Deux Méthodes de Recherche Locale pour Résoudre un Problème d'Horaire du Personnel Infirmier dans un Établissement Hospitalier

Belaïd Ahiod *Ilham Berrada* *Ismaïl Kassou*

LMD-ENSIAS, BP : 713
Agdal, Rabat, Maroc

Abstract

Dans cet article, nous traitons le problème de confection d'horaire du personnel infirmier dans un établissement hospitalier en tenant compte des contraintes issues de l'administration de l'hôpital et des désirs du personnel. Ce problème est modélisé comme un problème de programmation mathématique multi-objectifs non-linéaires. Nous proposons pour sa résolution deux techniques de recherche locale. La première est une adaptation de la méthode de recherche tabou qui a connu un grand succès pour résoudre plusieurs problèmes d'optimisation combinatoire à un seul objectif. La deuxième technique est une adaptation des algorithmes génétiques. Ces deux heuristiques ont été testées sur un jeu de données réelles issues de deux unités de soins de l'hôpital Hôtel-Dieu de Montréal.

Keywords: Horaire du personnel infirmier, Multi-objectifs, Recherche tabou, Algorithmes génétiques.

1 Introduction

On s'intéresse au problème de confection d'horaire du personnel infirmier dans un établissement hospitalier. Généralement, la préparation des grilles de travail se fait manuellement et requiert un temps considérable et un effort important de la part du planificateur. Ce dernier est chargé d'assurer un fonctionnement normal des différents services et d'assurer au personnel les meilleures conditions possibles de travail et ce, au moindre coût pour l'administration. Pour l'assister dans cette lourde tâche, nous avons mis au point un système interactif permettant de réduire considérablement et efficacement le temps de confection de l'horaire. Ce système est constitué de deux composantes principales. Dans ce papier, la première composante que nous décrivons

est une composante algorithmique intégrant deux techniques heuristiques. La seconde composante correspond à l'Interface Homme-Machine (IHM) permettant au planificateur d'intervenir dans le processus de résolution.

La composante algorithmique du système permet de choisir parmi deux heuristiques de recherche locale pour résoudre le problème d'horaire. Ces approches de résolution se prêtent particulièrement bien à des situations exceptionnelles (comme des absences non planifiées) où l'on désire ajuster l'horaire sans pour autant trop le modifier. En effet, on démarre à partir d'un horaire initial que l'on remplace par un horaire (qui devient un nouvel horaire) pris dans le voisinage du premier et ainsi de suite. Nous pouvons ainsi générer des horaires alternatifs de meilleure ou d'aussi bonne qualité que ceux obtenus manuellement par le planificateur.

Le problème d'horaire du personnel infirmier auquel nous nous intéressons sera défini dans la section 2. Dans la section 3, nous décrivons les deux techniques heuristiques de résolution que nous proposons pour résoudre ce problème. La première est une adaptation de la méthode de recherche tabou [10, 14] qui a connu un large succès pour la résolution de divers problèmes d'optimisation combinatoire à un seul objectif. La seconde approche concerne l'adaptation des algorithmes génétiques [15]. Enfin, dans la section 4, nous présentons plusieurs tests numériques effectués sur des données réelles, de tailles différentes et issues de deux unités de soins de l'hôpital Hôtel-Dieu de Montréal.

2 Définition du Problème d'Horaire

Le problème de confection d'horaire du personnel fait partie d'une classe de problèmes très complexes à modéliser et à résoudre. En effet, ce problème est de nature combinatoire et plusieurs contraintes conflictuelles entrent en ligne de compte.

Résoudre un problème de confection d'horaire du personnel infirmier dans un établissement hospitalier revient à spécifier les jours de travail et de congé de chaque employé pour chaque unité de soins. Un service permanent de 24 heures par jour et de 7 jours par semaine doit être assuré. Deux types d'horaires existent : les horaires cycliques où chaque employé choisit son ou ses horaires hebdomadaires parmi un ensemble d'horaires préétablis et les horaires non-cycliques. Dans le contexte infirmier, il est préférable d'utiliser des horaires non-cycliques afin d'augmenter le niveau de flexibilité pour satisfaire les désirs des employés.

Dans certains établissements hospitaliers, les horaires sont rotatifs en ce sens que les employés changent régulièrement de relèves. Dans ce travail, nous supposons que trois relèves de huit heures chacune sont quotidiennement en service : relève du jour (08:00 - 16:00), relève du soir (16:00 - 24:00) et relève de nuit (00:00 - 08:00). Les horaires considérés ne sont pas rotatifs, et par conséquent, chaque employé travaille

dans une seule relève. De plus, ces horaires sont établis sur des périodes de deux semaines.

La nature conflictuelle des contraintes du problème nous a naturellement amené à les diviser en deux types : des contraintes rigides et d'autres souples qui peuvent être violées mais dont nous minimisons la violation. Les contraintes rigides correspondent aux exigences administratives et concernent les fins de semaine de travail, le nombre hebdomadaire de jours de travail des employés, la répartition équilibrée du manque ou du surplus en personnel sur toute la semaine et les congés fériés. Les contraintes souples traduisent les désirs du personnel infirmier relatifs au nombre de jours consécutifs de travail, aux congés fériés et aux congés hebdomadaires.

Le modèle mathématique préconisé est un problème de programmation en nombres entiers multi-objectifs non-linéaires [4, 5]. Bien qu'il soit plus complexe que ceux rencontrés dans la littérature, il a le mérite de mieux représenter la réalité des hôpitaux et aussi de prendre en compte le traitement des congés fériés et des demandes spéciales qui sont, dans la plupart des cas, difficiles à modéliser. Notre modèle permet de tenir compte de tous les objectifs même conflictuels et offre la possibilité de modifier la priorité relative aux objectifs.

3 Approches de Résolution

Depuis de nombreuses années, plusieurs heuristiques de recherche locale ont été développées pour résoudre divers problèmes d'optimisation combinatoire. Le principe de ces heuristiques est le suivant : à partir d'une solution initiale réalisable, on applique itérativement des modifications locales à la solution courante, tant qu'elles permettent d'améliorer la valeur de l'objectif. L'inconvénient de ces heuristiques est que la recherche est bloquée par les optimums locaux dont la valeur peut être très éloignée de celle de l'optimum global.

Par conséquent, de nouvelles heuristiques de recherche locale plus générales pouvant surmonter l'obstacle de l'optimalité locale ont été développées au cours des dernières années. Nous citons notamment la méthode de recherche tabou qui est une méthode de voisinage et les algorithmes génétiques, que nous adaptons pour résoudre le problème d'horaire multi-objectifs.

3.1 Adaptation de la Méthode de Recherche Tabou

La méthode de recherche tabou a été développée par Glover[10] et parallèlement par Hansen[14]. Depuis, elle a connu un large succès pour résoudre divers problèmes d'optimisation combinatoire à un seul objectif (voir, par exemple, [7, 8, 9]). On trouvera plus de détails sur cette méthode dans [11, 12, 13].

L'adaptation de la méthode de recherche tabou est un processus itératif qui consiste à se déplacer d'un horaire réalisable à un autre tout en minimisant, selon le critère de l'ordre lexicographique dépendant des priorités initiales, les objectifs considérés à l'intérieur du voisinage de l'horaire courant [5]. Un tel critère tente de réduire, en premier, les violations des objectifs ayant la plus haute priorité.

Un horaire voisin est obtenu à l'aide d'une modification représentant l'échange d'un jour de congé avec un jour de travail pour un employé pendant une semaine donnée. Une telle modification est donc caractérisée par un triplet (i, j_1, j_2) où i est l'indice de l'employé dont l'horaire est modifié et j_1 est la journée de travail (ou de congé) qui est échangée avec la journée de congé (ou de travail) j_2 . Les journées j_1 et j_2 sont choisies dans une même semaine.

Compte tenu du fait que la taille du voisinage V_x d'une solution x (un horaire) peut être très grande, nous nous proposons de générer seulement un échantillon de voisins $V^* \subset V_x$ relativement à l'objectif le plus prioritaire dont la violation est maximale. Celui-ci appartient à l'une des deux catégories d'objectifs qu'on peut distinguer pour le problème d'horaire : les objectifs définis par les contraintes associées aux jours de la semaine (i.e. les contraintes où l'on fixe le jour de semaine et on fait varier l'indice relatif aux employés. A titre d'exemple, la contrainte: "satisfaire, exactement, la demande journalière en personnel prévue pour les lundis et vendredis") et les objectifs restants qui sont définis par les contraintes associées aux employés. La procédure de génération du voisinage retenue est décrite dans [5].

Pour déterminer le "meilleur" voisin x' dans V^* , nous évaluons l'influence de chaque modification sur tous les objectifs. La "meilleure" modification retenue sera celle qui viole le moins les objectifs les plus prioritaires. Les modifications sont alors comparées selon l'ordre lexicographique précité des priorités relatives aux objectifs. S'il y a plusieurs modifications de valeurs équivalentes, nous en choisissons une de manière aléatoire [5].

Il est important de noter que la méthode de recherche tabou permet le déplacement vers une solution voisine même si elle n'améliore pas la fonction objectif et ce, afin de s'éloigner des minimums locaux vers un optimum global. Cependant, ce processus de recherche introduit le risque de cycle dès que l'on sort d'un minimum local, en y retournant aussitôt. Pour remédier à ce risque, la méthode utilise une liste tabou T permettant d'interdire temporairement certains déplacements qui pourraient ramener la procédure vers des solutions déjà rencontrées. Pour notre problème d'horaire, les éléments de la liste tabou T sont des modifications plutôt que des solutions. Ainsi, si une modification (i, j_1, j_2) est utilisée pour se déplacer d'une solution x à une autre voisine x', alors les éléments (i, j_1, j_2) et (i, j_2, j_1) sont ajoutés dans la liste T.

Enfin, ce processus itératif est initialisé par un horaire réalisable généré à l'aide d'une heuristique similaire à celle du "first-fit decreasing" utilisée pour résoudre le

problème du "bin-packing", [5]. Cette heuristique fonctionne en deux étapes principales. Dans la première étape, on répartit uniformément le manque ou le surplus journalier en personnel tout en minimisant le manque pour les lundis et vendredis (les jours où le nombre d'absences est généralement élevé). Ensuite, on établit successivement l'horaire de chaque employé pris un à la fois en affectant ses journées de travail aux jours où la demande résiduelle est la plus élevée. Le processus se poursuit jusqu'à ce que l'horaire de tout le personnel soit établi.

Le critère d'arrêt retenu est le nombre maximal d'itérations sans amélioration de la solution.

3.2 Adaptation des Algorithmes Génétiques

Les algorithmes génétiques (Ags) sont des méthodes d'optimisation basées sur une analogie avec le phénomène de l'évolution et de la sélection naturelle. Cette approche développée par Holland [15], a été utilisée pour traiter plusieurs problèmes d'optimisation combinatoire (voir, par exemple, [7, 8, 9, 18]). Pour plus de détails sur les aspects théoriques et pratiques des Ags, voir [2, 3, 6, 17].

Les Ags se caractérisent par l'utilisation de populations d'individus représentant des solutions du problème étudié qui évoluent en appliquant des opérateurs génétiques de croisement et de mutation. Les populations suivent un processus de sélection naturelle qui permet aux individus les mieux adaptés (les meilleures solutions) de se reproduire le plus souvent et de contribuer davantage aux populations futures. Pendant la phase de reproduction, certains individus sélectionnés de la population courante, appelés "parents", sont combinés entre eux pour produire de nouveaux individus appelés "enfants". Ces derniers partagent plusieurs caractéristiques héritées des parents et forment la nouvelle population. De plus, certaines caractéristiques des enfants peuvent être modifiées en subissant des mutations aléatoires pour maintenir une diversité dans la population. Enfin, pour conserver une taille fixe des populations, certains individus, généralement les moins adaptés, seront éliminés de la nouvelle population. Une fonction d'évaluation ou fonction objectif joue le rôle d'environnement pour distinguer entre les individus les mieux adaptés et les moins adaptés.

Adapter un Ag à un problème d'optimisation combinatoire revient à spécifier les cinq composantes suivantes:

1. une représentation génétique des solutions du problème,

2. une façon de créer une population initiale de solutions,

3. une fonction d'évaluation jouant le rôle de l'environnement, distinguant les solutions en termes de leur adaptation,

4. des opérateurs génétiques qui modifient la composition des enfants pendant la reproduction,

5. des valeurs pour les divers paramètres que l'Ag utilise (taille de la population, probabilités d'application des opérateurs génétiques, etc.)

3.2.1 Représentation des Solutions

Dans le cas du problème d'horaire auquel nous nous intéressons, nous avons choisi une représentation matricielle binaire à deux dimensions. Soit une matrice $X = (x_{ij})$, avec

$$x_{ij} = \begin{cases} 1 & \text{si l'employé } i \text{ travaille le jour } j \\ 0 & \text{sinon} \end{cases} \qquad 1 \leq i \leq |I|,\ 1 \leq j \leq 14$$

où, $|I|$ désigne le cardinal de l'ensemble I des employés d'une relève donnée.

3.2.2 Population Initiale

Nous proposons deux procédures pour générer aléatoirement une solution initiale, chacune étant utilisée m fois pour créer une population initiale de taille m. La première procédure établit l'horaire de chaque employé en affectant aléatoirement ses jours de travail et de congé. La solution ainsi générée ne satisfait pas forcément la répartition uniforme du manque ou du surplus journalier en personnel sur toute la semaine. La deuxième procédure est similaire à la procédure proposée pour initialiser l'adaptation de la méthode de recherche tabou (voir section 3.1), sauf qu'on y introduit un caractère aléatoire dans le choix des employés et des jours de la semaine pour augmenter la diversité de la population initiale. Dans la première étape, on répartit uniformément le manque ou le surplus journalier en personnel par rapport à des jours choisis aléatoirement. Ensuite, on établit successivement l'horaire des employés sélectionnés aléatoirement en affectant leurs journées de travail aux jours où la demande résiduelle est la plus élevée. Le processus se poursuit jusqu'à ce que l'horaire de tout le personnel soit établi.

3.2.3 Evaluation des Solutions

Le critère d'évaluation des solutions est identique à celui utilisé dans la méthode tabou. On minimise les objectifs selon l'ordre lexicographique relatif aux priorités préétablies associées aux objectifs.

3.2.4 Sélection des Parents

La sélection des parents peut se faire de diverses façons (voir [2, 17]). Dans notre adaptation, nous nous sommes intéressés à une technique de sélection par tournoi proposée dans [2] et qui consiste à choisir aléatoirement T individus de la population courante. Le meilleur de ces individus est retenu comme premier parent. Ce processus est répété une deuxième fois pour sélectionner le deuxième parent.

Notons que lors de l'adaptation d'un Ag, on est confronté au problème de la convergence prématurée vers un optimum local. Ce problème est généralement dû au mécanisme de sélection utilisé qui peut favoriser certains individus comparativement bien adaptés (mais pas optimaux). Afin de mieux contrôler le taux de convergence, il est souvent utile de pouvoir mesurer de façon précise le degré de diversité d'une population. Cette mesure qui est généralement basée sur la notion d'entropie [7] dépend du problème traité. Fleurent et Ferland proposent une mesure de diversité adaptée pour les problèmes de coloration de graphes, d'affectation quadratique et de la clique maximale dans un graphe [9]. Nous l'avons adapté au problème d'horaire comme suit.

Nous définissons l'entropie de la population à l'aide de la formule suivante:

$$E = \frac{-\sum_{i=1}^{|I|}\sum_{j=1}^{14}\sum_{k=0}^{1}\frac{n_{ijk}}{m}\log\frac{n_{ijk}}{m}}{|I|\times 14\times\log 2} \tag{3.1}$$

avec la convention $0\log 0 = 0$.

n_{ijk} représente le nombre de fois, dans une population de taille m, que l'employé i prend la valeur k (0 ou 1) le jour j.

Cette entropie normalisée prend ses valeurs dans l'intervalle $[0,1]$. Une valeur $E = 0$ indique que la population est entièrement formée par des individus identiques, alors qu'une valeur proche de 1 suggère une grande diversité dans la population. Par conséquent, E peut être vue comme une mesure de diversité dans la population.

Pour augmenter le degré de diversité de la population, nous utilisons un mécanisme de sélection qui est ajusté dynamiquement durant l'évolution. Ce mécanisme est une variante de la sélection par tournoi et consiste à faire une sélection très "agressive" lorsque la diversité est grande et moins "agressive" une fois que l'Ag converge.

Pour ce faire, nous proposons les relations suivantes qui expriment la taille du tournoi T en fonction de la mesure de diversité E obtenue par la relation (3.1):

$$T = 1 + ((m-1)\times E) \tag{3.2}$$
$$T = \exp(E\times\log(m)) \tag{3.3}$$

Dans tout ce qui suit, nous appellerons sélection par tournoi linéaire (resp. exponentiel), tout mécanisme de sélection par tournoi basé sur la relation (3.2) (resp. basé sur la relation (3.3)).

3.2.5 Les Opérateurs Génétiques

a. Opérateurs de Croisement

Nous utilisons trois types d'opérateurs de croisement : **croisement "à un point"**, **"à deux points"** et **"uniforme"** . Les croisements sont effectués sur les lignes de chaque solution plutôt que sur les colonnes. Un croisement relatif aux colonnes conduit généralement à une violation de la contrainte rigide régissant le nombre obligatoire de jours de travail de chaque employé. Notons que le croisement par ligne peut engendrer un déséquilibre de la demande journalière en personnel. La procédure de réparation donnée dans le tableau 3.1 permet alors de rééquilibrer cette demande.

	Tant que la solution n'est pas équilibrée faire:
Etape 1 :	Déterminer le plus grand écart entre deux colonnes dans la ligne "équilibre" . Choisir la première paire de colonnes (j_1, j_2) produisant cet écart.
Etape 2 :	Choisir aléatoirement un employé i tel qu'on peut effectuer un échange entre les jours j_1 et j_2 afin de réduire cet écart. Effectuer l'échange entre j_1 et j_2 pour l'employé i.

Tableau 3.1 : Algorithme de réparation du déséquilibre d'une solution

Considérons, par exemple, les tableaux 3.2 et 3.3 qui représentent deux parents sélectionnés pour le croisement. On ne considère que la partie d'un parent relative à la première semaine, le même processus pourra être répété pour la deuxième semaine. La ligne "demande" indique la demande journalière en personnel formulée par l'administration. La ligne "équilibre" est introduite pour représenter la répartition du manque ou du surplus en personnel, la valeur -1 (resp. +1) indiquant un manque le jour correspondant (resp. un surplus).

	2	**3**	**4**	**5**	**6**
1	1	1	0	1	1
2	0	0	1	1	0
3	1	1	0	0	0
4	0	0	0	0	1
demande	**2**	**2**	**2**	**2**	**2**
équilibre	**0**	**0**	**-1**	**0**	**0**

Tableau 3.2 : Parent 1

	2	**3**	**4**	**5**	**6**
1	1	0	1	1	1
2	0	0	0	1	1
3	1	1	0	0	0
4	0	0	1	0	0
demande	**2**	**2**	**2**	**2**	**2**
équilibre	**0**	**-1**	**0**	**0**	**0**

Tableau 3.3 : Parent 2

Les tableaux 3.4 et 3.5 représentent le résultat de l'application de l'opérateur de croisement du type "à un point" . Pour ce croisement, nous supposons que la position de coupure générée aléatoirement correspond à la ligne 2.

	2	3	4	5	6
1	1	1	0	1	1
2	0	0	1	1	0
3	1	1	0	0	0
4	0	0	1	0	0
demande	2	2	2	2	2
équilibre	0	0	0	0	-1

Tableau 3.4 : Enfant 1 après croisement du type à un point

	2	3	4	5	6
1	1	0	1	1	1
2	0	0	0	1	1
3	1	1	0	0	0
4	0	0	0	0	1
demande	2	2	2	2	2
équilibre	0	-1	-1	0	+1

Tableau 3.5 : Enfant 2 après croisement du type à un point

Nous remarquons que le tableau 3.4 représente une solution "équilibrée" dans le sens que la répartition du manque ou du surplus est uniforme. La solution du tableau 3.5 n'est par contre pas équilibrée. A cet effet, nous appliquons l'algorithme de réparation du tableau 3.1. Nous avons recours à cet algorithme, si nécessaire, après chaque application des opérateurs génétiques de croisement et de mutation.

L'application d'un opérateur du type "uniforme" nécessite la disposition d'une chaîne aléatoire de bits pour engendrer les enfants. Nous supposons que la chaîne de bits (0 1 0 1) est générée. Cette chaîne est de longueur égale au nombre d'employés pour lesquels nous élaborons un horaire. Par conséquent, chaque bit de cette chaîne est associé à une ligne d'une solution. Lorsque le bit est égal à 0 (resp. égal à 1), le premier enfant hérite du premier père (resp. du deuxième père).

	2	3	4	5	6
1	1	1	0	1	1
2	0	0	0	1	1
3	1	1	0	0	0
4	0	0	1	0	0
demande	2	2	2	2	2
équilibre	0	0	-1	0	0

Tableau 3.6 : Enfant 1 après croisement du type uniforme

	2	3	4	5	6
1	1	0	1	1	1
2	0	0	1	1	0
3	1	1	0	0	0
4	0	0	0	0	1
demande	2	2	2	2	2
équilibre	0	-1	0	0	0

Tableau 3.7 : Enfant 2 après croisement à du type uniforme

Les tableaux 3.6 et 3.7 montrent le résultat de l'application d'un opérateur de croisement du type "uniforme" aux deux parents présentés dans les tableaux 3.2 et 3.3.

b. Opérateurs de Mutation

Nous avons défini deux opérateurs de mutation:

- Un opérateur de **mutation simple** qui consiste à échanger, lorsque c'est possible, un jour de travail et de congé d'un employé choisi aléatoirement dans une même semaine. Cet échange peut éventuellement perturber l'équilibre de la solution dans cette semaine. Dans ce cas, on applique l'algorithme de réparation du tableau 3.1.

- Un opérateur de **mutation spéciale** qui consiste à appliquer la deuxième procédure de génération d'une solution initiale proposée dans la section 3.2.2, pour déterminer l'horaire de p employés pendant k journées dans une même semaine. Ces p employés et k journées retenus sont aléatoirement choisis. Notons que cet opérateur est similaire à celui proposé pour le problème de transport dans [18].

Les tableaux 3.8, 3.9, 3.10 et 3.11 illustrent une application de l'opérateur de "mutation spéciale" . Le tableau 3.8, qui représente un enfant candidat à la mutation, correspond à l'horaire de 4 employés pendant la deuxième semaine. Nous supposons aussi que les colonnes (resp. les lignes) 9, 11 et 13 (resp. 2, 3 et 4) sont aléatoirement choisies. La sous-matrice résultante de ce choix correspond à un problème de planification d'horaire de trois employés pendant trois journées. Nous remarquons que l'application d'un opérateur de "mutation spéciale" conserve l'état de l'"équilibre" d'une solution.

	9	**10**	**11**	**12**	**13**
1	1	1	0	1	1
2	0	0	1	1	0
3	1	1	0	0	1
4	0	0	1	0	0
demande	**2**	**2**	**2**	**2**	**2**
équilibre	**0**	**0**	**0**	**0**	**0**

Tableau 3.8 : Enfant avant mutation "spéciale"

	9	**11**	**13**	**offre**
2	0	1	0	**1**
3	1	0	1	**2**
4	0	1	0	**1**
demande	**1**	**2**	**1**	**3**

Tableau 3.9 : Extraction d'une sous-matrice

	9	**11**	**13**	**offre**
2	1	0	0	**1**
3	0	1	1	**2**
4	0	1	0	**1**
demande	**1**	**2**	**1**	**3**

Tableau 3.10 : Réinitialisation et insertion de la sous-matrice

La colonne "offre" correspond au nombre de jours obligatoires de travail des p employés pendant les k journées.

	9	**10**	**11**	**12**	**13**
1	1	1	0	1	1
2	1	0	0	1	0
3	0	1	1	0	1
4	0	0	1	0	0
demande	**2**	**2**	**2**	**2**	**2**
équilibre	**0**	**0**	**0**	**0**	**0**

Tableau 3.11 : Enfant après mutation "spéciale"

Les opérateurs de croisement et de mutation sont appliqués respectivement avec une probabilité p_c et p_m. Les valeurs de ces probabilités sont ajustées dynamiquement

à chaque génération k selon les relations (3.4) et (3.5) (voir [16]).

$$p_c(k) = p_c(0) \times \exp(-\frac{k}{\alpha}) \tag{3.4}$$

$$p_m(k) = p_m(0) \times \exp(-\frac{k}{\beta}) \tag{3.5}$$

avec

$$\alpha = \frac{\text{Max\_Gener}}{\log \dfrac{p_c(0)}{p_c(\text{Max\_Gener})}} \quad \text{et} \quad \beta = \frac{\text{Max\_Gener}}{\log \dfrac{p_m(\text{Max\_Gener})}{p_m(0)}} \, .$$

où Max_Gener désigne le nombre maximal de générations à effectuer qui sert comme critère d'arrêt pour l'algorithme. $p_c(0), p_c(\text{Max\_Gener}), p_m(0)$ et $p_m(\text{Max\_Gener})$ désignent respectivement les probabilités initiales et finales de croisement et de mutation. Notons que ces probabilités, qui vérifient $p_c(0) > p_m(0)$ et $p_c(\text{Max\_Gener}) < p_m(\text{Max\_Gener})$, sont fixées au début de l'algorithme à des valeurs résultantes de plusieurs tests. Cet ajustement des probabilités de croisement et de mutation permet d'appliquer le croisement avec une forte probabilité au début de l'évolution en la diminuant progressivement au cours de l'évolution. L'ajustement de la probabilité de mutation s'effectue dans le sens inverse.

4 Résultats Numériques

Les données réelles qui ont servi dans nos tests sont issues de l'unité des soins intensifs et l'unité des urgences de l'hôpital Hôtel-Dieu de Montréal. On s'intéresse à l'élaboration des horaires des employés de chaque relève dans les deux unités pour la période allant du 25 Juin 1995 au 16 Septembre 1995. Compte tenu du fait que la période d'horaire est de deux semaines, nous identifions six catégories de problèmes tests, correspondant à la période précitée que nous notons par $C_i (1 \leq i \leq 6)$. Chaque catégorie représente une relève (Jour, Soir ou Nuit) d'une des deux unités pour les six périodes d'horaires. Il en découle 36 problèmes d'horaires à élaborer. En vue de faciliter l'interprétation des résultats, nous classons les catégories C_i dans un ordre croissant selon la dimension des problèmes, donnée par:

$$\text{dimension}(P) = (\#\text{ contraintes} \times \#\text{ employés}), \text{ pour tout problème } P \text{ donné.} \tag{4.1}$$

Le tableau 4.1 regroupe les informations concernant les dimensions moyennes des six catégories. Ces moyennes sont calculées sur les six problèmes de chaque catégorie.

(Relève, Unité)	Catégories	# employés	# contraintes	Dimension
(Nuit, Urgences)	C_1	15.00	115.17	1727.50
(Nuit, Soins Intensifs)	C_2	17.00	174.33	2963.67
(Soir, Soins Intensifs)	C_3	20.67	169.83	3503.83
(Jour, Soins Intensifs)	C_4	22.00	159.33	3505.33
(Soir, Urgences)	C_5	19.33	191.83	3712.83
(Jour, Urgences)	C_6	26.00	251.00	6526.00

Tableau 4.1 : Caractéristiques moyennes des problèmes tests

Les objectifs $O_i (1 \leq i \leq 7)$ considérés dans nos tests sont définis comme suit:

O_1 : le manque ou le surplus en personnel doit être réparti uniformément sur chaque semaine.

O_2 : le nombre de jours consécutifs de travail ne doit pas excéder un nombre fixé $\text{succ}_{\max}$.

O_3 : le nombre de jours consécutifs de travail doit être au moins égal à 2.

O_4 : la demande journalière en personnel du même groupe de substitution doit être satisfaite.

O_5 : les jours de vacances doivent être pris consécutivement et accolés à une fin de semaine.

O_6 : les demandes spéciales concernant les congés hebdomadaires et/ou les journées de travail doivent être satisfaites.

O_7 : la demande journalière en personnel pendant les lundis et vendredis doit être satisfaite.

L'ordre de priorité accordé à ces objectifs, retenu pour faire les tests, est le suivant: $O_1 > O_6 > O_7 > O_4 > O_2 > O_3 > O_5$. Il est important de noter, qu'excepté l'objectif O_1 qui devrait rester en première priorité, tous les autres objectifs peuvent voir leur ordre de priorité modifié. Ceci engendrera d'autres catégories de problèmes.

4.1 Méthodologie pour les Tests

La performance de l'adaptation de la méthode tabou et des algorithmes génétiques a été évaluée selon certains critères liés à la qualité de la solution [5]. Nous en présentons quelques uns:

CPUT : temps d'exécution total (en secondes) requis par les deux méthodes.

CPUT/Alt : temps d'exécution par solution alternative (Alt est le nombre de solutions alternatives).

Vmoy : somme pondérée de l'écart entre la valeur de l'objectif et le but à atteindre donnée par:

$$\text{Vmoy} = \frac{\sum_{i=1}^{p} \lambda_i \, (f_i - f_i^*)}{\sum_{i=1}^{p} \lambda_i} \tag{4.2}$$

p = nombre d'objectifs retenus ($p = 7$).

λ_i = poids associé à l'objectif ayant la priorité i. Il est égal à $(p - i + 1)$.

f_i = valeur de l'objectif ayant la priorité i.

f_i^* = valeur idéale de l'objectif de priorité i, obtenue par minimisation de cet objectif sous les contraintes rigides.

%A : pourcentage d'amélioration de la solution initiale donné par:

$$\%\text{A} = \frac{\text{Vmoy(Sol.Initiale)} - \text{Vmoy(Sol.Finale)}}{\text{Vmoy(Sol.Initiale)}} \tag{4.3}$$

Note : L'ensemble du programme mis au point a été écrit en Turbo Pascal (version 6.0) et les tests ont été effectués sur un compatible PC (processeur 486SX de fréquence 33 Mhz).

Dans la suite, nous désignons par solution de "bonne" qualité toute solution ayant la plus faible violation moyenne (min Vmoy).

4.2 Résultats Numériques de l'Adaptation de Tabou

Rappelons que l'efficacité de la méthode de recherche tabou est fortement liée au choix des valeurs de:

$|T|$: la taille de la liste tabou;
Vmax : la taille maximale du voisinage à générer. On génère au plus les Vmax premiers voisins ou Vmax voisins immédiats;
IterMax : le nombre maximal d'itérations sans amélioration de la meilleure solution.

Les tests effectués nous ont conduits aux conclusions suivantes:

- La meilleure combinaison ($|T|$, Vmax, IterMax) à choisir est celle qui minimise la violation moyenne Vmoy des objectifs, donnée par (4.2).

- En augmentant la valeur de $|T|$, la qualité des solutions générées s'améliore au sens de Vmoy. Nous remarquons aussi que la taille de la liste croît en fonction de la dimension des problèmes. En effet, sur 83% des problèmes testés et d'après la relation (4.1), nous constatons que lorsque T vérifie la relation:

$$|T| = \tfrac{1}{4}\sqrt{\text{dimension}(P)} \text{ , pour un problème } P \text{ donné,} \tag{4.4}$$

la solution est de "bonne" qualité.

- Les valeurs des paramètres Vmax et IterMax augmentent en fonction de la dimension moyenne des problèmes pour 83% des problèmes testés.

Les valeurs moyennes retenues pour ces trois paramètres sont indiquées dans la colonne "combinaison" du tableau 4.2. Nous appliquons la méthode de recherche tabou avec ces combinaisons moyennes sur tous les problèmes d'une même catégorie. Dans ce même tableau, nous rapportons dans les colonnes "sol. Initiale" et "sol. Finale" les valeurs moyennes du critère de qualité Vmoy de chaque solution initiale et finale. Les colonnes suivantes, regroupent les valeurs moyennes des résultats des tests pour les 6 problèmes de chaque catégorie relatives aux autres critères de qualité décrits précédemment.

Catégories	Combinaison			Sol. Initiale	Sol. Finale	%A	Alt	CPUT	CPUT/Alt
	$\|T\|$	Vmax	IterMax	Vmoy	Vmoy				
C_1	10	58	50	2.13	0.05	98	16.17	5.60	0.49
C_2	14	75	133	4.31	0.49	89	1.33	20.02	18.47
C_3	15	83	75	3.82	0.18	95	7.67	13.30	2.50
C_4	15	58	100	3.11	0.45	85	3.33	10.39	7.13
C_5	15	92	100	4.00	0.96	76	10.17	31.27	15.02
C_6	20	92	150	6.64	0.55	92	18.50	57.67	6.96

Tableau 4.2 : Résultats numériques de l'adaptation de tabou

Il ressort des résultats obtenus, pour toutes les catégories, nous réussissons à obtenir plusieurs solutions alternatives de très bonne qualité en terme Vmoy et %A en un temps raisonnable. Ces solutions alternatives, qui sont de qualité identique, sont présentées au planificateur pour choisir celle qui lui convient le mieux. Ce dernier peut arrêter l'algorithme dés que les premières solutions alternatives sont satisfaisantes. Ces solutions sont obtenues en un temps comparable à celui donné dans la dernière colonne du tableau 4.2.

En ce qui concerne le temps d'exécution CPUT, nous remarquons qu'il est plus important pour les catégories C_5 et C_6 que pour les autres. Ceci est essentiellement dû, d'une part, à la dimension relativement grande de ces deux catégories ainsi qu'à la moins bonne qualité de la solution initiale. Par ailleurs, l'augmentation du temps de calcul est liée aussi au nombre relativement grand de solutions alternatives différentes que peut générer la méthode tabou (le temps de calcul non négligeable consacré à la comparaison de la nouvelle solution alternative générée avec ses prédécesseurs).

En pratique, ce temps est à comparer avec celui requis par le planificateur pour établir manuellement un seul horaire. Rappelons que ce temps se mesure en nombre de jours plutôt qu'en secondes.

4.3 Résultats Numériques de l'Adaptation des Algorithmes Génétiques

Les résultats numériques que nous présentons ici concernent des tests effectués sur les six problèmes de chaque catégorie $C_i(1 \leq i \leq 6)$. Compte tenu du fait que les Ags génèrent des populations de solutions à la fin de l'exécution, nous retenons dans cette population la "meilleure" solution selon l'ordre lexicographique des priorités relatives aux objectifs et ce, afin de la comparer avec celle obtenue par la méthode tabou (pour plus de détails, se référer à [1]).

L'efficacité des algorithmes génétiques dépend du choix de la taille de la population m et du nombre maximal de générations Max_Gener.

Pour le même choix de paramètres, la qualité de la solution varie d'une exécution à une autre. Ainsi, nous rapportons dans le tableau 4.3 les résultats moyens obtenus de cinq exécutions de l'algorithme pour chaque problème $P_i(1 \leq i \leq 6)$ de la catégorie $C_j(1 \leq j \leq 6)$.

Les probabilités initiales et finales de croisement et de mutation ont été fixées comme suit :
$p_c(0) = p_m(\text{Max\_Gener}) = 0.6$ et $p_m(0) = p_c(\text{Max\_Gener}) = 0.4$
suite à de nombreux tests se basant sur la qualité des solutions.

La population issue de chaque génération est égale à la taille m de la population initiale. Les tests effectués conduisent aux résultats suivants:

- la qualité de la solution s'améliore lorsque l'on augmente la taille de la population. Néanmoins, le temps d'exécution requis augmente en conséquence.

- Le croisement du type "uniforme" donne généralement des résultats de bonne qualité comparativement aux opérateurs de croisement "à un point" et "à deux points".

- La mutation simple est meilleure que la mutation spéciale en terme de qualité de solutions.

	$m = 10$				$m = 20$			
Catégories	Meill. Sol. Initiale Vmoy	Meill. Sol. Finale Vmoy	%A	CPUT	Meill. Sol. Initiale Vmoy	Meill. Sol. Finale Vmoy	%A	CPUT
C_1	1.53	0.07	96	49.81	1.50	0.03	98	85.64
C_2	3.34	0.39	88	71.71	3.26	0.24	93	126.65
C_3	3.31	0.32	90	87.26	3.21	0.19	94	154.06
C_4	3.04	0.27	92	85.60	2.78	0.15	95	150.10
C_5	3.17	0.34	90	98.80	3.05	0.17	95	179.79
C_6	5.48	0.93	83	139.25	5.24	0.46	91	254.13

Tableau 4.3 : Résultats numériques de l'adaptation des Ags (Max_Gener = 100) .

Nous constatons par ailleurs, que la qualité des solutions obtenues par les Ags est meilleure que celle obtenue par tabou. Ceci peut s'expliquer, d'une part, par l'effet diversificateur des opérateurs génétiques et d'autre part, par le fait que la qualité des solutions initiales de l'Ag est meilleure que celle générée pour la méthode tabou. Néanmoins, les temps de calculs requis sont clairement moins bons que ceux de la méthode tabou.

5 Conclusion et Perspectives

La réaction positive de certaines infirmières (chargées de planifier l'horaire) de l'hôpital Hôtel-Dieu de Montréal face aux horaires proposés de manière interactive est encourageante. Une avenue intéressante que nous explorons actuellement est un algorithme génétique hybride où on utilise l'algorithme de la méthode de recherche tabou comme opérateur de mutation. Le but visé étant de générer des solutions d'aussi bonne qualité dans les Ags en un temps suffisamment raisonnable. Cette variante est en cours d'expérimentation. Une autre voie à explorer consiste à rendre notre système flexible à tout autre contexte hospitalier.

References

[1] B. Ahiod, *Adaptation de certaines heuristiques pour résoudre le problème d'horaire du personnel infirmier dans un établissement hospitalier*. Thèse de 3ème cycle, Université Mohammed V, Rabat, (1997).

[2] D. Beasley, D.R. Bull et R.R. Martin, "An Overview of Genetic Algorithms: Part 1, fundamentals". *University Computing*, **15**(2), p. 58-69, (1993).

[3] D. Beasley, D.R. Bull et R.R. Martin, "An Overview of Genetic Algorithms: Part 2, Research Topics". *University Computing*, **15**(4), p. 170-181, (1993).

[4] I. Berrada, J.A. Ferland et P. Michelon, "A Multi-objective Approach for Nurse Scheduling with both Hard and Soft Constraints". *Socio-Econ. Plann. Sci.*, **30**, p. 183-193, (1996).

[5] I. Berrada, J.A. Ferland et P. Michelon, "Multi-objective Models and Techniques for Nurse Scheduling Problem", *Publication No. 907*, Université de Montréal, Montréal, (1994).

[6] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, (1991).

[7] C. Fleurent et J.A. Ferland, "Genetic and Hybrid Algorithms for Graph Coloring". *Annals of Operations Research*, **63**, p. 437-461, (1996).

[8] C. Fleurent et J.A. Ferland, "Genetic Hybrids for the Quadratic Assignment Problem". In: *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, (P.M. Pardalos et H. Wolkowicz eds.)*, **16**, p.173-188, (1994).

[9] C. Fleurent et J.A. Ferland, "Object-Oriented Implementation of Heuristic Search Methods for Graph Coloring, Maximum Clique, and Satisfiability". *présenté au deuxième DIMACS Implementation Challenge*, (1994).

[10] F. Glover, "Futur Paths for Integer Programming and Links to Artificial Intelligence". *Computers and Operations Research*, **13**, p. 533-549, (1986).

[11] F. Glover, "Tabu Search-Part I". *ORSA Journal on Computing*, **1**, p. 190-206, (1989).

[12] F. Glover, "Tabu Search-Part II". *ORSA Journal on Computing*, **2**, p. 4-32, (1990).

[13] F. Glover et M. Laguna, *Tabu Search: Modern Heuristic Techniques for Combinatorial Problems*. C.R. Reeves éditeur), Blackwell Publishing, p. 70-150, (1992).

[14] P. Hansen, "The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming". *présenté au congrès sur les méthodes numériques en optimisation combinatoire*, Capri, Italie, (1986).

[15] J.H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, University of Michigan Press, (1975).

[16] E. Lutton et P. Martinez, *A Genetic Algorithm for the Detection of 2D Geometric Primitives in Images*. Rapport de recherche No. 2110, INRIA, (1993).

[17] Z. Michalewics, *Genetic Algorithms + Data Structures = Evolution Programs*. Springler-Verlag, Berlin, (1992).

[18] Z. Michalewics, G.A. Vignaux et M. Hobbs, "A Non-standard Genetic Algorithm for the Nonlinear Transportation Problem". *ORSA Journal on Computing*, **3**, p. 307-316, (1991).