 20. Notre activité rouge sera donc planifiée à la fin de l'exécution de celle-ci.

3.3 Calcul du coût de la solution

Lorsque tous les bolcs de solutions de la solution en codage indirect ont été ordonnancés, on va pouvoir calculer les temps d'attente de chacun des patients, en déduire le temps maximal relevé pour un patient dans l'ensemble de ceux-ci, et calculer les minutes de dépassement des fenêtres de disponibilité.

Ce coût est évalué à l'aide de la fonction objectif, définie comme telle :

$$cout(solution) = \alpha \times \sum_{j=1}^{n} w_j + (1-\alpha) \times w_j^{max} + \beta \times \sum_{j=1}^{n} dep_j \quad (1)$$

avec :
— $j \in [1, n]$ l'identifiant du patient j
— w_j le temps d'attente total du patient j
— w_j^{max} le temps d'attente maximal de l'ensemble des patients
— dep_j le dépassement de fenêtre de disponibilité du patient j

4 Parcours de voisinage

Pour trouver une solution ayant un coût plus faible, on applique une recherche locale permettant de trouver des solutions accessibles. Cela va permettre de trouver en un temps assez bref une solution non-optimale d'ordonnancement, mais qui améliorera celle trouvée précédemment. Ces heuristiques sont appliquées sur la solution en codage indirect, et veilleront à respecter les contraintes de précédences. Pour se faire, on va utiliser une recherche locale hybride, utilisant successivement deux opérateurs de voisinage différents.

Pour la suite de cette section, on va se baser sur un exemple de solution en codage indirect. Celle-ci comporte trois patients de couleurs différentes. Elle met en jeu huis blocs de solutions, dont quatre pour le blanc, deux pour le rouge et deux pour le vert. Les activités du patient blanc sont numérotées, et son parcours correspond à celui défini dans la Figure 1 (Chapitre 6).

Figure 3 – *Exemple de solution en codage indirect avec trois patients (Rouge, Vert et Blanc)*

4.1 Insertion

L'opérateur d'insertion permet d'atteindre des solutions voisines en extrayant un bloc de solution puis en l'insérant plus loin.

Pour chaque bloc de solution de la solution en codage indirect, on essaye de le décaler sur la droite. Si l'activité derrière laquelle il est inséré est effectuée par un autre patient que celui dont l'activité est en cours d'insertion, cela ne devrait pas poser de problème au sujet des contraintes de précédences.

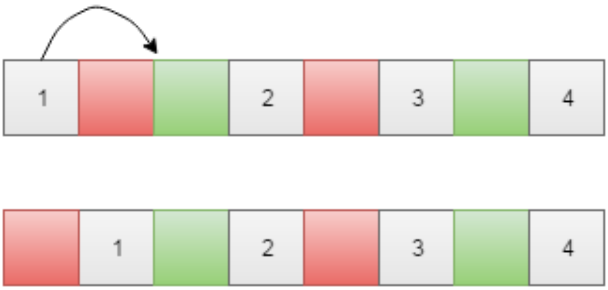


Figure 4 – *Illustration d'une insertion valable*

En revanche, si le bloc est inséré après un autre bloc de ce même patient, il faut veiller à ce qu'il n'y ait pas de viol de contraintes de précédences, auquel cas la phase d'insertion s'arrête, puisqu'une insertion encore plus à droite perpétuerait cette anomalie détectée.

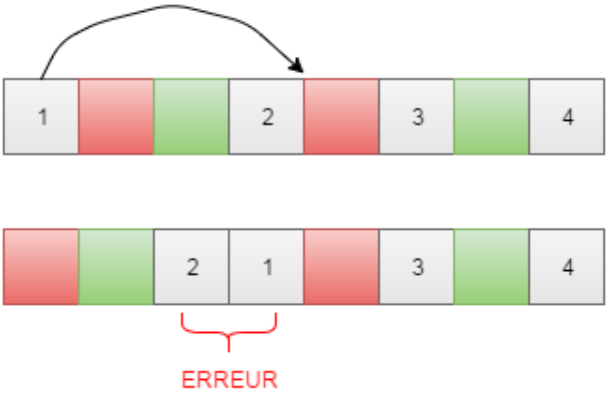


Figure 5 – *Illustration d'une insertion non valable : 1 soit s'effectuer avant 2*

Pour veiller au bon respect des contraintes de précédence, on va utiliser le tableau des successions défini dans la Figure 1 (Chapitre 6). On part du principe que la solution initiale a respecté les contraintes. On va vérifier que l'activité après laquelle on va effectuer l'insertion ne soit pas un successeur direct de celle qui va être insérée. Pour cela, on regarde la liste associée à l'activité dans le parcours. Si on reprend l'exemple de parcours, on remarque qu'effectivement, l'activité 2 est un successeur de l'activité 1. L'insertion est donc non conforme.

On effectue les insertions jusqu'à ce qu'on ne puisse pas trouver de meilleure solution via cet opérateur.

4.2 Permutation

La permutation est un second opérateur de voisinage que l'on a mis en pratique dans ce projet. Comme son nom l'indique, il consiste à permuter deux blocs de solution au sein de la solution en codage indirect.

Cela rajoute un niveau de complexité, puis qu'il s'agit là de veiller à ce que l'activité décalée sur la gauche respecte les contraintes de précédences aussi bien que celle décalée à gauche.

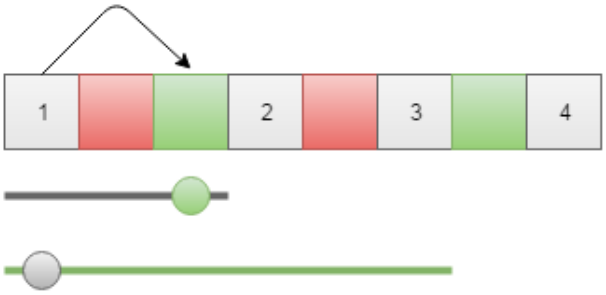


Figure 6 – *Illustration d'une permutation valable*

On va nommer l'activité blanche 1 comme étant l'activité de gauche, et la première activité verte comme l'activité de droite. On sait d'ors et déjà que l'activité de gauche est permutable, puisque les deux activités ne portent pas sur le même patient. Il faut maintenant veiller à ce que le décalage à gauche de l'activité de droite n'enfreigne pas les contraintes du patient vert. On parcourt donc la solution en codage indirect de droite à gauche, en partant de l'activité de droite, afin de vérifier que le décalage à gauche soit valable. Si lors de ce parcours on atteint notre activité de gauche, la permutation est viable. Pour cela, on va utiliser le tableau des précédences défini dans la Figure 1 (Chapitre 6).

Schématiquement, on va définir des fenêtres de permutations pour chaque bloc de la solution. Ces fenêtres correspondent aux blocs permutables pour chacun des blocs considérés. On considère que la permutation entre deux blocs peut s'effectuer si ceux-ci sont inclus dans leurs fenêtres respectives.

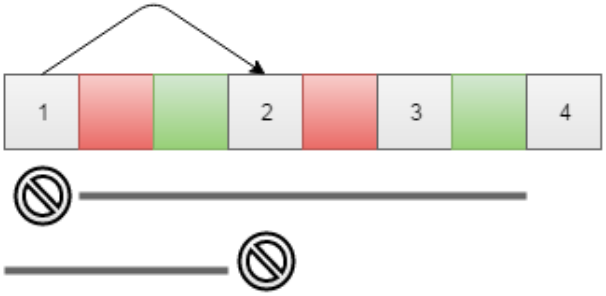


Figure 7 – *Illustration d'une permutation non valable : 1 doit s'effectuer avant 2*

L'avantage de cet opérateur est qu'il va permettre de ramener vers la gauche des blocs situés à la fin de la solution en codage indirect, chose que l'opérateur d'insertion ne peut pas faire.

8 Liaison avec le système d'information

Le programme est désormais fonctionnel, et peut proposer une solution d'ordonnancement réalisable, respectant au mieux les contraintes définies.

1 Export de la solution

Le programme va se charger de l'export de la solution dans un fichier texte, donc la structure est définie en annexe. Ce fichier sera ensuite parsé côté SI, permettant son important dans la base de données.

2 Déclenchement du programme avec le système d'information

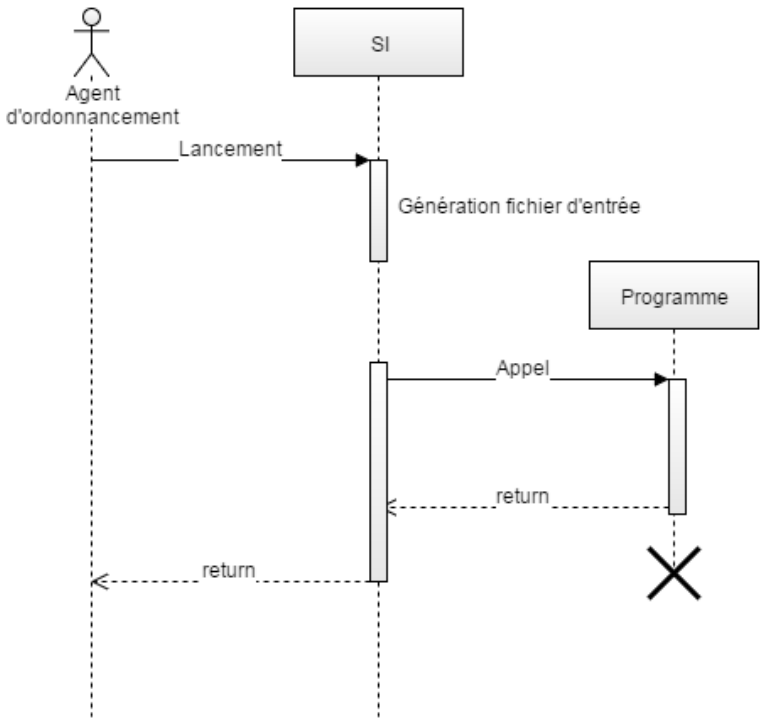


Figure 1 – *Diagramme d'interaction du système*

En amont du programme, il faut que le système d'information soit capable de générer le fichier d'entrée. Pour cela, on rajoute un bouton accessible uniquement pour les utilisateurs authentifiés et qui ont les droits d'accès sur l'ordonnancement.

Au déclenchement de ce bouton, le système d'information va récupérer toutes les données nécessaires, à savoir l'ensemble des patients prévus pour le jour donné, ainsi que l'ensemble des ressources disponibles ce jour là et les informations sur les parcours et activités. Le programme va s'exécuter, puis produire un fichier de solution qui sera par la suite parsé. Les activités planifiées seront insérées dans la base de données.

9 Validation et tests

1 Tests unitaires

Tout au long du développement, des tests unitaires ont été effectués, sans pour autant les avoir automatisés.

2 Tests fonctionnels

Pour tester les fonctionnalités, chaque fonction est passée au crible avec des petits jeux de données. La comparaison a été effectuée avec des solutions réalisées à la main, permettant de veiller à la cohérence des résultats.

Pour la solution fournie à la fin du programme, une fonction de validation permet de vérifier que les contraintes strictes sont respectées.

3 Tests de performances de la solution

Afin de tester l'efficacité de la méthode, il aurait fallu concevoir un modèle mathématique permettant de comparer les résultats avec une solution optimale pour chaque instance. Une alternative serait de comparer les résultats obtenus pour chaque instance avec le résultat obtenu si nous avions opté pour une solution acceptable générée aléatoirement. Dans le principe, il faudrait que notre solution générée par le programme soit bien meilleure que celle générée aléatoirement, pour considérer que notre méthode est efficace.

La génération va se faire à l'aide d'une heuristique gloutonne multistart. Le fonctionnement de cette heuristique est simple. On va tout d'abord créer une solution en codage indirect. Celle-ci va se voir attribuer aléatoirement les activités d'un ensemble de patient, tout en respectant les contraintes de précédences. Cette solution sera ensuite décodée lors de l'évaluation, et on en tirera un coût de la fonction objectif. Le décodage sera identique à celui utilisé dans notre méthode.

3.1 Plan de tests

On va concevoir 3 familles d'instances homogènes, contenant chacune 10 instances. On va calculer pour chacune des instances une solution à l'aide de notre méthode. Ces familles instances sont de tailles différentes, on va trouver :

— les petites instances : 10 patients et 15 ressources
— les moyennes instances : 20 patients et 30 ressources
— les grandes instances : 50 patients et 30 ressources

Avec les coûts de notre méthode et de l'heuristique gloutonne, on va en déduite un taux, qui permettra d'évaluer l'amélioration qu'apporte notre méthode. Ce taux est défini par :

$$\tau = \frac{cout_{glouton} - cout_{hybride}}{max(cout_{glouton}, cout_{hybride})} \quad (1)$$

Paramètres :

— $\alpha = 0.5$
— $\beta = 1000$
— $N = 4$

Les tests ont été effectués sur une machine de l'école, la configuration est la suivante :

— Système d'exploitation : Machine virtuelle sous Windows 7
— Mémoire vive (RAM) : 4Go
— Processeur Intel(R) Xeon(R) 3.07GHz (4 Cores)

3.2 Petites instances

Table 1 – *Résultats des tests sur 10 petites instances*

Instance	Coût algo	Coût random	Taux d'amélioration
Input_10_15_4992	20	260	92.30
Input_10_15_5155	20	193	89.63
Input_10_15_5161	41	345	88.11
Input_10_15_5165	25	264	90.53
Input_10_15_5168	5	290	98.11
Input_10_15_5171	19	390	95.12
Input_10_15_5175	31	158	80.37
Input_10_15_5178	51	281	81.85
Input_10_15_5181	22	176	87.50
Input_10_15_5328	28	210	86.66
Moyenne	26.20	254.90	89.01

Le temps d'exécution est en moyenne de 0.378 secondes sur ces petites instances.

3.3 Moyennes instances

Le temps d'exécution est en moyenne de 4.491 secondes sur ces moyennes instances.

Table 2 – *Résultats des tests sur 10 moyennes instances*

Instance	Coût algo	Coût random	Taux d'amélioration
Input_20_30_1	88	1926	95.43
Input_20_30_2	102	1763	94.21
Input_20_30_3	133	2396	94.44
Input_20_30_4	120	1687	92.88
Input_20_30_5	103	1830	94.37
Input_20_30_6	145	1779	91.84
Input_20_30_7	135	2226	93.93
Input_20_30_8	103	2325	94.74
Input_20_30_9	108	1950	94.46
Input_20_30_10	107	1667	93.58
Moyenne	114.4	1954.90	93.99

3.4 Grandes instances

Le temps d'exécution est en moyenne de 283.309 secondes (4 minutes et 43 secondes) sur ces grandes instances.

On note une disparité dans les résultats trouvés à l'aide de notre méthode, car le nombre de ressources est parfois insuffisant compte tenu des fenêtres de disponibilités des patients. On se retrouve donc avec des coûts dépassant les milliers, faute d'avoir pu trouver des ordonnancements respectant les fenêtres de disponibilité des patients.

3.5 Synthèse des tests

Lors des trois phases de tests, on peut s'apercevoir que notre méthode fournit des solutions qui sont bien meilleures que des solutions trouvées de façon aléatoire dans un laps de temps identique. En plus de donner des solutions réalisables, notre méthode va surtout permettre de discriminer celles qui enfreignent les contraintes qui peuvent être assouplies comme la date de sortie du patient du système. On obtient donc des résultats qui pourront être utilisés afin de satisfaire au mieux l'ensemble des patients qui seront présents dans notre système.

Table 3 – *Résultats des tests sur 10 grandes instances*

Instance	Coût algo	Coût random	Taux d'amélioration
Input_50_30_1	401	634658	99.93
Input_50_30_2	123340	1411960	91.26
Input_50_30_3	262776	1440619	81.75
Input_50_30_4	97301	743379	94.26
Input_50_30_5	163	1576960	99.97
Input_50_30_6	179778	1555565	88.59
Input_50_30_7	75100	1580941	95.17
Input_50_30_8	134929	1624351	91.46
Input_50_30_9	195595	1358773	85.60
Input_50_30_10	50291	1355552	96.29
Moyenne	111967	1362337	92.66

10 Reproductivité

1 Pré-requis & Installation

Le projet a été développé à l'aide de l'IDE Code :Blocks, compatible avec Windows, Mac OS et Linux. Dans le cadre de ce projet, j'ai utilisé le système d'exploitation Windows 7 sur machine virtuelle VM Ware. Le compilateur utilisé est MinGW, qui est la version Windows de GCC.

L'IDE est disponible en téléchargement sur le site officiel de Code :Blocks. (`http://www.codeblocks.org/`)

Le système d'information utilise CodeIgniter, un framework libre utilisant PHP 5. Pour l'utiliser en local, j'ai utilisé WAMP. (`http://www.wampserver.com/`)

2 Configuration

Le projet est prêt à l'emploi, il suffit de le compiler une fois pour générer les exécutables. Pour passer au programme les paramètres d'exécution, il faut aller dans le menu de Code :Blocks, dans l'onglet Projects puis Set programs' arguments. Il faut mettre dans l'ordre le chemin relatif vers le fichier d'entrée, la valeur de alpha, la valeur de beta puis la valeur de N.

3 Mode d'emploi

Le programme est composé d'une classe principale comportant les fonctions, ainsi que les différentes classes décrites dans ce rapport.

Les fonctions du fichier main sont les suivantes :

- parseInput : permet de lire le fichier d'entrée et de récupérer et d'initialiser toutes les données.
- creationSemiSolution : permet de créer la solution initiale
- chercher : fonction qui permet de retourner une date au plus tôt pour une ressource donnée dans l'évaluation de la solution
- evaluationSemiSolution : Permet d'évaluer une solution en codage indirect, et également de la stocker dans la variable solution.
- swapSemiSolution : fonction qui prend en entrée une solution en codage indirect, et qui retourne la meilleure solution trouvée une fois toutes les permutations effectuées. Elle s'arrête dès lors qu'aucune permutation ne puisse améliorer la solution.
- insertSemiSolution : idem que swapSemiSolution, mais en utilisant l'insertion.

— exportSolution : fonction qui va exporter dans un fichier la solution trouvée à l'aide de notre méthode.
— createrandomSemiSolution : fonction servant aux tests eprmettant de créer une solution respectant les contraintes de façon aléatoire.
— displayX : permet d'afficher en console la solution ou la solution en codage indirect selon la fonction appelée.
— validateurSolution : permet de vérifier qu'une solution est réalisable et respecte les contraintes

11 Conclusion

La phase 1 du projet est aboutie, le programme permet de fournir une solution réalisable, respectant les contraintes, pour une journée donnée. En revanche, tout reste à faire pour les réordonnancements en-ligne et hors-ligne.

Le livrable fourni peut être évidemment optimisé, on pourrait imaginer une parallélisation des traitements pour les recherche de voisinage, ainsi qu'une recherche de voisinage multi-start lorsque la méthode est bloquée dans un optimum local.

A l'avenir, il serait bénéfique d'utiliser ce programme avec des données du réel, permettant de tester à la fois la robustesse de l'algorithme, ainsi que sa validité dans la pratique.

Annexes

A Fichiers intermédiaires d'échanges

Code source A.1 – *Fichier d'entrée de l'ordonnancement*

```
[Patients]
idPatient, idParcours, heureDebut, heureFin
...
[Composer]
idParcours, idActivité, Pred1;Pred2;.., Succ1;Succ2;.., delaiMin, delaiMax
...
[Activités]
idActivité, durée
...
[Nécessiter]
idActivité, idTypeRessource, quantité
...
[Ressources]
idRessource, idTypeRessource
...
[Indisponibilités]
idRessource, heureDebut, heureFin
...
```

La base de données du système d'information a été conçue de telle sorte à ce qu'il n'y ait pas de difficulté à concevoir ce fichier.

Code source A.2 – *Fichier contenant l'ordonnancement*

```
[idRessource]
idPatient, idActivité, heureDebut, heureFin, deplacable
...
```

On note la présence d'un booléen **deplacable** qui indiquera si l'activité planifiée peut être déplacée dans le temps ou non.

Code source A.3 – *Fichier d'ajout d'un patient*

```
[Patients]
idPatient, idParcours, heureDebut, heureFin
```

B Diagramme de Gantt prévisionnel

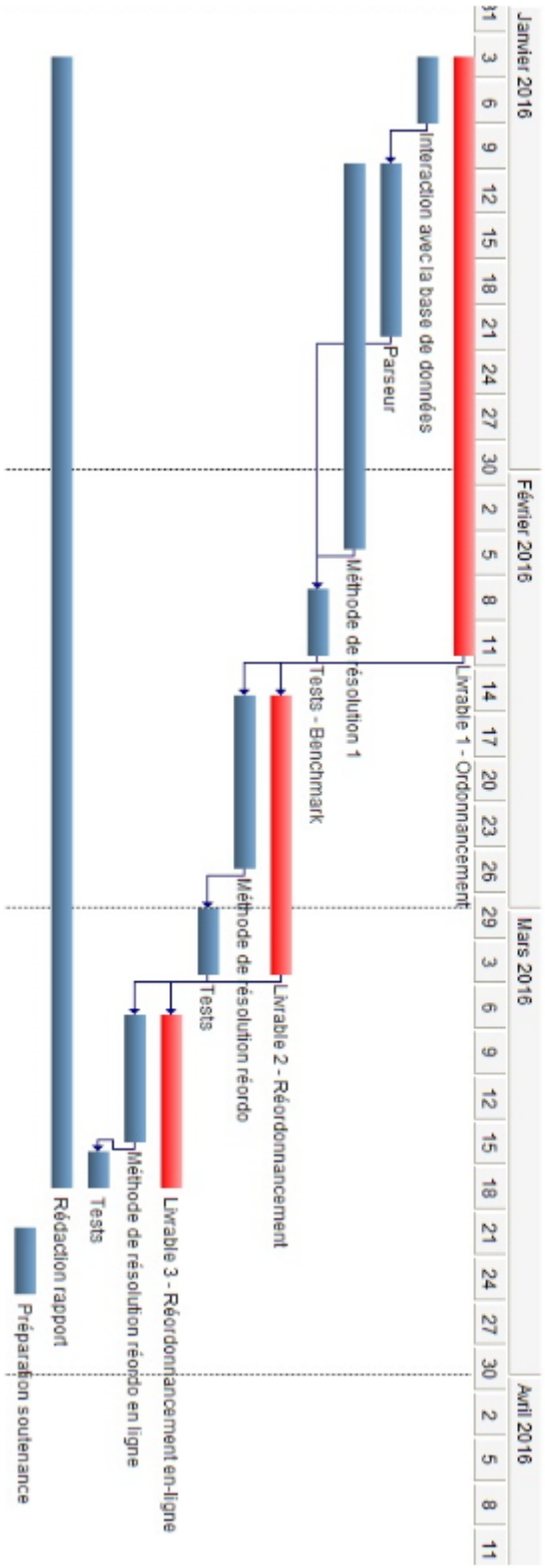


Figure 1 – *Diagramme de Gantt prévisionnel de développement*

Comptes rendus hebdomadaires

Compte rendu n°1 du 17/09/2015

Rencontre avec l'encadrant pour parler du sujet plus en détail.
Récupération des premiers articles de recherche. Lecture des articles, des revues de littérature, dans l'objectif de mieux cerner le sujet.

Compte rendu n°2 du 24/09/2015

Lecture des articles. Recherche d'articles basé sur les articles précédemment étudiés.

Compte rendu n°3 du 01/10/2015

Recherche d'articles basé sur les articles précédemment étudiés. Première rencontre avec les deux étudiants de DI4.

Compte rendu n°4 du 08/10/2015

Recherche d'articles basé sur les articles précédemment étudiés.

Compte rendu n°5 du 15/10/2015

Récupération de nouveaux articles de la part de monsieur Kergosien. Début du tri des articles en fonction de leur pertinence par rapport au problème. Début de la prise de notes concernant les articles à pertinence faible ou modérée. Recoupement et recherche de points communs entre ces articles et le problème.

Compte rendu n°6 du 22/10/2015

Début de la rédaction de l'état de l'art. Conception du plan de rédaction de l'état de l'art.
Recherche d'articles ayant un lien étroit avec notre sujet. Prise de notes concernant les articles à pertinence moins importante. Lecture d'articles.

Compte rendu n°7 du 05/11/2015

Première version de l'état de l'art mise au point.

Compte rendu n°8 du 12/11/2015
Premiers correctifs du plan de l'état de l'art.

Compte rendu n°9 du 19/11/2015
Rédaction de l'état de l'art. Début de la modélisation.

Compte rendu n°10 du 26/11/2015
Seconde version de l'état de l'art avec les correctifs. Veille technologique des logiciels du comemrce existants.

Compte rendu n°11 du 03/12/2015
Début de la rédaction du cahier des charges.

Compte rendu n°12 du 10/12/2015
Rédaction du rapport & du cahier des charges.

Compte rendu n°13 du 17/12/2015
Fin de la rédaction du cahier des charges. Rédaction du rapport. Estimation des durées des tâches et conception du planning prévisionnel de développement.

Compte rendu n°14 du 28/01/2016
Conception du modèle. Création du parseur du ficheir d'entrée.

Compte rendu n°15 du 04/02/2015
Conception et création de la méthode de création de la solution initiale. Première ébauche de la méthode d'évaluation.

Compte rendu n°16 du 11/02/2015
Tests et amélioration de la création de la solution initiale. Développement de la méthode d'évaluation de la solution.

Compte rendu n°17 du 25/02/2015
Fin du développement de la première méthode d'évaluation. Conception et développement de la méthode d'insertion.

Compte rendu n°18 du 03/03/2015
Conception et développement de la méthode de permutation. Conception et développement d'une seconde méthode d'évaluation.

Compte rendu n°19 du 10/03/2015
Développement de la seconde méthode d'évaluation. Développement d'une fonction de validation de solution. Début des tests de performance.

Compte rendu n°20 du 17/03/2015
Rédaction du rapport, développement de la jointure avec le SI.

Bibliographie

[1] Norman TJ Bailey. « A study of queues and appointment systems in hospital out-patient departments, with special reference to waiting-times ». In : *Journal of the Royal Statistical Society. Series B (Methodological)* (1952), p. 185–199.

[2] Aleida Braaksma, Nikky Kortbeek, GF Post et Frans Nollet. « Integral multidisciplinary rehabilitation treatment planning ». In : *Operations Research for Health Care* 3.3 (2014), p. 145–159.

[3] Brecht Cardoen, Erik Demeulemeester et Jeroen Beliën. « Operating room planning and scheduling : A literature review ». In : *European Journal of Operational Research* 201.3 (2010), p. 921–932.

[4] Elkin Castro et Sanja Petrovic. « Combined mathematical programming and heuristics for a radiotherapy pre-treatment scheduling problem ». In : *Journal of Scheduling* 15.3 (2012), p. 333–346.

[5] Tugba Cayirli et Emre Veral. « OUTPATIENT SCHEDULING IN HEALTH CARE : A REVIEW OF LITERATURE* ». In : *Production and Operations Management* 12.4 (2003), p. 519.

[6] Gang Du, Zhibin Jiang, Yang Yao et Xiaodi Diao. « Clinical pathways scheduling using hybrid genetic algorithm ». In : *Journal of medical systems* 37.3 (2013), p. 1–17.

[7] Jacob Feldman, Nan Liu, Huseyin Topaloglu et Serhan Ziya. « Appointment scheduling under patient preference and no-show behavior ». In : *Operations Research* 62.4 (2014), p. 794–811.

[8] Daniel Gartner et Rainer Kolisch. « Scheduling the hospital-wide flow of elective patients ». In : *European Journal of Operational Research* 233.3 (2014), p. 689–699.

[9] José Fernando Gonçalves, JJ de M Mendes et Mauricio GC Resende. « A genetic algorithm for the resource constrained multi-project scheduling problem ». In : *European Journal of Operational Research* 189.3 (2008), p. 1171–1190.

[10] Vernon Ning Hsu, Renato de Matta et Chung-Yee Lee. « Scheduling patients in an ambulatory surgical center ». In : *Naval Research Logistics (NRL)* 50.3 (2003), p. 218–238.

[11] Guido C Kaandorp et Ger Koole. « Optimal outpatient appointment scheduling ». In : *Health Care Management Science* 10.3 (2007), p. 217–229.

[12] David V Lindley. « The theory of queues with a single server ». In : *Mathematical Proceedings of the Cambridge Philosophical Society*. T. 48. 02. Cambridge Univ Press. 1952, p. 277–289.

[13] Nan Liu, Serhan Ziya et Vidyadhar G Kulkarni. « Dynamic scheduling of outpatient appointments under patient no-shows and cancellations ». In : *Manufacturing & Service Operations Management* 12.2 (2010), p. 347–364.

[14] Marie E Matta et Sarah Stock Patterson. « Evaluating multiple performance measures across several dimensions at a multi-facility outpatient center ». In : *Health care management science* 10.2 (2007), p. 173–194.

[15] Mahshid Salemi Parizi et Archis Ghate. « Multi-Class, Multi-Resource Advance Scheduling with No-Shows, Cancellations and Overbooking ». In : *Multi-Resource Advance Scheduling with No-Shows, Cancellations and Overbooking (January 15, 2015)* (2015).

[16] Katja Schimmelpfeng, Stefan Helber et Steffen Kasper. « Decision support for rehabilitation hospital scheduling ». In : *OR spectrum* 34.2 (2012), p. 461–489.

[17] Ayten Turkcan, Bo Zeng et Mark Lawley. « Chemotherapy operations planning and scheduling ». In : *IIE Transactions on Healthcare Systems Engineering* 2.1 (2012), p. 31–49.

[18] MF Van der Velde, N Kortbeek et N Litvak. « Organizing Multidisciplinary Care for Children with Neuromuscular Diseases ». In : (2012).

[19] Ivan B Vermeulen, Sander M Bohte, Sylvia G Elkhuizen, Piet JM Bakker et Han La Poutré. « Decentralized Online Scheduling of Combination-Appointments in Hospitals. » In : *ICAPS*. 2008, p. 372–379.

[20] Armin Wolf. « Constraint-based modeling and scheduling of clinical pathways ». In : *Recent Advances in Constraints*. Springer, 2011, p. 122–138.

[21] Christos Zacharias et Michael Pinedo. « Appointment Scheduling with No-Shows and Overbooking ». In : *Production and Operations Management* 23.5 (2014), p. 788–801.

[22] Bo Zeng, Ayten Turkcan, Ji Lin et Mark Lawley. « Clinic scheduling models with overbooking for patients with heterogeneous no-show probabilities ». In : *Annals of Operations Research* 178.1 (2010), p. 121–144.

Gestion et Optimisation des Parcours Patients

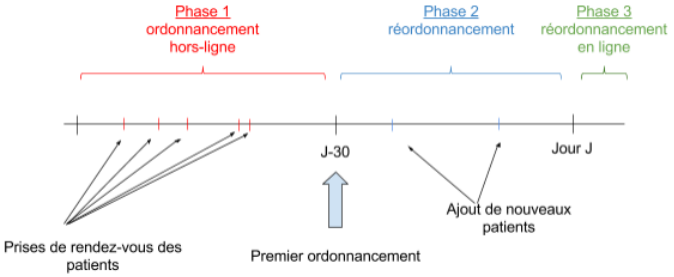
Jean Coquelet

Encadrement : Yannick Kergosien

Objectifs

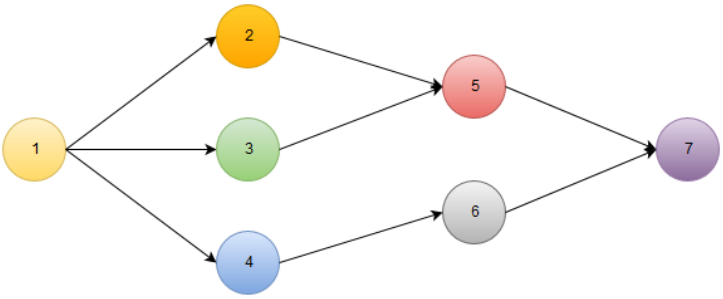
Pour un ensemble de couples patient-ressource :

- créer un 1er planning de rendez-vous
- pouvoir réordonnancer d'autres couples
- gérer en temps-réel les imprévus le jour J
- minimiser les temps d'attente des patients.



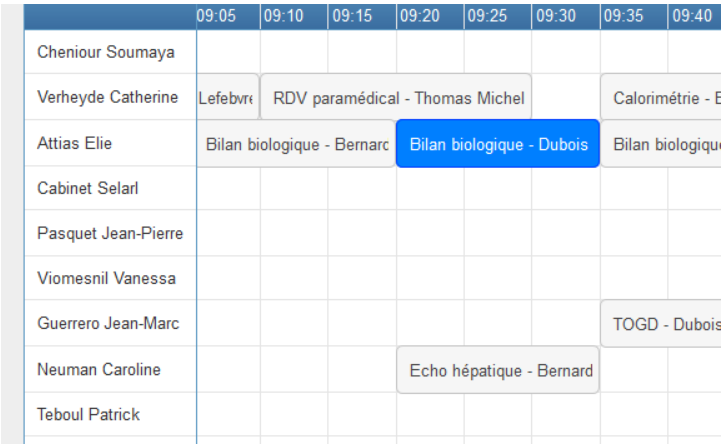
Mise en œuvre

- Développement de logiciels d'ordonnancement.
- Conception de méthodes de résolution heuristiques.
- Utilisation à l'aide d'un SI développé conjointement.



Résultats attendus

- Ordonnancement **réalisable** de rendez-vous
- **Flexibilité** des activités médicales
- Utilisation **accrue** des ressources
- Concentrer en **une journée** les séjours des patients
- Maximiser la **satisfaction** des patients et du personnel médical !



Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
www.polytech.univ-tours.fr

POLYTECH TOURS
Informatique

Gestion et Optimisation des Parcours Patients

Résumé

Pour désengorger les services hospitaliers écrasés sous la charge des patients venant pour des besoins différents avec des priorités différentes, les centres de soins mettent en place de plus en plus des services dédiés à l'accueil de patients pour une matinée, un après-midi ou même durant une journée complète. Ces établissements sont appelés hôpitaux de jour. C'est le cas de l'hôpital Antoine-Béclère, en région parisienne. A l'heure actuelle, chaque service dispose de son propre système de gestion des rendez-vous des patients. Cette décentralisation est à proscrire lors d'une mutualisation des services.

La mise en place de parcours cliniques permettrait d'optimiser l'utilisation des ressources médicales, tout en réduisant les temps d'attente des patients intégrés dans ces services.

Le but de ce projet est de concevoir un ensemble de programmes permettant l'ordonnancement et la planification de rendez-vous. Ces programmes devront être capable de fournir une solution acceptable d'ordonnancement, respectant les contraintes du problème d'ordonnancement général.

Mots-clés

Ordonnancement, Parcours cliniques, Recherche opérationnelle

Abstract

To relieve medical services, overpowered by the amount of outpatients coming for different needs with different priorities, health centers set up more and more dedicated services, to receive outpatient during a morning, an afternoon or even during a day. Those facitilies are called day clinics. Right now, every service has its own appoint scheduling system. This decentralization must be banned, with a services' mutualization.

The set up of clinical pathways would permit to optimize the utilisation of medical resources, while reducing the patients' waiting time.

The goal of this project is to conceive a set of softwares which can build an acceptable scheduling solution, which respects the problem's constraints.

Keywords

Scheduling, Clinical pathways, Operational research

Tuteurs académiques
Yannick Kergosien

Étudiants
Jean Coquelet (DI5)