Ordonnancement journalier dans un bloc opératoire dans le cadre d'une stratégie « open scheduling »

Fei H.[1], Duvivier D.[2], Meskens N.[1], Chu C[3].
1. Département GPO, Facultés Universitaires Catholiques de Mons (FUCaM), Mons, Belgique
2. LIL, Université du Littoral Côte d'Opale, Calais, France
3. ISTIT-OSI, Université de Technologie de Troyes (UTT), Troyes, France

Résumé
Un bloc opératoire se compose de deux parties importantes : les salles d'opération et la salle de réveil (SSPI). Il constitue non seulement le secteur le plus important et le plus coûteux dans la majorité des hôpitaux, mais il est également le service le plus complexe à gérer et à planifier, étant donnés les nombreux aléas, le nombre élevé d'acteurs, la difficulté de standardisation et de coordination des interventions chirurgicales.
Dans cette recherche, nous visons à construire un programme opératoire journalier pour un bloc opératoire constitué de plusieurs salles d'opération identiques et de lits communs en SSPI en prenant en compte le cas où les lits en SSPI ne sont pas toujours disponibles pour les demandes. Nous nous plaçons dans le contexte d'une stratégie de programmation opératoire d'« open scheduling ». Ce problème est résolu au moyen d'un algorithme génétique hybride.

Mots clés : Ordonnancement, bloc opératoire, open scheduling, algorithme génétique hybride.

1 Introduction

L'optimisation du fonctionnement des blocs opératoires est un problème extrêmement complexe. De multiples contraintes telles que l'emploi du temps des chirurgiens, leurs compétences spécifiques, la disponibilité des lits, etc. doivent être prises en compte. Dans cet article, nous nous concentrons sur l'ordonnancement journalier des interventions chirurgicales dans un bloc opératoire ayant plusieurs salles d'opération multifonctionnelles. Ce problème consiste à décider de l'ordre de passage des interventions ayant été affectées à une journée par le niveau de planification dans les salles d'opération en prenant en compte la disponibilité des chirurgiens et des lits de réveil dans la salle de réveil (SSPI).

Il y a deux politiques courantes de programmation opératoire : la programmation par allocation préalable de plages, « block scheduling », et la programmation ouverte, « open scheduling ». Dans cet article, nous nous concentrons sur la stratégie d' « open scheduling ». Notre objectif est de minimiser le coût des heures opératoires. Pour se faire, comme les ressources matérielles et humaines sont beaucoup plus coûteuses pour les salles d'opération que pour la SSPI, notre objectif est non seulement de minimiser la durée qui s'écoule entre l'heure de début de la première opération dans les salles d'opération et l'heure de fin de la dernière intervention dans la SSPI (le *makespan* du bloc opératoire), mais aussi de minimiser la durée qui s'écoule entre l'heure de début de la première intervention dans les salles d'opération et l'heure de fin de la dernière intervention dans les salles d'opération (le *makespan* des salles d'opération) en accordant à ce dernier objectif un poids plus important. De plus, nous nous employons à minimiser un objectif auxiliaire : le coût moyen des temps de passage des patients dans les salles d'opération.

Cet article se compose de quatre parties : dans la première partie, nous donnons la description du modèle et les notations utilisées. Ensuite, un algorithme génétique hybride est proposé pour résoudre le modèle

construit.

Puis, dans la troisième partie, les résultats des expérimentations numériques effectuées sur des données générées aléatoirement sont présentés et cet article se termine par les conclusions et perspectives.

2 Description du modèle et des notations

Pour résoudre notre problème d'ordonnancement, des hypothèses ont été émises :

- Chaque intervention a été affectée au préalable à une équipe chirurgicale ;
- Les ressources humaines (à l'exception du chirurgien) et matérielles sont toujours disponibles pour les demandes pendant cette journée. Nous tiendrons compte du fait qu'un chirurgien ne peut pas opérer deux interventions en même temps ;
- Toutes les salles d'opération ouvrent simultanément, l'horizon d'ordonnancement est fixé à un jour ;
- Tous les patients sont prêts pour leur intervention, c'est-à-dire que nous ne prenons pas en compte l'heure d'arrivée des patients ;
- Les cas urgents ne sont pas pris en compte ;
- Si aucun lit en SSPI n'est disponible à la fin d'une intervention alors le patient reste dans la salle d'opération jusqu'à ce qu'un lit se libère ou est transféré vers son lit d'hospitalisation s'il se réveille avant qu'un lit de réveil ne se libère. Les lits de réveil dans la SSPI sont identiques, c'est-à-dire que le patient peut être transféré vers n'importe quel lit de réveil disponible ;
- Il n'y a ni urgence ni préemption dans les salles d'opération ainsi la SSPI ;
- Le coût d'une heure d'ouverture d'une salle d'opération est beaucoup plus important qu'une heure d'ouverture de la SSPI ;
- Les durées de reconfection des lits de réveil et les durées de nettoyage des salles d'opération ne sont pas prises en compte. Précisons ici que les temps de « setup » sont indépendants de la séquence opératoire.

Certains travaux de recherche ont été réalisés à ce jour pour résoudre les problèmes d'ordonnancement journalier des blocs opératoires (Dexter et Traub, (2002) ; Fei *et al*., (2004) ; Jebali *et al*., (2006) ; Kharraja *et al*., (2002) ; Sier *et al*., (1997)). Cependant aucun, à notre connaissance, n'a traité le problème que nous considérons. Nous avons constaté certaines analogies entre le problème d'ordonnancement des blocs opératoires et celui d'ateliers de production pour lesquels de nombreuses recherches ont été réalisées. Nous nous basons sur ces travaux pour résoudre le problème d'ordonnancement des blocs opératoires. En fait, notre problème s'apparente à un modèle d'ordonnancement de type « flow-shop » hybride à deux étages où les salles d'opération multifonctionnelles constituent des « machines identiques parallèles » au premier étage et où les lits de la salle de réveil sont des « machines identiques parallèles » au deuxième étage. De plus, les durées opératoires sont dépendantes entre ces deux étages. Les interventions sont ordonnancées avec comme objectif la minimisation du coût total du bloc opératoire c'est-à-dire le coût des salles d'opération et de la SSPI, dont le cas le plus simple est déjà prouvé NP-complet (Sriskandarajah et Wagneur, (1991)).

Traditionnellement, les problèmes d'ordonnancement classique d'un « flow-shop » hybride visent à minimiser le *makespan*. Cependant, pour notre problème d'ordonnancement cette seule minimisation n'est pas suffisante car nous devons tenir compte du fait que le coût d'une heure d'ouverture de la salle d'opération est toujours beaucoup plus élevé que celui de la SSPI. Donc, notre fonction objectif consiste en une minimisation de la somme pondérée des *makespan*s : l'un deux est relatif aux salles d'opération, l'autre concerne le bloc opératoire. Un poids plus important est attribué au *makespan* des salles d'opération.

Avant de détailler l'algorithme génétique hybride proposé pour résoudre ce problème, décrivons tout d'abord

les notations importantes, illustrées dans la Figure 1.

Ω : L'ensemble de toutes les interventions (patients) à assigner pour ce jour d'ordonnancement;

$N_k^{(1)}$: Le nombre des patients (interventions) affectés à la salle d'opération *k*.

$N_k^{(2)}$: Le nombre des patients affectés au lit *k* dans la SSPI.

$C_i^{(s)}$: Le temps d'achèvement d'une intervention $i (i \in \Omega)$ à l'étage s. Ce temps représente le moment où ce patient sort de sa salle d'opération (pour *s* = 1) ou de la salle de réveil (pour *s* = 2).

$CI_i^{(1)}$ Le temps d'achèvement idéal de l'intervention $i (i \in \Omega)$ du premier étage (les salles d'opération). Un cas idéal se présente lorsque le patient i peut être transféré à la SSPI sans attente c'est-à-dire dès que son intervention est terminée dans la salle d'opération.

$s_i^{(1)}$: Le temps d'arrivée du patient (intervention) *i* au premier étage (les salles d'opération) dans le programme opératoire, c'est le temps de début de l'intervention *i* si les autres ressources nécessaires au premier étage (les salles d'opération) sont disponibles.

$s_i^{(2)}$: Le temps d'arrivée du patient *i* au deuxième étage (la SSPI) si les ressources nécessaires sont disponibles.

$C_{\max}^{(1)}$: Le temps de sortie du dernier patient du premier étage (de l'ensemble des salles d'opération) du programme opératoire π, $C_{\max}^{(1)} = \max\{C_i^{(1)} | i \in \Omega\}$.

$C_{\max}^{(2)}$: Le temps de sortie du dernier patient du deuxième étage (de la SSPI) du programme opératoire π, $C_{\max}^{(2)} = \max\{C_i^{(2)} | i \in \Omega\}$.

π : Un programme opératoire réalisable, c'est-à-dire, une séquence de tous les patients passant par une salle d'opération et la salle de réveil.

f : La valeur de la fonction objectif du programme opératoire π, $f = \omega C_{\max}^{(1)} + C_{\max}^{(2)}$ où ω représente le rapport entre le coût d'une heure d'ouverture des salles d'opération et le coût d'une heure d'ouverture de la SSPI.

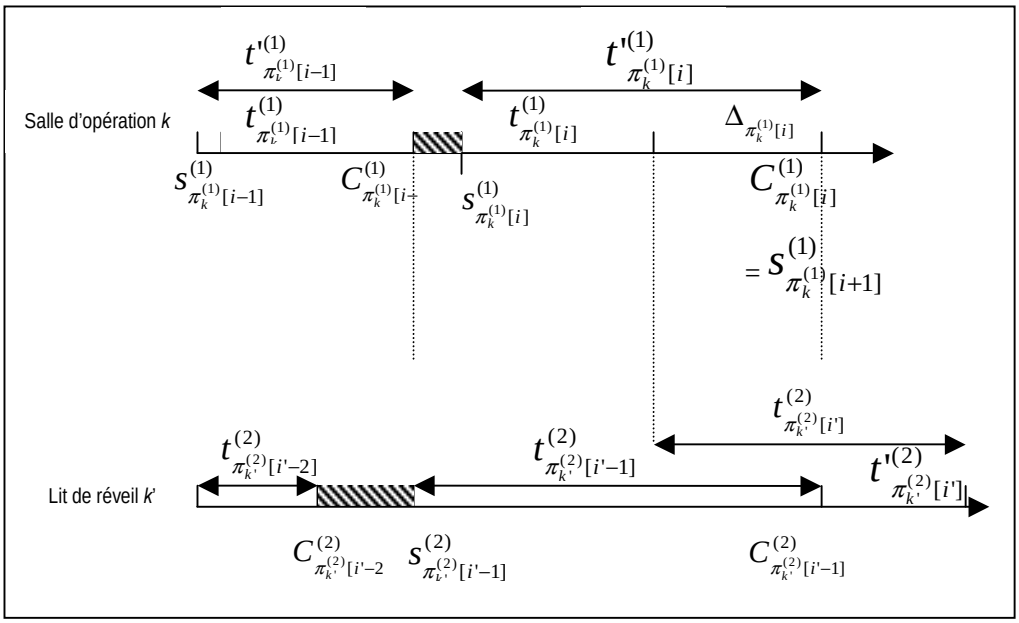

Figure 1 : Illustration des notations utilisées

Notre modèle peut être décrit comme suit :

Par jour, *N* patients sont à assigner dans le bloc opératoire composé de M_1 salles d'opération identiques et de M_2 lits de réveil identiques dans la SSPI. Chaque patient *i* va d'abord être traité par son chirurgien ch_i dans sa salle d'opération où sa durée opératoire $t_i^{(1)}$ est estimée à l'avance, et ensuite va être transféré vers un lit de réveil dans la SSPI. Il va y rester jusqu'à ce qu'il se réveille, sauf cas particulier, s'il n'y a pas de lit disponible dans la SSPI. En ce cas, il reste « bloqué » dans sa salle d'opération. La durée de réveil $t_i^{(2)}$ peut donc être partagée entre la salle d'opération et la salle de réveil. Ce problème d'ordonnancement

vise tout d'abord à minimiser le coût total du bloc opératoire. S'il existe un ensemble de solutions dont cette valeur est la même, un objectif secondaire est utilisé pour trouver la meilleure solution parmi elles. Celui-ci vise à la minimisation du coût moyen des heures d'occupation des salles d'opération ainsi que la durée d'occupation du bloc opératoire ($C_{\max}^{(2)}$), c'est-à-dire $\min\{\omega * moyenne(C_i^{(1)}) + C_{\max}^{(2)}\}$.

3 Résolution du problème d'ordonnancement par l'algorithme génétique hybride

Pour résoudre le problème d'ordonnancement des interventions dans le bloc opératoire, nous proposons un algorithme génétique couplé avec une procédure de recherche Tabou afin d'améliorer localement un individu choisi. Nous avons utilisé la procédure de recherche Tabou décrite par Nowicki et Smutnicki (1998). L'algorithme génétique hybride proposé se déroule comme ci-dessous :

Etape 1 : Génération d'une population initiale. Celle-ci contient un *pool* génétique qui représente un ensemble d'individus (solutions réalisables)). Nous avons utilisé une procédure de décomposition pour générer la population initiale des individus, où les patients (interventions) sont tout d'abord assignés au premier étage (les salles d'opération) et ensuite au deuxième étage (la SSPI).

Etape 2 : Calcul d'une valeur d'adaptation *(fitness)* pour chaque individu. Elle est fonction directe de la proximité des différents individus avec l'objectif de confirmer que les meilleurs individus ont les plus grandes probabilités d'être choisis.

Etape 3 : Sélection d'un couple de parents en utilisant une technique de sélection selon leurs valeurs d'adaptation. Afin de régénérer la population initiale pour la prochaine génération, nous utilisons une variante de la méthode de roulette (Roulette Wheel, RW) comme opérateur de sélection basé sur les valeurs d'adaptation (*fitness*) des candidats.

Etape 4 : Croisement des génomes des parents avec une probabilité pour générer deux enfants ;
Pour obtenir de nouveaux individus (enfants) à partir d'une population initiale d'une itération, nous utilisons un des deux opérateurs de croisement aléatoirement (Le croisement s'effectue soit sur le vecteur V_1 représentant l'ordre des interventions dans les salles d'opération, soit sur le vecteur V_2 représentant l'affectation des lits aux patients dans la SSPI. Le taux de croisement est de 85%). Il faut que les nouveaux individus obtenus par ces opérateurs de croisement soient aussi des solutions réalisables.

Etape 5 Choisir aléatoirement un enfant généré par l'opérateur de croisement dans l'étape 4 et lui appliquer l'opérateur de mutation avec une probabilité de mutation (P_m est fixée à 1%) pour obtenir un nouvel individu.

Etape 6 : Appliquer une variante de la procédure de recherche Tabou systématiquement, comme un opérateur d'amélioration locale, à l'autre enfant généré par l'opérateur de croisement non choisi par l'étape 5 pour la mutation. Un nouvel individu est obtenu.

Etape 7 : Choisir les individus dans la génération courante selon leurs valeurs d'adaptation pour régénérer la population initiale de la génération prochaine ; dans cette étape, nous appliquons une technique d'élitisme afin d'assurer que le meilleur individu actuel soit toujours présent dans la population.

Etape 8 : Itérer à partir de l'étape (2) jusqu'à ce qu'une des conditions d'arrêt soit satisfaite.

Nous allons juste détailler ici le codage utilisé dans notre algorithme génétique.

Au vu des spécificités de notre problème d'ordonnancement, le chromosome proposé pour représenter un individu (une solution réalisable) se constitue de cinq parties:

1. V_1 : le vecteur séquentiel, de N dimensions, des patients au premier étage représentant l'ordre des patients (interventions) dans les salles d'opération ;

2. V_2 : le vecteur d'affectation de *N* dimensions représentant les indices des lits de réveil au deuxième étage (la SSPI) correspondant à l'ordre des patients représenté par V_1 ;
3. V_3 : le vecteur de séparation des salles d'opération, qui est de $(M_1 - 1)$ dimensions, représentant le nombre de positions dans le vecteur V_1 affectées à la première salle d'opération, puis à la deuxième, … jusqu'à la $(M_1 - 1)$ salle d'opération ;
4. V_4 : le vecteur séquentiel des patients au deuxième étage, qui est de *N* dimensions, représentant l'ordre des patients sur chaque lit de réveil dans la SSPI ;
5. V_5 : le vecteur de séparation des lits de réveil, qui est de $(M_2 - 1)$ dimensions, indiquent combien de positions dans le vecteur V_4 sont affectées au premier lit de la SSPI.

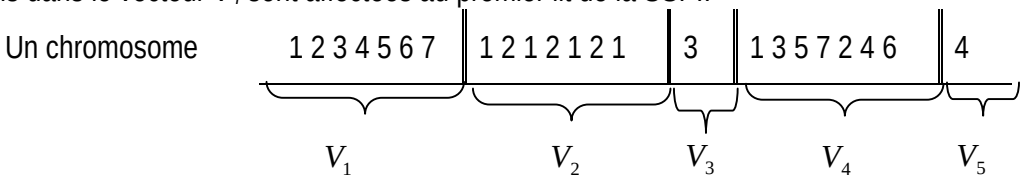

Figure 2: Un exemple de la stratégie du codage

La figure 2 nous permet de visualiser un exemple de la stratégie de codage. Ce chromosome représente le codage d'un programme opératoire réalisable pour un bloc opératoire constitué de deux salles d'opération et de deux lits de réveil dans la SSPI. Ce chromosome nous montre un programme opératoire où trois interventions (1, 2, et 3) sont opérées dans l'ordre {1-2-3} dans la première salle d'opération et quatre interventions (4, 5, 6 et 7) dans l'ordre {4-5-6-7} dans la deuxième salle d'opération. Après leurs interventions, les patients 1, 3, 5 et 7 sont transférés au lit de réveil 1 et les autres au lit de réveil 2 dans la SSPI. L'ordre des patients passant sur le lit de réveil 1 est 1-3-5-7 et l'ordre pour le lit de réveil 2 est 2-4-6. Dès qu'un chromosome est défini, nous obtenons un programme opératoire pour ce bloc opératoire.

4. Résultats expérimentaux

Pour réaliser nos expérimentations, nous avons utilisé un IBM ThinkPad T42 (CPU : PM 1,6 GHz, mémoire : 256 Mo) avec comme logiciel Microsoft VC++ 6.0.

4.1 Données

La plupart des études menées à ce jour utilisent une loi normale (Dexter,(2000)) ou Lognormale (Strum *et al*., (1998)) pour générer les données expérimentales concernant des durées opératoires et de réveil des interventions (Dexter et Tinker, (1995)).

Les données utilisées dans notre expérimentation sont les suivantes (les durées sont exprimées en minutes) :

- La durée opératoire de l'intervention *i*, t_i, est générée entre [30, 150] par la loi de Pearson III (Combes *et al*. (2004)) où la moyenne est 60 minutes, l'écart type est 15 minutes ;
- La durée de post-anesthésie de l'intervention *i*, $t_i^{(2)}$, est générée entre [30, 120] par une loi de Pearson III où la moyenne est égale à la durée opératoire correspondante en salle d'opération $t_i^{(1)}$ moins 10. Cette manière de générer la durée de post-anesthésie est utilisée par Dexter et Tinker (1995), Kharraja *et al.* (2002) et Jebali *et al.* (2006) sauf que nous utilisons la loi Pearson III au lieu de la loi lognormale. L'écart type est égal à 15 minutes ;
- Le rapport entre le coût d'une heure d'ouverture de la salle d'opération et d'une heure d'ouverture de la SSPI : ω= 5,8 ;
- Il y a 6 salles d'opération disponible ;

- Il y a 10 lits de réveil dans la SSPI ;
- Toutes les interventions sont affectées aléatoirement à 8 chirurgiens à l'avance ;
- Le nombre d'interventions *N* à affecter sont de 20, 30, 40 et 50.

Nous avons exécuté pour chaque valeur de *N* 10 exemples, donc au total 40 exemples ont été testés.

4.2 Borne inférieure pour l'évaluation de la performance de l'algorithme proposé

Afin d'avoir un bon indicateur pour l'évaluation de la performance de notre algorithme, une borne inférieure est proposée. Gupta, en 1988, a été le premier à avoir proposé une borne inférieure pour un modèle de « flow-shop » hybride à deux étages. Puis, Lee et Vairaktarakis (1994) ont proposé une autre borne inférieure meilleure que celle de Gupta (1988) ; Haouari et M'Hallah (1997) l'ont encore améliorée. Il faut noter que leurs bornes inférieures sont développées pour le cas où les machines à chaque étage sont identiques et où les durées opératoires à chaque étage sont connues. Ce n'est pas le cas pour notre problème d'ordonnancement où les durées de réveil peuvent être partagées entre les salles d'opération et la SSPI. Nous avons donc développé une nouvelle borne inférieure globale *OLB*, adaptée à notre problème, en nous basant sur les idées développées par Haouari et M'Hallah (1997).

Lemme 1 : $$OLB = \omega * \max\left\{ t^1_{i^0}, \frac{\sum_{i\in\Omega} t_i^{(1)}}{M_1} \right\} + \max\left\{ t^1_{i^0} + t^{(2)}_{i^0}, \frac{SPT(M_2) + \sum_{i\in\Omega}(t_i^{(2)} + t_i^{(2)})}{M_1 + M_2} \right\}$$

est une borne inférieure de la valeur optimale de la fonction objectif du problème d'ordonnancement d'une stratégie d' « open scheduling ».

4.3 Résultats numériques

Nous avons fait une comparaison entre la valeur de la fonction objectif des solutions obtenues par l'algorithme génétique hybride et la borne inférieure *OLB* que nous avons proposé dans la section 4.2. Notre algorithme hybride se compose de deux parties : l'algorithme génétique et la recherche Tabou. Donc, nous avons aussi fait les comparaisons entre les résultats de l'algorithme génétique sans la recherche Tabou et ceux de la pure recherche Tabou. Nous avons également fait une comparaison entre la valeur de critère auxiliaire des solutions obtenues par les trois algorithmes.

Tout d'abord, nous présentons les indicateurs de comparaison utilisés :

GATS : la valeur de la fonction objectif, $f = \omega * C_{max}^{(1)} + C_{max}^{(2)}$, de la solution obtenue par l'algorithme génétique hybride ;

GATS_A : la valeur du critère auxiliaire, $f' = \omega * moyenne(C_i^{(1)}) + C_{max}^{(2)}$, de la solution obtenue par l'algorithme génétique hybride ;

GA : la valeur de la fonction objectif, $f = \omega * C_{max}^{(1)} + C_{max}^{(2)}$, de la solution obtenue par l'algorithme génétique sans l'amélioration locale ;

GA_A : la valeur du critère auxiliaire, $f' = \omega * moyenne(C_i^{(1)}) + C_{max}^{(2)}$, de la solution obtenue par l'algorithme génétique sans l'amélioration locale ;

TS : la valeur de la fonction objectif de la solution obtenue par la recherche Tabou utilisée seule ;

TS_A : la valeur du critère auxiliaire obtenue par la recherche Tabou utilisée seule ;

OLB : la borne inférieure proposée du problème d'ordonnancement;

CPUGATS : le temps moyen de calcul de l'algorithme génétique hybride proposé (en secondes) ;

CPUGA : le temps moyen de calcul de l'algorithme génétique simple (sans l'amélioration locale) (en secondes) ;

CPUTS : le temps moyen de calcul de la recherche Tabou (en secondes).

N	GATS	GA	TS	OLB
20	2432,16	2642,78	2721,74	2397,42
30	3672,16	4475,58	4420,94	3382,12
40	4959,04	6275,04	6296,18	4378,24
50	6455,74	8078,04	8213,96	5144,68

Tableau 1 : Comparaison des valeurs moyennes de la fonction objectif des solutions obtenues par les différentes méthodes

N	GATS A	GA A	TS A
20	1860,96	2240,74	2395,01
30	3310,14	4021,23	3945,34
40	4592,39	5791,42	5727,69
50	5955,39	7462,29	7516,61

Tableau 2 : Comparaison des valeurs moyennes du critère auxiliaire des solutions obtenues par les différentes méthodes

N	CPUGATS	CPUGA	CPUTS
20	162,6	1,079	0,984
30	476,111	1,622	1,555
40	1098,708	2,381	2,339
50	2662,973	4,444	4,224

Tableau 3 : Comparaison entre les temps de calcul (en secondes)

Les tableaux 1 et 2 montrent que l'algorithme génétique hybride peut toujours trouver la meilleure valeur de la fonction objectif ainsi que la valeur de critère auxiliaire par rapport à d'autres algorithmes bien qu'il n'y ait pas de grande différence entre les résultats des trois algorithmes. De plus, tous les écart-types sont du même ordre de grandeur pour toutes les méthodes appliquées. Cependant dans le cas où le nombre d'interventions est grand, la solution obtenue par l'algorithme hybride n'est pas très proche de la borne inférieure proposée, dans le cas de petite taille (N = 20), cette borne inférieure est de bonne qualité. Cela signifie que les solutions de notre algorithme génétique hybride sont très satisfaisantes pour les problèmes de petite taille mais il est moins performant pour les problèmes de grande taille par rapport aux bornes inférieures proposées. Par conséquent, nous allons mener une étude complémentaire afin de déterminer si c'est la borne inférieure développée ou l'algorithme proposé qu'il faut améliorer.

Concernant le temps de calcul, le tableau 3 montre que notre algorithme génétique hybride consomme beaucoup plus de temps de calcul que les autres mais reste néanmoins raisonnable. Par conséquent, il s'agit d'un bon compromis entre la rapidité et la qualité de la solution.

5 Conclusion et perspectives

Dans cet article, nous avons présenté un problème d'ordonnancement journalier du bloc opératoire. Ce problème d'ordonnancement est modélisé comme un problème de « flow-shop » hybride à deux étages et nous avons proposé un algorithme génétique hybride permettant de trouver un programme opératoire réalisable et efficace pour une journée.

En comparant les résultats obtenus par notre méthode avec notre borne inférieure, l'algorithme génétique sans l'amélioration locale et la pure recherche Tabou, nous trouvons que notre méthode peut trouver une solution de bonne qualité dans un temps raisonnable.

Nos recherches futures porteront sur :

- La prise en compte des temps de changement entre deux interventions, notamment les temps nécessaires au remplacement du matériel et à la préparation des instruments ;
- La prise en compte des disponibilités de toutes ressources humaines et matérielles ;
- La prise en compte des cas d'urgence ;
- Améliorer/accélérer l'Algorithme Génétique et ses constituants.

Références bibliographiques

Combes C., A . Dussauchoy, S. Chaabane, N. Smolski, J.P. Viale et J.C. Souquet (2004). Démarche méthodologique d'analyse des données pour la planification des blocs opératoires : une application à un service d'endoscopie, *Actes GISEH'04, Mons, Belgique*.

Dexter F. (2000). A strategy to decide whether to move the last case of the day in an operating room to another empty operating room to decrease overtime labour cost, *Anesthesia & Analgesia*, 91.

Dexter F., J. Tinker (1995). Analysis of strategies to decrease postanesthesia care unit costs, *Anesthesiology*, 82(1), 94-101.

Dexter F., R.D. Traub (2002). How to schedule elective surgical cases into specific operating rooms to maximize the efficiency of use of operating room time, *Anesthesia & Analgesia*, 94, 933-942.

Fei H. A. Artiba, C. Chu (2004). A memetic algorithm for operating room scheduling, *International Conference on Computational & Experimental Engineering and Sciences, Portugal*.

Gupta J.N.D. (1988). Two-stage hybrid flowshop scheduling problem, *Journal of Operation Research Society*, 39, 359-364.

Haouari H., R. M'Hallah (1997). Heuristic algorithms for the two-Stage hybrid flowshop problem, Operations Research Letters, 21(1), 43-53.

Jebali A., A.B. Hadj Alouane, P. Ladet (2006). Operating room scheduling, *International Journal of Productions Economics*, 99, 52-62.

Kharraja S., S. Chaabane, E. Marcon (2002). Evaluation de performances pour deux stratégies de programmation opératoire de bloc, *Proceeding de la 2ème conférence Internationale Francophone d'Automatique, France*.

Lee C.Y., Vairaktarakis (1994): Minimizing makespan in hybrid flowshops, *Operations Research Letters*, 16(3), 149-158.

May J.H., D.P. Strum, L.G. Vargas (2000). Fitting the LogNormal distribution to surgical procedure times, *Decision Sciences,* 31(1), 129-148.

Nowicki E., C. Smutnicki (1998). The flow shop with parallel machines: A Tabu Search approach, *European Journal of Operational Research*, 106, 2, 226-253.

Sier D., P. Tobin, C. McGurk (1997). Scheduling surgical procedures, *Journal of Operating Research Society*, 48, 884-891.

Sriskandarajah C., E. Wagneur (1991). Hierarchical control of the two processor flow-shop with state dependent processing times: complexity analysis and approximate algorithms, INFOR, *Canadian Journal of Information Systems and Operational Research*, 29(3), 193-205.

Strum D.P., L.G. Vargas, J.H. May, G. Bashein (1997). Surgical suite utilization and capacity planning : a minimal cost analysis model, *Journal of Medical systems,* 22(5).