République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 8 Mai 1945 Guelma

Faculté des mathématiques et de l'informatique et des sciences de la matière
Département d'informatique
Laboratoire des sciences et technologies de l'information et de la communication

THÈSE

EN VUE DE L'OBTENTION DU DIPLOME DE DOCTORAT EN 3ème CYCLE

Domaine : Mathématiques et informatique. Filière : Informatique
Spécialité : Informatique

Présentée par

GUESSOUM Fatima

Intitulée

Planning de rotation des infirmières

Soutenue le : 12/10/2020 Devant le Jury composé de :

Nom et Prénom	Grade		
Pr. SERIDI Hamid	**Prof**	Univ. de 8 Mai 1945, Guelma	Président
Pr. HADDADI Salim	**Prof**	Univ. de 8 Mai 1945, Guelma	Directeur de thèse
Pr. BOUHADADA Tahar	**Prof**	Univ. de Badji Mokhtar, Annaba	Examinateur
Pr. FARAH Nadir	**Prof**	Univ. de Badji Mokhtar, Annaba	Examinateur

Année Universitaire 2019/2020.

Dédicace

Cette thèse de doctorat est particulièrement dédiée :

À mes chers parents.

À mes chers frères et à ma chère sœur.

À mon mari et à ma belle-famille.

À mon neveu et mes nièces.

À mes proches, mes amies et mes collègues.

À tous ceux qui m'aiment.

À tous ceux que j'aime.

À tous ceux qui m'ont aidé de près ou de loin.

Remerciements

Tout d'abord, je remercie Dieu tout-puissant de m'avoir donné la force, le courage, la volonté, et la patience pour réaliser ce travail de thèse.

Je tiens à remercier, mon directeur de thèse, Monsieur, ***Salim HADDADI*** professeure à l'université de Guelma pour ses conseils constants, ses encouragements, son aide et surtout son expertise, ainsi que son soutien tout au long de cette recherche.

Je tiens à remercier tout particulièrement, Monsieur, ***Hamid SERIDI***, Professeur à l'Université de Guelma, pour m'avoir accueillie au sein du laboratoire (LabSTIC) et permis de réaliser cette thèse dans les meilleures conditions, aussi pour l'honneur qu'il m'a fait, en acceptant la présidence de ce jury, et pour l'intérêt qu'il porte à ce travail.

Mes remerciements vont aussi aux membres du jury : Monsieur, ***Tahar BOUHADADA***, Professeur à l'université de Badji Mokhtar, Annaba, et Monsieur, ***Nadir FARAH***, Professeur à l'université de Badji Mokhtar, Annaba, d'avoir bien voulu accepter de lire et de rapporter cette thèse et de faire partie de ce jury en tant qu'examinateurs.

Je désire exprimer ma gratitude à mes chers parents, pour leurs permanents encouragements, leurs immenses soutiens, leurs grandes amours, leurs sacrifices et leurs prières.

Je remercie mes chers frères et ma chère sœur pour leurs soutiens, et Particulièrement, je remercie mon mari, qui m'a soutenu et encouragé tout au long de la réalisation de cette thèse, même dans les moments les plus difficiles.

Enfin, je tiens à remercier toutes mes amies et mes collègues, pour leurs sympathies et leurs amitiés en leur souhaitant une bonne continuation et une bonne chance.

Je tiens aussi à remercier tous les membres du laboratoire LabSTIC permanents ou thésards, avec qui j'ai passé des bons moments.

ملخص

تعتبر الجداول الزمنية لمناوبات الممرضات نوع خاص من جداول الزمن تهدف لإنشاء جدول زمني يلبي متطلبات المؤسسة بالإضافة إلى مطالب الموظفين مع احترام مجموعة من القيود المشروطة مسبقا.

تختلف الجداول الزمنية لمناوبات الممرضات اختلافًا كبيرًا من مؤسسة إلى أخرى نظرًا للكم الواسع من القيود الخاصة (متطلبات الممرضات ، سياسة المستشفى ، التنظيم العام ، إلخ).

من المعروف أن الجداول الزمنية لمناوبات الممرضات هو مشكلة من مشاكل التحسين التوافقي الصعبة من الناحية النظرية والعملية ، ومن ثم فإن إنشاء جدول زمني مثالي يعتبر مهمة شاقة ومرهقة للغاية وتستلزم وقتا طويلا للقيام بها ولهذا نختار بدلا من ذلك خوارزميات سريعة بما فيه الكفاية والتي تنتج حلول تقريبية في وقت معقول. بشكل عام ، تتجه جميع الأبحاث في هذا المجال نحو محاولة التوفيق بين سرعة التنفيذ وجودة الحل المقدم.

الهدف الرئيسي من هذه الأطروحة هو اقتراح طرق جديدة لحل المشكلة المدروسة تكون قادرة على إنتاج حلول ذات جودة عالية، وضمان الطابع العام للطرق المقترحة بحيث تكون قابلة للتطبيق على مشكلات أخرى للتحسين التوافقي.

وتشمل ثلاث مساهمات:

- الأولى خوارزمية هجينة جديدة تدمج دراسة بحثية واسعة النطاق في مجال الجوار في إطار تحسين التدرج الفرعي.
- الثانية عبارة عن أدلة عليا مؤلفة من مرحلتين: المرحلة الأولى هي مساعدة على الكشف ذات طابع عام لتثبيت المتغيرات تهدف لتخفيض حجم المشكلة .المرحلة الثانية تقوم بتطبيق معالج لبرمجة الأعداد الصحيحة المختلطة لحل المشكلة المخفضة .
- الثالثة تتمثل في تطبيق خوارزمية الانتشار النباتي على مشكلة مخفضة و استغلال مجموعة من الحلول النخبوية لتشكيل مشكلة جديدة صغيرة الحجم يتم حلها بتطبيق معالج لبرمجة الأعداد الصحيحة المختلطة في وقت قياسي.

تم اختبار الطرق الثلاث المقترحة على أمثلة من قاعدة بيانات خاصة و النتائج المحصل عليها كانت تنافسية ومثيرة للاهتمام مقارنة مع الطرق المقترحة سابقا. المساهمة الأولى و الثانية تم نشرهما في مجلة علمية محكمة.

الكلمات المفتاحية: الجدول الزمني لمناوبات الممرضات، تحسين التدرج الفرعي، البحث في الجوار الواسع النطاق، البحث الموجه، تثبيت المتغيرات، الاستدلال الجشع، البرنامج الثنائي، الأدلة العليا، خوارزمية انتشار النبات.

Résumé

Le problème de l'élaboration d'un planning pour la rotation des infirmières (NRP) est un problème particulier d'emploi du temps dont l'objectif est la production d'un planning qui satisfait les exigences de l'établissement ainsi que les demandes du personnel requis tout en respectant un certain nombre de contraintes.
Les NRP varient considérablement d'un établissement à l'autre, en raison du large éventail de contraintes spécifiques (les demandes des infirmières, politique hospitalière, réglementation, etc.). Il est bien connu que le NRP est un problème d'optimisation combinatoire difficile du point de vue théorique et pratique, d'où la mise en place d'un planning optimal est considérée comme une très lourde et fastidieuse tâche et nécessite un temps excessivement long, pour cela on opte généralement des algorithmes suffisamment rapides qui produisent des solutions approchées dans des délais raisonnables. Tout l'art réside dans ce compromis entre rapidité d'exécution et qualité de la solution fournie.

L'objectif principal de cette thèse est la proposition de nouvelles approches pour résoudre le problème étudié capable de produire des résultats de bonne qualité, et d'assurer que les méthodes proposées sont génériques et applicables à d'autres problèmes d'optimisation.

Il comprend trois contributions:

- La première contribution présente un nouvel algorithme hybride qui intègre une méta-heuristique de recherche du voisinage à très grande échelle (VLSN) dans un cadre d'optimisation de sous gradient.

- La deuxième contribution présente une méthode à deux phases pour résoudre le NRP.La première phase est une heuristique générique de fixation de variable (VFH) pour réduire la taille du problème. La deuxième phase utilise un solveur MIP (Cplex) pour résoudre le NRP réduit (RNRP).

- La troisième contribution présente une méthode basée sur l'algorithme de propagation des plantes, le PPA est appliqué sur des données introduite par l'heuristique de fixation des variable présentée ci-dessus, ensuite, des solutions élites générées par le PPA sont utilisées pour réduire la taille du problème qui sera ensuite résolu facilement et très rapidement pas un solveur MIP (Cplex).

Les trois approches proposées sont testées sur des instances de la base de données NSPLib et fournissent des résultats intéressants. La comparaison montre que nos approches s'avèrent compétitives et concurrentes avec l'état de l'art.

Les deux premières approches ont fait l'objet de deux publications dans la revue (American Journal of Mathematical and Management Sciences).

Mots-clés: planning de rotation des infirmières, optimisation de sous-gradient, recherche du voisinage à très grande échelle, programme binaire, recherche guidée, fixation des variables, méta-heuristique, heuristique gloutonne, algorithme de propagation de plantes.

Abstract

The nurse rostering problem (NRP) is a particular timetabling problem whose aim is the production of a roster for a given set of nurses subjected to a predefined set of requirements. NRPs vary considerably from an institution to another, due to the wide range of specific constraints (nurses' requests, hospital policy, regulations, etc…).
It is well known that the NRP is a difficult combinatorial optimization problem from both theoretical and practical points of view. Hence, the implementation of an optimal schedule is considered as a very difficult task that requires an excessively long time. Generally, we look for sufficiently fast algorithms producing an approximate solution within reasonable running time. All the art is in this compromise between speed of execution and quality of the solution provided.

The main and important objectives of this thesis are the proposal of new approaches able to solve the studied problem that produces results of good quality and to ensure that the proposed methods are generic and can be applied for other optimization problems.

It includes three contributions:

- The first contribution presents a new hybrid algorithm that integrates a very large scale neighborhood (VLSN) search metaheuristic within a subgradient optimization framework.

- The second contribution proposes a two-phase method for solving the NRP. The first phase utilizes a subgradient method applied to the Lagrangian relaxation of the coverage constraints. From the information provided, a variable-fixing heuristic reduces the problem size. The second phase applies a MIP solver (Cplex) to solve the reduced NRP.

- The third contribution describes a new method based on the plant propagation algorithm, the PPA is applied to the reduced instances introduced by the variable fixing heuristic presented above, then elite solutions generated by the PPA are used to reduce the size of the problem which can be solved easily and very quickly

All the proposed approaches are tested on benchmark instances from NSPLib dataset and provide interesting results. The comparison shows that our approaches are competitive and compete favorably with the state of the art.

Both the first and the second contributions were published in the scientific journal (American Journal of Mathematical and Management Sciences).

Keywords: Nurse rostering, subgradient optimization, very large scale neighborhood search, guided search, Variable-fixing, binary program, greedy heuristic, metaheuristic, Plant propagation algorithm.

Table des matières

Table des figures

Liste des tableaux

Acronymes

NRP	**N**urse **R**ostering **P**roblem
VLSN	**V**ery **L**arge **S**cale **N**eighborhood
LS	**L**ocal **S**earch
VFH	**V**ariable **F**ixing **H**euristic
PPA	**P**lant **P**ropagation **A**lgorithem
RL	**R**elaxation **L**agrangienne
PR	**P**roblème **R**elaxé
DL	**D**ual **L**agrangien
PO	**P**roblème d'**O**ptimisation
PL	**P**rogrammation **L**inière
PB	**P**rogramme **B**inaire

Introduction

La recherche opérationnelle (RO) est une discipline qui s'occupe de l'application des méthodes analytiques de résolution de problèmes et de prise de décision. Les projets de recherche sont souvent pluridisciplinaires (sciences de la décision, ingénierie, économie, médecine, etc...).

Comme tout progrès scientifique, la RO a été utilisée d'abord à des fins militaires, et grâce au succès obtenu, elle a été par ailleurs adoptée dans d'autres domaines d'application tels que l'économie, l'industrie, le commerce, physique, l'urbanisme, les finances, probabilité et statistique, le service social, le service sanitaire, etc..., actuellement, en l'utilise dans tous les domaines pour optimiser les ressources disponibles et d'améliorer l'efficacité pour obtenir des bénéfices, il s'agit souvent de déterminer un maximum (tel que le profit, la performance ou le rendement) ou le minimum (comme la perte, le risque ou le coût).

L'optimisation combinatoire est une voie d'études occupant une place très importante en recherche opérationnelle et en informatique, c'est le domaine des mathématiques discrètes qui traite de la résolution du problème dont l'objectif est de chercher la meilleure solution parmi l'ensemble de toutes les solutions possibles.

La gestion des emplois du temps (Timetabling) est l'allocation des ressources aux objets au long d'un horizon temporel, de telle façon à satisfaire un ensemble de contraintes et atteindre des objectifs prédéfinis. Ces problèmes font partie des problèmes d'optimisation combinatoire pour lesquels, dans la majorité des cas, il est très difficile de trouver la solution optimale.

Il est très important d'avoir un bon planning pour les sociétés, hôpitaux ou universités quoiqu'elle soit une tâche prenante et qui exige souvent un effort considérable pour

satisfaire les exigences et les besoins, car il ne peut en aucun cas sous-estimer l'effort nécessaire pour leur trouver une solution.

Un problème d'emplois du temps est composé habituellement de deux types de contraintes : dures et molles. Les contraintes dures doivent être satisfaites pour avoir une solution faisable (réalisable) pour le problème, par contre les contraintes molles peuvent être violées, mais la satisfaction de ces contraintes est souhaitable et habituellement utilisée pour évaluer la qualité de la solution où chaque violation implique une pénalité sur la solution qui est ajoutée au coût.

Au cours des dernières décennies, les problèmes d'emplois du temps du personnel ont été largement étudiés. L'augmentation d'intérêt pourrait être motivée par des considérations économiques. Dû au nombre croissant d'institutions et du personnel à travers le monde, l'établissement d'un planning du personnel est devenu plus difficile que jamais. Pour beaucoup d'entreprises, le coût de la main d'œuvre est la principale composante des coûts directs. La réduction d'un pourcentage de ce coût mettant en œuvre un nouveau calendrier de personnel pouvait être bénéfique.

Planifier les horaires du personnel, implique d'assigner un ensemble de personnes, considérées comme des ressources, à un ensemble de tâches ou de services durant un horizon de planification, tenant en compte plusieurs contraintes, telles que les contraintes de temps, de qualification, les contraintes contractuelles, etc. . . .

L'emploi du temps du personnel des secteurs de santé est parmi les domaines les plus étudiés et constitue un fameux défi.

Dans cette thèse, on considère le problème de l'élaboration d'un planning pour la rotation des infirmières, qui est un problème particulier de construction d'emploi du temps dans les milieux hospitaliers, il consiste à concevoir un planning qui satisfait les exigences de l'établissement ainsi que les demandes du personnel requis tout en respectant un certain nombre de contraintes.

La thèse est organisée ainsi de la façon suivante.

le chapitre 1, décrit le contexte scientifique de ce mémoire, Nous commençons par une introduction à l'optimisation combinatoire, puis nous présentons quelques notions de base de la théorie de la complexité.

Le chapitre 2, présente le problème de planification de rotation des infirmières. On étudie sa complexité et on passe en revue quelques modèles du NRP, et on propose un état de l'art.

Le chapitre 3 est un résumé de la panoplie des méthodes utilisées : la relaxation lagrangienne et la méthode de sous-gradient, les notions de voisinage et d'amélioration locale, l'algorithme de propagation des plantes.

Le chapitre 4 présente une méta-heuristique consistant en une hybridation de la recherche du voisinage à très grande échelle (VLSN) avec l'optimisation de sous-gradient pour le problème de NRP. Celle-ci est fondée sur deux étapes clés. tout d'abord, un planning initial réalisable est présenté à l'aide d'un solveur commercial (Cplex 12.6). Ensuite, le planning construit est soumis à une procédure d'amélioration qui consiste à la recherche VLSN guidée par une méthode de sous gradient appliqué à la relaxation lagrangienne des contraintes de couverture. L'heuristique est codée en C, et est testée sur une grande masse de données benchmark obtenues auprès d'une bibliothèque bien connue, NSPlib. Les résultats obtenus sont comparés à d'autres méthodes trouvées dans l'état de l'art. Au vu de la comparaison, l'heuristique proposée se montre compétitive avec ces dernières, et très supérieure aussi bien du point de vue de la qualité des solutions obtenues que du point de vue de la vitesse d'exécution. La méthode a fait l'objet d'une publication dans la revue « American Journal of Mathematical and Management Sciences» publiée par Taylor & Francis[83].

Le chapitre 5 suggère une méthode à deux phases. La première phase est une heuristique générique de fixation de variable (VFH) pour réduire la taille du problème. La deuxième phase résout le problème réduit résultant de la première phase par un solveur MIP à usage général (Cplex). La méthode à deux phases représente l'application séquentielle de l'heuristique VFH et de Cplex.

L'approche proposée testée sur l'ensemble de données NSPLib fournit des résultats intéressants. La comparaison montre que la nouvelle contribution est en concurrence avec quatre méthodes publiées récemment. cette approche aussi a fait l'objet d'une publication dans la revue « American Journal of Mathematical and Management Sciences » [81].

Le chapitre 6 présente une méthode à trois phases, basé sur une méta-heuristique inspirée d'un algorithme de propagation de plante (PPA). l'approche proposée est composée de trois phases principales. La première phase est une heuristique générique de fixation de variable (la même heuristique présentée dans le chapitre 5). La deuxième phase propose un algorithme de propagation des plantes (PPA), cet algorithme est appliqué sur le problème réduit résultant de la première phase. La troisième phase exploite les solutions élites générées de la deuxième phase pour construire un très petit problème qui sera résolu facilement et très rapidement par un solveur MIP (Cplex). La méthode proposée est testée sur l'ensemble des instances NSPLib, et les résultats obtenus justifiés l'efficacité de l'approche proposée. Le contenu de ce chapitre a été soumis pour publication.

Et enfin on termine par une conclusion qui résume notre apport et propose quelques pistes de recherche pour le futur.

Deux choses avant de conclure. Malheureusement, en raison de plusieurs obstacles, on n'a pas pu réaliser notre étude sur des cas réels des hôpitaux de Guelma. Enfin, nous n'oublierons pas de remercier Maenhout et Vanhoucke pour maintenir le site web qui contient le pack de données NSPlib, et en laissant presque tout librement téléchargeable, et qui seront testés aux chapitres 4 et 5 et 6 .

Avertissement Comme certains concepts n'ont pas une expression équivalente en français, on y fait référence en utilisant leur terme d'origine.

Chapitre 1

Optimisation combinatoire et théorie de la complexité

1.1 Introduction

Au cours des dernières décennies, l'optimisation combinatoire a reconnu des progrès très positifs, et devenu un domaine très mature occupant une place très importante en RO et en informatique, c'est le domaine des mathématiques discrètes avec un grand nombre d'applications, où de nombreuses applications pratiques, souvent des problèmes complexes de la vie réelle dans différents secteurs (télécommunications, électronique, mécanique, chimie, transport, ...) peuvent être formulées sous la forme d'un problème d'optimisation. Dans ce premier chapitre, nous présentons le domaine de l'optimisation combinatoire et sa relation avec la théorie de la complexité, dont l'objectif est de situer le contexte scientifique de notre travail de thèse et non plutôt de présenter une synthèse exhaustive sur le domaine.

Ce chapitre est divisé en deux parties principales. Au début, dans la première partie 1.2, nous décrivons de manière concise les concepts de base de cette discipline. En outre, nous décrivons sommairement quelques problèmes classiques de l'optimisation combinatoire, ainsi que les techniques de base et les méthodes les plus représentatives pour les résoudre. La deuxième partie 1.3 est consacrée à la représentation de la théorie de la complexité. D'abord nous présentons une introduction informelle à la théorie de la

complexité, ensuite, nous présentons quelques rudiments nécessaires à la compréhension du domaine, et en terminons par quelques types de problèmes existants et ses propriétés.

1.2 Problème de l'optimisation combinatoire

1.2.1 Définition

l'optimisation combinatoire est le domaine des mathématiques discrètes qui traite de la résolution du problème dont l'objectif est de maximiser (le gain, la performance, etc. : problème de maximisation) ou de minimiser (la perte, le coût, etc. : problème de minimisation) une fonction objective sous certaines contraintes [147], c'est le fait de chercher la meilleure solution (ou l'ensemble des meilleures solutions) parmi l'ensemble de toutes les solutions possibles dont le but est de trouver une solution optimale ou proche de l'optimale dans un temps d'exécution raisonnable. Ce meilleur est estimé par un ou plusieurs critères de qualité formulés dans une fonction objective [111]. Quoique l'ensemble de solutions réalisables soit, généralement, fini mais, potentiellement, de cardinalité très élevée, ce qui justifier la difficulté d'énumération de telle solution qui nécessite des temps d'exécution très longs. Les problèmes d'optimisation se divisent, principalement, en deux catégories : celles avec des variables continues et d'autres avec des variables discrètes, dans le cas des problèmes d'optimisation « combinatoire» on parle des variables discrètes [143]. Le type des variables et la nature des contraintes permettent de déterminer la complexité du problème. Ces problèmes sont généralement difficiles à résoudre et la plupart appartiennent à la classe des problèmes NP-difficiles.

1.2.2 Modélisation des problèmes d'optimisation combinatoire

Une bonne conception des données descriptives d'un problème est très utile pour une modélisation adéquate permettant de résoudre le problème d'une manière efficace, cela nécessite l'étude de ces trois concepts :

- la modélisation du problème par la description de l'espace de recherche et la définition de l'ensemble des solutions réalisables.
- la proposition d'une formulation mathématique par l'expression de l'objectif à optimiser, ainsi que les contraintes à satisfaire.

— le choix d'une méthode d'optimisation a appliqué pour trouver une meilleure solution parmi l'ensemble des solutions possibles.

La modélisation inclut les deux premiers concepts et le dernier représente la résolution [111].

D'une façon générale un problème d'optimisation combinatoire $P(S, f)$ peut être défini comme suite [143] :

$$\begin{aligned} & Opt \quad f(x) \quad telque \\ & g_i(x) \geq 0, \quad i = 1, \dots m \\ & h_j(x) = 0, \quad j = 1, \dots, p \\ & x \in S \subset \mathbb{R}^n \end{aligned}$$

où Opt représente $Minimiser$ ou $Maximiser$. $f(x)$ représente la fonction objective (fonction du coût) à optimiser. x représente le vecteur des variables à optimiser $X = x_1, x_2, ...x_n$. L'ensemble S représente l'espace de recherche (l'espace de variables) où chaque variable x_i est associée à un domaine D_i. g_i et h_i sont les contraintes reliant les variables. Dans un problème d'optimisation on peut trouver des contraintes d'égalité ou d'inégalité ou même les deux. Un éliment x_i est une solution réalisable pour le problème P si $x_i \in S$, et si et seulement si toutes les contraintes sont satisfaites. Une solution (x^*) est dite solution optimale pour le problème P tel que $\forall x \in S \ f(x^*) \leq f(x)$. Les problèmes d'optimisation peuvent être des problèmes mono-objectifs ou des problèmes multi-objectifs, dans les cas multi-objectifs la fonction objective à optimiser est formulée comme un vecteur regroupant plusieurs fonctions objectives [111], les problèmes multi-objectifs sont, souvent, plus difficile à résoudre que les problèmes mono-objectifs.

1.2.3 Les méthodes d'optimisation combinatoire

D'une façon générale, les problèmes d'optimisation combinatoire sont, souvent, simples et faciles à définir mais, pratiquement, très difficiles à résoudre surtout à l'optimalité [143]. Jusqu'à ce jour, il n'existe aucune solution algorithmique efficace qui est valable pour toutes les données [71]. La complexité d'un tel algorithme dépend de la taille de l'espace de recherche et peut s'augmenter d'une façon exponentielle, pour cela, les chercheurs s'orientent vers l'approximation pour trouver des solutions acceptables et de bonne qualité qui sont proches de l'optimale, dans un temps d'exécution raisonnable

au lieu de chercher la solution optimale qui nécessite des temps de calcul exagérément longs. Enfin, les méthodes d'optimisation sont réparties en deux groupes :

— **Méthode exacte :** destinée à introduire des solutions optimales, pour cela, l'espace de recherche est exploré d'une manière exhaustive pour trouver la solution optimale si elle existe, ou de prouver autrement l'inconsistance du problème[47] .

— **Méthode approchée :** si le problème est très compliqué pour avoir une solution exacte, alors on cherche une solution approximative pour gagner du temps [90]. ces méthodes explorent une sous-partie de l'espace de recherche.

Au cours des ans, plusieurs méthodes exactes et approchées ont été proposées, une classification de telles approches est résumée dans la figure 1.1.

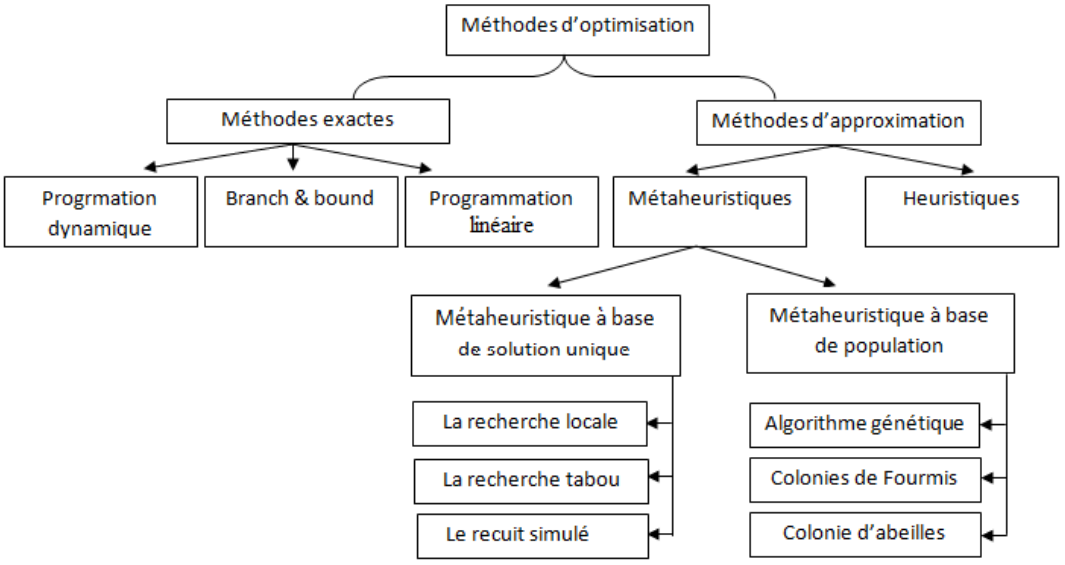

FIGURE 1.1: Classification des méthodes d'optimisation combinatoire.

1.2.4 Quelques problèmes classiques de l'optimisation combinatoire

L'optimisation combinatoire présente une voie d'étude qui tient une importance primordiale dans la recherche opérationnelle, en mathématiques discrètes et en informatique. Malgré le caractère difficile de ce type de problème, un succès très important sur la qualité des solutions fournies est obtenu. Plusieurs applications pratiques des problèmes réels tels que le transport, l'économie, la gestion, l'industrie, l'ingénierie..., peuvent s'exprimer sous la forme d'un problème d'optimisation combinatoire [156]. On peut citer brièvement quelques exemples les plus connus comme suit :

— **Problème d'emballage (bin packing) :** Le problème d'emballage représente le fait de rangement d'un certain nombre d'objets dans un nombre minimum de boîtes. Ce problème existe dans de nombreux domaines industriels et informatiques, tels que le rangement des données numériques sur un support informatique, remplissage de conteneurs, placement des objets physique dans un entrepôt, etc. Ce problème est en forte relation avec le problème du sac à dos (Knapsack) qui représente le fait du remplissage d'un sac à dos avec un ensemble d'objets dont chacun ayant un poids et une valeur, et dont le but est de maximiser la valeur totale des objets sans dépasser la capacité maximum du sac à dos [1, 119, 152].

— **Le problème du voyageur de commerce (Traveling Salesman Problem) :** Le problème du voyageur de commerce est de trouver l'itinéraire pour visiter un certain nombre de villes, tel que chaque ville doit être visitée une et une seule fois, et de revenir au point de départ d'origine, l'objectif est de minimiser la distance parcourue par le voyageur de commerce, le TSP est souvent modélisé par un graphe valué dont les villes sont représentées par les sommets, et la distance séparant deux villes est représentée sous forme d'une arête valuée reliant les deux sommets [29, 126].

— **le problème de couverture par ensembles (Set Covering problem) :** Étant donné un ensemble U de n éléments, et S une collection de sous-ensembles de U, $S = S_1, ..., S_m$, un élément e est couvert par un ensemble S_i si e appartient à S_i, généralement, chaque élément de S est associé à un coût (poids), l'objective est de trouver un sous-ensemble de S couvrant tous les éléments de U avec un coût minimal [84–86].

— **Le problème de l'emploi du temps (Timetabling) :**

Un problème de l'emploi du temps est l'affectation d'un certain nombre de ressources aux objets durant une période de temps, et de satisfaire un ensemble de contraintes pour atteindre des objectifs prédéfinis. Les problèmes d'emploi du temps existent abondamment dans plusieurs domaines d'application, planning des vols aériens [159], planning des programmes télévisés [18], planning des examens [51], planning de rotation des infirmières [81, 83] qui fait l'objet de notre sujet de thèse .

1.3 Théorie de la complexité

Dans cette section nous présentons quelques rudiments de la théorie de la complexité comme décrit par Papadimitriou [145], celle-ci a pour but de savoir s'il existe des algorithmes efficaces pour certains problèmes combinatoires, et de discriminer les problèmes en ceux qui sont faciles et ceux qui sont difficiles ou durs selon qu'ils peuvent être résolus par des algorithmes efficaces. Ici, «efficace» signifie que le temps de fonctionnement d'un algorithme est une fonction de croissance lente dans la taille des données d'entrée.

La notion de complexité des algorithmes est utilisée pour préciser l'idée selon laquelle un algorithme est caractérisé par un coût mesuré essentiellement par rapport à l'utilisation des deux ressources qui sont le temps et l'espace mémoire [172].

Il est important de distinguer la complexité pratique qui est une mesure précise des temps de calcul et des tailles de mémoire pour un modèle de machine bien défini, de la complexité théorique qui donne des ordres de grandeur des coûts, de façon plus indépendante des conditions pratique de l'exécution de l'algorithme.

Lorsqu'on considère un problème, la tâche consiste généralement à trouver soit un algorithme efficace pour le problème, soit, au contraire, une preuve de l'inexistence de tel algorithme. En effet il y a des problèmes pour lesquels on ne peut pas trouver un algorithme efficace pour les résoudre. Lorsqu'on arrive à identifier ou à dire qu'on se trouve confronté à de tels cas de problème, il n'existe aucune méthode connue pour prouver qu'un algorithme efficace ne peut exister [101]. Par conséquent, on est d'habitude satisfaits de la comparaison d'un tel problème difficile avec d'autres problèmes, ce qui permet de dire qu'un problème est «au moins aussi difficile que plusieurs d'autres problèmes déjà supposer difficiles». Un des buts de la théorie de la complexité est de classifier les problèmes selon leurs difficultés de calcul intrinsèques. À cette fin, on regroupe les problèmes en classes de complexité, jusqu'à nos jours, on n'a pas une réponse précise pour la question sur les ressources et les puissances exigées pour résoudre un problème donné, mais on a fait une grande progression dans la classification des problèmes en classes de complexité.

La rédaction de cette section s'inspire substantiellement de la référence [59].

1.3.1 Problèmes indécidables et problèmes intraitables

Le calcul du coût d'un algorithme est inhérent pour l'analyse de sa complexité, celui-là est calculé par la double mesure du temps et d'espace mémoire utilisée. Le résultat d'une analyse de complexité est une estimation de la rapidité avec laquelle le temps de la solution augmente à mesure que la taille du problème augmente, ce qui permet d'analyser les problèmes et d'aider à concevoir des algorithmes pour les résoudre. Ici un algorithme est une procédure systématique qui produit dans un nombre fini d'étapes la réponse à une question, ou la solution d'un problème avec un ensemble fini de cas ou de valeurs. Parfois, un algorithme ne peut pas exister pour résoudre une classe infinie de problèmes.

À l'essor du $20^{\text{éme}}$ siècle, dans un effort infructueux pour déterminer les propositions qui peuvent être intraitable, l'influent mathématicien et logicien anglais Alan Turing commence à s'intéresser aux propriétés mathématiques des algorithmes. Turing [184] introduit un modèle de machine abstrait (machine de Turing) dont l'objectif est de proposer une réponse au problème de décidabilité qu'il s'agit de savoir s'il existe un algorithme qui peut décider qu'un énoncé est vrai ou faux. Bien que Turing ait fini par prouver qu'il doit exister des problèmes indécidables qui ne peuvent être résolus ni par la machine de Turing ni par conséquent par l'ordinateur actuel, de tels problèmes sont dit indécidables ou sans preuve, un autre exemple est donné dans la référence [100].

La description des caractéristiques indispensables de la machine de Turing est en effet largement utilisée en théorie de la complexité, en théorie de la calculabilité et devenait le fondement de l'informatique. Aujourd'hui, les problèmes de décision et de calculabilité sont essentiels à la conception d'un programme informatique.

Les problèmes dits faciles, ou traitables, peuvent être résolus par des algorithmes informatiques exécutés en temps polynomial, c'est-à-dire, pour un problème de taille n, le temps, ou le nombre d'étapes nécessaires pour trouver la solution est une fonction polynomiale de n.

Les algorithmes pour résoudre des problèmes difficiles, ou intraitables, nécessitent des temps qui sont des fonctions exponentielles de la taille du problème n. Les algorithmes de temps polynomial sont considérés comme efficaces, tandis que les algorithmes de temps exponentiel sont considérés comme inefficaces, car les temps d'exécution de ces derniers

augmentent beaucoup plus rapidement à mesure que la taille du problème augmente. Des exemples de problèmes intraitables sont présentés dans les références [69, 131, 171].

1.3.2 Problèmes de décision

Un problème de décision est un problème dont la réponse de la question posée appartient à l'ensemble {*oui, non*} [71]. Chaque problème de décision est défini par un énoncé (instance du problème) et formulé sous la forme d'une question. À titre d'exemple, considérons le problème de décision suivant :

Soit X défini comme une paire (I_X, q_X) constituée d'un ensemble I_X d'instances et une question q_X. Une instance de problème $x \in (I_X$ est appelée instance à réponse oui de X si la réponse à la question q_X est oui pour x, alors, la résolution de tel type de problème se limite à savoir l'existence d'une solution au problème et non plus de trouver la solution, autrement, une instance-non consiste à prouver l'inexistence d'un algorithme pour le résoudre (une tâche très difficile).

Par exemple pour un problème de décision, considérez le problème de couverture de sommets **(Vertex Cover)** Figure 1.2, qui est défini comme suit et sera utilisé comme un exemple dans cette section.

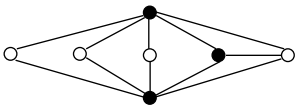

FIGURE 1.2: Exemple de couverture de sommets.

Problème de couverture de sommets (Vertex Cover)

Instance : *Un graphe non orienté $G = (V, E)$, et un entier positif k.*

Question : *Existe-t-il un sous-ensemble $C \subseteq V$ d'au plus k sommets **(vertex)** de telle sorte que chaque arête **(edge)** de E ait au moins un point final en C ?*

Dans le cas de couverture de sommets, chaque instance de problème est constitué d'une paire $(G = (V, E), k)$; un ensemble de sommets $C \subseteq V$ avec la propriété que chaque

arête de E a au moins un point final dans C est appelé une couverture de sommet pour le graphe G (voir la figure 1.2). Les sommets noirs forment une couverture de sommets de taille trois, chaque arête a au moins un nœud noir.

1.3.3 Classes de complexité

Les problèmes peuvent être classés selon les temps de calcul nécessaires pour les résoudre. Pour cela, un modèle informatique abstrait est utilisé qui s'appelle *la machine de Turing (déterministe)* et dont la puissance de calcul est identique à celui de la plupart des programmes informatiques du monde réel. par exemple, Chaque problème qui peut-être résolu par un programme Java peut être aussi résolu dans un temps de fonctionnement «similaire» par une machine de Turing adéquate et vice-versa.

Le temps d'exécution nécessaire pour résoudre un problème X est mesuré en fonction du nombre d'étapes que doit être effectué par une machine de Turing M pour résoudre le problème. Ce nombre d'étapes est toujours donné en fonction de $t_M(n)$ dans la taille n de l'entrée. Par exemple, le temps de fonctionnement d'une machine de Turing pour résoudre un *problème de couverture de sommets* serait donné en fonction du nombre de sommets et des arêtes du graphe d'entrée.

Plus précisément, la fonction $t_M(n)$ exprime toujours ce qu'on appelle le pire des cas des temps de fonctionnement de la machine de Turing M, c'est-à-dire que $t_M(n)$ est la durée maximale requis par M, pris par toutes les entrées de taille n.

Afin de constituer un outil de classement, les problèmes qui peuvent être résolus dans des temps de fonctionnement similaires sont regroupés et triés en classes de complexité, avec P et NP étant deux des plus importantes de ces classes.

1.3.3.1 La classe P

La classe P contient tous les problèmes de décision relativement faciles, c'est-à-dire ceux qui peuvent être résolus dans un temps de fonctionnement polynomial [71], où des algorithmes efficaces sont connus pour les résoudre. Plus formellement, ce sont les problèmes pour lesquels une machine déterministe peut être construite, c'est-à-dire, pour chaque problème $X \in$ P, il existe une machine de Turing M_X dont le temps d'exécution

$t_{M_X}(n)$ est de complexité polynomiale n. Pour la plupart des applications pratiques, un tel temps de fonctionnement polynomial signifie que le problème est résolu en un temps raisonnable.

1.3.3.2 La classe NP

La classe de complexité NP est définie comme la classe des problèmes de décision X qui peuvent être résolus de manière non déterministe dans un temps de fonctionnement polynomial [71]. La classe NP est donc une extension de la classe P où on ne doit pas chercher une solution mais de vérifier si la solution donnée est correcte ou non. Pour chaque problème $X \in$ NP, il existe une machine de Turing non déterministe M_X dont le temps d'exécution $t_{M_X}(n)$ est polynomial dans n. Comme dans le cas des machines de Turing déterministes, le temps de fonctionnement d'une machine de Turing non déterministe M pour un problème X est exprimé comme une fonction $t_M(n)$ dans la taille de donnée n ; il donne le nombre maximal d'étapes dont la machine de Turing peut avoir besoin pour résoudre une instance de problème de taille n.

1.3.3.3 La classe NP-complets

La classe NP-complet est un sous-ensemble de la classe NP contenant des problèmes plus difficiles où en ne peut trouver pour aucun d'entre eux un algorithme de temps polynomial [71]. Mais, l'impossibilité de trouver de tels algorithmes n'a pas été prouvée. Un problème est NP-complet s'il est complet pour NP, c'est-à-dire que tous les problèmes appartenant à NP lui sont réductibles en temps polynomial. Cependant, si on trouve un algorithme polynomial pour un seul problème NP-complet, il y aura alors automatiquement des algorithmes rapides pour tous les problèmes de la classe NP.

1.3.3.4 La classe NP-difficiles (NP-dur)

Les problèmes NP-difficiles sont plus difficiles que les problèmes NP-complets. La classe NP est limitée aux problèmes de décision, un problème d'optimisation associé à un problème de décision NP-complet est NP-difficile [112]. D'une manière générale, un problème X est NP-difficile s'il existe un certain problème Y appartenant à la classe

NP-complet qui pouvait être transformé en temps polynomial de Y à X par la réduction de Turing .

Pratiquement, il est très peu probable qu'un algorithme en temps polynomial pour n'importe quel problème NP-dur puisse être trouvé, car cela impliquerait immédiatement que tous les problèmes de NP (en particulier, tous les problèmes NP-complets) pourraient être résolus en temps polynomial, ce qui signifie que $NP = P$. Il existe des milliers de problèmes NP-complets, et ils se posent dans tous les domaines de la vie [71, 146]. La question de savoir si $NP = P$ est l'un des sept ≪problèmes du prix du millénaire≫ nommé par The Clay Mathematics Institute [187].

1.3.4 Problèmes de recherche

dans les applications pratiques, la résolution d'un problème ne s'arrête pas seulement de savoir si une instance de problème admit-il une solution ou non, mais on s'intéresse à la recherche d'une telle solution. Ainsi, un problème de recherche peut être étendu à partir d'un problème de décision. Par exemple, dans le cas de couverture de sommet, la définition suivante de problème pourrait être plus utile pour de nombreuses applications.

Problème de couverture de sommets II (Vertex Cover II)

Instance : *Un graphe non orienté $G = (V, E)$ et un entier positif k.*

Question : *Trouvez une couverture de sommet C pour G qui se compose au plus de k sommets, ou indiquez qu'il n'existe pas de couverture de sommet.*

Des problèmes de ce type sont appelés problèmes de recherche ou de fonction. étant donné que les définitions de P et NP ne s'appliquent pas à ces problèmes, il existe des classes de complexité spécifiques pour les problèmes de fonction. En particulier, la classe de problèmes où la tâche est de calculer un certificat pour une instance d'un problème de décision X de NP s'appelle FNP (Problème de couverture de sommets II serait un représentant typique pour cette classe). Si un problème dans FNP peut être résolu de manière déterministe en temps polynomial, c'est-à-dire qu'un certificat pour une instance donnée peut-être calculé en temps polynomial s'il existe, puis il appartient à la classe FP.

De nombreux (voir aussi [24]) problèmes de fonction dans FNP ne sont pas vraiment ≪plus difficiles≫ que leurs versions de décision correspondantes dans NP, un fait appelé auto-réductibilité : un certificat pour une instance x d'un problème de décision X de NP peut être construit en appelant de manière polynomiale plusieurs fois un algorithme pour X répondant uniquement avec *oui* ou *non*.

Par exemple, pour construire une couverture de sommet de taille k pour un graphe G, essayez tous les sommets $v \in V$ et vérifiez s'il y a une couverture de sommet de taille $k-1$ pour le graphe G avec élimination de v. Si pour un sommet v la réponse est *oui*, mettez v dans la solution à construire et continuez en construisant un sommet de taille $k-1$ pour le graphe G avec v supprimé.

Bien sûr, si une machine de Turing M résout un problème de fonction de FNP, alors M peut être également utilisé pour résoudre le problème de décision correspondant de NP : il suffit de vérifier si M a renvoyé un certificat ou non.

Les problèmes de fonction dont les machines de Turing peuvent être utilisées de cette façon pour résoudre des problèmes de décision NP-dur sont généralement, appelés NP-dur (bien que le terme ≪NP-dur ≫ ait été à l'origine défini pour des problèmes de décision seulement) et nous ne pouvons pas espérer trouver des algorithmes de temps polynomial pour eux.

1.3.5 Problèmes d'optimisation

Comme déjà vue dans la section 1.2, un problème d'optimisation représente la tâche de trouver parmi l'ensemble de solutions possibles (appelées aussi des solutions ≪faisables ou réalisables≫) celle qui maximise ou minimise une fonction objective, c'est-à-dire on cherchera la solution avec la valeur optimale.

Un problème d'optimisation peut être obtenu à partir d'un problème de recherche par l'association d'une valeur (coût ou poids) à chaque solution.

La version d'optimisation de problème de couverture de sommet par exemple, est définie comme suit :

Problème de couverture de sommets III (Vertex Cover III)

Instance : *Un graphe non orienté G = (V, E).*

Question : *Trouvez une couverture de sommet de taille minimale C pour G.*

Ici, l'ensemble de solutions faisables consiste en toutes couvertures de sommets pour le graphe d'entrée, et le coût d'une solution est simplement le nombre de ses sommets.

La classe de complexité qui contient les problèmes d'optimisation où on doit calculer une solution optimale est NPO.

La classe PO contient les problèmes de NPO qui peuvent être résolus de manière déterministe en temps polynomial, c'est-à-dire les problèmes pour lesquelles une solution optimale pour une instance donnée peut-être trouvée en temps polynomial.

La classe NPO est étroitement liée à la classe NP, le problème de décision suivant :

instance : *soit $x \in I_X$ un problème de minimisation (maximisation) $X \in NPO$ et un nombre k.*

Question : *décider si x a une solution faisable $s \in SOL_X(x)$ dont le coût est au plus (au moins) k.*

Ce problème appartient clairement à NP et s'appelle « problème de décision dans NP qui correspond à X ». De nombreux problèmes d'optimisation $X \in NPO$ peuvent être résolus en appelant plusieurs fois un algorithme polynomial pour le problème de décision correspondant dans NP.

Dans le cas de **couverture de sommets III**, par exemple, on peut d'abord calculer la taille d'une couverture de sommet minimale pour le graphe donné par des appels répétés d'un algorithme de couverture de sommets et mettant k à $1, 2, 3, \ldots$ jusqu'à ce qu'il réponde avec *oui*.

La couverture de sommet peut ensuite être construite en utilisant la méthode décrite dans le paragraphe des problèmes de fonction.

Dans l'autre sens, une machine de Turing M qui résout *le problème de couverture de sommet III* peut également être utilisée pour résoudre le problème de décision NP-dur *couverture de sommet*, il suffit de vérifier si la couverture de sommet renvoyé par M a

une taille d'au plus k. De tels problèmes d'optimisation s'appellent NP-dur, et, encore une fois, nous ne pouvons pas espérer de trouver des algorithmes en temps polynomial pour eux.

Le fait que des algorithmes pour les problèmes NP-complets peuvent être utilisés pour résoudre des problèmes dans FNP et dans NPO et vice-versa est également exprimé dans les notions suivantes : problème couverture de sommet, problème couverture de sommet II et problème couverture de sommet III, ainsi que tous les autres problèmes de décision NP-complets et NP-dur.

Les problèmes dans FNP et NPO sont appelés des équivalents polynomiaux, ce qui signifie que, d'une part, ils ne sont pas censés être résolus en temps polynomial, mais, d'autre part, si l'un de ces problèmes peut être résolu en temps polynomial, alors les autres problèmes sont résolus aussi en temps polynomial.

1.3.6 La réduction en temps polynomial

Bien qu'il existe des problèmes trop difficiles pour être résolu de manière non déterministe en temps polynomial, la classe NP contient un très grand nombre de problèmes, beaucoup d'entre eux d'une importance pratique considérable. En particulier, tous les problèmes qui appartiennent à la classe P sont également contenus dans NP. Cependant, il existe de nombreux problèmes importants dans NP qui ne peuvent pas être résolus de façon déterministe en temps polynomial. L'essai de deviner ou d'essayer plusieurs possibilités à plusieurs étapes déterministes, conduit souvent à des temps d'exécution exponentiels (appelés «explosion combinatoire»), ce qui rend ces problèmes souvent difficiles à résoudre dans la pratique.

Afin de grouper ces problèmes «difficiles»dans une classe spéciale, et de produire des preuves de leur caractère intraitable en montrant que toutes, ou aucun de ces problèmes n'est résolu de façon déterministe en temps polynomial, le concept de réductions peut être introduit comme suit :

On dit qu'un problème $P1$ se réduit polynomialement en un problème $P2$, s'il existe un algorithme polynomial A construisant à partir d'une donnée $D1$ de $P1$, une donnée $D2$ de $P2$ telle que la réponse pour $D1$ soit *oui* si et seulement si la réponse pour $D2$ est *oui*.

La réduction polynomiale permet de comparer les problèmes de décision ainsi que la réduction de Turing permet de comparer les problèmes de recherche.

On dit qu'un problème de recherche $P1$ se réduit polynomialement à un problème de recherche $P2$ par la réduction de Turing s'il existe un algorithme $A1$ pour résoudre $P1$ utilisant comme sous–programme un algorithme $A2$ pour résoudre $P2$ de telle sorte que la complexité de $A1$ est polynomiale quand on évalue chaque appel de $A2$ en temps constant.

Le principe de la réduction d'un problème à un autre peut être illustré par la plus simple réduction du problème *Ensemble Indépendant sur la Couverture de Sommet* (Independent Set to Vertex Cover) : Le problème d'ensemble indépendant représenté par un graphe $G = (V, E)$ et un entier positif k, si G est un ensemble indépendant $V\prime \subseteq V$ d'au moins k sommets. (Un ensemble indépendant est un ensemble des sommets qui sont des paires non connectés par des arêtes).

L'ensemble indépendant peut être réduit à une couverture de sommet en reliant chaque instance (G, k) de problème d'ensemble indépendant à une instance $(G, |V| - k)$ de problème de couverture de sommet. Si $(G, |V| - k)$ est une instance à réponse oui de couverture de sommet, alors G a une couverture de sommet C de taille au plus $|V| - k$ et, par conséquent, (G, k) est une instance à réponse oui de l'ensemble indépendant.

Les sommets n'appartenant pas à C ne sont pas connectés par des arêtes et forment un ensemble indépendant de taille au moins k (pour voir ceci, considérons les sommets blancs de la figure 1.2). Si, cependant, l'instance $(G, |V| - k)$ est une instance de non de couverture de sommet, alors (G, k) est une instance de non de l'ensemble indépendant, car s'il y avait un ensemble indépendant V' de taille Au moins k, alors les sommets dans $V \backslash V'$ forment une couverture de sommet de taille au plus $|V| - k$.

Chapitre 2

Problème étudié

2.1 Problème de l'élaboration d'un planning pour la rotation des infirmières

Le problème de l'élaboration d'un planning pour la rotation des infirmières (NRP ou gestion d'horaires des infirmières), est un problème particulier de construction d'emploi du temps. Il consiste à concevoir un planning qui satisfait les exigences de l'établissement ainsi que les demandes du personnel requis, tout en respectant un certain nombre de contraintes. Ces dernières sont divisées en deux catégories : les contraintes dures, qui doivent être satisfaites pour garantir la faisabilité, et les contraintes dites molles, qui représentent les préférences et aversions exprimées par les infirmières, la distribution équitable des vacations, etc.... Les contraintes molles peuvent être violées, mais la qualité du planning est en relation proportionnelle avec leur taux de satisfaction.

2.1.1 Description du problème NRP

un planning de rotation des infirmières est une affectation de quadruple (infirmière, jour, type de shift, type de compétences). Il consiste à assigner chaque infirmière à un type de shift (matin, soir, nuit) pour chaque jour ouvrable durant la période de planification. Cette affectation se faite en correspondance avec leur niveau d'expérience et leur compétence afin de garantir la couverture des demandes du personnel, en prenant en

compte les contraintes dures et molles, avec différents objectifs (par exemple, maximiser les préférences des infirmières, minimiser la violation des demandes du personnel, minimiser les heures supplémentaires, etc. . .) et une fonction objective est utilisée pour évaluer la qualité de la solution.

2.1.2 Complexité

Le problème de rotation des infirmières est un problème d'optimisation combinatoire très difficile sa complexité est théoriquement NP-dur [58, 140]. En pratique, il est très difficile à résoudre en raison de la grande masse de données et de l'explosion combinatoire. De tels problèmes sont compliqués et difficiles, en particulier pour les résoudre à l'optimalité [34]. Cela signifie que l'on n'obtient une solution exacte qu'avec l'usage d'un algorithme prenant un temps exagérément long. Généralement, on opte pour un algorithme relativement rapide mais qui produit une solution approximative.

Tout l'art est dans ce compromis entre la vitesse d'exécution et la qualité de la solution fournie.

2.1.3 Modélisation du NRP

2.1.3.1 Les variables de décisions

Le NRP est habituellement modélisé selon le point de vue à partir de laquelle on peut agir sur les variables. Les principaux modèles adoptés fréquemment dans la littérature sont : infirmière- jour, infirmière-shift et infirmière-shift pattern [48].

Supposons que N infirmières doivent être planifiées. Les infirmières sont affectées à un shift pour chaque jour. Supposons que la période du planning implique D jours, chacun composé de S shifts.

Une vue infirmière-jour est représentée par un planning à deux dimensions. En effet, les variables de décision sont définies pour chaque infirmière à chaque jour comme x_{ij}, où $1 \leq i \leq N$ indexe les infirmières et $1 \leq j \leq D$ indexe les jours dans une période de planification, Burke *et al.* [40], Post and Veltman [149] ont appliqué ce modèle dans leurs propositions. Les domaines de ces variables consistent en des périodes de travail et des périodes de repos.

Les périodes de travail peuvent inclure plusieurs types de shift par jour, le plus courant est de partager le jour en trois shifts de huit heures de travail pour chacun, un shift du matin (M), un shift d'après-midi (S) et un shift de nuit (N).

Les périodes de repos incluent les jours de congé pour plusieurs raisons : les jours de repos (R), les jours de récupération, les jours fériés, les congés annuels, les jours d'étude, le congé de maternité, le congé sans solde, etc.... Les variables de décision peuvent généralement prendre 10 valeurs ou plus ce qui augmente les efforts de calcul. Pour cela, plusieurs chercheurs s'orientent à réduire toutes les valeurs des périodes de repos à une valeur (R) de sorte que la variable de décisions prendra quatre valeurs possibles :

$$x_{ij} = \begin{cases} \text{(R)} & \text{Si l'infirmière } i \text{ est affecté a un quart de repos le jour } j \\ \text{(M)} & \text{Si l'infirmière } i \text{ est affecté a un quart du matin le jour } j \\ \text{(S)} & \text{Si l'infirmière } i \text{ est affecté a un quart d'après-midi le jour } j \\ \text{(N)} & \text{Si l'infirmière } i \text{ est affecté a un quart de nuit le jour } j \end{cases}$$

	S	D	L	M	M	J	V
N1	M	N	S	N	M	N	S
N2	S	R	R	R	S	R	R
N3	R	M	N	M	R	M	N
N4	N	S	M	S	N	S	M

FIGURE 2.1: Exemple de la vue infirmière-jour.

Pour les modèles 0/1, les variables de décision peuvent être personnalisées pour être x_{ijk}, où i, j sont les mêmes index que celle pour x_{ij}, et $1 \leq k \leq S$ indexe les S shifts possibles en une journée.

$$x_{ijk} = \begin{cases} 1 & \text{Si l'infirmière } i \text{ travail le shift } k \text{ le jour } j \\ 0 & \text{autre.} \end{cases}$$

	S				D				L				M				M				J				V			
	M	S	N	R	M	S	N	R	M	S	N	R	M	S	N	R	M	S	N	R	M	S	N	R	M	S	N	R
Nurse1	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0
Nurse2	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0
Nurse3	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0
Nurse4	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1

FIGURE 2.2: Exemple de la vue infirmière-jour (modèle binaire).

Une vue infirmière-shift est une variante étroite de la vue infirmière- jour. La variable de décision peut être définie pour chaque infirmière à chaque shift comme x_{is}, où

$1 \leq i \leq N$ indexe les infirmières et $1 \leq k \leq S$ des shifts indexées dans une période de planification, Hicks and Hennessy [96], Moz and Pato [135] ont proposé des modèles selon ce point de vue. La seule différence entre la vue infirmière-jour et la vue infirmière-shift est qu'un shift définie dans la vue infirmière-shift peut ne pas correspondre nécessairement à un «jour». Le modèle binaire est comme suite :

$$x_{is} = \begin{cases} 1 & \text{Si l'infirmière } i \text{ travail le shift } k \\ 0 & \text{autre.} \end{cases}$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	M	S	N	R	M	S	N	R	M	S	N	R	M	S	N	R	M	S	N	R	M	S	N	R	M	S	N	R
Nurse1	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0
Nurse2	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0
Nurse3	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0
Nurse4	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1

FIGURE 2.3: Exemple de la vue infirmière-shift.

Une vue d'infirmière-shift pattern est différente des deux vues ci-dessus. Un shift pattern est représenté comme un planning individuel réalisable qui peut être affecté à une infirmière en tant que bloc de shifts. Comme le personnel peut préférer avoir des plannings simples et plus conforts, il est souhaitable d'avoir les shift patterns les moins laborieux. [138],[20], [89] ont appliqué ce modèle dans leurs travaux, Aickelin and Dowsland [8, 9] ont modélisé leurs problèmes en tant qu'IP et définissent des variables de décision comme x_{ip}, dont $1 \leq i \leq N$ indexe les infirmières et $1 \leq p \leq M$ indexe les shift patterns.

$$x_{ip} = \begin{cases} 1 & \text{Si l'infirmière } i \text{ travail le shift pattern } j \\ 0 & \text{autre.} \end{cases}$$

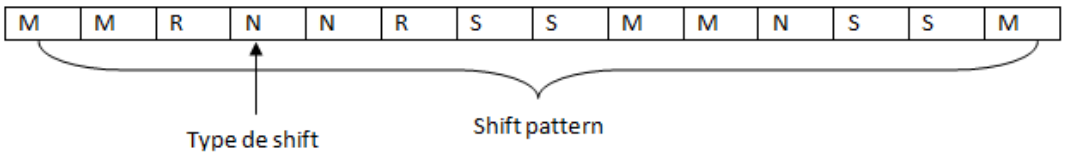

FIGURE 2.4: Exemple d'un shift pattern.

2.1.3.2 Les contraintes

chaque établissement a ses propres besoins et caractéristique suivant leur réglementation et politique de gestion ce qui engendre la différence des contraintes d'un établissement à l'autre, classiquement, les contraintes se regroupent en trois catégories de base qui sont : les contraintes d'affectation, les contraintes de couvertures et les contraintes d'horaires [129].

Les contraintes d'affectations : représentent les affectations obligatoires ou interdites d'une tâche j bien définie pour un employé i durant une période bien précise d, généralement chaque affectation est couplée avec une pénalité, l'affectation avec une faible pénalité est plus souhaitable d'être considéré.

Les contraintes de couvertures : assurent les demandes de personnel requis pour la couverture d'une charge de travail pendant chaque unité de temps en fonction de leurs qualifications, responsabilités et expérience, ce type de contrainte est défini souvent par une borne inférieure et supérieure, la couverture minimale du staff est fréquemment définie comme étant une contrainte dure.

Les contraintes d'horaires : sur chaque ligne de l'horizon de plannings, des contraintes horizontales sont imposées pour le contrôle de la séquence des charges de travail individuelle pour chaque employé. Un poids ou une pénalité est associé à chaque contrainte pour minimiser le taux de violation, quatre types de contraintes d'horaires sont omniprésents pour l'élaboration d'un planning [25, 38] comme suit :

- **-La consécutivité :** ces contraintes limitent le nombre d'affectations consécutives d'un employé ou d'un type de charge pour un employé.
- **-Le temps libre :** la quantité de temps libre nécessaire entre deux blocs d'assignations est limitée par ces contraintes.
- **-Le nombre total :** ces contraintes limitent le nombre total d'affectations d'un employé durant l'horizon de travail.
- **-Les successions interdites :** quelques successions des shifts de travail ne sont pas autorisées, intuitivement, il est impérativement non accepté d'affecter une charge de matinée après une nuit de travail, de telles contraintes sont généralement classées comme contraintes dures.

Le NRP se caractérise habituellement comme un problème sur-contraint, une large gamme de contraintes est trouvé dans la littérature, ces contraintes sont classées selon leur nature en deux types de base :

Les contraintes dures (Hard) : sont celles qui doivent être impérativement satisfaites pour avoir un planning réaliste, telles que :

- Une infirmière doit travailler un et un seul type de shift par jour.
- Nombre total d'infirmières nécessaire pour chaque catégorie de compétences et chaque type de shift de travail pendant toute la durée de planification.
- Nombre minimal et maximal de shift de nuit travaillé par chaque infirmière par période.
- Période de repos minimale entre les shifts de travail.

Les contraintes molles (Soft) : déterminent les contraintes souhaitables à être satisfaites autant que possible, en pratique, il est généralement impossible de trouver des solutions réalisables qui répondent à toutes les contraintes molles, la qualité du planning généré est mesurée avec le taux de satisfaction de ces contraintes, une pénalité ou un coût est attribué à chacune d'eux, la violation de ces contraintes induise un coût supplémentaire sur le coût final de planning.

Par exemple les contraintes qui se produisent sont énumérées comme suite :

- Affectation a la compétence principale,
- Les shifts patterns indésirables.
- Préférences ou exigences des infirmières.
- Nombre minimal ou maximal des shifts de travail consécutifs.
- Nombre minimal et maximal d'affectation par période.

2.1.3.3 Les objectifs

Comme les besoins et les contraints les objectifs aussi se différent d'un établissement à l'autre. La qualité des solutions est mesurée par un ou plusieurs critères généralement intégrés à une fonction objective. Le but est de trouver une solution qui maximise ou minimise la valeur de la fonction objective.

Le NRP est souvent exprimé comme un problème multi-objective dont les contraintes molles sont considérées comme des objectifs à atteindre où chaque contrainte est liée à une variable de décision et l'objectif général est de minimiser le plus possible le coût global de ces sous-objectifs.

Les objectifs qui peuvent être considérés par exemple sont :

- Réduction des coûts de la main-d'œuvre infirmière
- Minimisation de l'insatisfaction du patient
- Maximisation des préférences des infirmières
- Minimisation des heures supplémentaire

2.2 Etat de l'art

Dans les dernières décennies le problème de rotation des infirmières est devenu un sujet très important. En raison de sa complexité et de son caractère difficile Il a reçu une considérable attention au fil des ans, il a été l'un des problèmes les plus étudiés vu que plusieurs chercheurs s'orientent à ce domaine[56]. Plusieurs études qui fournissent l'état de l'art ont été présentés.

Cheang *et al.* [48] ont présenté une brève et distinctive étude bibliographique, dans laquelle une description de plusieurs façons de modélisation a été présentée. Différentes méthodes de classification en fonction de modèles et de types de contraintes ont été proposées, ainsi que les différentes approches de résolutions existantes.

L'un des plus complets est présenté dans Burke *et al.* [38], représentant l'état de l'art depuis près de 40 ans, dans lequel une description de différentes méthodes et modèles a été présentée avec différentes classifications, en fonction des spécifications du problème et des techniques couramment appliquées pour le résoudre.

Ernst *et al.* [66] ont présenté une étude bibliographique approfondie sur le problème général de planification du personnel, les caractéristiques générales et les terminologies fréquemment utilisées ont été définies, dans lequel le problème a été classé selon les modèles, les méthodes de solution utilisées, et les domaines d'application. Le problème de la planification des infirmières a été motionné dans le domaine de la santé.

Généralement, les objectifs principaux des problèmes de planification du personnel hospitalier, ou plus générale, du personnel sont les mêmes, mais, malgré ça, les problèmes sont souvent différents aussi bien que les approches de résolution[56]. Dans cette section, on va présenter un aperçu général sur les différents modèles du problème ainsi que les différentes approches de résolution.

2.2.1 Classification du problème

Les contraintes et les objectifs se différent d'un hôpital à l'autre. Cela a pour effet d'engendrer une pléthore de modèles et de méthodes. En fonction de cette diversité, un problème de rotation des infirmières peut être classé selon différents critères.

— Type de problème

— Type de planning

— Approche de résolution

1 Type de problème

Selon les modèles et les contraintes, un NRP peut être classés comme un problème d'optimisation ou comme un problème de décision. Les problèmes d'optimisation sont basés généralement sur des techniques de la programmation mathématique, alors que les problèmes de décision habituellement se basent sur des heuristiques et autres outils d'IA [48].

a Les problèmes d'optimisation :

le NRP a été traitée comme un problème d'optimisation pour minimiser ou maximiser une fonction objective, où des méthodes exactes de l'optimisation combinatoire ont été utilisées pour le résoudre, ces méthodes sont basé sur la programmation mathématique, les méthodes traditionnelles de programmation linéaire et la programmation en nombres entiers.

b Les problèmes de décision :

Dans les situations où il ya un grand nombre de contraintes à traiter pour les modèles de la programmation mathématique, il peut être plus approprié de modéliser le problème comme un problème de satisfaction de contraintes (PSC), et les solutions possibles sont les affectations de valeurs à des variables de décision satisfaisant toutes les contraintes.

c Problèmes d'optimisation de contraintes :

La majorité de NRP du monde réel sont sur-contraints, où l'affectation de valeurs à des variables de décision sans violer aucune contrainte est très difficile et même, souvent, impossible, dans ce cas-là, un problème de satisfaction de contraintes (PSC) peut être considéré comme un problème d'optimisation de contraintes, où on ne cherche pas uniquement une solution mais plutôt une meilleure solution qui minimise le taux de violation des contraintes. La qualité des solutions est mesurée par un ou plusieurs critères qui sont généralement incorporés dans une fonction objective. Le but est de trouver une solution qui maximise ou minimise la valeur de la fonction objective. Un problème modélisé de cette manière est appelé problème d'optimisation de la satisfaction des contraintes (POSC), qui consiste en un PSC standard associé à une fonction objective.

2 Type de planning

En général, il existe deux types de base de planification utilisés qui sont la planification cyclique ou non cyclique. La plupart des approches heuristiques se concentrent sur la résolution de problèmes cycliques, tandis que les approches basées de la programmation mathématique et de l'intelligence artificielle peuvent être utilisées pour les deux types cycliques et non cycliques [48].

a La planification cyclique :

Le même programme est répété aussi longtemps que les conditions ne changent pas. Ces types de plannings sont faciles à construire et ne dépendent pas des affectations anticipé, mais peu flexible, car peu évolutif, et peut-être très rigide et s'adaptent difficilement aux changements.

b La planification non cyclique :

Un nouveau planning est généré pour chaque période de planification. Ce processus nécessite plus de temps pour l'élaboration, mais plus flexible aux changements tels que la variabilité de la demande [27].

3 Approche de résolution

On applique généralement deux méthodes, exact ou approché.

— Méthode exacte (exhaustive) : explore l'espace de recherche dans sa totalité pour trouver la solution optimale si elle existe, ou de prouver autrement l'inconsistance du problème.

— Méthode approchée (approximative) : explore une sous-partie de l'espace de recherche. Si le problème est très compliqué pour avoir une solution exacte, alors on cherche une solution approximative pour gagner du temps [90].

Méthodes exactes :

Parmi les méthodes exactes que l'on trouve en abondance dans la littérature, on a la programmation linéaire, Branche & Baund, Branche & price, etc. . . .

Ces méthodes de la programmation mathématique sont appropriées pour trouver des solutions optimales. La plupart des approches mathématiques sont basées sur l'optimisation d'une fonction objective unique. Un certain nombre d'expériences ont également été menées avec les méthodes de la programmation mathématique, Thornton and Sattar [179] ont utilisé une approche d'optimisation à base de la programmation en nombres entiers (PNE), Maenhout and Vanhoucke [123] ont proposé une approche de branch-and-price, Trilling *et al.* [181, 182] ont présenté une méthode basée sur la programmation entière pour résoudre le problème de planification des infirmières anesthésiste.

Bien que le temps de calcul nécessaire pour obtenir une solution optimale augmente de façon exponentielle avec la taille du problème, de nombreux chercheurs continuent de proposer des approches exactes. La dernière compétition internationale concernant les tableaux de service des infirmières a été dominée par de telles méthodes. Certains d'entre eux sont rapportés dans ce qui suit.

En commençant d'une formulation compacte, Santos *et al.* [162] ont introduit une technique de la programmation entière avec des procédures de génération de coupures (cut generation procedures) améliorées, où le problème primaire est décomposé en un ensemble de sous problèmes. Deux phases successives sont proposées. Une simple heuristique constructive gloutonne est utilisée pour générer une solution initiale réalisable. Ensuite, la méthode de la programmation mathématique proposée explore une procédure de recherche locale pour améliorer la solution construite à l'aide d'une technique de descente à voisinage variable. Ce dernier explore deux structures de voisinage basées sur la résolution des sous problèmes à l'optimalité en utilisant la formulation IP fournie. En conséquence, ils ont gagné le challenge avec des résultats meilleurs que celle obtenue par les premiers gagnants de la compétition INRC2010, où la majorité des cas sont résolus à l'optimalité, et de nouvelles meilleures solutions sont découvertes.

Baeklund [17] a développé une méthode branch-and-Price où le problème principal (maître) affecte les horaires aux infirmières, et la relaxation linéaire est résolue par la génération de colonnes. Les résultats montrent l'efficacité de la méthode pour des instances NRP avec une période de planification de quatorze jours.

Les gagnants de la compétition INRC II Römer and Mellouli [157] ont proposé une nouvelle formulation MILP basée sur une approche de flux réseau. L'incertitude des demandes concernant la fonctionnalité en plusieurs étapes du problème pendant les semaines à venir a été gérée avec une stratégie prospective permettant de prévoir les besoins d'une période de planification anticipée, dans laquelle des contraintes supplémentaires ont été intégrées. Le solveur Coin CBC a été utilisé pour résoudre le programme MILP décrit à l'optimalité.

Legrain *et al.* [113] ont abordé le problème présenté dans INRC II avec une procédure de branchement et évaluation (Branch & Bound). Un bloc de shifts de travail séparés par un jour de repos représentant les rotations possibles pour une infirmière, a été modélisé comme un problème du flux permettant de constituer des plannings individuels à l'aide d'un réseau dirigé. La formulation IP conçue sur la base des rotations possibles générées est résolue avec l'application de l'algorithme B&B pour les petits problèmes, tandis que les grandes instances sont traitées en intégrant une recherche adaptative dans le grand voisinage (adaptive large neighborhood search). L'étude expérimentale montre des résultats prometteurs, particulièrement, pour les instances de grande taille.

Mischek and Musliu [134] ont étudié une nouvelle formulation IP dans laquelle une extension de contraintes supplémentaires a été incorporée dans le modèle IP de base pour traiter la propriété en plusieurs étapes des instances de l'INRC II. C'est une extension du travail présenté dans [133]. L'idée principale est d'étendre certaines contraintes molles qui ne sont pas vraiment nouvelles, mais ils sont envisagés pour le planning global, tandis que le modèle IP a été appliqué pour générer une solution optimale pour chaque semaine. La contribution consiste à générer une extension des contraintes à partir des contraintes globales pour qu'ils soient considérés pour chaque semaine. Les résultats expérimentaux sur l'ensemble de données "Hidden" montrent la bonne performance de la nouvelle formulation IP par rapport à la formulation de base.

Méthodes approchées :

Malheureusement les méthodes exactes ne sont efficaces que pour les instances de problèmes de petite taille ou elles donnent un très bon succès. par contre les méthodes approchées sont capables de trouver des solutions pour les problèmes de grande taille dans un temps de calcul tolérable. Parmi ces méthodes on a les méthodes d'intelligence artificielle, les heuristiques et hyper-heuristiques, et les meta-heuristiques.

a Méthodes d'intelligence artificielle :

Les méthodes d'intelligence artificielle son basé en général sur la programmation logique à base de règle, où chaque contrainte représente une règle. Parmi ces méthodes on a la programmation à base de contrainte et les systèmes expert :

Li *et al.* [114] ont proposé une approche d'intelligence artificielle hybride, qui combine des techniques à base de contraintes avec des techniques de la recherche locale.

Cheng *et al.* [50] ont décrit la conception et la mise en œuvre d'un système d'emploi du temps des infirmières, ils proposent une méthode de modélisation redondante et ils utilisent une approche à base de contraintes.

Okada and Okada [139] ont implémenté un système expert basé sur Prolog pour le problème de planification des infirmières

Chen and Yeung [49] ont développé un système expert hybride (INFIRMIÈRE-HELP) qui combine un système expert avec la programmation objectif linéaire zéro-un (0/1-LGP) pour résoudre le problème d'affectation des infirmières

b Les heuristiques :

Les heuristiques ont été conçue pour un problème d'optimisation particulier permettant de trouver des solutions avec un temps de calcul acceptable traduisant une stratégie ou une manière de penser, généralement les heuristiques ne garantissent pas l'optimalité, la plupart des heuristiques sont développées pour générer des plannings cycliques, tel qu'il est utilisé par Smith and Wiggins [169], aussi Constantino *et al.* [53] ont proposé une heuristique simple est efficace où la phase constructive et la phase d'amélioration sont incorporées. La première construit un planning en résolvant un problème d'affectation pour chaque jour de l'horizon de planification, tandis que la deuxième consiste à résoudre les problèmes d'affectation pour produire de meilleurs programmes. Les résultats obtenus sur les instances de NSPLib montrent l'efficacité de la méthode.

Zhong *et al.* [191] ont proposé une heuristique à deux phases, l'algorithme proposé génère des affectations des shifts aux infirmières selon le niveau du personnel requis, et minimise la différence de charge de travail, dont la deuxième phase répartit les quarts de week-end le plus uniformément possible pour assurer la production des plannings équitable.

Wong *et al.* [188] ont présenté une nouvelle heuristique simple et efficace dans un service d'urgence local à Hong Kong, au début une solution initiale est générée par une simple procédure d'affectation satisfaisant toutes les contraintes dures, ensuite un algorithme de recherche locale séquentielle est utilisé pour améliorer le planning construit en prenant en compte toutes les contraintes molles.

Komarudin *et al.* [108] ont décrit une approche en trois étapes pour le problème de la qualité du personnel, dans laquelle le planning est construit en tenant compte de la structure de personnel adaptée. Le personnel a été regroupé en sous-groupes selon ses caractéristiques particulières (niveau de compétence, type de contrat). l'étape de vérification de la cohérence, une vérification est effectuée pour déterminer si la structure de personnel proposée peut répondre ou non aux besoins du personnel Les deux étapes suivantes consistent à évaluer la qualité de la structure du personnel et de ses voisinages. L'approche proposée vise à améliorer la qualité du planning en modifiant, si nécessaire, la structure du personnel au niveau de la dotation.

La société ORTEC [103] a participé à la compétition INRC II avec leur solveur qui a été directement appliquée pour générer des plannings hebdomadaires. L'incertitude liée à la demande de shifts et de compétences a été traitée avec la stratégie de la liaison des semaines, dans laquelle ils intègrent de nouvelles contraintes (soft) artificielles liées aux aperçus hebdomadaires programmés, ces contraintes servent à prévoir les besoins estimés pour la semaine prochaine. Le solveur ORTEC a été classé cinquième à la compétition, donnant des résultats prometteurs.

c Hyper-heuristiques :

Les hyper-heuristiques sont des méthodes itératives d'optimisation qui comprend un ensemble d'heuristiques ou de méta heuristiques permettant à l'aide d'un mécanisme de sélectionner et exécuter à chaque itération les heuristiques les plus puissantes pour résoudre le problème, les résultats sont acceptés ou rejetés selon un critère d'acceptation ; ainsi les mécanismes de sélection et d'acceptation sont

indépendants du problème [37]. Comme il est proposé par Anwar *et al.* [12] où un algorithme de recherche harmonique est intégré dans le cadre d'une hyper-heuristique, une séquence d'heuristiques de bas niveau est appliquée, la recherche harmonique consiste d'un espace de recherche de solutions comprend des solutions réalisables et un espace de recherche heuristique contient des ensembles d'heuristiques.

Bilgin *et al.* [26] ont déployé un ensemble de voisinages dans le cadre d'une hyper-heuristique. Une stratégie de tournoi a été utilisée pour basculer entre les ensembles de voisinages. Des méthodes de recuit simulé (SA)et de grand déluge (GD)ont été utilisées comme critères d'acceptation. Les résultats expérimentaux montrent que l'hyper-heuristique avec SA donne de meilleurs résultats qu'avec GD, et surpassent les résultats obtenus avec VNS.

Asta *et al.* [14] ont proposé une hyper-heuristique composée de quatre groupes d'heuristique de bas niveau (mutation, ruine et recréation, croisement et recherche locale), L'approche proposée incorporant un algorithme d'apprentissage automatique basé sur le tenseur.

Bilgin *et al.* [28] ont présenté une hyper-heuristique combinée avec une heuristique gloutonne, une méthode simple de sélection aléatoire est appliqué pour le choix de l'heuristique de bas niveau, avec une méthode de recuit simulé en tant que critère d'acceptation de mouvement, le résultat obtenu est amélioré à l'aide de l'heuristique gloutonne.

Gretsista and Burke [80] ont proposé une hyper-heuristique sélective dans le cadre d'une recherche locale itérée (ILS). L'ILS est représenté comme une heuristique de haut niveau composée d'une recherche locale et une heuristique de perturbation. Cette dernière est développée comme une hyper-heuristique sélective composée de plusieurs heuristiques de perturbation de bas niveau. L'approche proposée est testée sur un large éventail d'instances. Différentes stratégies de sélection ont été testées et comparées. Les résultats expérimentaux montrent que l'approche proposée est plus performante que d'autres hyper-heuristiques existant dans la littérature.

Kheiri *et al.* [106] ont décrit une approche hyper-heuristique sélective. La sélection est appliquée sur neuf heuristiques de bas niveau, le but est de construire une séquence d'heuristique sélectionnée qui sert à améliorer une solution candidate.

Malgré la simplicité de l'hyper-heuristique générique présentée, cette approche a été classé la troisième dans la compétition INRC II et présente un impact remarquable sur le taux de faisabilité.

d Méta-heuristique :

Les méta-heuristiques sont plus générales que les heuristiques et souvent inspirées de la nature, elles peuvent être adaptées à plusieurs problèmes d'optimisation, elles offrent des solutions «acceptables» dans un délai raisonnable pour résoudre des problèmes complexes et difficiles ; elles sont parmi les techniques les plus prometteuses et réussies qui gagnent beaucoup de popularité dans les deux dernières décennies [42].

Selon la stratégie suivie à la recherche les méta-heuristiques sont classées en deux catégories : à base de solution unique ou à base de population de solutions.

— **Les méta-heuristiques à solution unique**

Les méta-heuristiques à base de solution unique ou méthodes de trajectoire écartent progressivement à partir d'une seule solution initiale en construisant une trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, la méthode de recuit simulé, la recherche tabou, la méthode GRASP, la recherche à voisinage variable, la recherche locale itérée, et leurs variantes [31], dans la suite, nous proposons de passer en revue quelques-unes.

Burke *et al.* [39] ont proposé une recherche dans le voisinage variable. Ils ont appliqué deux différents algorithmes de recherche pour créer une solution initiale avant le processus de recherche, ces méthodes sont : la méthode de plus profonde descente (Steepest Descent) et la recherche tabou, plusieurs dispositions ont été mises en œuvre pour la permutation entre ces algorithmes.

Solos *et al.* [170] ont appliqué une méthode de recherche stochastique de voisinage variable à deux phases. Les résultats sur des instances NRP réelles montrent que l'algorithme produit de bons résultats.

Bilgin *et al.* [27] ont proposé un nouveau modèle générique pour la représentation des contraintes, ils ont présenté une nouvelle approche de la recherche locale dans le voisinage impliquant six types de voisinage qui sont exploré selon la technique de la recherche dans le voisinage variable.

Bellanti *et al.* [23] ont proposé un algorithme de recherche tabou basée sur le voisinage, au début une solution initiale est construite à l'aide d'un algorithme glouton, cette solution est améliorée par l'exploration de quatre types de voisinage, pour garder la faisabilité uniquement une solution partielle liée aux shifts de nuit et de repos est considérée, l'algorithme glouton est utilisé ensuite pour attribuer les shifts de jour.

Amponsah *et al.* [11] ont proposé une approche de coloration de graphique des conflits pour le service de maternité d'Ejura au Ghana, où les sommets représentent les infirmières et chaque deux infirmières qui sont en conflits selon leurs contraintes molles sont relié par un arc, les sommets sont colorés avec des couleurs différentes s'ils sont adjacents les uns aux autres, une technique a été conçue pour transformer le graphe de conflit coloré en un tableau de shift après l'application de toutes les contraintes dures.

Ramli and Ahmad [154] ont utilisé une technique de la recherche tabou pour résoudre un problème particulier de rotation des infirmières, ils ont introduit une représentation spéciale avec la construction d'une génération innovante de voisinage.

Liu *et al.* [118] ont proposé un algorithme de recuit simulé avec trois structures de voisinage pour résoudre un cas réel de problème de génération de plannings des infirmières à plusieurs niveaux dans un service d'hémodialyse à Wuhan, Chine.

Brucker *et al.* [33] ont proposé une méthode de construction adaptative où les contraintes sont regroupées en trois classes : liée à la séquence, liée aux infirmières et liée au planning ; une approche de décomposition en deux étapes est utilisée, en considérant seulement les contraintes de séquence, la première étape construit des séquences pour les infirmières et sur cette base, la deuxième étape construit des plannings en considérant le reste des contraintes. Une recherche locale gloutonne est effectuée pour l'amélioration après chaque construction du planning, de nouveaux ensembles de données de référence sont introduits dans cette recherche.

Afin d'améliorer à la fois la qualité et l'équité d'un planning, Komarudin *et al.* [109] ont introduit un ensemble de nouvelles fonctions objectives d'équité fondées sur des règles lexicographiques. Un algorithme de recherche Tabou déployant un ensemble de voisinages a été présenté. La méthode proposée se compose principalement de deux phases consécutives. Au cours de la première phase, seule la

fonction objective fondée sur la qualité est prise en compte ; par la suite, l'équité du planning est traitée au cours de la deuxième phase en tenant compte de l'une des fonctions objectives d'équité. En raison de la difficulté de produire des plannings équitables de haute qualité, trois critères ont été examinés pour évaluer la qualité et l'équité d'un planning. Les différentes évaluations appliquées ont montré que la méthode à deux phases proposée peut produire une solution équitable avec une légère diminution de la qualité totale.

Ceschia and Schaerf [45] ont proposé un algorithme simple de recuit simulé, dans lequel deux voisinages ont été utilisés pour explorer l'espace de recherche. Ce travail présente une extension de l'approche introduite dans [54], dans laquelle le swap unique et le changement ont été remplacés par un swap multiple entre un ensemble de jours, en fonction de types de shifts et des compétences, et un changement multiple entre un ensemble de jours et de types de shifts.

— **Les méta-heuristiques à population de solutions**

Les méta-heuristiques à population de solutions améliorent durant plusieurs itérations une population de solutions. Parmi ces méthodes On distingue, les algorithmes évolutionnaires issus de la théorie de l'évolution énoncée par Darwin [55] et les algorithmes d'intelligence en essaim, généralement, ces algorithmes sont inspirés de la nature par analogie avec des phénomènes biologiques naturels [177], les plus importantes sont rapportées dans ce qui suit :

Burke *et al.* [42] ont proposé une approche de la recherche dispersée (scatter search) où de nouvelles implémentations de sous-programmes sont développées ; les résultats obtenus sur des benchmarks, ainsi que sur des instances NRP réel, révèlent que l'approche proposée est efficace et robuste.

Aussi Maenhout and Vanhoucke [122] ont présenté un algorithme de recherche dispersé qui s'avère robuste face à des contraintes spécifiques à chaque cas. L'approche électromagnétique de Maenhout and Vanhoucke [124], expérimentée sur des instances NRP de NSPLib, peut également être démontrée comme robuste par rapport à des contraintes spécifiques.

Gutjahr and Rauner [82] ont implémenté une approche d'optimisation par colonie de fourmis qui est évaluée sur un problème réel qui se pose à l'hôpital de Vienne.

Todorovic and Petrovic [180] ont proposé une méthode d'optimisation des colonies d'abeilles qui alternent les phases de construction et d'amélioration. Certaines

parties de ses voisinages sont intelligemment écartées. La méthode est évaluée sur des données réelles provenant d'hôpitaux belges et les résultats obtenus montrent l'efficacité de l'approche proposée.

Tsai and Li [183] ont présenté un algorithme génétique utilisé pour la construction et l'amélioration des plannings, une étude de cas dans une salle d'ORL à Tainan, Taïwan, est traitée efficacement.

Zheng and Gong [190] ont introduit une méthode d'optimisation de la réaction chimique, simulant le mouvement des molécules ; l'approche est testée sur des ensembles de données provenant de la première compétition internationale de NRP en 2010.

Özcan [141] ont utilisé un ensemble d'algorithmes mimétiques, combinant un ensemble d'opérateurs génétiques et une méthode hiérarchique d'escalade colline dirigé par la violation auto-adaptative (self adaptive violation directed hierarchical hill climbing method).

De nos jours, la tendance est vers l'hybridation. Pour améliorer la performance d'une méthode de classement stochastique,Bai *et al.* [19] ont hybridé le recuit simulé dans le cadre d'algorithme génétique et de la recherche locale ; les résultats obtenus sur un problème réel d'un grand hôpital au Royaume-Uni, montrent que l'algorithme proposé dépasse les méthodes publiées.

Burke *et al.* [43] ont présenté un modèle multi-objectif hybridant la programmation entière et la recherche du voisinage variable ; un programme entier constitué des contraintes dures et d'une partie des contraintes molles est résolu, et la solution obtenue est améliorée par une recherche du voisinage variable, une autre heuristique de recherche dans le voisinage variable est utilisée pour satisfaire le reste des contraintes.

Burke and Curtois [36] ont hybridé la méthode Branch-and-Price avec une méthode de la chaîne d'éjection (ejection chain), l'algorithme est testé sur un grand nombre de problèmes du monde réel et aussi de la première compétition internationale de NRP.

He and Qu [93] ont présenté un algorithme combinant la programmation par contraintes et la génération de colonnes, basé sur des informations rassemblées au cours de la recherche, de nouvelles techniques sont utilisées lors de la génération de colonnes, pour générer des colonnes de bonne qualité.

Jaradat *et al.* [99] ont proposé un système de fourmis élitiste hybride (Elitist-Ant System) avec l'ajout d'une mémoire externe dont le contenu est différent, des matrices de phéromones pour stocker des solutions élites servant de guider la recherche, une recherche locale itérée (ILS) est incorporée pour améliorer les solutions tout en maintenant la diversité pour échapper de bloquer dans des optimums locaux.

Jin *et al.* [104] ont hybridé la recherche harmonique (HS) avec une technique de système immunitaire artificiel (AIS), les opérations de clonage et de mutation de la procédure AIS sont appliquées sur la meilleure solution générée dans la procédure HS pour aider l'amélioration de la nouvelle harmonie par la mise à jour de la population existante.

Le NRP a été étudié depuis plus de 40 ans, il a fait l'objet d'une littérature abondante qui dépasse ce qu'on a pu citer. Une grande diversité d'approches a été proposée, où la majorité des paradigmes de la recherche opérationnelle ont été appliqués.

2.2.2 Benchmark pour le problème de rotation des infirmières

Généralement, les chercheurs ont tendance à traiter des problèmes réels issus des hôpitaux locaux, souvent, avec simplifications de ces problèmes , pour le problème de rotation des infirmières les contraintes et les objectives se défaire entre les organisations hospitalières ce qui engendre une pléthore de modèle et de méthode puisqu'il n'y a pas un modèle standard pour modéliser le problème. Cela rend la comparaison et l'évaluation des différentes approches très difficiles, si pas impossible [33]. Dans un effort pour aider à minimiser l'écart entre la théorie et la pratique, des collections d'instances de référence (benchmark) ont été rendues accessibles au public. Ces ensembles de données de référence offrent un grand nombre de cas variés, dont les chercheurs peuvent l'utiliser pour évaluer la performance ou la robustesse de leurs nouvelles approches de solutions [166].

Vanhoucke and Maenhout [186] ont proposé un grand ensemble d'instances de données basées sur différents indicateurs de complexité pour le problème de planification des infirmières pour faciliter l'évaluation des approches proposées. La base de données des

instances et les meilleures solutions trouvées peuvent être téléchargées à partir du site http://www.projectmanagement.ugent.be/research/personnel_scheduling/nsp.

Aussi Burke *et al.* [41] ont introduit une collection des instances pour le problème de planification du personnel, dont la majorité des problèmes sont des NRP. Les instances collectées proviennent de publications scientifiques, de sociétés de logiciels, d'instances artificielles, et la plupart sont des exemples réels recueillis auprès de treize pays différents dans le monde entier. Les ensembles de données ont été unifié et présentés au format XML standard afin de produire des données de référence variées et stimulantes. Ces données sont accessibles via la page Web : http://www.schedulingbenchmarks.org/ qui a été conçue et gérée par l'Université de Nottingham.

Il ya encore d'autres instances de données disponibles sur Internet comme celles proposées par Bilgin *et al.* [27], représentant un ensemble d'instances des problèmes complexes, où différents scénarios sont associés à chaque ensemble. Les scénarios décrivent des vrais problèmes du monde réel. ces instances sont disponibles et téléchargeables à partir de la page Web : https://gent.cs.kuleuven.be/nurserostering.html.

Haspeslagh *et al.* [92] ont proposé la première compétition internationale pour les plannings des infirmières ≪INRC-2010 ≫, cet ensemble de données vise à fournir des problèmes avec des contraintes très compliquées du monde réel en intégrant des contraintes supplémentaires manquées dans les littératures précédentes. cette base est accessible sur ce site http://www.kuleuven-kortrijk.be/nrpcompetition.

Fondé sur le succès de la première compétition INRC 2010, Ceschia *et al.* [46] ont introduit la deuxième édition de la compétition ≪INRC-II ≫. Contrairement aux variantes de problèmes de l'INRC 2010, l'INRC II est caractérisé par la formulation en plusieurs étapes, telle qu'une étape représente un programme d'une semaine, et la liste finale considère une séquence d'horaires hebdomadaires dépendant les uns des autres. la base des instances est disponible sur ce lien http://mobiz.vives.be/inrc2/

Le tableau suivant regroupe les bases les plus utilisées dans la littérature ainsi que leur description et les travaux en relation.

Les bases de données	Description	Les travaux en relation
NSPLIB	[186]	[124] [122] [125] [123] [168] [53] [94]
Nottingham	[25]	[33] [42] [35] [44] [76] [89] [153] [80]
KAHO	[27]	[165] [166] [167] [128] [109] [26] [108]
INRC-2010	[92]	[160] [26] [121] [185] [130] [13] [91]
INRC II	[46]	[134] [133] [157] [45] [113] [103] [106] [54]

TABLE 2.1: Table benchmark et travaux en relation

2.3 Spécification du problème étudié

Comme on a vu précédemment, plusieurs bases de données, modèles et formulations ont été présentées par divers chercheurs. Dans la pratique, l'importance des contraintes molles diffère d'un établissement à l'autre, et un bon planning pour un établissement peut être inacceptable pour un autre, dans ce cas-là, la position d'une approche générale pour résoudre ce genre de problèmes est impossible.

Dans cette thèse, on va étudier le problème issu de la base de données NSPlib, pour plus de détails sur la technique de génération des instances et leurs niveaux de complexité voir [186].

2.3.1 Formulation mathématique de NRP

Supposons que n infirmières doivent être planifiées. Les infirmières travaillent sur des shifts (fractions d'un jour). Supposons que l'horizon du planning implique d jours, chacun composé de s shifts, $s-1$ shifts de travail et un shift de repos (l'attribution d'une infirmière à un shift de repos signifie qu'elle obtient un jour de congé pour une raison quelconque).

Une contrainte dure, inhérente à chaque NRP, est :

— Contrainte 1 : chaque infirmière doit être affectée à exactement un shift de travail par jour.

Les infirmières ont généralement des niveaux de compétence différents. Cependant, comme les infirmières avec le même niveau de compétence peuvent être programmées de façon indépendante, on peut supposer, sans perte de généralité, que toutes les infirmières ont le même niveau de compétence. Il y a plusieurs points de vue à partir de

laquelle on peut agir sur les variables [38]. Ici, nous adoptons le point de vue de modèle infirmière-shift pattern.

Soit $p = s \times d$. Un shift pattern pour une infirmière est un p vecteur binaire où les s première coordonnées référencent aux shifts de la première journée, et ainsi de suite. Par conséquent, la k^{iem} coordonnée, où $k = \alpha \times s + \beta$, se réfère au déplacement k^{iem} qui est le shift numéro β (du jour numéro α) + 1.

Par exemple, le vecteur $(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1)$ est un shift pattern, où la période de planification se compose de $d = 4$ jours, et chaque jour se compose de $s = 3$ shifts, le troisième étant un shift de repos. Cet exemple indique que l'infirmière en question est affecté au shift 1 du jour 1, shift 2 du jour 2, et est libre au cours des deux derniers jours.

La première contrainte prise isolément résulte s^d de shifts patterns possibles pour chaque infirmière, le nombre augmente de façon exponentielle avec la longueur de la période.

Heureusement, il existe plusieurs contraintes spécifiques liées au temps qui sont considérés comme des contraintes dures, telles que les limites sur :

- le nombre minimum/maximum d'affectations par période,
- le nombre minimum/maximum d'affectations par type de shift,
- le nombre minimum/maximum de shifts de travail consécutif,
- le nombre minimum/maximum de même type de shifts consécutifs,
  * Plus quelque succession interdites de shifts (Soir/Matin, Nuit/Matin, Nuit/Soir).

Par la suite, ces contraintes permettent de filtrer les shifts patterns de la manière suivante. Les shifts patterns sont pris dans l'ordre. Tout shift pattern qui viole l'une des contraintes est annulé. On suppose que m est le nombre des shifts patterns valides restants. Nous mettons tous les s^d shifts patterns valides dans une $m \times p$ matrice binaire définie par :

$$a_{ik} = \begin{cases} 1 & \text{si shift pattern } i \text{ couvre shift } k \\ 0 & \text{autre} \end{cases}$$

D'ordre pratique, soit $M = \{1, \cdots, m\}$, $N = \{1, \cdots, n\}$, et $P = \{1, \cdots, p\}$. L'infirmière j exprime ses préférences en donnant un e_{jk} matrice des coûts (le moins le meilleur) pour travailler sur le shift k. Selon les préférences pour les shifts, les ≪coûts de préférence≫ pour les shifts patterns sont calculés pour les infirmières comme suite :

$$c_{ij} = \sum_{k \in P} a_{ik} e_{jk}, \quad i \in M, \quad j \in N$$

Le coefficient c_{ij} est interprété comme le coût total de préférence pour l'infirmière j lorsqu'elle est affectée au shift pattern i.

Une autre contrainte dure, intrinsèque à chaque NRP, est la contrainte de couverture. Il exige un nombre minimal d'infirmières pour chaque shift.

— Contrainte 2 : le nombre d'infirmières requises au shift k est au moins b_k (nul pour les shifts de repos).

L'objectif est d'optimiser les préférences des infirmières, en attribuant un shift pattern unique à chaque infirmière et en satisfaisant les contraintes de couverture.

Soit la variable binaire x_{ij} spécifie si le shift pattern i est affecté à l'infirmière$j$.

$$x_{ij} = \begin{cases} 1 & \text{si shift pattern } i \text{ est affecté à l'infirmière } j \\ 0 & \text{autre.} \end{cases}$$

Le modèle mathématique est :

$$\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} \tag{2.1}$$

$$\sum_{i \in M} x_{ij} = 1 \qquad j \in N \tag{2.2}$$

$$\sum_{i \in M} a_{ik} \sum_{j \in N} x_{ij} \geq b_k \qquad k \in P \tag{2.3}$$

$$x_{ij} \in \{0, 1\} \qquad i \in M, \quad j \in N \tag{2.4}$$

L'objectif (2.1) est d'optimiser les préférences des infirmières. Les contraintes (2.2) indiquent que chaque infirmière est affectée à un seul shift pattern, tandis que les contraintes (2.3) précisent que le nombre d'infirmières affectées au shift k n'est pas inférieur à b_k.

2.4 Conclusion

Le problème de rotation des infirmières est un problème d'optimisation combinatoire très difficile, aussi bien du point de vue de la complexité théorique que de la mise au point pratique.

Dans la littérature existante, beaucoup de modèles ont été mis en évidence. Pratiquement, toute la panoplie des méthodes de la recherche opérationnelle a été testée, quelquefois avec succès. La tendance lourde étant axée sur les méta-heuristiques.

Chapitre 3

Outils théoriques

Ce chapitre présente des notions de base nécessaires à la compréhension de notre contribution dans les chapitres 4, 5 et 6. Sa rédaction est basée sur plusieurs références bibliographiques.

3.1 Relaxation lagrangienne et méthode de sous-gradients

3.1.1 Introduction

La relaxation lagrangienne a été développée au début des années 1970 avec le travail pionnier de Held and Karp [95] sur le problème du voyageur de commerce. la plupart des problèmes d'optimisation difficiles peuvent être considérés comme des problèmes faciles compliqués par l'ajout d'un ensemble relativement restreint de contraintes secondaires. L'idée de base de la relaxation lagrangienne est de relâcher les contraintes difficiles et les réinjecter dans la fonction objective sous la forme de pénalités combinant linéairement les contraintes relaxées. Les coefficients de cette combinaison linéaire sont les variables duales associées à la relaxation lagrangienne, appelées multiplicateurs lagrangiens.

la dualisation des contraintes difficiles produit un problème appelé dual lagrangien où une méthode de sous-gradients est souvent utilisée pour le résoudre. La valeur de la solution optimale obtenue du problème dual donne une borne inférieure (dans un cas de minimisation) de la solution optimale du problème original.

Cette section présente la relaxation lagrangienne dans un cadre général (voir [22, 67, 68, 73, 137, 164]). Pour une application de cette technique à un problème précis, le NRP, voir les chapitres 4 et 5.

3.1.2 problème primal et problème dual

Considérons le problème (P) un problème d'optimisation combinatoire avec n variables et m contraintes. Soient $\mathbf{A} \in \mathbb{R}^{m \times n}$ avec les vecteurs $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$ et $\mathbf{b} \in \mathbb{R}^m$. (P) est connu comme le problème primal par opposition au dual (D) qui sera introduit ensuite.

$$(\mathbf{P}) \begin{cases} min & \mathbf{c}^t\mathbf{x} & \\ s.t & \mathbf{Ax} & = \mathbf{b} \\ & \mathbf{x} & \geq 0 \\ & \mathbf{c}, \mathbf{x} & \in \mathbb{R}^n \end{cases}$$

Si le problème primal est présenté sous la forme standard alors le problème dual (D) s'exprime sous la forme suivante :

$$(\mathbf{D}) \begin{cases} max & \mathbf{b}^t\mathbf{y} & \\ s.t & \mathbf{A^t y} & \leq \mathbf{c} \\ & & \\ & \mathbf{b}, \mathbf{y} & \in \mathbb{R}^m \end{cases}$$

avec m variables, n contraintes, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b}, \mathbf{y} \in \mathbb{R}^m$ et $\mathbf{c} \in \mathbb{R}^n$.

Par opposition, s'il y a des contraintes d'inégalité dans le problème primal, son dual s'exprime comme suite :

Problème primal			**Problème dual**		
min	$\mathbf{c}^t\mathbf{x}$		max	$\mathbf{b}^t\mathbf{y}$	
$s.t$	$\mathbf{Ax}$	$\geq \mathbf{b}$	$s.t$	$\mathbf{A^t y}$	$\leq \mathbf{c}$
	$\mathbf{x}$	≥ 0		$\mathbf{y}$	≥ 0
	$\mathbf{c}, \mathbf{x}$	$\in \mathbb{R}^n$		$\mathbf{b}, \mathbf{y}$	$\in \mathbb{R}^m$

En résumé, pour chaque problème primal, on associe un problème dual, on permutant les variables et les contraintes de sorte que chaque variable du primal correspond à une contrainte du dual et à chaque contrainte du primal correspond une variable du dual, on conclut que le dual du dual est le problème primal.

3.1.3 problème dual lagrangien

Le principe de la relaxation lagrangienne consiste à éliminer les contraintes difficiles et les inclure sous forme de pénalités dans la fonction objective, on dit qu'on dualise ces contraintes. Pour cela, on associe à chaque contrainte un multiplicateur de Lagrange ou variable duale et en définissant la fonction de Lagrange (ou problème relaxé).

Supposons le problème (P) suivant :

$$(\mathrm{P}) \begin{cases} \min & \mathbf{z} & = \mathbf{cx} \\ \text{s.t} & \mathbf{Qx} & \geq \mathbf{g} \\ & \mathbf{Sx} & \geq \mathbf{b} \\ & \mathbf{x} & \in \mathbb{N}^n \end{cases} \tag{3.1}$$

avec deux ensembles de contraintes $Q \in \mathbb{R}^{m_Q \times n}$ et $S \in \mathbb{R}^{m_S \times n}$. La RL résultant de relâcher les contraintes ($\mathbf{Sx} \geq \mathbf{b}$) serait

$$(\mathrm{D}) \begin{cases} \min & \mathbf{z} & = \mathbf{cx} + \lambda(\mathbf{b} - \mathbf{Sx}) \\ \text{s.t} & \mathbf{Qx} & \geq \mathbf{g} \\ & \mathbf{x} & \in \mathbb{N}^n \end{cases} \tag{3.2}$$

Le programme relaxé est appelé le problème Dual Lagrangien (DL) ou programme lagrangien de la borne inférieure (Lagrangian Lower Bound Program (LLBP)). Il fournit une borne inférieure à la solution optimale du problème original (P) pour tout $\lambda \in \mathbb{R}^{m_S}_{\geq 0}$.

Intuitivement, on peut voir que λ impose une pénalité aux contraintes violées. La valeur objective optimale d'un problème DL est une borne inférieure pour la valeur objective optimale du problème primal (P), Beasley démontre ce fait dans [22] comme suit :

La valeur objective optimale z^* de (P) est supérieure à la valeur objective optimale de

$$\begin{array}{lll} \min & \mathbf{z} & = \mathbf{cx} + \lambda(\mathbf{b} - \mathbf{Sx}) \\ \text{s.t} & \mathbf{Qx} & \geq \mathbf{g} \\ & \mathbf{Sx} & \geq \mathbf{b} \\ & \mathbf{x} & \in \mathbb{N}^n \end{array}$$

puisque $\lambda \geq 0$ et $(b - Sx) \leq 0$, un terme qui est ≤ 0 et simplement ajouté à la fonction objective. Ceci n'est pas inférieur que la valeur objective optimale de

$$\begin{array}{lll} \min & \mathbf{z} & = \mathbf{cx} + \lambda(\mathbf{b} - \mathbf{Sx}) \\ \text{s.t} & \mathbf{Qx} & \geq \mathbf{g} \\ & \mathbf{x} & \in \mathbb{N}^n \end{array}$$

comme l'élimination d'un ensemble de contraintes pour un problème de minimisation ne peut que réduire la valeur de la fonction objective.

Pour une relaxation lagrangienne utile, deux points-clés doivent être mise en évidence, une stratégie pour choisir la meilleure contrainte à relâcher de telle sorte que le problème relaxé soit plus facile à résoudre que le problème original, la deuxième étape consiste à trouver les bonnes valeurs numériques pour les multiplicateurs lagrangiens, qui produisent la borne inférieure la plus élevée. Il s'agit de trouver des multiplicateurs qui correspondent à :

$$\max \lambda \geq 0 \left\{ \begin{array}{lll} \min & \mathbf{z} & = \mathbf{cx} + \lambda(\mathbf{b} - \mathbf{Sx}) \\ \text{s.t} & \mathbf{Qx} & \geq \mathbf{g} \\ & \mathbf{x} & \in \mathbb{N}^n \end{array} \right\}$$

Un algorithme heuristique pour trouver de bonnes valeurs pour les multiplicateurs lagrangiens et approcher λ^*, est l'optimisation de sous-gradient décrite dans la section suivante.

3.1.4 Résolution du problème dual : algorithme du sous-gradient

L'optimisation de sous-gradient (SG), est une méthode pour résoudre de manière heuristique un problème dual lagrangien. les multiplicateurs lagrangiens sont ajusté itérativement

pour trouver les valeurs qui produisent la meilleure borne inférieure ou la plus proche. cette méthode s'appuie sur la solution du problème DL, et sur une borne supérieure Z_{UB} pour la valeur de la fonction objective optimale du problème original [151].

Supposons un ILP comme dans les équations 3.1. nous voulons relâcher les contraintes ($\mathbf{Sx} \geq \mathbf{b}$). Ceci résulte en un problème DL comme dans les équations 3.2. L'optimisation du sous-gradient proposé par Beasley [22] peuvent être décrite comme suite :

1. Initialiser une borne supérieure Z_{UB} qui pourrait, par exemple, être calculée en trouvant une solution réalisable avec une heuristique pour le problème original.

   En débutant avec un ensemble initial (λ_i) de multiplicateurs tel que $\lambda_i = 0$ $\forall 1 \leq i \leq m_s$.

2. Soit $Z_{LB} = c.x^* + \lambda(b - S.x^*)$ correspond à la valeur de la fonction objective du problème DL avec l'ensemble actuel des multiplicateurs (λ_i). Les sous-gradients $\delta_i \in \mathbb{R}^{m_s}$ définis pour les contraintes relâchées sont évalués à la solution actuelle par :

   $$\delta_i = (b - S.x^*)$$

3. Définir une taille de pas (scalaire) $\Delta \in \mathbb{R}^{m_s}$ par :

   $$\Delta = \frac{\pi(Z_{UB} - Z_{LB})}{\sum_{=1}^{m_s} \delta_i^2}$$

   Le coefficient π, est un paramètre défini par l'utilisateur, appelé coefficient de relaxation satisfaisant $0 \geq \pi \geq 2$. L'usage est de débuter avec $\pi_0 = 2$ puis le diviser par 2 après β itérations, puis $\lceil \beta/2 \rceil$ itérations, etc....

   Cette taille de pas dépend de l'écart entre la limite inférieure actuelle Z_{LB} et la limite supérieure Z_{UB} et le coefficient π, avec $\sum_{=1}^{m_s} \delta_i^2$ étant un facteur d'échelle.

4. Mettez à jour λ_i en utilisant

   $$\lambda_i = max(0, \lambda_i + \Delta.\delta_i) \forall 1 \leq i \leq ms$$

   et passez à l'étape 2 pour résoudre le problème DL avec ce nouvel ensemble de multiplicateurs. Comme nous l'avons déjà indiqué, cette procédure itérative ne s'arrêtera jamais, nous introduisons donc une règle de terminaison basée soit sur

la limitation du nombre d'itérations possible, soit sur la valeur de π où π est réduit au cours de la procédure.

Finalement, on prend la valeur maximale du problème dual lagrangien comme meilleure approximation du problème primal

3.2 Voisinages et amélioration locale

3.2.1 Introduction

La bonne exploitation de l'espace de recherche nécessite des stratégies qui peuvent guider le processus de recherche afin de trouver des solutions au moins proches de l'optimale pour des problèmes d'optimisation difficiles lorsqu'il n'y a aucune approche classique efficace connue [105] (telle que les problèmes de recherche opérationnelle, de l'intelligence artificielle, bio-informatique, ...).

Les méta-heuristiques sont largement reconnues comme des stratégies manipulant aisément l'espace de recherche. Il existe généralement deux classes de méta-heuristiques : les méthodes à population de solutions qui ont tendance à diversifier la recherche par le parcourent des lots différents de l'espace de recherche, tandis que les méthodes à solution unique ou les recherches locales tendent à intensifier la recherche par l'exploration d'une partie de l'espace des solutions candidates [31].

La recherche locale est une méta-heuristique utilisée pour améliorer une solution initiale à l'aide d'un algorithme de recherche du voisinage, en choisissant à chaque itération une nouvelle solution dans son voisinage. L'espace de recherche associé à un problème d'optimisation combinatoire est souvent très vaste et non calculable en un temps raisonnable [127]. Les techniques de la recherche du voisinage essayent de trouver une nouvelle solution par la transformation de la solution courante, à plusieurs reprises, dans le voisinage de cette dernière, jusqu'à ce qu'une meilleure solution soit trouvée, ou que le critère d'arrêt soit atteint. Le voisinage d'une solution, est l'ensemble de toutes les solutions qui peuvent être obtenues par une simple modification d'une petite partie de la solution candidate (par exemple, la modification d'un seul nœud dans un arbre recouvrant engendre un arbre différent), pour cela, on doit définir une notion de voisinage sur l'espace de recherche. Les méthodes de la recherche locale sont fondées sur la notion de

voisinage et sur la procédure d'exploitation de ce voisinage et ,généralement, à base de cette procédure les recherches locales se différencient. Pour une recherche du voisinage à très grande échelle, le voisinage est vaste et peut-être de taille exponentielle. Nous présenterons ici les méthodes les plus utilisées : la méthode de descente, la recherche locale itérée, le recuit simulé, la recherche tabou et la recherche dans le voisinage à très grand échelle.

3.2.2 La méthode de descente

La méthode de descente est une méthode classique et assez ancienne, sa simplicité et rapidité sont le point fort de sa popularité et son succès [143, 144]. Chaque étape de la recherche consiste à exploiter le voisinage pour progresser à une nouvelle solution de meilleure qualité.

À partir d'une solution initiale s, une solution voisine s' est choisie, puis s sera remplacé par s' si $f(s') \leq f(s)$, ce processus est répété jusqu'à ce que toutes les solutions voisines sont moins bonnes que la solution actuelle, c'est-à-dire, un optimum local est obtenu. Les méthodes de descente peuvent être différencié selon la stratégie de génération de la solution de départ et du parcours du voisinage, on peut distinguer trois types fondamentaux : la descente stochastique, la descente déterministe et la descente vers le premier meilleur.

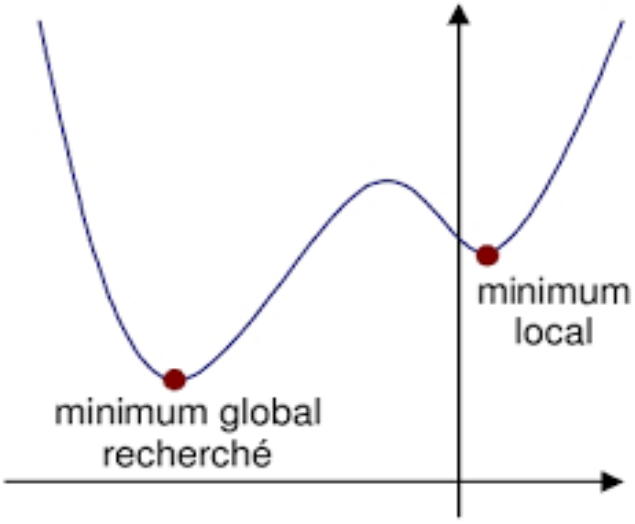

FIGURE 3.1: La méthode de descente

3.2.3 La recherche locale itérée (Iterated Local Search, ILS)

Les méthodes de descente sont faciles à mettre en œuvre et rapides, mais malgré la simplicité ces méthodes n'aboutissent généralement pas des optima locaux de meilleure qualité puisque la recherche s'arrête au premier optimum local trouvait. Pour échapper de stagner à un optimum local, plusieurs stratégies sont mises en place pour poursuivre la recherche dans d'autres régions dans le voisinage, cependant, un critère d'arrêt doit être défini, ce critère peut-être un nombre maximal d'itérations, un temps d'exécution, ou encore un nombre d'itérations sans aucune amélioration. La recherche locale itérée est une bonne technique pour pallier l'arrêt brutal sur un optimum local. En appliquant une stratégie de perturbation de la solution actuelle suivie d'une méthode de descente. La décision d'accepter ou de rejeter la nouvelle solution est faite selon un critère d'acceptation. Ces étapes se poursuivent jusqu'à ce qu'un critère d'arrêt soit atteint.

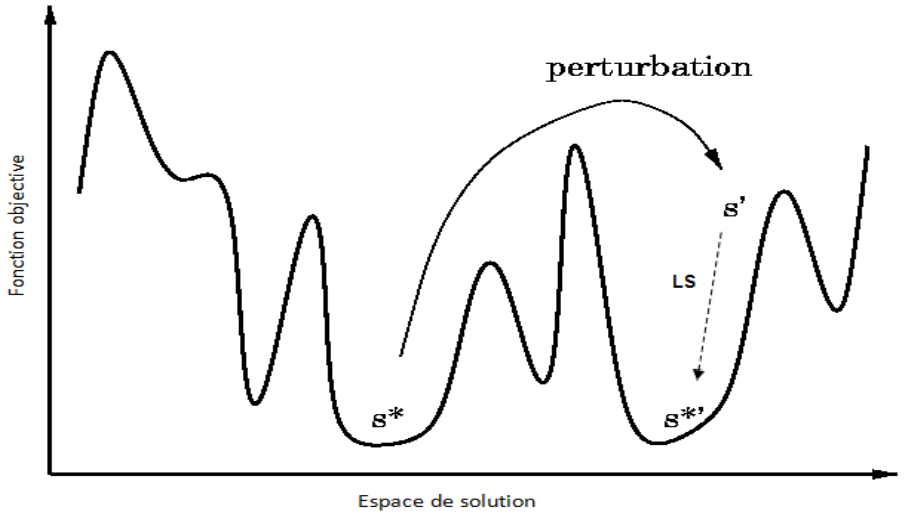

FIGURE 3.2: La recherche locale itérée

3.2.4 La recherche tabou (Tabu Search, TS)

La recherche tabou est une heuristique de recherche locale introduite par Glover [77] comme une amélioration de la méthode de descente, où la recherche est poursuivie (même si un optimum local est rencontré) par l'acceptation des déplacements vers des meilleurs voisins qui n'améliorent pas forcément la solution courante. Pour éviter les mouvements cycliques (retours en arrière) en utilisant une liste tabou (notion de mémoire) contenant les attributs des dernières solutions rencontrées pour l'exclusion des déplacements vers

des solutions récemment visitées, dont le principe est d'enlever la plus ancienne solution à chaque fois qu'une nouvelle solution est ajoutée à la liste tabou. Ainsi, la recherche est poursuivie dans le voisinage de la solution courante sans considération des déplacements existant dans la liste tabou pour échapper des optima locaux. Néanmoins, le réglage des paramètres de la méthode telle que la taille de la liste tabou, ainsi que la définition d'un critère d'arrêt, s'avère difficile à déterminer.

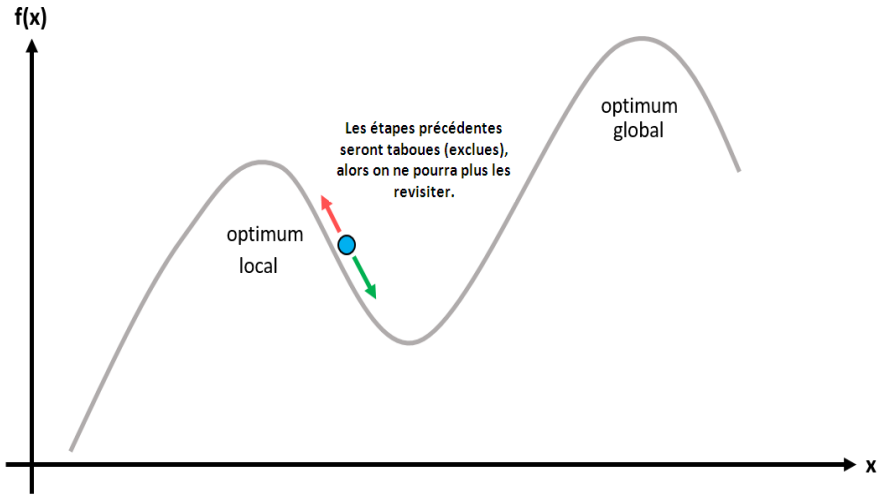

FIGURE 3.3: La recherche tabou

3.2.5 Le recuit simulé (Simulated Annealing, SA)

Le recuit simulé est une méthode destinée à résoudre au mieux des problèmes d'optimisation difficiles, cette méthode est inspirée des techniques de simulation de Metropolis (proposées en 1953) en mécanique statistique. C'est une généralisation de la méthode Monte-Carlo, à laquelle un paramètre de température est introduit et il sera ajusté pendant la recherche [107].

Cette méthode se repose sur l'exploitation du voisinage par une procédure qui permet d'échapper des optima locaux, par la considération des voisins de moins bonne qualité qui font diminuer la fonction objective, et elle est acceptée avec une probabilité non nulle. Au début de l'algorithme, on définit un paramètre de température T qui se décroît au cours de l'exécution de l'algorithme pour tendre vers le zéro. La probabilité p d'acceptation des solutions qui se dégradent dépend de la valeur de ce paramètre (cette probabilité se baise en correspondance avec la décroissance de la température). L'intérêt de recuit simulé, est qu'il converge à de bonnes solutions par rapport aux algorithmes

de recherche classiques. Malheureusement, le réglage des paramètres recommandés est difficile à déterminer et se fait souvent d'une façon empirique, aussi un critère d'arrêt est nécessaire si le paramétrage "optimal" n'a pas été trouvé.

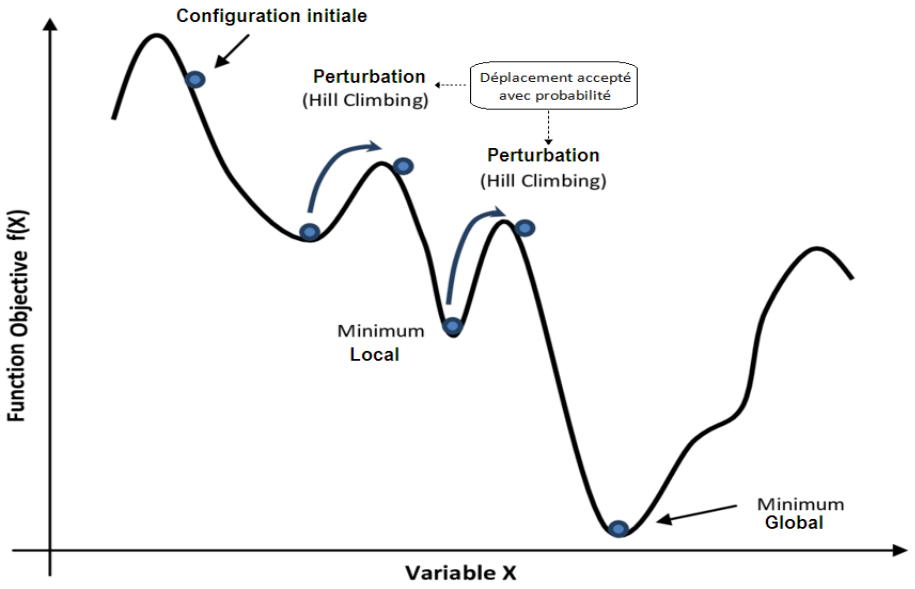

FIGURE 3.4: Le recuit simulé

3.2.6 La recherche dans le voisinage à très grande échelle (Very Large Scale Neighborhood, VLSN)

Les heuristiques sont généralement regroupées dans deux classes principales : soit des algorithmes de construction ou d'amélioration, d'habitude les algorithmes de construction produisent des solutions initiales construites étape par étape avec l'affectation des valeurs aux variables de décision (une à chaque fois), par contre les algorithmes d'amélioration débutent généralement à partir d'une solution initiale faisable et tentent d'améliorer leur qualité par une petite modification de la solution originale. Les méthodes de la recherche locale ou ce que l'on appelle les recherches dans le voisinage sont largement présentées comme des algorithmes d'amélioration. Ces méthodes obtiennent à chaque itération une solution améliorée par la recherche dans le voisinage de la solution courante. La définition des voisins constitue le défi majeur lors de développements des techniques de recherche dans le voisinage ; le point faible de ces méthodes est de piéger (stagner) dans un optimum local, mais d'une façon générale, on peut constater que les voisinages larges ont tendance de trouver des optimums locaux de haute qualité par rapport à d'autres issues d'un petit voisinage, malgré ça, malheureusement, le processus de parcourir ses voisins

prend des temps très couteux que de petits voisins et s'augmentent exponentiellement avec la taille du problème. Alors, la recherche dans un grand voisinage n'est pas pratique si la recherche n'est pas guidée par une méthode de recherche efficace. Ainsi, un algorithme de recherche rapide est indispensable pour l'utilisation efficace des grands voisinages.

Selon la description d'Ahuja *et al.* [4], les algorithmes de recherche du voisinage à très grande échelle sont classés en trois catégories principales. La première catégorie regroupe les méthodes à profondeur variable (variable-depth methods) appliquant une heuristique pour faire une recherche partielle dans un voisinage très large qui grandit d'une façon exponentielle. La deuxième catégorie contient des algorithmes d'amélioration basée sur le flux de réseau (network flow based improvement algorithms) pour la recherche dans le voisinage et l'identification des voisins améliorés par l'utilisation des techniques de flux de réseaux. La troisième catégorie se compose des voisinages basés sur des restrictions des problèmes NP-difficiles qui peuvent être résolus en temps polynomial. L'étude d'Ahuja *et al.* [4, 6] sur les méthodes de voisinage à très grand échelle fournit une exposition approfondie des algorithmes dans ces trois catégories.

Les techniques de recherche de voisinage à très grande échelle ont été appliquées à une gamme très large de problèmes d'optimisation combinatoire. Il s'agit notamment du problème général d'affectation (the generalized assignment problem) [74, 189], du problème d'affectation quadratique (the quadratic assignment problem) [7], du problème de tournées de véhicules (vehicle routing problems) [110, 155],du problème de voyageur de commerce(traveling salesman problem) [57, 64], du problème d'arbre couvrant de poids minimal (minimum spanning tree problem) [3, 5],du problème de regroupement (clustering problems) [62], du problème de planification des machines parallèles (parallel machine scheduling problems) [2], du problème d'emploi du temps (timetabling problem) [132],du problème de planification des satellites (satellite scheduling) [117], problème de ramassage et livraison (pickup and delivery problem) [116] et du problème de la planification de chargement de fret aérien (air cargo loading planning) [115].

Dans la majorité des problèmes traités avec des méthodes de recherche VLSN, les résultats de calcul sont presque toujours les meilleurs connus, ce qui rend le développement de tels algorithmes pratiquement très souhaitables. La réussite de la conception d'un algorithme de recherche VLSN dépend du choix de la fonction de voisinage adéquate et

de l'élaboration d'une méthode heuristique efficace pour la recherche dans le voisinage et l'amélioration des solutions.

Les méthodes de recherche VLSN peuvent être hybridées également avec d'autres méthodes heuristiques et méta-heuristiques afin d'aboutir à des nouvelles solutions améliorer, telles que la recherche à voisinage variable (variable neighborhood search) [63], la recherche tabou (tabu search) [97, 102],la recherche locale itérée(Iterated local search) [178], et la recherche dispersée (scatter search) [158], Ahuja *et al.* [4, 6] ont présenté des techniques bien expliquées servant à aider le développement des algorithmes de recherche VLSN robustes et efficaces.

3.3 L'algorithme de propagation des plantes (PPA)

3.3.1 Introduction

La plupart des problèmes d'optimisation réels sont souvent difficiles à résoudre, surtout à l'optimalité, pour cela, de nombreux chercheurs s'orientent vers des méthodes méta-heuristiques en exploitant la similitude entre l'optimisation et les problèmes de la vie réelle. Ces méthodes simulent le comportement intelligent des êtres vivants pour résoudre des problèmes similaires existant dans la nature, L'algorithme de propagation des plantes (PPA) introduit par Salhi and Fraga [161] est l'une des plus récentes méta-heuristiques reconnues avec leur succès considérable pour la résolution des problèmes d'optimisation combinatoire complexes.

Le PPA a récemment été étudié dans un nombre appréciable d'articles de recherche, révélant les promesses de ce nouveau méta-heuristique. Voici quelques domaines d'application réussie de PPA : L'optimisation de l'ingénierie « Engineering Optimization » (Sulaiman *et al.* [175]), voyageur de commerce « Traveling Salesman » (Selamoglu and Salhi [163]), Répartition du chargement « Load Dispatch » (Nag [136]), planification de parcours « Path Planning » (Zhou *et al.* [192]) Planification de l'examen « Exam Scheduling » (Cheraitia *et al.* [52]). De plus, plusieurs modifications et améliorations ont été ajoutées au PPA (Sulaiman and Salhi [173, 174] et Sulaiman *et al.* [176]). Une étude récente plus détaillée sur les PPA est proposée dans Borzog-Haddad *et al.* [30], et les articles les plus récents traitant de cette méta-heuristique, en raison de (Geleijn *et*

al. [72] et Paauw and Van Den Berg [142]), sont instructifs. Cette section présente un aperçu général de la méta-heuristique (PPA), décrit sa simplicité conceptuelle et expose les détails de son implémentation.

3.3.2 Aperçu général du PPA

L'algorithme de propagation des plantes (PPA), illustrée par l'algorithme de fraisier, est un algorithme de recherche dans le voisinage inspiré de la nature, c'est une méta-heuristique basée sur la population. Il appartient à une famille d'algorithmes nommés « algorithme inspirés par les plantes» « Plant-inspired algorithms » (Brabazon *et al.* [32]). ou «algorithme basés sur l'intelligence végétale» « Plant intelligence based algorithms » (Akyol and Alatas [10]) Les plantes, si nous les examinons attentivement, doivent survivre et se reproduire. Tous ont une stratégie de propagation sous-jacente qui montre ce qui peut être appelé comportement intelligent. La plupart des plantes se propagent à travers les graines. Par contre le fraisier se reproduire par des stolons (voir Figure 3.5). Un stolon est une branche (rhizome), provenant de la plante-mère, qui pousse sur le sol, lorsqu'elle touche le sol, elle produit des racines qui donnent naissance à une autre plante.

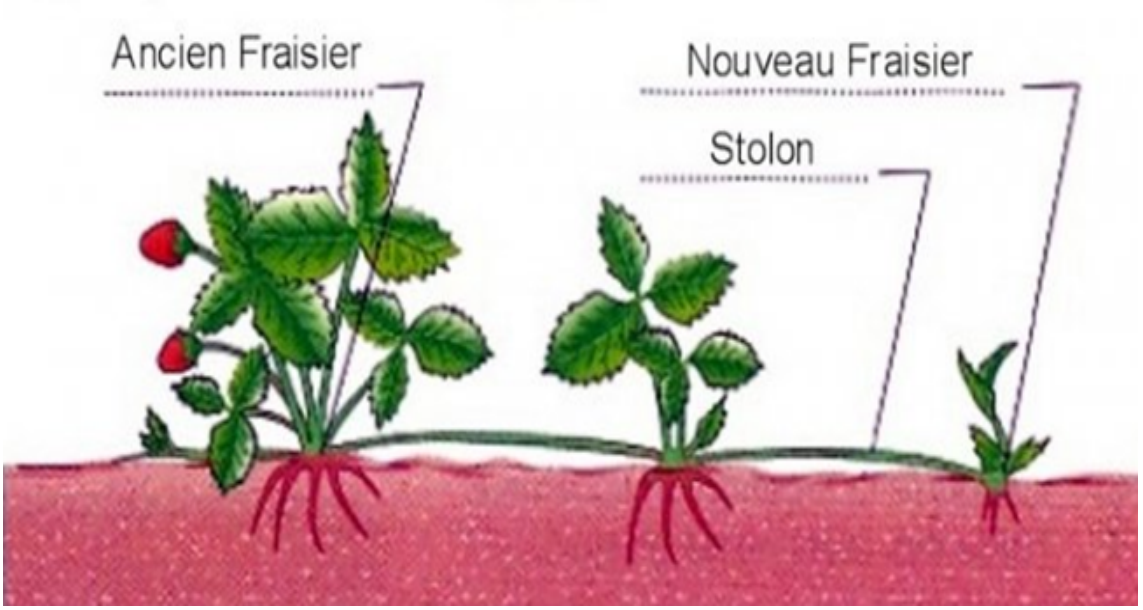

FIGURE 3.5: Reproduction végétative du fraisier

3.3.3 Principe fondamentale du PPA

Certaines plantes, telles que le fraisier, utilisent leurs stolons pour explorer le voisinage où elles se trouvent afin de trouver de bons endroits pour pousser et se propager. Bien que les stolons soient envoyés dans toutes les directions, il y a une orientation vers les bons endroits, c'est-à-dire, les zones où l'éclairage et l'humidité sont les plus élevés. Ce phénomène est dû à ce que l'on appelle, en langage végétal, le tropisme ou la réponse de croissance aux stimuli [75]. En outre, Il est donc supposé que le fraisier comme d'autres plantes, a une stratégie de propagation sous-jacente. Cette stratégie est développée au fil du temps pour assurer la pérennité de l'espèce. En résumé, une plante tente d'avoir sa progéniture dans des endroits qui sont suffisamment fertiles et qui offrent le plus important de nutriments pour sa croissance. Pour augmenter ses chances de survie, le fraisier réalise cette stratégie de base :

— Si la plante-mère se trouve à un bon endroit, elle génère le plus grand nombre possible de stolons de courte distance dans son voisinage pour donner à sa progéniture la chance de profiter pleinement des conditions avantageuses.

— Au contraire, si elle est dans un endroit pauvre, elle ne s'intéresse pas à garder ses descendants trop près. Au lieu de cela, elle envoie de longs stolons pour explorer les zones les plus loin à la recherche de conditions plus appropriées. En outre, seuls quelques stolons sont envoyés puisque les ressources dont ils disposent sont limitées et que la recherche de meilleures conditions n'est pas assurée de réussir.

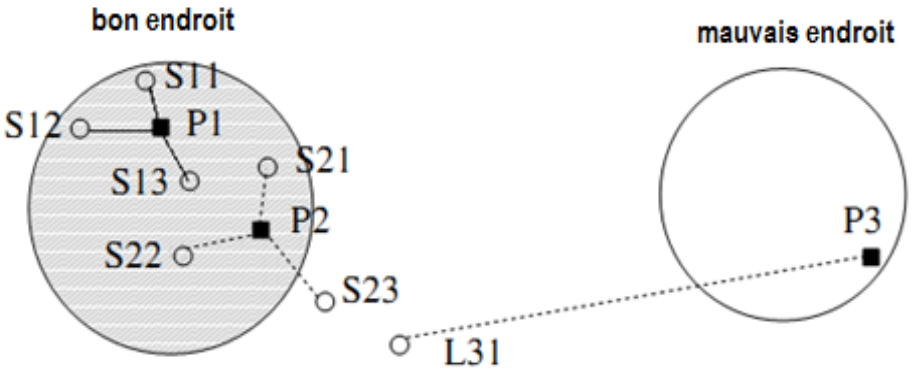

FIGURE 3.6: Stratégie de propagation du fraisier

Le PPA imite cette stratégie illustrée à la figure 3.6. Comme la plante-mère $P1$ est dans un bon endroit, elle envoie trois petits stolons donnant naissance à trois descendants

$S11$, $S12$ et $S13$. De sa part, la plante-mère $P3$ est dans un mauvais endroit (une terre aride), par conséquent, elle n'envoie qu'un seul long stolon $L31$ à la recherche de meilleures conditions.

Le principe sous-jacent de base est énoncé dans un pseudo-code, présenté dans l'algorithme 3.1. Le PPA est lancé avec une population de plantes répartie aléatoirement dans l'espace de recherche (ligne 1 de l'algorithme 3.1). Le nombre $\%bon$ (ligne 4) est le pourcentage de plantes considérées comme vivant dans de bons endroits. Le tri dans les lignes 2 et 7 est essentiel pour distinguer les plantes S dans les bons endroits et les plants $N-S$ dans les moins bonnes places. La ligne 8 de l'algorithme 3.1 n'accepte qu'un pourcentage des plantes générées et de leurs parents. Cette sélection est nécessaire pour maintenir la population avec une taille maniable.

Comme toute méta-heuristique, le PPA demande la personnalisation d'un ensemble de paramètres. Des décisions génériques doivent être prises pour certaines d'entre elles : la taille de la population N, la proportion de bonnes plantes considérées comme étant bien placées, le nombre r de stolons courts envoyait de bons parents et le nombre R de stolons longs à envoyer des parents dans les mauvais endroit, $r > R$. En revanche, il existe des paramètres qui nécessitent des décisions spécifiques à un problème : la "distance" parcourue par le stolon, qu'elle soit courte ou longue, et le nombre de plantes à rejeter (Ligne 8) selon un critère d'acceptation.

Algorithme 3.1 Pseudo-code de base pour le PPA

1. Générer la population initiale $p = [p_1, ..., p_N]$.
2. Ordonner p par ordre croissant en fonction de la fonction objective
3. **Répété**
4. $S = \%bon * N$
5. **Pour** $i = 1, ..., S$ la plante p_i génère r stolons courts.
6. **Pour** $i = S + 1, ..., N$ la plante p_i génère R stolons courts.
7. Ordonner la nouvelle population (plus large).
8. Sélectionner N plantes de la nouvelle population selon un critère d'acceptation.
9. **Jusqu'au critère d'arrêt.**
10. **Retourner** p_1**.**

3.3.4 PPA dans le contexte des problèmes d'optimisation

Étant donné l'espace fini de solution S et la fonction f, qui consiste à associer chaque solution s au nombre réel $f(s)$, la stratégie ci-dessus du fraisier peut être implémenté

dans le PPA pour résoudre le problème de l'optimisation combinatoire (COP) suivant : trouver $s* \in S$ de telle sorte que $f(s*) = min_{s \in S} f(s)$.

Les plantes en PPA correspondent à la solution s en COP, et l'aptitude (fitness) de l'individu s à la fonction objective $f(s)$. Le fitness est représenté par le concept de la qualité du spot. Le changement d'état correspond au concept d'envoi d'un stolon, tandis que la notion de perturbation ou de déplacement est illustrée par la distance qu'un stolon a parcourue depuis sa plante-mère. Tandis qu'un stolon court est atteint de son parent par une petite perturbation, un stolon long est obtenu par un changement substantiel. La motivation, lorsqu'on la traduit en matière de COP, est qu'une bonne solution peut-être atteinte soit à partir d'une autre bonne solution seulement par une petite perturbation, soit à partir d'une mauvaise mais par une modification substantielle.

L'intensification et la diversification sont les deux éléments fondamentaux de toute méta-heuristique efficace. Le plus important est d'établir un équilibre entre eux, puisqu'ils se comportent en sens inverse. En effet, la phase d'intensification se focalise sur la recherche au voisinage des optima locaux, et la phase de diversification a pour but de permettre à la recherche d'échapper de la zone d'attraction d'optima locaux. Ces deux concepts, ont une importance cruciale dans le PPA. L'intensification est réalisée par l'envoi de stolons courts à partir des plantes mères qui se trouvent dans des lieux profitables : c'est-à-dire qu'un stolon court exploite l'espace de recherche à proximité de sa mère à la recherche d'un optimum local. La diversification est implémentée par l'envoi des longs stolons à partir des plantes mères situées dans des endroits inadéquats : c'est-à-dire qu'un long stolon explore l'espace de recherche loin de ses parents afin d'échapper de l'optimum local. La stratégie de survie du fraisier dans son environnement peut donc être considérée comme celle de la recherche de points dans l'espace de solution qui aboutit finalement à des meilleures valeurs de fitness.

En effet, le PPA généralise la recherche locale itérée [120]. Pratiquement, lorsque la population se réduit à un individu (N = 1) et que le descendant, au lieu de son parent, est toujours sélectionné pour être le parent de la génération suivante (qui se réduit à une seule plante), le PPA se réduit clairement à la recherche locale itérée.

Chapitre 4

Une heuristique hybride d'optimisation de sous-gradient et de la recherche du voisinage à très grand échelle pour le NRP

4.1 Introduction

Le problème de rotation des infirmières nécessite la production d'un planning périodique pour un ensemble d'infirmières soumises à un ensemble donné d'exigences. Le NRP, dont le modèle mathématique est donné à la section 2.3.1, est un problème d'emploi du temps particulier dont le but est d'assigner des périodes de travail (shifts) à un ensemble donné d'infirmières pour satisfaire un ensemble de contraintes prédéfini. Les NRPs varient considérablement d'un établissement à l'autre, en raison d'un large éventail de contraintes spécifiques (les demandes des infirmières, politique hospitalière, réglementation, etc...). Le NRP est intraitable sur le plan informatique Osogami and Imai [140]. Malgré cela, il a reçu une attention considérable au fil des ans.

Le temps de calcul nécessaire pour obtenir une solution optimale au NRP augmente exponentiellement avec la taille du problème. Contrairement aux méthodes exactes, les heuristiques et les méta-heuristiques sont capables de trouver des solutions proches de

l'optimale dans des temps de calcul tolérable. La plupart des articles de recherche proposent de telles approches approximatives.

Dans ce travail, nous suivons la même ligne de résolution en proposant une méta-heuristique rapide capable de générer rapidement de bonnes solutions. La méthode que nous présentons est une méta-heuristique d'amélioration. On commence par une solution initiale calculée en utilisant Cplex12.6 (ILOG, Inc [98]) un solveur à usage général. Ce dernier, appliqué au NRP, s'arrête dès qu'une première solution réalisable est trouvée. C'est une procédure rapide pour avoir une solution initiale.

La méthode proposée hybride l'optimisation de sous-gradients et la recherche du voisinage à très grande échelle (VLSN). L'hybridation est aujourd'hui largement utilisée dans le contexte du NRP (voir par exemple les références suivantes : Awadallah *et al.* [15, 16],Rahimian *et al.* [153]. L'algorithme est hybride dans le sens qu'il est basé sur la recherche itérative de VLSN, dont la définition est guidée par la méthode de sous-gradient appliquée à la relaxation lagrangienne des contraintes de couverture. Plus précisément, à chaque itération de la méthode de sous-gradient, les coûts réduits sont utilisés pour définir un voisinage de la meilleure solution réalisable courante. La taille du voisinage est $O(2^n)$, où n est le nombre d'infirmières. La recherche dans le voisinage exponentiel revient à résoudre un petit PB avec n variables et p contraintes, où p est le nombre total de shifts dans l'horizon sélectionné. Cplex est utilisé à cette fin. Nous verrons que NRP est un PB avec $m \times n$ variables et $n+p$ contraintes. Puisque le nombre de shifts patterns m peut être exponentiel dans la taille de l'instance, la taille de NRP peut être très grande. Par conséquent, le principal avantage de notre approche est que la taille du PB, résolue dans chaque itération de la méthode de sous-graduation pour trouver un optimum local dans le voisinage exponentiel, ne dépend pas du nombre de shifts patterns. Bien que le voisinage soit exponentiel, cette façon d'exploration de cela peut être considérée comme efficace. Un inconvénient de notre méthode est qu'elle s'appuie sur un solveur (Cplex dans notre cas). Un tel solveur est nécessaire, comme une boîte noire, pour résoudre les PBs successifs. Notre méthode est testée sur l'ensemble de données divers de NSPLib Vanhoucke and Maenhout [186] et est comparée avec trois algorithmes existants Constantino *et al.* [53], Maenhout and Vanhoucke [122, 124], Ce sont les seules méthodes que l'on sait traitant l'ensemble de données NSPLib. La comparaison d'un point de vue de la précision, ainsi que de la vitesse, tourne à l'avantage de notre algorithme.

Ce chapitre est organisé comme suit. La section 4.2 décrit les techniques de recherche de VLSN. La section 4.3 présente notre approche et détails la méthode proposée. La section 4.4 est consacrée à l'expérimentation et à la comparaison.

4.2 Un aperçu sur la recherche locale et VLSN

Considérons un problème d'optimisation combinatoire avec l'espace de solution S et la fonction objective $g : S \to \mathbb{R}$.

Un élément $s \in S$ est une solution. Le voisinage d'une solution donnée s est l'ensemble $N(s)$ de toutes les solutions qui peuvent être obtenues à partir de s par une petite "perturbation" (la définition précise d'un voisinage est spécifique au problème).

Étant donné $s \in S$, n'importe quel $s' \in N(s)$) est appelé voisin de s, et une transition de s à s' est appelée mouvement.

La recherche locale (LS) est la méthode d'amélioration la plus naturelle. Il commence par une solution initiale $s_0 \in S$ que l'on considère temporairement comme la meilleure solution courante ($s^* = s_0$). Dans toute itération, la meilleure solution s^* est examinée pour l'amélioration, en considérant uniquement les solutions qui se trouvent dans son voisinage.

S'il existe $t \in N(s^*)$ telle que $g(t) < g(s^*)$ alors nous obtenons une amélioration ; La solution t devient la meilleure courante ($s^* = t$), et la méthode réitère. Sinon $g(t) \geq g(s^*)$ pour tout $s \in N(s^*)$ et la LS se termine. L'inconvénient de cette stratégie qui consistant d'accepter uniquement les mouvements d'amélioration est qu'elle se termine piégée dans un minimum local en raison de la nature myope de la recherche. Plusieurs stratégies d'amélioration existent (meilleure, aléatoire, glouton). La question cruciale dans la recherche locale est la structure de voisinage.

La recherche VLSN représente la recherche où la taille de voisinage s'augmente d'une façon exponentielle avec la taille du problème.

En peut trouver un bon tutoriel sur la VLSN dans Pisinger and Ropke [148], et un Survey très compréhensive dans Ergun [65, chap. 1].

la communauté de la recherche montre un intérêt croissant de la recherche VLSN [60, 70, 79]. La propriété importante de la taille exponentielle du voisinage est la possibilité de produire un bon optimum local. Cependant, une contrepartie est qu'il prend du temps beaucoup plus long à les chercher. La recherche VLSN est attractive quand le voisinage peut être cherché efficacement.

4.3 Méthodologie de la solution

La section 4.3.1 montre comment la solution initiale faisable est obtenue, la section 4.3.2 rappelle à quelques faits basiques de la relaxation Lagrangienne et la méthode de sous gradients, la section 4.3.3 présente une restriction de NRP, en la appeler 2NRP qui est le problème central dans notre approche. La section 4.3.4 traite la structure du voisinage choisi, chercher le voisinage de taille exponentielle d'une solution faisable donnée s'avère d'être équivalent à résoudre un 2NRP faisable, appelé F2NRP. La section 4.3.5 montre comment transformer un F2NRP à un PB, la section 4.3.6 proposent l'algorithme de recherche VLSN, et finalement la section 4.3.7 présente un exemple illustratif.

4.3.1 Solution initiale

Au début, sans aucune solution à notre disposition, on doit construire une. Cplex 12.6 (ILOG, Inc [98]), un solveur commercial, est appliqué au programme binaire NRP pour identifier une solution initiale faisable (s'il existe). Le solveur est utilisé avec la règle suivante : trouvez une seule affectation possible et quittez. La solution initiale est de mauvaise qualité, mais elle est obtenue très rapidement et est suffisante pour lancer le processus de recherche VLSN. Un échec de Cplex signifie que le problème est vraiment infaisable et que l'instance est proclamée comme telle.

4.3.2 Relaxation lagrangienne et la méthode de sous-gradients

On associe un vecteur de multiplicateur Lagrangien $\pi \in \mathbb{R}^p$ pour la contrainte relaxée (2.3), le dual lagrangien de NRP est le problème $\max_{\pi \in \mathbb{R}^p} w(\pi)$ où $w(\pi)$ est la valeur optimale du problème $LR(\pi)$

$$
\begin{aligned}
w(\pi) &= -\sum_{k\in P}\pi_k b_k + \min \sum_{i\in M}\sum_{j\in N}\left(c_{ij} + \sum_{k\in P}\pi_k a_{ik}\right)x_{ij} \\
&\sum_{i\in M} x_{ij} = 1 && j \in N \\
&x_{ij} \in \{0,1\} && i \in M,\quad j \in N
\end{aligned}
$$

Nous savons que $w(\pi)$ est une borne inférieure de la valeur optimale du NRP, pour tout $\pi \in \mathbb{R}^p$. De ce fait, la solution optimale du dual lagrangien fournit la meilleure borne inférieure. Mais trouver le vecteur Lagrangien optimal π^* est une tâche difficile. Heureusement, il existe une méthode rapide et très populaire pour trouver une solution approchée de la solution optimale.

En commençant avec $\pi^{(0)} = 0$, la méthode du sous-gradient génère une séquence $\pi^{(t)}$, $t \geq 1$, et prend $\pi^* = \pi(t^*)$ où $t^* = \text{argmax}_{t\geq 0} w(\pi^{(t)})$. Il est bien connu que $w(\pi^*)$ est égal à la valeur optimale de la relaxation_PL de NRP.

Dans notre approche, on n'a jamais intéressé à la qualité de la borne inférieure, puisque le seul but de la méthode de sous-gradient est de guider la recherche VLSN.

Pour $\pi^{(t)}$ fixe, le problème $LR\left(\pi^{(t)}\right)$ est facilement calculé : pour tout j, $j \in N$, calculer :

$$i_j = \text{argmin}_{i\in M}\left\{c_{ij} + \sum_{k\in P}\pi_k^{(t)} a_{ik}\right\} \tag{1}$$

Et mettre

$$x_{i_j j}^{(t)} = 1 \tag{2}$$

La valeur $w\left(\pi^{(t)}\right)$ est calculée en conséquence.

Il est bien connu que les composantes du sous-gradient $\sigma\left(\pi^{(t)}\right)$ sont :

$$\sigma_k\left(\pi^{(t)}\right) = \sum_{i\in M}\sum_{j\in N} a_{ik}x_{ij}^{(t)} - b_k, \quad k \in P \tag{3}$$

Et les composantes du vecteur lagrangien sont actualisées itérativement

$$\pi_k^{(t+1)} = \max\left\{0, \pi_k^{(t)} + \rho\frac{ub - w\left(\pi^{(t)}\right)}{\left\|\sigma\left(\pi^{(t)}\right)\right\|^2}\sigma_i\left(\pi^{(t)}\right)\right\}, k \in P \tag{4}$$

Où ρ est un paramètre satisfaisant $0 < \rho \leq 2$ et $\rho \longrightarrow \infty$ comme $t \longrightarrow \infty$. La valeur ub est une surestimation de la valeur optimale de NRP (fréquemment, la valeur ub est fixée au coût de la meilleure solution courante), et la valeur

$$\rho \frac{ub - w\left(\pi^{(t)}\right)}{\left\|\sigma\left(\pi^{(t)}\right)\right\|^2}$$

est la taille de pas dans la direction du sous-gradient.

4.3.3 Une restriction du NRP

Considérez la restriction de NRP, appelé 2NRP, où chaque infirmière est autorisée seulement à choisir entre deux shifts patterns différents.

Soit r_j et s_j les deux modèles de shifts patterns qui peuvent être assignés à l'infirmière j, avec $r_j, s_j \in M$ et $r_j \neq s_j$. Pour i fixé, $i \in M$, soit $J_i^{(r)} = \{j \in N | r_j = i\}$ et $j_i^{(s)} = \{j \in N | s_j = i\}$. $J_i^{(r)} \cup J_i^{(s)}$ est l'ensemble de toutes les infirmières qui peuvent choisir le shift pattern i. Notez que cet ensemble peut être vide (exe : le shift pattern i n'est jamais affecté). Soit S l'ensemble des shifts patterns faisables, $S = M - \{i | J_i^{(r)} \cup J_i^{(s)} = \emptyset\}$.

Puisque 2NRP ne se compose que de $2n$ variables, soient les $y_j, z_j, j \in N$. Le modèle mathématique de 2NRP est

$$\min \sum_{j \in N} c_{r_j j} y_j + \sum_{j \in N} c_{s_j j} z_j$$

$$\sum_{i \in S} a_{ik} \left(\sum_{j \in J_i^{(r)}} y_j + \sum_{j \in J_i^{(s)}} z_j \right) \geq b_k, \qquad k \in P \tag{5}$$

$$y_j + z_j = 1, \qquad j \in N \tag{6}$$

$$y_j, z_j \in \{0, 1\}, \qquad j \in N \tag{7}$$

4.3.4 Voisinages de taille exponentielle

Soit $r = (r_1, \cdots, r_n)$ une solution réalisable, où $r_j = i$ est équivaut à $x_{ij} = 1$ et signifie que le shift pattern i est attribué à l'infirmière j.

Pour définir le voisinage $N(r)$ de r, on commence tout d'abord de choisir pour chaque infirmière j un autre shift pattern appelé s_j, avec $s_j \in M, s_j \neq r_j$.

Le voisinage de r est l'ensemble de toutes les solutions réalisables qui peuvent être obtenues à partir de r en remplaçant quelques shifts patterns r_j par shifts patterns s_j, à condition que les shifts patterns soient réalisables.

Étant donné que chaque infirmière j a deux possibilités, soit garder le shift pattern r_j ou réassigner à shift pattern s_j, la taille du voisinage est $O(2^n)$.

Il est clair que le voisinage $N(r)$ est la région réalisable définie par les contraintes (5-7), et le meilleur voisin (optimum local) peut être obtenu directement en résolvant le problème, appelé F2NRP :

$$\min \sum_{j \in N} c_{r_j j} y_j + \sum_{j \in N} c_{s_j j} z_j \quad \text{st} \quad (5-7).$$

Alors que 2NRP est défini avec r_j et s_j arbitraire, F2NRP est légèrement plus facile puisque nous savons une solution réalisable $r = (r_1, \cdots, r_n)$ i.e. une solution faisable $y_j, z_j, j \in N$ satisfaisant(5-7). Plus précisément, nous savons que si nous fixons $y_j = 1, z_j = 0, j \in N$, nous obtenons une solution faisable dont le coût est $c^* = \sum_{j \in N} c_{r_j j}$. Par conséquent, le coût d'une solution optimale de F2NRP n'est pas supérieur à c^*. Rappelons que $y_j = 1$ indique que le shift pattern r_j est affecté à l'infirmière j, tandis que $y_j = 0$ (ce qui est équivalent à $z_j = 1$) signifie que le shift pattern r_j est retiré de l'infirmière j qui reçoit le shift pattern s_j.

4.3.5 Transformation de F2NRP en PB

Soit $z_j = 1 - y_j, j \in N$, F2NRP se transforme clairement en PB avec n variables et p contraintes :

$$\sum_{j \in N} c_{s_j j} + \min \sum_{j \in N} \left(c_{r_j j} - c_{s_j j} \right) y_j \tag{8}$$

$$\sum_{i \in S} a_{ik} \left(\sum_{j \in J_i^{(r)}} y_j - \sum_{j \in J_i^{(s)}} y_j \right) \geq b_k - \sum_{i \in S} a_{ik}, \qquad k \in P \tag{9}$$

$$y_j \in \{0, 1\}, \qquad j \in N \tag{10}$$

Rappelez-vous que $y_j = 1, j \in N$, est une solution réalisable triviale de PB. Ici $y_j = 1$ pour certains j signifié également que le shift pattern r_j est maintenu pour l'infirmière j, tandis que $y_j = 0$ signifie que l'infirmière j reçoit le shift pattern s_j.

4.3.6 Hybridation de l'optimisation de sous-gradient et la recherche VLSN

La recherche VLSN est guidée par la méthode du sous-gradient. Soit $(r_j)_{j\in N}$, la meilleure solution faisable courante. On calcule les coûts réduits à l'itération t

$$c_{ij} + \sum_{k\in P} \pi_k^{(t)} a_{ik}, \quad i \in M, \quad j \in N \tag{11}$$

et les coefficients :

$$s_j = \text{argmin}_{\{i\in M,\ i\neq r_j\}} \left\{ c_{ij} + \sum_{k\in P} \pi_k^{(t)} a_{ik} \right\}, \quad j \in N \tag{12}$$

et on définit le PB comme dans (8-10).

Par construction, le PB est faisable. Par conséquent, il doit avoir une solution réalisable dont le coût n'est pas moins bon que le coût de la meilleure solution faisable r courante.

On donne un pseudo-code pour la description de l'heuristique de recherche VLSN dans l'algorithme 4.1.

Un planning réalisable initial est construit dans la ligne 1. La boucle dans les lignes 4-15 effectue la méthode de sous-gradient appliquée à la relaxation Lagrangienne des contraintes de couverture (2.3) du NRP. La définition et la résolution des PBs sont incorporées dans les lignes 5-7.

Le calcul d'une solution initial exige, comme déjà indiqué, $O(n\times m\times \max\{n,p\})$ opérations.

Le calcul des coûts réduits dans la Ligne 5 nécessite $O(m \times n \times p)$. Le calcul dans Ligne 6 demande $O(n \times p)$ opérations. La mise à jour à la Ligne 8 est insignifiante.

Résoudre le problème relaxé à la ligne 9, le calcul de $w(\pi^{(t)})$ à la ligne 10, et le sous gradient à la ligne 11,$O(m \times n \times p)$ chacun.

La mise à jour du vecteur Lagrangien dans la Ligne 12 nécessite $O(p)$ opérations une fois que le sous-gradient est connu. Les lignes 13 et 14 valent respectivement $O(p)$ et $O(1)$. Par conséquent, s'il ya une charge de calcul, il doit se produire dans la ligne 7, qui s'exécute $NBIT$ fois.

Heureusement, un PB NP-difficile a une très petite taille (n variables et p contraintes) par rapport à NRP qui a $m+p$ contraintes et $m \times n$ variables.

En fait, le PB n'a que des contraintes $p-d$ car les contraintes de couverture avec $b_k = 0$ sont valides et peuvent être omises. Par conséquent, la résolution du PB dans la ligne 7 peut être considérée comme un moyen efficace pour fouiller dans le voisinage de taille exponentielle.

Un nombre maximal d'itérations $NBIT$ est autorisé pour contrôler la boucle exécutant la méthode de sous-gradient. Le coefficient de relaxation ρ commence par la valeur 2 est réduit de moitié à chaque nombre fixe d'itérations (50 dans notre implémentation). Tous les faits discutés dans cette section tendent à conclure que notre approche est une méthode rapide pour aborder le NRP.

Algorithme 4.1 La recherche VLSN

entrée les donnée de NRP
sortie meilleur solution trouvé jusqu'à présent $(r_1...r_n)$ et leur Coût ub
Initialisation
1. Calculer une solution initiale faisable$(r_1...r_n)$ et son Coût ub
2. $\rho \leftarrow 2$
3. $\pi^{(0)} \leftarrow 0$

Optimisation avec sous-gradient
4. **Pour** $t = 0, ..., NBIT$ **faire**
5. Calculer le cout réduit comme dans (11)
6. Calculer $(s_1, ..., s_n)$ comme dans (12)
7. Résoudre PB // PB est défini comme dans (8-10)
8. Mettre à jour $(r_1...r_n)$ et ub si une amélioration est trouvée
9. Résoudre le problème relaxé comme dans (1-2) // soit $x_{ij}^{(t)}$ est la solution du problème relaxé $LR(\pi^{(t)})$)
10. Calculer la valeur optimal $w(\pi^{(t)})$
11. Calculer les composants de sous gradient comme dans (3)
12. Mettre à jour le vecteur lagrangien comme dans (4)
13. **Pour** $k \in P$ **faire**
14. **si** $\pi_k^{(t+1)} > 0$ **alors** $\pi_k^{(t+1)} \leftarrow 0$
15. Ajuster ρ si nécessaire
16. **Fin pour**

4.3.7 Un exemple illustratif

La méthode proposée est illustrée avec $m = 13$ shifts patterns, $n = 4$ infirmières, $d = 4$ jours, et $s = 4$ shifts par jour (le quatrième étant le jour libre). Par conséquent, il y a $P = d \times s = 16$ shifts au total. Le vecteur de couverture $b = (2, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0)$ donne pour chaque $k \in P = 1, ..., 16$ le nombre b_k d'infirmières requis pour le shift k. Les 13 shifts patterns sont donnés dans la matrice suivante.

$$a_{ik} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Le premier shift pattern (première ligne) indique que l'infirmière qui l'obtient est libre le premier jour, travaille le premier shift du deuxième jour, le premier shift du troisième jour et le deuxième shift du quatrième jour. Les coûts totaux de préférence sont donnés dans la matrice suivante.

$$c_{ij} = \begin{pmatrix} 10 & 11 & 11 & 12 \\ 8 & 11 & 8 & 13 \\ 10 & 12 & 9 & 13 \\ 11 & 9 & 13 & 12 \\ 9 & 9 & 10 & 13 \\ 11 & 10 & 11 & 13 \\ 9 & 8 & 13 & 11 \\ 11 & 9 & 14 & 11 \\ 7 & 11 & 10 & 13 \\ 7 & 10 & 13 & 11 \\ 6 & 9 & 12 & 14 \\ 6 & 8 & 15 & 12 \\ 8 & 9 & 16 & 10 \end{pmatrix}$$

La première colonne, par exemple, donne les coûts de préférence pour les shifts patterns de l'infirmière 1. Par conséquent, cette dernière préfère les shifts patterns 11 et 12.

NRP est un PB avec $m \times n = 52$ variables et $P = 16$ contraintes. Cependant, 7 contraintes sont valides (avec $b_k = 0$) et peuvent être omises. Nous commençons par l'application de Cplex à NRP avec la stratégie suggérée : trouver la première solution possible et arrêter.

La solution initiale obtenue est $r = (2, 13, 1, 7)$, ce qui signifie que l'infirmière 1 reçoit le shift pattern 2, l'infirmière 2 le shift pattern 13, et ainsi de suite. Le coût total de cette solution est de 39.

Dans la première itération de la méthode de sous-gradient, nous avons $\pi = 0$. Ainsi, les coûts réduits dans (11) sont les coûts originaux c_{ij}. Maintenant, nous devons calculer les shifts patterns s_j, $j \in N = 1, ..., 4$. Pour l'infirmière 1, par exemple, s_1 correspond au shift pattern $i \neq r_1 = 2$. Nous trouvons $s = 11, 7, 2, 13$. Ainsi, dans le problème F2NRP, l'infirmière 1 a le choix entre le shift pattern 2 et 11, l'infirmière 2 doit choisir entre le shift pattern 13 et 7, et ainsi de suite. Le problème F2NRP est converti en PB suivant avec 4 variables et $p = 16$ contraintes, sauf les redondantes qui sont supprimées.

$$
\begin{array}{lrrrrrr}
\min & 2y_1 & +y_2 & +3y_3 & +y_4 & & \\
 & & -y_2 & -y_3 & +y_4 & \geq & -1 \\
 & & y_2 & & -y_4 & \geq & 0 \\
 & & & y_3 & & \geq & 1 \\
 & y_1 & -y_2 & -y_3 & +y_4 & \geq & -1 \\
 & -y_1 & +y_2 & & -y_4 & \geq & -1 \\
 & & -y_2 & & +y_4 & \geq & 0 \\
 & -y_1 & +y_2 & & -y_4 & \geq & -2 \\
 & y_1, & y_2, & y_3, & y_4 & \in & \{0,1\}
\end{array}
$$

Sa solution optimale est $x = (0, 0, 1, 0)$ avec la valeur 3. Rappelons que $y_j = 1$ signifie que l'infirmière j retient le shift pattern r_j, alors que $y_j = 0$ indique que le shift pattern r_j est retiré d'elle et elle obtient s_j à sa place. La nouvelle solution est donc $r = (11, 7, 1, 13)$. Puisque $\sum_{j \in N} c_{sj} j = c_{11,1} + c_{7,2} + c_{2,3} + c_{13,4} = 6 + 8 + 8 + 10 = 32$, le coût de la solution r est de $32 + 3 = 35$. Pour l'achèvement, une solution optimale $r = (12, 4, 2, 13)$ de coût 33 est obtenue à l'itération 3.

4.4 Expérimentation numérique

Dans cette section, des expériences sont effectuées afin d'évaluer la performance de notre approche par des tests empiriques. Le plus important ici est la qualité des solutions et l'effort de calcul. Avant de tester, nous pouvons affirmer que la méthode proposée est simple, innovante et facile à mettre en œuvre (voir l'algorithme 4.1). elle est également parfaitement déterministe et reproductible. Pour cette raison, une seule exécution est autorisée pour chaque instance de l'ensemble de données.

La section 4.4.1 présente la configuration expérimentale. La section 4.4.2 propose les détails de l'implémentation. La section 4.4.3 présente les résultats obtenus et compare notre approche avec trois méta-heuristiques existantes récentes.

4.4.1 Configuration expérimentale

La méthode est codée en C et exécutée sur un HP Intel Core i5, 2.4GHz. elle est testé sur les instances NRP de NSPLib (pour plus de détails sur ces benchmarks voir [186]).

Il existe deux ensembles de tests. Le premier, l'ensemble divers, contient quatre sous-ensembles avec $n = 25$, 50, 75, et 100 infirmières respectivement. Tous les quatre ensembles ont été étendus par 8 combinaisons de contraintes spécifiques qui apparaissent fréquemment dans la littérature.

Pour chacun des quatre sous-ensembles et pour chaque type de contraintes spécifiques à chaque cas, il ya $7,290$ instances NRP Donc, il y a $4 \times 8 \times 7,290 = 233,280$ occurrences dans l'ensemble divers.

Les préférences des infirmières et les contraintes de couverture sont caractérisées par des niveaux systématiquement variés d'indicateurs de complexité NRP divers [186].

Dans cette base de données, le nombre de shifts est $s = 4$ y compris le shift libre la périodicité considérée ici est $d = 7$ jours, donc $p = 28$.

Le deuxième ensemble, appelé réaliste, est omis pour l'instant puisque la période considérée est $d = 28$ jours résultant en un grand nombre m de shifts patterns faisables.

4.4.2 Détails de l'implémentation

Le nombre d'itérations $NBIT$ est le seul paramètre qui doit être défini, dans notre cas il est fixé à 200. On a trouvé que ce nombre est suffisant puisque le grand nombre d'itération mène seulement à des temps de calcul plus importants.

Les PBs successifs sont résolus en utilisant Cplex (ligne 7 de l'algorithme 4.1). Rappelons que PBs ont n variables et $p = 28$ contraintes. En fait, nous pouvons ignorer les 7 contraintes liées aux shifts libres), pour que le nombre total des contraintes de PB soit 21. Même les PBs les plus importants avec $n = 100$ sont résolus en moins de 0.01 secondes.

4.4.3 Résultats et comparaison avec les méthodes existantes

Les résultats obtenus sont présentés dans le tableau 4.1. Les colonnes 3 et 4 se réfèrent respectivement au coût et au temps de calcul de la solution initiale. Les temps de calcul sont exprimés en secondes. Les autres colonnes concernent la meilleure solution trouvée. La colonne 6 se réfère à l'itération à laquelle la meilleure solution a été obtenue. Bien que le nombre d'itérations permises soit 200, on constate que la meilleure solution est obtenue très tôt en moyenne. La dernière colonne indique le temps de calcul total.

Observez que le coût initial de la solution s'écarte d'environ 6% du meilleur. Par conséquent, la solution initiale peut être considérée comme mauvaise. Ce fait tend à démontrer la robustesse de la méthode proposée puisque celle-ci ne dépend pas de la qualité de la solution initiale.

L'approche proposée est comparée avec trois méthodes récentes (voir le tableau 4.2). À partir de la qualité des solutions et du point de vue du temps de calcul. Chacune des quatre méthodes effectue une seule exécution sur chaque instance. Ces méthodes sont l'état de l'art en ce qui concerne l'ensemble de données NSPLib. Nous ne connaissons aucune autre méthode traitant le même ensemble de données.

Le tableau 4.3 présente les résultats des quatre méthodes. La colonne étiquetée ≪Coût≫ fait référence à la valeur moyenne (plus de 7290 occurrences) de la fonction objective dans (2.1), et la colonne intitulée ≪% Feas≫ indique la proportion d'instances NRP qui sont faisables. Les meilleures valeurs sont en gras.

Du point de vue de la qualité de la solution, il semble que VLSN ni pas la meilleure sauf dans deux cas ($n = 25$, cas 6 et $n = 50$, cas 5). Mais du point de vue du taux de faisabilité, VLSN est toujours meilleur. mais ce n'est pas surprenant. En effet, nous appuyons sur Cplex pour trouver la solution initiale faisable. Donc,% Feas dans la dernière colonne du tableau 4.3 est le vrai taux de faisabilité.

D'après le tableau 4.4 qui propose de récapituler le tableau 4.3, en donnant les valeurs moyennes pour la comparaison, Clairement, que ce soit du point de vue de la précision ou le taux de faisabilité il apparaît que notre approche est la meilleure, il reste à considérer les temps de calcul moyens respectifs.

TABLE 4.1: Résultats (valeurs moyennes sur 7290 occurrences)

		Solution initiale		Meilleure solution		
n	Cas	Coût	Temps	Coût	Itération	Temps
25	1	265.00	0.10	250.43	27.76	0.65
	2	252.93	0.16	239.46	24.81	1.01
	3	280.14	0.06	265.72	26.98	0.61
	4	263.62	0.16	248.64	28.32	0.88
	5	277.72	0.05	263.16	33.17	0.58
	6	254.08	0.15	240.44	25.80	0.91
	7	292.83	0.01	279.21	30.41	0.55
	8	270.15	0.04	256.18	31.11	0.62
50	1	522.65	0.31	499.10	12.84	1.10
	2	500.30	0.43	477.66	11.26	1.64
	3	547.73	0.15	524.15	11.78	0.79
	4	520.40	0.55	495.78	14.20	1.67
	5	544.67	0.15	521.84	12.89	0.75
	6	502.78	0.43	479.82	11.65	1.63
	7	567.96	0.03	547.07	10.16	0.57
	8	529.73	0.12	507.17	11.60	0.73
75	1	792.26	0.45	755.75	12.92	1.41
	2	768.80	0.72	732.54	14.10	2.28
	3	829.19	0.22	793.78	11.06	0.95
	4	783.87	0.77	745.49	16.03	2.19
	5	827.23	0.20	792.52	13.22	0.92
	6	770.43	0.72	733.88	14.58	2.27
	7	863.87	0.04	833.79	10.68	0.65
	8	813.03	0.17	777.15	13.38	0.90
100	1	1281.14	0.73	1214.29	17.42	1.98
	2	1240.82	1.05	1173.87	18.19	3.03
	3	1350.10	0.36	1287.14	15.52	1.32
	4	1271.03	1.28	1201.19	20.01	3.10
	5	1327.53	0.33	1265.38	16.76	1.24
	6	1244.03	1.09	1176.61	18.58	3.09
	7	1387.44	0.06	1332.23	13.70	0.83
	8	1306.48	0.27	1242.86	17.34	1.21

TABLE 4.2: Méthodes en compétition

Label	Méthodologie	Référence
MV2006	Scatter search	[122]
MV2007	Electromagnetic-like	[124]
CAL2013	Linear assignment	[53]
VLSN	Our approach	[83]

TABLE 4.3: Résultats des quatre méthodes

		MV2006		MV2007		CAL2013		VLSN	
n	Cas	Coût	%Feas	Coût	%Feas	Coût	%Feas	Coût	%Feas
25	1	250.68	**88.27**	250.89	**88.27**	251.39	**88.27**	**250.43**	**88.27**
	2	239.45	**88.27**	239.61	**88.27**	239.69	**88.27**	**239.46**	**88.27**
	3	266.64	88.08	266.98	87.91	267.68	**88.09**	**265.72**	**88.09**
	4	248.71	**88.27**	248.97	**88.27**	249.09	**88.27**	**248.64**	**88.27**
	5	263.63	**85.88**	263.88	**85.83**	265.23	**85.88**	**263.16**	**85.88**
	6	**240.43**	**88.27**	240.61	**88.27**	240.64	**88.27**	240.44	**88.27**
	7	280.14	80.10	279.98	79.84	282.04	77.39	**279.21**	**80.18**
	8	256.66	85.60	256.88	85.53	257.50	85.49	**256.18**	**86.69**
50	1	500.30	**90.03**	500.27	89.90	500.02	**90.03**	**499.10**	**90.03**
	2	478.19	**90.03**	478.41	**90.03**	477.91	**90.03**	**477.66**	**90.03**
	3	526.48	89.66	526.51	89.26	525.69	89.77	**524.15**	**89.82**
	4	496.68	**90.03**	496.87	89.93	496.61	**90.03**	**495.78**	**90.03**
	5	523.42	85.25	**521.75**	74.43	523.64	**85.34**	521.84	**85.34**
	6	480.39	**90.03**	481.52	89.96	480.07	**90.03**	**479.82**	**90.03**
	7	549.48	78.29	548.67	77.72	549.28	76.46	**547.07**	**78.61**
	8	508.77	85.50	508.68	84.91	508.06	85.39	**507.17**	**85.65**
75	1	758.15	**88.70**	759.42	88.63	756.83	**88.70**	**755.75**	**88.70**
	2	733.58	**88.70**	734.37	88.68	732.80	**88.70**	**732.54**	**88.70**
	3	797.75	88.37	797.17	87.82	795.46	88.53	**793.78**	**88.57**
	4	749.09	**88.70**	747.79	88.66	746.00	**88.70**	**745.49**	**88.70**
	5	795.71	86.06	795.25	85.58	794.51	**86.09**	**792.52**	**86.09**
	6	735.03	**88.70**	735.46	88.68	734.13	**88.70**	**733.88**	**88.70**
	7	837.85	79.48	835.42	78.07	835.54	77.56	**833.88**	**79.93**
	8	780.40	85.78	779.84	84.99	778.14	85.76	**777.15**	**85.93**
100	1	1218.45	90.49	1218.90	90.22	1215.34	**90.53**	**1214.29**	**90.53**
	2	1176.08	90.51	1176.60	90.48	1174.09	**90.53**	**1173.87**	**90.53**
	3	1293.94	90.01	1293.09	89.19	1289.11	90.37	**1287.14**	**90.38**
	4	1204.34	90.48	1205.31	90.32	1201.74	**90.53**	**1201.19**	**90.53**
	5	1270.29	86.28	1269.53	85.89	1267.56	**86.54**	**1265.38**	**86.54**
	6	1178.97	90.51	1179.62	90.41	1176.83	**90.53**	**1176.61**	**90.53**
	7	1338.26	79.49	1335.57	78.49	1335.19	78.59	**1332.23**	**80.34**
	8	1247.93	86.53	1247.15	85.91	1243.78	86.74	**1242.86**	**86.94**

TABLE 4.4: Comparaison des quatre méthodes du point de vue de la précision

	MV2006		MV2007		CAL2013		VLSN	
n	Coût	%Feas	Coût	%Feas	Coût	%Feas	Coût	%Feas
25	255.79	86.59	255.97	86.52	256.66	86.24	**255.41**	**86.62**
50	507.96	87.35	507.84	85.77	507.66	87.14	**506.57**	**87.44**
75	773.20	86.81	773.09	86.39	771.68	86.59	**770.61**	**86.91**
100	1241.03	88.04	1240.72	87.56	1237.96	88.05	**1236.70**	**88.29**

Comme d'habitude, la comparaison des temps de calcul est plus problématique, puisque, différentes plateformes, langages et compilateurs sont utilisés (voir tableau 4.5). Les temps de calcul doivent être mis à l'échelle avant de pouvoir les comparer. Donc, nous devons savoir la performance de chaque machine en question.

Dongarra [61] étudie les performances d'un grand nombre d'ordinateurs à l'aide du logiciel d'algèbre linéaire. Bien que la base de données contienne des milliers d'ordinateurs, nous ne trouvons pas les trois machines, mais seulement les semblables. Le pic théorique de chaque machine est rapporté.

Ces ordres de grandeur approximatifs montrent que le SPA Toshiba10 est comparable à notre HP, alors qu'un Quad core Xeon est environ 2.67 fois plus rapide.

TABLE 4.5: Machines utilisées par les quatre méthodes

	Machine utilisée	La fréquence	Machine similaire trouvée dans la base de données	Pic théorique (Mflops)	Facteur d'échelle
MV2006	Toshiba SPA10	2.4 GHz	Intel Xeon 2.4 GHz	4800	1.00
MV2007	Idem				
CAL2013	Quad-Core Xeon (2 cores)	3.2 GHz	Fujitsu Siemens hpcLine (2 proc Intel Xeon 3.2GHz)	12800	2.67
VLSN	HP Intel Core i5	2.4 GHz	Intel Xeon 2.4 GHz	4800	1.00

Les temps de calcul sont donnés dans le tableau 4.6, où il apparaît, à partir des valeurs moyennes dans la dernière ligne, que notre méthode est environ dix fois plus rapide que les trois autres méthodes (surtout étant d'accord avec la conclusion de la section 4.3.6). Cette affirmation devrait cependant être prise avec prudence, car les temps de calcul signalés sont petits.

TABLE 4.6: Comparaison des quatre méthodes du point de vue du temps de calcul

	MV2006	MV2007	CAL2013		VLSN
n	Temps	Temps	Temps	Temps échelonné	Temps
25	2.00	2.07	0.72	1.92	0.73
50	5.79	4.52	2.83	7.56	1.11
75	10.99	12.22	6.83	18.24	1.45
100	22.58	17.76	13.63	36.39	1.98
Moy	10.34	9.14		16.03	**1.32**

4.5 Conclusion

NRP est l'un des problèmes les plus importants dans la planification et l'optimisation combinatoire. Une heuristique, basée sur la recherche itérative de VLSN est proposée

pour le résoudre. La construction des voisinages exponentiels successifs est intégrée dans le cadre d'une méthode de sous-gradient appliquée à la relaxation lagrangienne des contraintes de couverture.

La recherche dans le voisinage s'effectue de manière efficace puisqu'elle s'agit de résoudre un très petit PB. À notre connaissance, c'est la première fois que cette approche est présentée ; le voisinage exponentiel et la méthode proposée pour l'explorer sont les contributions originales ici.

L'approche a été testée sur l'énorme ensemble de diverses instances NRP d'NSPLib. Les résultats obtenus et la comparaison avec les heuristiques existantes montrent que notre contribution est significative, car la méthode proposée semble être la plus rapide et la plus précise. Par conséquent, Donc, nous pouvons conclure que notre approche constitue un complément à l'arsenal d'outils pour résoudre les NRPs.

Chapitre 5

Méthode simple, rapide et efficace à deux phases pour le NRP

5.1 Introduction

Dans la présente étude, nous proposons une méthode composée principalement de deux phases pour le problème de rotation des infirmières issue de la base de données NSPlib, dont le modèle mathématique est spécifié à la section 2.3, on a appliqué des idées très simples mais très utiles pour le résoudre, représenté ci-dessous.

La première phase représente une heuristique générique à fixation de variable (VFH). C'est générique dans le sens qu'il peut être appliqué à n'importe quel problème d'optimisation combinatoire avec des variables binaires.

VFH réduit le nombre d'infirmières, le nombre de shift-patterns, et élimine la plupart des variables binaires, en conséquence, un problème très petit et très réduit est obtenu, appelé RNRP. Bien que ce dernier soit une version très restreinte du NRP d'origine, les expériences menées montrent qu'une solution optimale au RNRP est toujours prolongée à une solution optimale ou très proche de l'optimale du NRP.

La deuxième phase consiste simplement à résoudre le RNRP en utilisant un solveur MIP à usage général. La méthode à deux phases proposée est testée sur l'ensemble de données NSPLIB et comparée face aux quatre dernières méthodes existantes, ainsi qu'à un logiciel commercial à usage général. (Cplex12.6) appliqué aux instances NRP

originales. Ce qui est intéressant, c'est que cette méthode simple à deux phases s'avère très rapide et efficace pour obtenir de solutions optimales ou très bonnes.

Ce qui suit motive l'approche proposée et nous a incités à poursuivre cette approche pour résoudre le problème. Une idée simple et évidente consiste à réduire la taille du problème avant de le soumettre à Cplxe, l'inconvénient de notre approche est qu'elle s'appuie sur un solveur commercial !, cependant, elle présente plusieurs avantages, comme suit :

- C'est simple et très facile à mettre en œuvre.
- Capable de fournir des solutions optimales (dans la plupart des cas), ou au moins des solutions de haute qualité.
- C'est très rapide.
- C'est sans paramètre.

Ce chapitre est organisé comme suit : la section 5.2 décrit la méthode proposée, la section 5.3 présente l'heuristique générique VFH, et la section 5.4 explique le problème réduit, la section 5.5 est consacrée aux résultats expérimentaux et aux comparaisons.

5.2 La méthode proposée

Notre contribution dans ce travail peut être résumée comme suit : éliminer les variables peu prometteuses de NRP et résoudre le problème réduit résultant avec Cplex. La section 5.3 détaille l'heuristique de fixation de variable (VFH) où la section 5.3.1 est consacrée à la création d'une solution initiale, la section 5.3.2 rappelle quelques notions de l'optimisation de sou-gradient, la section 5.3.3 montre la stratégie de fixation des variables, et la section 5.4 traite le problème réduit.

5.3 Heuristique de fixation de variables

Les méthodes de fixation de variables présentent un grand intérêt et jouent un rôle éminent dans l'optimisation combinatoire. Ils cherchent à réduire la taille du problème initial pour accélérer la mise en œuvre de la méthode de résolution.

Dans certains cas, des règles logiques sont utilisées. Voir, par exemple, [21] dans le contexte du problème de la couverture d"ensemble, où les colonnes et les lignes de la

matrice binaire de contraintes sont ignorées conformément aux règles de dominance des colonnes et d'inclusion de lignes.

Dans d'autres cas, la méthode de fixation est basée sur les coûts réduits, où une méthode basée sur la relaxation LP ou une optimisation de sous-gradient est utilisée. Voir, par exemple [150]. Dans tous les cas, le problème original et la version réduite sont mathématiquement équivalents.

Notre proposition est totalement différente puisque c'est purement heuristique. On ne garantit pas l'équivalence mathématique du NRP original et la version réduite appelée RNRP. D'où, si une solution optimale est demandée, la VFH proposé devrait être abandonné. Mais si nous cherchons une approximation, au lieu de résoudre la grande instance originale, nous préférons traiter le problème réduit plus petit et plus clairsemé.

De plus, la VFH proposé dans ce travail est générique et peut être appliqué à n'importe quel problème d'optimisation combinatoire avec des variables binaires.

Cette méthode a été utilisé avec succès pour le problème de la couverture d'ensemble Haddadi *et al.* [84] et pour le problème générale d'affectation Haddadi [88]. Dans ce travail aussi, les expériences numériques montrent que cette idée s'avère efficace.

Comme la méthode de fixation de variables proposée s'appuyer sur l'application d'une méthode de sous-gradient à la relaxation lagrangienne des contraintes de couverture, nous devons construire une solution initiale pour lancer la méthode avec (voir section 5.3.1), une heuristique gloutonne est conçue pour cette fin, en fait, nous n'avons pas besoin de la solution elle-même, mais juste son coût. Quelques faits de base de la méthode de sous-gradient sont rappelés à la section (section 5.3.2). La section 5.3.3 détaille la règle de fixation des variables en montrant comment les infirmières, les shift patterns, et les variables sont écartées de toute considération ultérieure, laissant une version clairsemée et condensée de l'instance NRP. La section 5.3.4 présente un petit exemple illustratif utile.

5.3.1 Construire une solution initiale

Une heuristique gloutonne construit une solution initiale. Si c'est irréalisable, une procédure de recherche locale essaie de la réparer. Dans la suite, nous représentons une solution à

NRP par n-vecteur s avec $s(j) \in M, j \in N$ tel que $S_j = i \Longleftrightarrow x_{ij} = 1$. Des pseudo-codes détaillés sont donnés pour rendre notre méthode reproductible.

5.3.1.1 Heuristique gloutonne

Au début, nous ne disposons pas d'une solution à notre disposition. Nous devons construire une. À cet effet, nous adoptons une heuristique simple, qui assigne d'une manière gloutonne les shifts patterns aux infirmières, un à la fois, sans violer les contraintes (2.2 et 2.3). Par conséquent, n itérations sont obligatoires. À chaque itération, chaque shift pattern est évalué pour sélectionner le plus adéquat en calculant le nombre de shifts de travail qui peuvent être couverts par un shift pattern i ce qu'on appelle un poids.

$$w_i = \sum_{k \in P} \max \{0, a_{ik} \times b_k\}, i \in M \tag{1}$$

Ensuite, nous calculons un ≪score ≫qui correspond à l'affectation du shift pattern i pour l'infirmière j

$$r_{ij} = \frac{c_{ij}}{w_i}, \quad i \in M, \quad j \in N \tag{2}$$

Ensuite, nous calculons la meilleure sélection de shift pattern/infirmière

$$r_{i^*j^*} = \min_{i,j} r_{ij} \tag{3}$$

Et en affecte le shift pattern i^* à l'infirmière j^* si la contrainte (2.3) est satisfaite. L'infirmière j^* est retirée de la liste et les données sont mises à jour. La raison est que nous espérons que l'infirmière j^* reçoit le shift pattern i^* couvrant le plus grand nombre de shifts demander, et en minimisant le coût de préférence. La méthode se termine lorsque chaque infirmière reçoit son shift pattern.

Notez que la solution obtenue en appliquant l'heuristique constructive gloutonne dépend de l'ordre d'affectation des infirmières dans le planning.

Clairement, cette méthode ne garantit pas toujours une solution faisable. En effet, pendant une certaine itération, il peut arriver que, pour toutes les infirmières restantes et tous les shifts patterns, nous ne pouvons pas satisfaire la contrainte (2.3). Si ce cas se

produit, nous essayons de le corriger dans la section suivante. le pseudo-code de l'heuristique gloutonne est donné dans l'algorithme 5.1.

Une fois qu'une solution faisable r est obtenue, son coût (valeur de la fonction objective dans (2.1)) est calculée.

proposition 1 : L'heuristique gloutonne s'exécute en temps $O(n \times m \times \max\{n, p\})$.
Dimenstration : Le calcul en (1) (qui correspond aux lignes 6-7 de l'algorithm 5.1) se déroule en temps $O(m \times p)$. Calculer la fonction de score dans (2) (Ligne 10 de l'Algorithme 5.1) et trouver la plus petite valeur de score dans (3) (Ligne 11 de l'Algorithme 5.1) a besoin de temps $O(m \times n)$ pour chacun. La preuve vient du fait que les trois étapes sont répétées n fois.

Algorithme 5.1 Heuristique gloutonne

entrée les donnée de NRP
sortie Planning $(s_1...s_n)$ et leur Coût ub
1. $ub \leftarrow 0$
2. $\beta_k \leftarrow b_k, k \in P$
3. $free_j \leftarrow 1, j \in N$ ▷ pour le moment, chaque infirmière est libre
4. $iter \leftarrow 0,$
5. **répéter** ▷Faites ceci n fois
6. **Pour** $i \in M$ **faire** ▷ Calculer un poids pour chaque shift pattern
7. $w_i \leftarrow \sum_{k \in P} \min\{0, a_{ik} \times \beta_k\}$
8. **si** $w_i = 0$ **alors**
9. $w_i = 1$ ▷Pour éviter la division par zéro
10. $score_{ij} \leftarrow c_{ij}/w_i, i \in M, j \in N$
11. Trouver la valeur minimale $score_{ij}$ parmi tous $i \in M, j \in N$ tel que $free_j = 1$
12. $s_{j^*} \leftarrow i^*$ ▷Assigne shift-pattern i^* à l'infirmière j^*
13. $free_{i^*} = 0$ ▷l'infirmière j^* n'est plus libre
14. $ub \leftarrow ub + c_{i^*j^*}$ ▷Mettre à jour le coût de la solution
15. **Pour** $k \in P$ **faire**$\beta_k \leftarrow \beta_k - a_{i^*k}$▷ Mettre à jour la demande de couverture
16. $iter \leftarrow iter + 1$
17. **jusqu'à** $iter = n$ ▷L'heuristique gloutonne se termine ici
18. $feasibility\_measure \leftarrow \sum_{k \in P} \max\{0, \beta_k\}$ ▷Tester la faisabilité de la solution
19. **si** $feasibility\_measure = 0$ **alors** La solution est faisable
20. **sortie** $(s_1...s_n), ub$
21. **sinon** appliquer l'algorithme 5.2.

5.3.1.2 Procédure de réparation

Si la solution construite par l'heuristique gloutonne est infaisable, alors il y a certains shifts k pour lesquels le nombre d'infirmières assignées est inférieure à b_k. La mesure

de faisabilité calculée à la fin de l'algorithme 5.1 donne l'insuffisance totale du nombre d'infirmières.

Notre objectif dans cette section est d'essayer de diminuer la mesure de faisabilité jusqu'à ce qu'elle disparaisse (si possible) en améliorant localement la solution. Pour ceci, nous définissons un voisinage de la solution infaisable s comme étant l'ensemble de toutes les solutions qui peuvent être obtenues à partir de s en libérant une infirmière j et en lui réaffectant un autre shift pattern $i \neq s_j$. Puisqu'il y a n infirmières et comme une fois l'infirmière j choisie, on peut lui assigner $m-1$ shifts patterns, la taille du voisinage est $O(m \times n)$.

La méthode d'amélioration locale cherche l'infirmière j^* et le shift pattern $i^* \neq s_{j^*}$ telle que la mesure de faisabilité diminue le plus. S'il y a une diminution stricte, le déplacement est effectué et la méthode est itérée. La procédure d'amélioration locale s'arrête quand la mesure de faisabilité disparaît, dans le cas où la solution est faisable, soit parce que aucune amélioration n'est possible. Dans le dernier cas, l'infaisabilité est proclamée. Le pseudo-code de la procédure de réparation est donné dans l'algorithme 5.2.

Algorithme 5.2 Procédure de réparation

entrée les donnée de NRP, $s_1, ..., s_n$, $\beta_1, ..., \beta_p$, ub, $feasibility\_measure$

sortie $s_1, ..., s_n$, ub

1.**répéter**

2 : $best\_improvement = \infty$

3 : $old\_feas = feasibility\_measure$

4 : **pour** $j \in N$ **faire**

5 : $e \leftarrow s_j$

6 : **pour** $i \in M$; $i \neq e$ **faire**

7 : $improvement \leftarrow 0$ ▷ Trouver la meilleure amélioration possible

8 : **pour** $k \in P$ **faire**

9 : **si** $a_{ik} = 1$ **et** $\beta_k > 0$ **alors**

10 : $improvement \leftarrow improvement + \beta_k - a_{ik}$

11 : **pour** $k \in P$ **faire**

12 : **si** $a_{ek} = 1$ **et** $\beta_k \geq -1$ **alors**

13 : $improvement \leftarrow improvement + \beta_k + a_{ek}$

14 : **si** $best\_improvement > improvement$ **alors**

15 : $best\_improvement \leftarrow improvement$

16 : $i^* \leftarrow i$

17 : $j^* \leftarrow j$

18 : **si** $feasibility\_measure > best\_improvement$ **alors** ▷ mises à jour

19 : $feasibility\_measure \leftarrow best\_improvement$

20 : $e \leftarrow s_{j^*}$

21 : $s_{j^*} \leftarrow i^*$

22 : **pour** $k \in P$ **faire**

23 : $\beta_k \leftarrow \beta_k + a_{ek}$

24 : $\beta_k \leftarrow \beta_k - ai^*k$

25 : $ub \leftarrow ub + c_{i^*j^*} - c_{ej^*}$

26 : **jusqu'à** $feasibility\_measure = 0$ ou $best\_improvement = old\_feas$

27 : **si** $feasibility\_measure = 0$ **alors**

28 : **sortie** $s_1, ..., s_n$, ub

29 : **sinon**

30 : **sortie** 'infaisabilité'

proposition 2 : La procédure de réparation s'exécute en temps $O(m \times n^2 \times p^2)$.

Démonstration : Les trois boucles imbriquées commencent respectivement dans les lignes $4, 6, 8$ et se terminent dans la ligne 17 de l'algorithme 5.2 ont besoin d'$O(m \times n \times p)$ opérations. Puisque $b_k \leq n$, $k \in P$, nous avons $\sum_{k \in P} b_k \leq n \times p$. Par conséquent, la boucle ≪répéter≫est exécutée $n \times p$ fois dans le pire des cas.

Puisque dans la pratique le nombre d'exécutions de la boucle ≪répéter≫est plutôt petit, on peut considérer que le temps empirique surestimé lié à la proposition 2 est $O(m \times n \times p)$

5.3.2 Optimisation avec sous-gradient : un rappel

En associant un vecteur de multiplicateur Lagrangien $\pi \in \mathbb{R}^p$ aux contraintes de couverture relaxées (2.3), nous définissons le dual lagrangien de NRP comme étant le problème $\max_{\pi \in \mathbb{R}^p} w(\pi)$ où $w(\pi)$ est la valeur optimale du problème $LR(\pi)$.

$$
\begin{aligned}
& w(\pi) = -\sum_{k \in P} \pi_k b_k + \min \sum_{i \in M} \sum_{j \in N} \left(c_{ij} + \sum_{k \in P} \pi_k a_{ik} \right) x_{ij} \\
& \sum_{i \in M} x_{ij} = 1 \qquad j \in N \\
& x_{ij} \in \{0, 1\} \qquad i \in M, \quad j \in N
\end{aligned}
$$

La découverte de la solution optimale π^* de $LR(\pi)$ est une tâche dure. Un algorithme rapide, la méthode de sous-gradient, est généralement utilisé pour trouver une solution proche de l'optimale. En commençant avec $\pi^{(0)} = 0$, , il génère une séquence $\pi^{(t)}, t >= 1$ et prend $\pi^* = \pi(t^*)$ où $t^* = \text{argmax}_{t>=0} w(\pi^{(t)})$. Il est clair que $w(\pi^*)$ est égal à la valeur optimale de la relaxation LP de NRP. Mais on n'est pas intéressés à la qualité de la borne inférieure puisque le seul but de la méthode de sous-gradient est d'identifier les variables ≪peu prometteuses≫comme on verra dans la section suivante. Pour $\pi^{(t)}$ fixe, le problème $LR(\pi^{(t)})$ est résolu avec une règle simple : pour chaque $j \in N$ trouver

$$
i_j = \text{argmin}_{i \in M} \left\{ c_{ij} + \sum_{k \in P} \pi_k^{(t)} a_{ik} \right\} \tag{4}
$$

ensuite prendre

$$x_{ijj}^{(t)} = 1 \tag{5}$$

et calculer $w(\pi^{(t)})$ en conséquence. Les composantes du vecteur de sous-gradient $\sigma(\pi^{(t)})$ sont :

$$\sigma_k\left(\pi^{(t)}\right) = \sum_{i\in M}\sum_{j\in N} a_{ik}x_{ij}^{(t)} - b_k, \quad k \in P \tag{6}$$

ensuite, le vecteur lagrangien est mis à jour.

$$\pi_k^{(t+1)} = \max\{0, \pi_k^{(t)} + \rho\frac{ub - w\left(\pi^{(t)}\right)}{\left\|\sigma\left(\pi^{(t)}\right)\right\|^2}\sigma_k\left(\pi^{(t)}\right)\}, k \in P \tag{7}$$

Ici ρ est un paramètre satisfaisant $0 < \rho \leq 2$ et $\rho \longrightarrow \infty$ comme $t \longrightarrow \infty$. D'habitude la valeur d'*ub* prend le coût de la meilleure solution en cours et

$$\rho\frac{ub - w\left(\pi^{(t)}\right)}{\left\|\sigma\left(\pi^{(t)}\right)\right\|^2}$$

est la taille de pas dans la direction de sous-gradient.

Une description de pseudo-code de la méthode de sous-gradient est donnée dans l'algorithme (5.3). NBIT est un entier positif contrôle le nombre d'itérations. Une solution initiale est calculée (ligne 1) en utilisant l'heuristique gloutonne (et éventuellement après la réparation de la solution infaisable). Les lignes 1-13, à l'exception des lignes 2 et 7, décrivent la méthode de sous-gradient.

5.3.3 Règle de fixation de variable

L'idée d'écarter des variables est assez simple. Il résulte de l'information fournie par la méthode de sous-gradient. Certaines des variables x_{ij} ne prennent jamais la valeur 1 dans la solution du problème relaxé $LR(\pi)$. Par conséquent, Il est peu probable que ces variables prennent la valeur 1 dans une solution optimale du NRP. Cette affirmation n'a aucune preuve formelle puisqu'il s'agit d'une règle heuristique. Pour cette raison, le lecteur devrait être averti : le NRP original et le problème réduit ne sont pas

mathématiquement équivalents. Cependant, la réduction s'avère très efficace comme on verra dans la section consacrée à l'expérimentation computationnelle. La fixation des variables démarre à la fin de la méthode de sous-gradient à la ligne 14 de l'algorithme (5.3) où toutes les variables x_{ij} telles que $\sigma = 0$ sont rejetées (en fixant leur coût à ∞). Cette élimination incite d'autres réductions. Soit $I_j = \{i \in M | \sigma_{ij} \neq 0\}$ l'ensemble des shifts patterns qui peuvent être attribués à l'infirmière j après la limitation mentionnée ci-dessus.

Soit $J_i = \{j \in N | \sigma_{ij} \neq 0\}$ l'ensemble des infirmières qui peuvent recevoir le shift pattern i. S'il y a une infirmière j telle que I_j contient un seul shift pattern i^* (lignes 23-24), alors celle-ci devrait lui être affectée (ligne 27) dans une solution partielle. Par conséquent, le nombre d'infirmières se diminue (ligne 25), l'infirmière j est supprimée du problème (ligne 26), le coût partiel (ligne 28) et la demande de couverture (lignes 29-30) sont mises à jour.

Lorsque $J_i = \phi$ pour un certain shift pattern i (Lignes 31-32), alors ce dernier peut être supprimé (Ligne 34) et le nombre de shifts patterns se diminue d'un (Ligne 33). Si la réduction induit des demandes de couverture négatives, on pourrait les considérer comme nulles (lignes 35-37). À la fin, VFH sort les données du problème réduit RNRP (ligne 38).

Comme le nombre d'itérations NBIT est fixé, la complexité de VFH est dominée par le calcul d'une solution initiale dans la ligne 1 qui est $O(m \times n \times max\{n, p\})$. on peut donc le considéré comme un algorithme rapide.

Algorithme 5.3 VFH

entrée les données de NRP NBIT

sortie les données de RNRP

Calculer une solution initiale faisable $(s_1...s_n)$ ▷ soit ub leur Coût

$\sigma_{ij} \leftarrow 0, i \in M; j \in N$

$\rho \leftarrow 2$

$\pi_i^{(0)} \leftarrow 0$

pour $t = 0$,...,NBIT-1 **faire**

  Résoudre le problème relaxé $LR(\pi^{(t)})$ comme dans(4-5) ▷soit $x_{ij}^{(t)}$ la solution

  $\sigma_{ij} \leftarrow \sigma_{ij} + x_{ij}^{(t)}$ pour $i \in M, j \in N$

  Calculer la valeur optimale $w\pi_{(t)}$

  Calculer les composants du sous-gradient comme dans (6)

  Mettre à jour le vecteur lagrangien comme dans (7)

  pour $k \in P$ **faire**

    if $\pi_k^{(t+1)} > 0$ **alors** $\pi_k^{(t+1)} = 0$

  Ajuster ρ si nécessaire

$c'_{ij} \leftarrow \infty, \forall i \in M; j \in N$ tel que $\sigma_{ij} = 0$ ▷ suppression des variables peu prometteuses

$p\_cost \leftarrow 0$

$m' \leftarrow m$

$n' \leftarrow n$

$a'_{ik} \leftarrow a_{ik}, i \in M; k \in P$

$c'_{ij} \leftarrow c_{ij}, i \in M; j \in N$

$b'_k \leftarrow b_k, k \in P$

$I_j \leftarrow \{i \in M | c'_{ij} < \infty\} j \in N$

$J_i \leftarrow \{j \in N | c'_{ij} < \infty\} i \in M$

pour $j \in N$ **faire**

  si $|I_j| = 1$ **alors** ▷ soit i^* le shift pattern unique appartenant à I_j

    $n' \leftarrow n' - 1$

    Supprimer la j^{eme} colonne de la matrice c'

    $s_j \leftarrow i^*$

    $p\_cost \leftarrow p\_cost + c_{i^*j}$

    pour $k \in P$ faire

    $b_k \leftarrow b_k - a_{i^*k}$

pour $i \in M$ **faire**

  si $J_i = \emptyset$ **alors**

    $m' \leftarrow m' - 1$

    Supprimer la i^{eme} ligne de la matrice a' et la i^{eme} ligne de la matrice c'

pour $k \in P$ **faire**

    si $b'_k < 0$ **alors**

      $b'_k \leftarrow 0$

sortie m',n' , p, a', b', c'

5.3.4 Exemple illustratif

VFH est illustré avec un exemple de petite taille avec le $m = 7$,$n = 4$ et l'horizon de planification se comporte $d = 4$ jours, chacun est composé de 3 shifts de travail et un libre (shift de repos). D'où $p = 4 \times 4 = 16$. Les matrices (a_{ik}), (c_{ij}) et (b_k) de $i = 1...7$, $j = 1...4$ et $k = 1...16$ sont données dans le tableau 5.1.

TABLE 5.1: La matrice (a_{ik}) et le vecteur (b_k) du NRP (exemple).

	Jour 1				Jour 2				Jour 3				Jour 4			
shift-pattern	Shift 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
2	1	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0
3	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0
4	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0
5	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1
6	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1
7	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0
b_k	2	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0

$$c_{ij} = \begin{pmatrix} 21 & 2 & 11 & 5 \\ 8 & 7 & 31 & 33 \\ 17 & 2 & 12 & 11 \\ 26 & 9 & 22 & 41 \\ 7 & 8 & 11 & 32 \\ 22 & 8 & 16 & 15 \end{pmatrix}$$

Une solution initiale $s = (s_1 = 2, s_2 = 7, s_3 = 6, s_4 = 1)$ avec un coût $= 30$ est construite en utilisant l'heuristique gloutonne. Ainsi, la matrice σ_{ij} commence avec les valeurs suivantes :

$$\sigma_{ij} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

et à la fin de la méthode de sous gradient, nous obtenons la matrice suivante :

$$\sigma_{ij} = \begin{pmatrix} 0 & 0 & 38 & 201 \\ 199 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 163 & 0 \\ 0 & 201 & 0 & 0 \end{pmatrix}$$

À partir de cette information, chaque variable x_{ij} telle que $\sigma_{ij} = 0$ peut être éliminée. Par conséquent, on obtient seulement 6 sur $7 \times 4 = 28$ variables. On observe que l'infirmière 1 doit choisir seulement entre deux shifts patterns, le deuxième et le cinquième . L'infirmière 3 doit aussi choisir entre le premier et le sixième. Comme Infirmière 2 n'a qu'un seul choix (shift pattern 7), on le lui attribue, et on rappelant son coût $p\_cost = 1$. Une fois le shift pattern 7 est affecté à l'infirmière 2, il n'est plus nécessaire et peut être supprimé. Aussi l'infirmière 4 n'a qu'un seul choix et devrait recevoir le shift pattern 1 avec un coût = 5 ($p\_cost = 1 + 5 = 6$). La même chose est vraie pour les shifts patterns 3 et 4 qui peuvent être exclus de toute nouvelle considération.

Par conséquent le problème réduit résultant (RNRP) se comporte uniquement de $m' = 4$ shifts patterns et $n' = 2$ infirmières. Les matrices a', b' et c' sont données ci-dessous.

TABLE 5.2: La matrice (a'_{ik}) et le vecteur (b'_k) du problème réduit.

	Jour 1				Jour 2				Jour 3				Jour 4			
shift-pattern	Shift 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
2	1	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0
5	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1
6	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1
b'_k	1	1	0	0	0	1	1	0	0	0	1	0	0	1	0	0

$$c'_{ij} = \begin{pmatrix} \infty & 11 \\ 8 & \infty \\ 7 & \infty \\ \infty & 16 \end{pmatrix}$$

Une solution optimale de RNRP est obtenue en assignant le shift pattern 2 à l'infirmière 1 et le shift pattern 6 à l'infirmière 3 avec un coût de 24. Rappelant la solution partielle trouvée par VFH (affectation du shift pattern 7 à infirmière 2 et shift pattern 1 à

infirmière 4), nous obtenons une solution complète (qui est optimale) au NRP avec un coût $= 24 + p\_cost = 30$.

5.4 Le problème réduit

Supposant $M' = \{1, ..., m'\}$ et $N' = \{1, ..., n'\}$. Pour l'infirmière $j \in N'$, supposant $S_j = \left\{i \in M' | c'_{ij} < \infty\right\}$ soit l'ensemble de shift-patterns qu'elle peut choisir. Le problème réduit RNRP peut être défini comme suit :

$$\begin{aligned} \min \sum_{j \in N'} \sum_{i \in S_j} c'_{ij} x_{ij} & \\ \sum_{i \in S_j} x_{ij} = 1, & \qquad j \in N' \\ \sum_{j \in N'} \sum_{i \in S_j} a'_{ik} x_{ij} \geq b'_k, & \qquad k \in P \\ x_{ij} \in \{0, 1\} & \qquad j \in N', i \in S_j \end{aligned}$$

Il est clair que le RNRP est une restriction du NRP et par conséquent, toute solution optimale au RNRP peut être étendue à une solution réalisable pour le NRP, simplement en ajoutant la solution partielle obtenue lors de l'application du VFH pour avoir une solution complète pour le problème original.

Les résultats obtenus lors de l'application sur des instances de la base NSPLib benchmark montrent que le RNRP est un NRP très petit et clairsemé. Il est clairsemé au sens où chaque infirmière $j \in N'$ ne peut choisir que des shift patters dans S_j, un ensemble dont la cardinalité est considérablement très inférieure au nombre total de shift patterns m'. Le programme binaire RNRP est clairsemé au sens où sa matrice de contraintes binaires l'est, ce que signe que le programme binaire est facile à résoudre. Cela justifie notre idée simple de résoudre le RNRP avec Cplex.

5.5 Résultat expérimental

L'objectif de cette section est de démontrer l'efficacité de la méthode à deux phases proposée, comme notre approche est une application séquentielle de VFH et de Cplex, on

va l'appelé VFH⟶Cplex de manière typique. Deux tests sont nécessaires pour conclure à son efficacité par des tests empiriques : la qualité de la solution et la vitesse de calcul. La section 5.5.1 présente le protocole expérimental. La section 5.5.2 propose d'analyser les résultats de l'heuristique gloutonne. sur un petit échantillon d'instances. La section 5.5.3analyse les résultats de VFH sur le même échantillon. La section 5.5.4 présente la comparaison entre l'algorithme proposé et quatre autres méthodes récentes.

5.5.1 Configuration expérimentale

La méthode à deux phases est codé en C et s'exécute sur un processeur Intel Core i5 @ 2,4 GHz. En fait, le code est divisé en deux codes, un appelé VFH, qui effectue la fixation des variables et stocke les données du RNRP (en tant que sortie de VFH) dans un format approprié pour un solveur MIP ; par la suite, ce fichier (RNRP instance) est lu et résolu par Cplex. La méthode à deux phases est l'application séquentielle de VFH et cplex et est notée VFH⟶Cplex.

Le nombre d'itérations de la méthode du sous gradient est fixé à 200. Ce nombre est suffisant pour assurer la convergence.

La méthode proposée est testée sur l'ensemble appelé ≪diverse≫de la base de données NSPLib [186], cet ensemble contient quatre sous-ensembles de $n = 25$, 50, 75 et 100 infirmières respectivement. Dans chacun des quatre sous-ensembles, il existe 8 groupes de contraintes spécifiques à chaque cas, avec 7290 instances dans chacune. Par conséquent, il y a $4 \times 8 \times 7290 = 233280$ instances dans l'ensemble divers. Quatre contraintes spécifiques à chaque cas sont prises en compte, comme déjà indiqué à la section 2.3.1 :

- le nombre minimum/maximum d'affectations par période.
- le nombre minimum/maximum d'affectations par type de shift.
- le nombre minimum/maximum de shifts de travail consécutifs.
- le nombre minimum/maximum de même type de shifts consécutifs.

L'horizon de temps considéré est $d = 7$ jours et le nombre de shifts est $s = 4$ y compris le shift libre. Plus de détails sur la description de la base NSPlib ainsi que les contraintes du problème sont disponibles dans Vanhoucke and Maenhout [186].

5.5.2 Analyse des résultats de l'heuristique gloutonne et de la procédure de réparation

Il est important d'analyser les résultats de l'heuristique gloutonne présentés à la section 5.3.1.1, ainsi que l'algorithme de réparation présenté à la section 5.3.1.2. Nous considérons d'abord le cas où l'heuristique gloutonne est appliquée sans la procédure de réparation (sous l'étiquette « heuristique gloutonne seul »). Ensuite nous examinons le cas où l'heuristique gloutonne est suivie par la procédure de réparation (sous « heur-gloutonne- Réparation »). Comme il existe des instances irréalisables dans l'ensemble de données NSPLib, un autre critère important à étudier est le taux de faisabilité (noté% FEAS) de VFH⟶Cplex ; c'est-à-dire sa capacité de trouver des solutions réalisables. Dans notre méthode, la conclusion sur la faisabilité est dévolue à VFH (plus précisément à l'heuristique gloutonne suivie éventuellement par l'étape de réparation). En effet, si l'heuristique gloutonne ne parvient pas à construire une solution et si la procédure de réparation ne répare pas la solution infaisable, l'instance est proclamée infaisable même si ce n'est pas le cas. Comme la base de données est très volumineuse, nous avons choisi au hasard un échantillon de 32 instances pour notre étude ; un de chacun des 32 cas. Par conséquent, une instance est choisie au hasard sur 7290. Les résultats sont présentés dans le tableau 5.3. Les quatre premières colonnes caractérisent l'instance. Les colonnes 5 et 6 fournissent le coût de la solution (où Inf indique une infaisabilité) et le temps de calcul de l'heuristique gloutonne. Les deux colonnes suivantes font référence à la procédure de réparation, et les deux dernières à la solution optimale trouvée par Cplex et à son temps de calcul. Ces informations sont utiles pour déterminer la différence par rapport aux solutions optimales. Il semble que l'heuristique gloutonne n'ait pas efficace puisqu'elle échoue à la moitié du temps de trouver des solutions réalisables. De sa part, la procédure de réparation réussit à améliorer l'heuristique gloutonne de telle sorte qu'elle n'échoue que sur des instances vraiment irréalisables (Cplex agit ici comme un certificat), mais du point de vue de la qualité de la solution la méthode de réparation est moins précise car elle est orientée vers la faisabilité et ne tient pas compte de l'affectation des coûts et néglige la fonction objective. En ce qui concerne le temps de calcul, l'heuristique gloutonne ainsi que la méthode de réparation sont très rapides et s'accordent la plupart du temps avec les limites de temps théoriques. Par conséquent, l'application séquentielle de l'heuristique gloutonne et de la méthode de réparation peut être considérée comme un algorithme rapide et efficace pour identifier de bonnes solutions initiales.

TABLE 5.3: Résultats de l'heuristique gloutonne, avec et sans réparation.

				Gloutonne seule		Gloutonne avec réparation		Cplex solution optimal	
n	Cas	Instance	Shift-patterns	Coût	Temps	Coût	Temps total	Coût	Temps
25	1	9	1287	Inf	0.02	282	0.02	278	1.23
	2	1409	504	Inf	0.00	423	0.00	405	0.09
	3	2553	66	Inf	0.00	317	0.01	301	0.06
	4	3498	3058	Inf	0.02	Inf		Inf	
	5	4108	3114	227	0.02			223	0.67
	6	4264	423	229	0.00			219	0.38
	7	5895	2631	Inf	0.01	Inf		Inf	
	8	6531	449	245	0.00			238	0.13
50	1	109	449	Inf	0.00	623	0.02	598	0.66
	2	1269	504	491	0.00			483	1.11
	3	2216	66	693	0.00			661	0.08
	4	3743	3058	Inf	0.03	443	0.06	418	6.13
	5	5179	423	499	0.02			495	0.53
	6	5442	3114	489	0.03			472	1.30
	7	5998	1287	448	0.01			436	0.70
	8	6261	2631	513	0.03			494	1.41
75	1	34	449	Inf	0.00	895	0.01	877	0.94
	2	666	1287	1083	0.03			1042	4.59
	3	868	66	854	0.00			812	0.08
	4	1073	504	939	0.00			903	1.42
	5	1209	2631	768	0.06			727	12.09
	6	2656	3114	Inf	0.08	Inf		Inf	
	7	3249	3058	705	0.05			698	8.84
	8	7206	423	Inf	0.02	852	0.03	817	1.86
100	1	64	1287	1196	0.03			1185	1.19
	2	1211	66	1417	0.00			1375	0.14
	3	2754	3114	1467	0.09			1417	2.70
	4	3875	504	Inf	0.02	1404	0.02	1309	2.02
	5	4163	2631	1114	0.09			1046	5.13
	6	4386	423	Inf	0.02	1075	0.02	1031	0.56
	7	5642	2631	Inf	0.09	1808	0.17	1711	81.24
	8	7282	1287	Inf	0.09	1368	0.33	1350	4.95

5.5.3 Analyse des résultats de VFH⟶cplex

Il est important de se rappeler que VFH prend en entrée une instance NRP et génère une instance RNRP. Les questions qui se posent sont abordées ci-dessous :

1 Quelle est la taille du RNRP par rapport à celui du NRP ?

2 Bien que le RNRP soit une version très réduite et clairsemée du NRP, sa solution optimale peut-elle encore être étendue à une solution optimale du NRP ?

3 Si la réponse de la question (2) est oui, pourquoi cela se produit-il ?

Nous n'avons aucune explication théorique, nous allons donc essayer des arguments empiriques. Une réponse à la première question est donnée dans le tableau 5.4. Alors que le nombre d'infirmières est légèrement réduit, on observe que le nombre de shift patterns est largement diminué. Ce qui est remarquable, c'est la diminution spectaculaire du nombre de variables, environ 1000 fois moins dans certains cas. En plus les temps de calcul, indiqués dans la dernière colonne, sont inférieurs à une seconde.

Le tableau 5.5 fournit une réponse à la deuxième question. Les colonnes 4 et 5 donnent le coût partiel et le temps de calcul de la première phase. Les résultats de la phase 2 sont affichés dans les colonnes 6 et 7. Les colonnes 8 et 9 proposent de résumer les résultats VFH⟶Cplex. On peut voir que pour chaque 28 des 29 cas possibles, VFH⟶Cplex réussit très rapidement à trouver la solution optimale (ce qui est confirmé par les résultats obtenus par Cplex dans les deux dernières colonnes).

Enfin, nous proposons une explication à la troisième et dernière question du point (3). Il existe deux bornes inférieures et supérieures triviales sur les solutions optimales pour NRP et RNRP : $LB = \sum_{j \in N} min_{i \in M} c_{ij}$ et $UB = \sum_{j \in N} max_{i \in M} c_{ij}$. Ces limites sont calculées dans le tableau 5.6 (pour les instances RNRP, elles sont en fait LB + p-cost et UB + p-cost). Bien que l'écart entre les deux limites soit grand pour les instances de NRP, il est relativement très petit pour les instances de RNRP, ce qui renforce la solution optimale. Par conséquent, RNRP semble être une version condensée du PNR, qui présente ses caractéristiques les plus importantes.

TABLE 5.4: Résultats de VFH sur un échantillon de 32 instances

			Taille de l'instance NRP originale			Solution initiale		Taille de l'instance RNRP				
n	Cas	Instance	nurses	shift-patterns	variables	Coût	Temps	nurses	shift-patterns	variables	p-cost	Temps
25	1	9	25	1287	32175	282	0.01	21	35	95	54	0.11
	2	1409		504	12600	423	0.00	22	19	50	96	0.05
	3	2553		66	1650	317	0.00	21	34	92	50	0.02
	4	3498		3058	76450	Inf						
	5	4108		3114	77850	227	0.00	21	45	51	43	0.27
	6	4264		423	10575	229	0.00	16	39	43	73	0.05
	7	5895		2631	65775	Inf						
	8	6531		449	11225	245	0.00	6	12	13	186	0.05
50	1	109	50	449	22450	623	0.01	50	28	254	0	0.06
	2	1269		504	25200	491	0.00	45	96	189	48	0.08
	3	2216		66	3300	693	0.00	45	18	104	61	0.00
	4	3743		3058	152900	443	0.06	49	150	216	9	0.42
	5	5179		423	21150	499	0.00	42	90	137	80	0.05
	6	5442		3114	155700	489	0.03	40	81	112	92	0.41
	7	5998		1287	64350	448	0.02	34	80	86	140	0.16
	8	6261		2631	131550	513	0.03	44	106	156	57	0.33
75	1	34	75	449	33675	895	0.03	74	29	198	9	0.09
	2	666		1287	96525	1083	0.02	69	220	364	83	0.22
	3	868		66	4950	854	0.00	46	25	112	328	0.00
	4	1073		504	37800	939	0.01	74	143	582	11	0.09
	5	1209		2631	197325	768	0.06	71	120	229	43	0.45
	6	2656		3114	233550	Inf						
	7	3249		3058	229350	705	0.05	18	34	36	5660	0.55
	8	7206		423	31725	852	0.03	73	125	355	16	0.08
100	1	64	100	1287	128700	1196	0.05	89	41	346	168	0.28
	2	1211		66	6600	1417	0.00	90	40	268	166	0.02
	3	2754		3114	311400	1467	0.11	99	68	342	16	0.81
	4	3875		504	50400	1404	0.03	86	106	336	160	0.11
	5	4163		2631	263100	1114	0.09	88	184	303	116	0.64
	6	4386		423	42300	1075	0.02	57	77	125	436	0.09
	7	5642		2631	263100	1808	0.17	100	352	1030	0	0.73
	8	7282		1287	128700	1368	0.09	100	86	459	0	0.33

TABLE 5.5: Résultats de VFH⟶Cplex (sur le même échantillon)

			Phase1 : VFH		Phase 2 : résoudre RNRP avec Cplex		Résultats de la méthode à deux-phase		Cplex appliqué sur NRP	
n	Cas	Instance	p-cost	Temps1	Coût	Temps2	Coût total p-cost + Coût	Temps total Temps1+Temps2	Coût	Temps
25	1	9	54	0.11	224	0.05	278	0.16	278	1.23
	2	1409	96	0.05	309	0.02	405	0.07	405	0.09
	3	2553	50	0.02	251	0.03	301	0.05	301	0.06
	4	3498	Inf						Inf	
	5	4108	43	0.27	180	0.03	223	0.30	223	0.67
	6	4264	73	0.05	146	0.03	219	0.08	219	0.38
	7	5895	Inf						Inf	
	8	6531	186	0.05	52	0.02	238	0.07	238	0.13
50	1	109	0	0.06	598	0.05	598	0.11	598	0.66
	2	1269	48	0.08	435	0.16	483	0.24	483	1.11
	3	2216	61	0.00	600	0.03	661	0.03	661	0.08
	4	3743	9	0.42	409	0.25	418	0.67	418	6.13
	5	5179	80	0.05	415	0.05	495	0.10	495	0.53
	6	5442	92	0.41	380	0.06	472	0.47	472	1.30
	7	5998	140	0.16	296	0.03	436	0.19	436	0.70
	8	6261	57	0.33	440	0.11	497	0.44	494	1.41
75	1	34	9	0.09	868	0.06	877	0.15	877	0.94
	2	666	83	0.22	959	0.52	1042	0.74	1042	4.59
	3	868	328	0.00	484	0.01	812	0.01	812	0.08
	4	1073	11	0.09	892	0.39	903	0.48	903	1.42
	5	1209	43	0.45	684	0.25	727	0.70	727	12.09
	6	2656	Inf						Inf	
	7	3249	566	0.55	132	0.02	698	0.57	698	8.84
	8	7206	16	0.08	801	0.61	817	0.69	817	1.86
100	1	64	168	0.28	1017	0.05	1185	0.33	1185	1.19
	2	1211	166	0.02	1209	0.08	1375	0.10	1375	0.14
	3	2754	16	0.81	1401	0.14	1417	0.95	1417	2.70
	4	3875	160	0.11	1149	0.30	1309	0.41	1309	2.02
	5	4163	116	0.64	930	0.30	1046	0.94	1046	5.13
	6	4386	436	0.09	595	0.05	1031	0.14	1031	0.56
	7	5642	0	0.73	1711	2.53	1711	3.26	1711	81.24
	8	7282	0	0.33	1355	0.50	1355	0.83	1355	4.80

5.5.4 Comparaison avec les méthodes existantes

VFH⟶Cplex est testé sur l'ensemble de données NSPLib. Les résultats sont affichés dans les deux dernières colonnes du tableau 5.7 et les deux dernières colonnes du tableau 5.9. La méthode proposée est comparée à quatre méthodes récentes et à Cplex. Ce sont les seules méthodes traitant cette base de données NSPLib dont nous avons connaissance.

Les valeurs moyennes des six méthodes concurrentes sont données pour huit cas et les meilleures valeurs moyennes (sauf celles relatives à Cplex) sont en gras. En comparant les méthodes du point de vue de la précision, il apparaît clairement (voir le tableau 5.7) que VFH⟶Cplex est la meilleure solution si nous excluons Cplex. Du point de vue du taux de faisabilité, c'est HG2018 [83] qui obtient les meilleurs résultats puisqu'il s'appuie sur Cplex pour trouver une solution initiale (voir à nouveau le tableau 5.7). En fait, Cplex est évidemment le plus précis et le meilleur taux de faisabilité (le vrai). Maintenant, nous comparons les six méthodes du point de vue de la vitesse. En règle générale, la comparaison des temps de calcul est problématique, car les environnements informatiques utilisés

TABLE 5.6: Bornes inférieures et supérieures pour les instances d'échantillons NRP et RNRP.)

			NRP instance			RNRP instance		
n	Cas	Instance	LB	UB	Gap	LB	UB	Gap
25	1	9	278	492	214	278	282	4
	2	1409	405	637	232	405	423	18
	3	2553	298	503	205	298	322	24
	4	3498	Inf					
	5	4108	221	624	403	221	233	12
	6	4264	217	578	361	217	231	14
	7	5895	Inf					
	8	6531	237	580	343	237	247	10
50	1	109	591	917	326	591	636	45
	2	1269	483	890	407	483	491	8
	3	2216	644	1012	368	644	721	77
	4	3743	386	1004	618	386	467	81
	5	5179	495	858	363	495	499	4
	6	5442	472	885	413	472	489	17
	7	5998	436	1064	628	436	448	12
	8	6261	465	1129	664	465	554	89
75	1	34	877	1394	517	877	895	18
	2	666	1042	1526	484	1042	1083	41
	3	868	806	1469	663	806	858	52
	4	1073	857	1428	571	857	958	101
	5	1209	725	1425	700	725	768	43
	6	2656	Inf					
	7	3249	698	1599	901	698	705	7
	8	7206	718	1424	706	718	895	177
100	1	64	1185	1970	785	1185	1196	11
	2	1211	1375	2260	885	1375	1417	42
	3	2754	1404	2365	961	1404	1486	82
	4	3875	1243	2464	1221	1243	1491	248
	5	4163	973	2564	1591	973	1188	215
	6	4386	1004	2434	1430	1004	1089	85
	7	5642	1605	2428	823	1605	1866	261
	8	7282	959	2534	1575	959	1439	480

sont différents (voir le tableau 5.8). Les chercheurs ont l'habitude de normaliser les temps de calcul via des valeurs de spécification. Le site Web SPEC (Standard Performance Evaluation Corporation) https://www.spec.org/cpu2006/results/cint2006.html fournit une estimation de la vitesse d'exécution de nombreux ordinateurs modernes. Une valeur Specint 2006 est attachée à chaque ordinateur ; plus la valeur est élevée, plus l'ordinateur est rapide. Malheureusement, les machines indiquées dans le tableau 5.8 et utilisées par les heuristiques concurrentes ne sont pas incluses dans le site Web. Cependant, nous avons pris les valeurs associées à des machines ayant des spécifications similaires et dont les valeurs de spécification sont à nouveau indiquées dans le tableau 5.8. À partir de celles-ci, les facteurs d'échelle sont estimés et utilisés pour normaliser les temps de calcul dans le tableau 5.9. La dernière ligne du tableau 5.9 montre que VFH⟶Cplex [81] est environ 37 fois plus rapide que MV2006 [122], 33 fois plus rapide

que MV2007 [124], 48 fois plus rapide que CAL2013[53], 5 fois plus rapide que HG2018 [83] et 8 fois plus rapide que Cplex. Étonnamment, Cplex est meilleur que MV2006, MV2007 et CAL2013, à tout point de vue.

TABLE 5.7: Comparaison des méthodes concurrentes du point de vue du coût et du taux de faisabilité. (moyennes sur 7290 occurrences)

		MV2006		MV2007		CAL2013		HG2018		Cplex		VFH⟶Cplex	
n	Cas	Coût	%Feas	Coût	%Feas	Coût	%Feas	Coût	%Feas	Coût	%Feas	Coût	%Feas
25	1	250.68	88.27	250.89	88.27	251.39	88.27	250.43	88.27	250.14	88.27	250.18	87.97
	2	239.45	88.27	239.61	88.27	239.69	88.27	239.46	88.27	239.24	88.27	239.28	88.16
	3	266.64	88.08	266.98	87.91	267.68	88.09	265.72	88.09	265.38	88.09	265.40	87.11
	4	248.71	88.27	248.97	88.27	249.09	88.27	248.64	88.27	248.30	88.27	248.34	88.00
	5	263.63	85.88	263.88	85.83	265.23	85.88	263.16	85.88	262.79	85.88	262.84	85.13
	6	240.43	88.27	240.61	88.27	240.64	88.27	240.44	88.27	240.22	88.27	240.26	88.15
	7	280.14	80.10	279.98	79.84	282.04	77.39	279.21	80.18	278.57	80.18	278.63	72.63
	8	256.66	85.60	256.88	85.53	257.50	85.49	256.18	86.69	255.81	85.69	255.86	84.94
Valeurs moyennes		255.79	86.59	255.97	86.52	256.66	86.24	255.41	**86.62**	255.06	86.62	**255.10**	86.01
50	1	500.30	90.03	500.27	89.90	500.02	90.03	499.10	90.03	499.01	90.03	499.09	89.86
	2	478.19	90.03	478.41	90.03	477.91	90.03	477.66	90.03	477.60	90.03	477.67	89.92
	3	526.48	89.66	526.51	89.26	525.69	89.77	524.15	89.82	523.74	89.82	523.80	89.15
	4	496.68	90.03	496.87	89.93	496.61	90.03	495.78	90.03	495.71	90.03	495.79	89.82
	5	523.42	85.25	521.75	74.43	523.64	85.34	521.84	85.34	521.45	85.34	521.57	84.68
	6	480.39	90.03	481.52	89.96	480.07	90.03	479.82	90.03	479.74	90.03	479.81	89.89
	7	549.48	78.29	548.67	77.72	549.28	76.46	547.07	78.61	546.48	78.61	546.60	77.42
	8	508.77	85.50	508.68	84.91	508.06	85.39	507.17	85.65	506.85	85.65	506.95	85.03
Valeurs moyennes		507.96	87.35	507.84	85.77	507.66	87.14	506.57	**87.44**	506.32	87.44	**506.41**	86.97
75	1	758.15	88.70	759.42	88.63	756.83	88.70	755.75	88.70	755.70	88.70	755.79	88.63
	2	733.58	88.70	734.37	88.68	732.80	88.70	732.54	88.70	732.48	88.70	732.59	88.68
	3	797.75	88.37	797.17	87.82	795.46	88.53	793.78	88.57	793.56	88.57	793.62	88.05
	4	747.09	88.70	747.79	88.66	746.00	88.70	745.49	88.70	745.43	88.70	745.55	88.63
	5	795.71	86.06	795.25	85.58	794.51	86.09	792.52	86.09	792.43	86.09	792.53	85.82
	6	735.03	88.70	735.46	88.68	734.13	88.70	733.88	88.70	733.81	88.70	733.92	88.68
	7	837.85	79.48	835.42	78.07	835.54	77.56	833.88	79.93	833.95	79.93	833.08	78.79
	8	780.40	85.78	779.84	84.99	778.14	85.76	777.15	85.93	776.98	85.93	777.09	85.61
Valeurs moyennes		773.20	86.81	773.09	86.39	771.68	86.59	770.61	**86.91**	770.42	86.91	**770.52**	86.61
100	1	1218.45	90.49	1218.90	90.22	1215.34	90.53	1214.29	90.53	1214.13	90.53	1214.31	90.38
	2	1176.08	90.51	1176.60	90.48	1174.09	90.53	1173.87	90.53	1173.75	90.53	1173.94	90.43
	3	1293.94	90.01	1293.09	89.19	1289.11	90.37	1287.14	90.38	1286.97	90.38	1287.08	89.96
	4	1204.34	90.48	1205.31	90.32	1201.74	90.53	1201.19	90.53	1200.99	90.53	1201.22	90.38
	5	1270.29	96.28	1269.53	85.49	1267.56	86.54	1265.38	86.54	1264.93	86.54	1265.18	86.05
	6	1178.97	90.51	1179.62	90.41	1176.83	90.53	1176.61	90.53	1176.48	90.53	1176.67	90.43
	7	1338.26	79.49	1335.57	78.49	1335.19	78.59	1332.23	80.34	1331.46	80.34	1331.59	79.33
	8	1247.93	86.53	1247.15	85.91	1243.78	86.74	1242.86	86.94	1242.34	86.94	1242.59	86.32
Valeurs moyennes		1241.03	88.04	1240.72	87.56	1237.96	88.05	1236.70	**88.29**	1236.38	88.29	**1236.59**	87.91

TABLE 5.8: Méthodes concurrentes et leurs environnements d'exécution

label	Méthodologie	Référence	Machine utilisée	Fréquence
MV2006	Scatter search	Maenhout and Vanhoucke [122]	Toshiba SPA10	2.4 GHz
MV2007	Electromagnetism	Maenhout and Vanhoucke [124]	Idem	Idem
CAL2013	Linear assignment	Constantino *et al.* [53]	Quad-Core Xeon	3.2 GHz
HG2018	VLSN search	Haddadi and Guessoum [83]	HP Intel Core i5	2.4 GHz
VFH⟶cplex	Two-phase method	Guessoum *et al.* [81]	Idem	Idem
Cplex	MIP solver		Idem	Idem

TABLE 5.9: Comparaison des méthodes concurrente du point de vue de la vitesse.

		MV2006	MV2007	CAL2013		HG2018		Cplex		VFH⟶Cplex	
					Temps		Temps		Temps		Temps
n	Cas	Temps	Temps	Temps	normalisé	Temps	normalisé	Temps	normalisé	Temps	normalisé
25	1	1.66	1.70			0.65	0.68	0.55	0.58	0.10	0.11
	2	1.01	2.70			1.01	1.06	1.11	1.17	0.18	0.19
	3	2.42	1.65			0.61	0.64	0.24	0.25	0.06	0.06
	4	1.62	2.04			0.88	0.92	0.98	1.03	0.17	0.18
	5	2.27	1.46			0.58	0.61	0.20	0.21	0.05	0.05
	6	1.02	2.66			0.91	0.96	1.09	1.14	0.18	0.19
	7	4.42	2.65			0.55	0.58	0.04	0.04	0.02	0.02
	8	1.56	1.72			0.62	0.65	0.19	0.20	0.04	0.04
Valeurs moyennes		2.00	2.07	0.72	1.63		0.77		0.58		0.11
50	1	5.40	4.10			1.10	1.16	1.51	1.59	0.20	0.21
	2	3.70	4.50			1.64	1.72	3.11	3.27	0.31	0.33
	3	6.48	4.48			0.79	0.83	0.58	0.61	0.12	0.13
	4	5.43	4.94			1.67	1.75	2.80	0.29	0.32	0.34
	5	6.62	4.23			0.75	0.79	0.48	0.50	0.10	0.11
	6	3.72	4.47			1.63	1.71	3.07	3.22	0.31	0.33
	7	10.46	6.00			0.57	0.60	0.08	0.08	0.04	0.04
	8	4.53	3.48			0.73	0.77	0.45	0.47	0.08	0.08
Valeurs moyennes		5.79	4.52	2.83	6.42		1.17		1.59		0.19
75	1	9.94	11.54			1.41	1.48	2.65	2.78	0.36	0.38
	2	10.45	14.01			2.28	2.39	5.36	5.63	0.54	0.57
	3	14.20	11.10			0.95	1.00	0.97	1.02	0.20	0.21
	4	9.75	14.38			2.19	2.30	4.77	5.01	0.54	0.57
	5	11.18	10.85			0.92	0.97	0.81	0.85	0.18	0.19
	6	8.32	14.81			2.27	2.38	5.28	5.54	0.53	0.56
	7	15.09	10.55			0.65	0.68	0.15	0.16	0.06	0.06
	8	8.98	10.51			0.90	0.95	0.75	0.79	0.15	0.16
Valeurs moyennes		10.99	12.22	6.83	15.50		1.52		2.72		0.34
100	1	22.61	16.85			1.98	2.08	3.60	3.78	0.52	0.55
	2	20.53	16.20			3.03	3.18	6.93	7.28	0.76	0.80
	3	24.33	17.94			1.32	1.39	1.38	1.45	0.30	0.32
	4	21.57	20.17			3.10	3.26	6.32	6.64	0.77	0.81
	5	22.88	17.01			1.24	1.30	1.10	1.16	0.25	0.26
	6	20.59	20.62			3.09	3.24	6.86	7.20	0.75	0.79
	7	27.27	16.64			0.83	0.87	0.18	0.19	0.08	0.08
	8	20.90	16.66			1.21	1.27	1.03	1.08	0.21	0.22
Valeurs moyennes		22.58	17.76	13.63	30.94		2.08		3.60		0.47
Moyenne totale		10.34	9.14		13.62		1.39		2.12		0.28
		37	33		48		5		8		1

5.6 Conclusion

Une méthode à deux phases VFH⟶Cplex est proposée. on commence par réduire la taille du NRP. L'heuristique de réduction de la taille est très rapide. La deuxième étape consiste simplement à appliquer Cplex pour résoudre le petit problème réduit et condensé obtenu à partir de la première étape (RNRP).

VFH⟶Cplex est testé sur l'ensemble de données NSPLib et comparé avec quatre méthodes publiées récemment et avec Cplex. Globalement, il est bien classé du point de vue des trois critères retenus (qualité de la solution, taux de faisabilité et temps de calcul). Une caractéristique intéressante de notre méthode proposée est sa grande vitesse. Il peut donc être considéré comme une bonne méthode pour résoudre le NRP.

Chapitre 6

Algorithme de propagation des plantes pour le NRP

6.1 Introduction

Le NRP est un problème d'optimisation combinatoire extrêmement important qui se pose fréquemment dans tous les établissements de soins de santé. Le NRP vise à créer des plannings pour les infirmiers soumis à différents types de contraintes, telles que les réglementations et de nombreux autres contraints spécifiques. La génération des bons plannings a une incidence considérable sur la motivation des infirmières et, par conséquent, sur la qualité de soins de santé.

En raison de sa pertinence pratique, de nombreuses méthodes ont été appliquées pour le résoudre, et les méta-heuristiques sont largement utilisées dans ce contexte. Dans ce chapitre, un algorithme de propagation des plantes, qui est une méthode méta-heuristique basée sur la population est proposée. Cette méthode est employée avec succès pour résoudre différents problèmes d'optimisation combinatoire. Comme beaucoup de méta-heuristiques réussies, le PPA est inspiré par un processus de vie. Il imite la stratégie de reproduction et de propagation du fraisier.

Ce chapitre est organisé comme suit. La section 6.2 décrit l'approche proposée, la section 6.3 présente la méthodologie suivie pour résoudre le problème, tandis que la section 6.4 est consacrée aux résultats de calcul et à la comparaison.

6.2 Approche proposé

Dans la présente étude, nous proposons une heuristique à trois phases pour résoudre le problème de rotation des infirmières, l'approche proposée se base sur l'utilisation d'un algorithme de propagation de plante (PPA). Notre méthode pour résoudre le NRP est résumée dans les trois points suivants :

— Une heuristique générique de fixation de variable est appliquée pour réduire la taille du problème par l'élimination des variables concéderaient comme étant non prometteuses, l'heuristique utilisée est la même décrite dans la section 5.3 du chapitre 5. dans la présente étude en vas l'exploiter dans la première phase pour générer le problème NRP réduit (RNRP).

— Le PPA s'applique au RNRP, qui est beaucoup plus petit. Pour des raisons de reproductibilité, tous les détails de l'implémentation sont donnés.

— Les solutions successives (rosters) fournies par PPA constituent un pool de solutions «élites» servant à éliminer d'autres variables et réduire davantage la taille du problème laissant un NRP très clairsemé qui peut être résolu directement par un solveur IP. Notez que cette idée a déjà été exploitée avec succès à Haddadi [87].

Cette approche en trois phases (variable-fixing puis PPA puis exploitation de solutions élite) est testée sur trente cas de référence et comparée à trois méthodes récentes. Il s'avère une méthode rapide et précise pour résoudre le NRP.

Nous soulignons que notre contribution à ce chapitre se situe dans le deuxième point ci-dessus et que notre objectif principal est de nous concentrer sur le PPA et de démontrer, une fois encore, son applicabilité étendue et son efficacité. En d'autres termes, la méthode est, à notre avis, plus importante que le problème qu'elle veut résoudre.

6.3 Méthodologie de la solution

Cette section explique en détail tous les éléments de la méthodologie de la solution appliquée dans le présent chapitre. La section 6.3.1 résume l'heuristique de fixation des variables. La section 6.3.2 décrit chaque étape du PPA, conçue pour résoudre le NRP, tandis que la section 6.3.3 montre comment on peut utiliser les solutions générées par le PPA pour réduire plus encore la taille du problème.

6.3.1 Heuristique de fixation des variables : (résumé)

L'heuristique de fixation de variable est une heuristique générique permettant de réduire la taille de l'instance de NRP de manière heuristique. Il exploite une information utile fournie par la méthode du sous-gradient appliquée à la relaxation lagrangienne des contraintes de couverture. En pratique, cela permet d'éliminer jusqu'à 90% des variables sans compromettre la qualité de la solution, réduisant ainsi considérablement la taille du problème. Cette heuristique de fixation de variable est très attrayante dans la pratique et peut être appliquée à n'importe quel problème d'optimisation combinatoire, à condition que les variables soient binaires, pour plus de détaille voir la section 5.3 .

6.3.2 La conception du PPA pour résoudre le RNRP

Comme le NRP est connu comme étant un problème NP difficile, les méthodes exactes nécessitent des temps d'exécution inabordables pour assurer une solution optimale. Par conséquent, elles ne s'appliquent qu'aux instances de petite ou moyenne taille. Pour traiter des instances très larges qui se produisent dans la pratique, des algorithmes heuristiques sont nécessaires pour trouver des rotations proches de l'optimum dans un temps d'exécution raisonnable.

Ce travail vise à résoudre de manière heuristique le NRP, et au-delà, on va se concentrer uniquement au problème réduit RNRP obtenu grâce à l'heuristique de fixation des variables, et on le considère comme problème principal.

On commence par une description dans l'algorithme 6.1 qui présente le pseudo-code de PPA adapté à la résolution de RNRP. Ensuite, les étapes importantes seront détaillées.

La boucle dans les lignes $3-6$ correspond à la construction de la population initiale (voir section 6.3.2.1). Les générations K sont construites dans la boucle des Lignes $10-24$. L'intensification, ou la construction de stolons courts, est effectuée dans les lignes $12-16$, tandis que la diversification, ou la construction de stolons longs, est abordée dans les lignes $17-21$. Ces étapes sont expliquées à la section 6.3.2.2. Le critère d'acceptation à la ligne 22 est expliqué à la section 6.3.2.3.

Algorithme 6.1 PPA conçu pour la résolution de RNRP.

1 : Entrée : Données RNRP, Δ ; $\%bon$; K . Δ est la taille de la population, K est le nombre de générations.
2 : Sortie : La meilleure solution et son coût
3 : **pour** $i = 1, ..., \Delta$; **faire** .
4 : Construire une solution α_i
5 : Améliorer la solution α_i
6 : Insérer la solution α_i dans une liste chaînée triée par ordre croissant du coût de la solution.
7 : Soit $\Theta = [\alpha_1, ..., \alpha_\Delta]$ la population triée
8 : $\Delta_1 \longleftarrow \%bon \times \Delta$
9 : $k \longleftarrow 1$
10 : **tant que** $k \leq K$ **faire**
11 : $\Gamma \longleftarrow \Theta$
12 : **pour** $i = 1, ..., \Delta_1$ **faire**
13 : **pour** $j = 1, ..., r$ **faire**
14 : Générer la solution β_j en perturbant "légèrement" la solution α_i
15 : Améliorer la solution obtenue β_j
16 : $\Gamma \longleftarrow \Gamma \cup \{\beta_j\}$
17 : **pour** $i = \Delta_1 + 1, ..., \Delta$ **faire**
18 : **pour** $j = 1, ..., Rd$ **faire**
19 : Générer la liste β_j en perturbant " substantiellement " la liste a_i
20 : Améliorer la solution obtenue β_j
21 : $\Gamma \longleftarrow \Gamma \cup \{\beta_j\}$
22 : garder en Γ seulement Δ solution selon un critère d'acceptation
23 : $\Theta \longleftarrow \Gamma = [\alpha_1, ..., \alpha_\Delta]$. Liste triée des listes sélectionnées
24 : $k \longleftarrow k + 1$
25 : Sortie : La meilleure solution α_1 et son coût

6.3.2.1 Génération de la population initiale

Cette fonction est considérée dans la première partie (lignes $3 - 6$) de l'algorithme 2. Une population initiale aléatoire, dans laquelle chaque solution initiale faisable (α_i à la ligne 4) est construite en deux étapes. Dans la première étape, une solution aléatoire est générée par une affectation au hasard d'un shift pattern à chaque infirmière. Évidemment, la construction aléatoire ne garantit pas la satisfaction de la contrainte de couverture. Si la solution construite dans la première étape est infaisable, la deuxième étape essaie de la réparer. La procédure de réparation est une méthode d'amélioration de la faisabilité locale. Son but est de diminuer la mesure de faisabilité jusqu'à ce qu'elle soit éliminée (si possible), pour cela en applique la même fonction de réparation décrite à la section 5.3.1.2.

Une fois qu'une solution (faisable) est construite, la tâche suivante (ligne 5) est de

l'améliorer en utilisant une méthode d'amélioration locale dont le but est de réduire le coût (si possible) en modifiant localement la solution α_i. La question cruciale dans la recherche locale est la structure du voisinage. Dans ce travail, on applique deux voisinages différents, le premier est un interchange de shift patterns qui permet à deux infirmières d'échanger leur shift pattern si cela améliore le coût de la solution. On procède l'interchange entre les infirmières jusqu'à ce que l'amélioration ne soit plus possible. La taille du voisinage est $O(n^2)$ puisqu'il existe $O(n^2)$ possibilités de choisir deux infirmières. Le deuxième cas du voisinage est appelé déplacement du shift pattern. Un déplacement dans ce voisinage consiste à choisir une infirmière, en la libérer, puis en lui assignant un autre shift pattern si la solution obtenue est faisable, et si le déplacement fait diminuer la valeur de la fonction objective. Le déplacement est appliqué jusqu'à ce que l'amélioration ne soit plus possible. La taille du déplacement du shift pattern est $O(m \times n \times p)$. En raison de leur taille, la recherche dans le voisinage est très rapide. De plus, au lieu de chercher séparément dans les deux voisinages, on a constaté que l'exploration de leur union donne de meilleurs résultats.

À la fin de cette phase, on obtient Δ solutions faisables à notre disposition. Les solutions sont enregistrées dans une table chaînée. Chaque solution est enregistrée dans un tableau en spécifiant son coût et les valeurs s_j, $j = 1...n$, où s_j indique le shift pattern attribué à l'infirmière j. Cette structure simplifie la gestion (insertion, suppression) des populations successives. La boucle dans les lignes $3-6$ de l'algorithme 2 n'est exécutée qu'une seule fois. En effet, le calcul d'une solution à la ligne 4 nécessite des opérations $O(m \times n \times p)$, ce qui correspond à la complexité de la procédure de réparation. De plus, l'amélioration dans la ligne 5 prend le même temps $O(m \times n \times p)$.

6.3.2.2 Intensification et diversification

Ces facteurs sont les plus importants et constituent le cœur du PPA. L'intensification et la diversification sont décrites respectivement aux lignes $12-16$ et $17-21$ de l'algorithme 6.1.

Que signifie le fait d'envoyer des stolons courts en termes de NRP ?

Supposons que la solution α_i soit la meilleure solution. Dans les lignes 12-16, la solution β_j est générée à partir de la solution α_i de cette manière (la solution β_j correspond à un stolon court envoyé par la plante-mère qui est la solution α_i) :

— Sélectionner au hasard %FreedNurses (un pourcentage d'infirmières à libérer) dans la solution α_i puis assigner à chacune un shift pattern choisi aléatoirement ;

— Si la solution produite est infaisable, essayez de la réparer (appelez cette solution β_j faisable).

À la ligne 15, l'amélioration de la solution β_j est identique à celle décrite à la ligne 5, expliquée ci-dessus. La diversification est similaire. Ce qui la différencie de l'intensification est la perturbation qui est contrôlée par la valeur %FreedNurses. Dans la phase d'intensification (qui correspond à la génération de stolons courts dans le paradigme du PPA), cette valeur est faible (environ 5%), alors qu'elle est plus élevée (environ 20%) dans la phase de diversification (envoi de stolons longs depuis la plante-mère).

6.3.2.3 Critère d'acceptation et critère d'arrêt

Il existe plusieurs méthodes de sélection pour choisir parmi les solutions de la nouvelle population celles qui seront reproduites dans la prochaine génération, il est possible de proposer de nombreux critères d'acceptation. Par exemple, une possibilité est de n'accepter que les Δ meilleurs éléments. Toutefois, ce critère est orienté vers l'intensification. De plus, il réduit la diversité des populations comme le justifieraient les généticiens. En effet, les plantes mères dans les mauvais endroits peuvent encore engendrer des descendes dans les bons endroits (en envoyant des stolons longs). En d'autres termes, des solutions de moins bonne qualité peuvent quand même produire des solutions de bonne qualité par des perturbations importantes.

À partir de cette observation, et afin d'équilibrer l'intensification et la diversification, la règle de sélection suivante est adoptée (ligne 22 de l'algorithme 6.1). Les Δ premiers éléments de la nouvelle population triés sont sélectionnés pour la reproduction. Un pourcentage (%good) de ces meilleures solutions sont prises dans la prochaine génération en tant que bons parents et le reste sont tenus comme mauvais parent dans la population. Voir la figure 6.1 pour voir un exemple de l'évolution de la population au cours des générations en matière de meilleurs et de mauvais parents.

Un nombre maximum de générations K est fixées dans le PPA. S'il y a une charge de calcul, il se produira dans la boucle dans lignes $10-21$. Cependant, l'établissement des solutions dans les lignes 14 et 19 se fait rapidement puisqu'il consiste uniquement à perturber les solutions existantes. L'amélioration dans les lignes 15 et 20 nécessite très peu de temps. Les opérations effectuées dans les lignes 16 à 21 sont insignifiantes. Par conséquent, nous pouvons affirmer que le PPA proposé est théoriquement une méthode rapide.

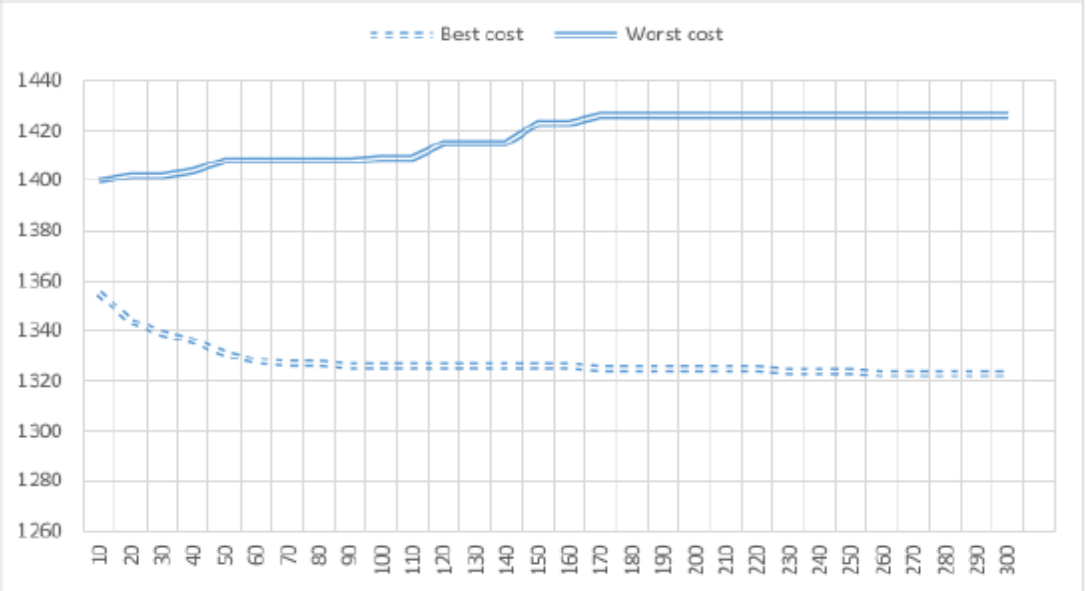

FIGURE 6.1: Écart entre les meilleurs et les pires coûts selon toutes les générations (instance 7209, cas 6).

6.3.3 Utilisation de solutions élites

Glover [78] a intelligemment observé que « l'information utile portant sur la forme (ou l'emplacement) des solutions optimales, se trouve généralement dans une collection diversifiée et appropriée de toutes les solutions élites ». Cette constatation nous encourage à utiliser le PPA pour produire des solutions élites.

En effet, lorsqu'il est appliqué au RNRP, le PPA crée des solutions à chaque génération. Au lieu de considérer toutes ces solutions, on considère seulement celles qui ont un petit coût et qui ont des valeurs de fonction objectives différentes. Ces solutions spécifiques peuvent être considérées comme des solutions élites au sens de Glover [78].

On définit les variables de référence de RNRP comme celles ayant la valeur 1 dans au moins une solution élite et on détermine la matrice v comme suit :

$$v_{ij} = \begin{cases} 1 & \text{si } x_{ij} \text{ est une variable de référence} \\ 0 & \text{autre} \end{cases}$$

On définit un nouveau problème appelé SmallNRP en ajoutant à RNRP les contraintes $x_{ij} = 0$ pour tous les i, j de sorte que $v_{ij} = 0$. Ainsi, on peut considérer l'utilisation de solutions élites comme une autre heuristique de fixation de variable. Il est clair que SmallNRP est une restriction de RNRP. En fait, c'est un programme binaire très clairsemé. Deuxièmement, il est toujours faisable puisque sa région faisable contient toutes les solutions élite. De plus, sa valeur optimale est par définition n'est pas moins bonne que celle de la meilleure solution élite.

Puisque SmallNRP est un programme binaire très petit et clairsemé, n'importe quel solveur MIP peut être utilisé pour le traiter. On verra dans la section suivante que l'incorporation de cet algorithme plutôt " évolutif " (il construit une solution en en combinant d'autres) aboutit toujours à une solution optimale ou très proche à l'optimale au NRP.

6.4 Résultats expérimentaux et analyse

Cette section vise à valider l'applicabilité et la pertinence de notre approche par des tests empiriques. Comme d'habitude, deux facteurs clés doivent être étudiés : la précision (le coût) et la vitesse de calcul.

Avant de commencer l'expérimentation, quelques observations peuvent être faites. Notre approche est simple et facile à implémenter. Mais comme toute méta-heuristique, elle a deux inconvénients. Elle souffre de la non-reproductibilité en raison de son caractère aléatoire intrinsèque et il n'y a aucun argument théorique pour le justifier.

La section 6.4.1 est consacrée à la configuration expérimentale. Les détails de l'exécution du PPA sont donnés à la section 6.4.2. Enfin, les résultats de calcul et les comparaisons du PPA avec les méta-heuristiques les plus récentes sont présentées à la section 6.4.3.

6.4.1 Configuration expérimentale

Notre méthode comporte deux parties. La première effectue l'heuristique de fixation des variables et sauvegarde les données RNRP du problème réduit dans un fichier. La deuxième partie du code lit les données du fichier généré et applique le PPA. Chacun des deux codes est codé en C et fonctionne sur un HP Intel Core i5, 2.4 GHz.

Des tests sont effectués sur une base de données de référence bien connue ≪NSPLib≫ (voir la section 4.4.1, cet ensemble de données est conçu par Vanhoucke and Maenhout [186] afin de faciliter l'évaluation des méthodes actuelles et futures dans le domaine. Le nombre d'instances à traiter est très important dans le but de pouvoir faire des statistiques. Notre objectif dans ce document n'est que de démontrer l'efficacité du PPA. Pour plus de détails sur cet ensemble de données voir [186].

6.4.2 Implémentation du PPA et configuration des paramètres

Un algorithme est une abstraction. Donc, son évaluation ne peut se faire que de manière indirecte à travers une implémentation. Les valeurs des paramètres relatifs à notre application particulière du PPA sont présentées au tableau 6.1. ces valeurs sont obtenues empiriquement. Comme d'habitude, plusieurs valeurs sont expérimentées et les meilleures dans les expériences sont utilisées.

La taille de la population est restée constante avec $\Delta = 5$ individus. Trois d'entre eux (60%) avec les meilleures valeurs de coût sont considérées comme de bonnes solutions (vivant dans de bons endroits selon le jargon du PPA). Les autres sont considérés comme mauvais. Chacune des trois bonnes solutions produira $r = 3$ nouvelles solutions comme expliqué à la section 6.3.2.2 (voir Figure 3.6). De la même manière, chacune des mauvaises solutions permet de générer une seule nouvelle solution, comme expliqué à la section 6.3.2.2.

6.4.3 Résultats et comparaisons

Notre objective consiste à comparer notre nouvelle approche par rapport aux techniques existantes (voir tableau 6.2). Ces dernières sont les méthodes les plus récentes dont nous disposons. Ils constituent l'état de l'art (au moins sur l'ensemble de données NSPLib)

et fournissent des points de référence pertinents. Évidemment, ils sont tous testés sur la même plateforme que celle mentionnée dans le plan expérimental.

TABLE 6.1: Les valeurs des paramètres

Paramètre	=	Valeur
Taille de la population Δ	=	5
% bon	=	60%
r	=	3
R	=	1
% NursesFreed	=	5% pour la phase d'intensification (ligne 14)
	=	20% pour la phase de diversification (ligne 19)

TABLE 6.2: Méthodes concurrentes

Label	Méthodologie	Référence
HG2018	VLSN search	Haddadi and Guessoum [83]
Had2019	Three-phase method	Haddadi [87]
GH2019	Two-phase method	Guessoum *et al.* [81]
PPA+Cplex	PPA followed by Cplex on SmallNRP	
Pure PPA	PPA alone	
Cplex 12.6	General purpose IP solver	

6.4.3.1 Les résultats sur un petit échantillon des instances difficiles

Le nombre d'instances à traiter est très important. Pour tester l'efficacité du PPA, on a sélectionné au hasard quelques instances au lieu de traiter l'ensemble des données. Les instances les plus difficiles et les plus importantes sont ceux impliquant n = 100 infirmières. Parmi ces derniers, les plus difficiles sont les cas 2, 4 et 6. Chaque cas contient 7290 instances similaires. On sélectionne au hasard 10 instances parmi 7290 dans chaque cas. De cette manière, on a choisi un échantillon de 30 cas parmi les plus importants et les plus difficiles. À notre avis, cet échantillon parle mieux que les valeurs moyennes obtenues sur l'ensemble du vaste ensemble de données. De plus, on estime que si la méthode proposée fonctionne bien sur cet échantillon, elle doit fonctionner au moins aussi bien sur les autres cas.

Le tableau 6.3 donne les résultats (coût et temps de calcul) de chacune des méthodes concurrentes. Les valeurs en gras se réfèrent à des solutions optimales. Des valeurs moyennes sont données pour chacun des dix instances et pour chacun des trois cas, ces valeurs moyennes représentent l'écart moyen par rapport à l'optimum (en pourcentage) et temps de calcul moyen. De plus, la dernière ligne fournit les valeurs moyennes globales sur les trente instances.

On peut constater à partir du tableau 6.3 que la méthode proposée (appelée PPA+Cplex) avec GH2019 (évidemment avec Cplex) sont les plus précises car elles trouvent toujours la solution optimale. Observez que Had2019 ainsi que l'application du PPA seul sont des méthodes légèrement moins précises, alors que HG2018 est largement dominé du point de vue de la précision.

Les méta-heuristiques sont souvent utilisées pour résoudre des problèmes où le temps est plus important que la qualité de la solution. De ce fait, l'effort de calcul est un facteur fondamental pour la comparaison entre les méta-heuristiques concurrentes. De ce point de vue, Cplex nécessite des temps de calcul importants, et cela n'est pas surprenant puisqu'il s'agit d'un solveur à usage général. Bien que le HG2018 soit environ trois fois plus rapide que le Cplex, il prend encore plus de temps que les autres méthodes. Les deux méthodes basées sur le PPA semblent être les plus rapides, en particulier le PPA pur qui n'utilise pas de solutions élite et donc ne résout pas le problème du SmallNRP. En résumé, il apparaît clairement que le PPA+Cplex est la meilleure méthode en vue de la précision ainsi que le temps de calcul.

TABLE 6.3: Résultats du PPA sur un échantillon d'instances difficiles

	Instance	HG2018		Had2019		GH2019		Pure-PPA		PPA+Cplex		Cplex	
Cas		Coût	Temps	Coût	Temps	Coût	Temps	Coût	Temps	Coût	Temps	Coût	Temps
2	261	1347	8.14	**1345**	1.09	**1345**	0.64	**1345**	0.11	**1345**	0.14	1345	9.03
	1589	**1167**	11.33	**1167**	1.66	**1167**	1.00	**1167**	1.17	**1167**	1.31	1167	7.39
	1613	**957**	9.98	**957**	1.25	**957**	0.77	**957**	0.25	**957**	0.28	957	6.22
	1614	**1441**	9.03	**1441**	1.37	**1441**	0.88	**1441**	0.34	**1441**	0.38	1441	9.67
	2149	1149	9.53	**1142**	1.14	**1142**	0.81	**1142**	0.14	**1142**	0.17	1142	8.83
	2354	1301	9.14	1299	2.84	**1296**	1.64	**1296**	2.70	**1296**	2.81	1296	11.90
	2369	1530	16.34	**1527**	1.50	**1527**	1.01	**1527**	0.80	**1527**	0.84	1527	15.19
	4048	1474	8.58	**1466**	1.78	**1466**	1.03	**1466**	0.49	**1466**	0.53	1466	7.73
	5645	**1542**	23.45	**1542**	1.96	**1542**	1.69	**1542**	0.72	**1542**	0.77	1542	19.28
	7002	1137	9.55	**1105**	1.14	**1105**	1.02	**1105**	0.09	**1105**	0.11	1105	7.66
Moyenne		0.48%	11.51	0.02%	1.57	0.00%	1.05	0.01%	0.68	0.00%	0.73	0.00%	10.29
4	754	**1985**	10.47	1987	2.60	**1985**	2.15	1991	1.86	**1985**	1.94	1985	38.42
	2374	1277	10.59	**1274**	1.64	**1274**	1.44	1275	0.77	**1274**	0.83	1274	66.91
	2916	**1521**	10.77	1522	2.19	**1521**	1.75	**1521**	1.20	**1521**	1.28	1521	113.15
	2944	1387	21.56	**1378**	2.06	**1378**	1.89	**1378**	1.67	**1378**	1.78	1378	115.00
	3995	1107	10.70	**1104**	3.86	**1104**	3.26	**1104**	3.68	**1104**	3.81	1104	40.77
	3996	**1581**	11.11	**1581**	2.79	**1581**	3.16	**1581**	2.08	**1581**	2.22	1581	99.65
	4023	1172	10.86	1151	1.78	**1150**	1.73	1151	0.84	**1150**	0.92	1150	92.67
	5642	1716	23.91	**1711**	3.20	**1711**	2.19	1714	1.84	**1711**	1.94	1711	83.01
	5645	1580	10.92	**1567**	1.92	**1567**	1.39	**1567**	0.97	**1567**	1.02	1567	32.16
	7203	1338	24.64	1337	1.93	**1335**	1.46	1339	0.97	**1335**	1.03	1335	46.47
Moyenne		0.43%	14.55	0.04%	2.40	0.00%	2.04	0.09%	1.59	0.00%	1.68	0.00%	72.82
6	1299	**1175**	12.17	1177	1.50	**1175**	0.86	1177	0.47	**1175**	0.52	1175	39.37
	2072	**991**	12.28	**991**	1.75	**991**	1.44	**991**	0.67	**991**	0.74	991	33.61
	2374	1281	11.75	**1274**	1.86	**1274**	1.35	1275	0.83	**1274**	0.89	1274	30.09
	2375	1354	12.11	**1343**	1.88	**1343**	1.49	1344	0.82	**1343**	0.91	1343	28.00
	3151	1185	25.20	**1172**	2.23	**1172**	1.75	**1172**	1.42	**1172**	1.55	1172	33.50
	3995	**1098**	23.73	**1098**	3.56	**1098**	4.26	1099	3.41	**1098**	3.52	1098	20.11
	4803	1261	24.70	**1257**	2.91	**1257**	2.10	**1257**	1.67	**1257**	1.73	1257	49.03
	4807	**1304**	29.36	**1304**	2.75	**1304**	1.78	**1304**	1.52	**1304**	1.59	1304	58.64
	7209	1329	12.22	1324	2.45	**1322**	1.62	1324	1.17	**1322**	1.22	1322	42.39
	7238	**1183**	12.16	**1183**	2.46	**1183**	1.97	**1183**	1.09	**1183**	1.13	1183	28.88
Moyenne		0.33%	17.57	0.03%	2.34	0.00%	1.86	0.05%	1.31	0.00%	1.38	0.00%	36.36
Moyenne Générale		0.41%	14.54	0.03%	2.10	0.00%	1.65	0.05%	1.19	0.00%	1.26	0.00%	39.82

6.4.3.2 Résultats sur l'ensemble global des données

Dans cette partie, le test est effectué sur la totalité de la base de données. Le tableau 6.4 présent les résultats. La colonne intitulée "Coût" fait référence à la valeur moyenne de la fonction objective, "%Feas" indique la proportion d'instances faisables. "Temps" représente le temps d'exécution moyen.

Trois méthodes concurrentes (voir tableau 6.2) sont comparées à la méthode proposé (PPA+Cplex). La comparaison est présentée dans le tableau 6.4. Les valeurs moyennes sont indiquées en gras.

On observe que HG2018 a le meilleur taux de faisabilité (ce qui est vrai pour le NRP du fait que la méthode fait appel à Cplex pour trouver les solutions initiales) suivie par GH2019 qui a aussi le taux de faisabilité réel puisque RNRP est résolu avec CPLEX directement. Pour cela, la comparaison équitable ne peut être faite qu'entre Had 2019 et PPA+Cplex, on observe que Had 2019 à un taux de faisabilité légèrement inférieur à celui des autres méthodes en raison de la difficulté pour trouver les solutions initiale pour RNRP.

Les temps de calcul sont rapportés dans la deuxième colonne, puisque toutes les méthodes sont exécutées sur la même machine, on peut dire que GH2019 est le plus rapide alors que PPA+Cplex et Had 2019 sont presque équivalents et beaucoup plus rapides que HG2018. En ce qui concerne les temps de calcul, la conclusion est simple. Mais concernant la qualité de la solution, la comparaison n'est pas si évidente. À l'exception de la méthode HG2018 qui est dominée par les autres méthodes, les méthodes Had 2019, GH2019 et PPA+Cplex ne présentent aucune différence significative entre elles. Cependant, notre méthode reste significative du fait que les solutions trouvées ne s'écartent pas de plus de $0,17\%$ en moyenne des meilleures solutions.

TABLE 6.4: Résultats des méthodes concurrentes (moyennes sur 7290 instances)

Instance		HG2018			H2019			GH2019			PPA+Cplex		
n	cas	coût	Temps	%feas	coût	Temps	%feas	coût	Temps	%feas	coût	Temps	%feas
25	1	250,43	0,65	88,27	250,28	0,22	87,96	250,18	0,10	87,97	250,25	0,20	87,97
	2	239,46	1,01	88,27	239,36	0,41	88,15	239,28	0,18	88,16	239,35	0,41	88,15
	3	265,72	0,61	88,09	265,46	0,13	87,11	265,40	0,06	87,12	265,46	0,09	87,12
	4	248,64	0,88	88,27	248,37	0,39	88,00	248,34	0,17	88,00	248,42	0,36	88,00
	5	263,16	0,58	85,88	262,93	0,12	85,10	262,84	0,05	85,13	262,93	0,08	85,13
	6	240,44	0,91	88,27	240,33	0,40	88,13	240,26	0,18	88,15	240,32	0,40	88,13
	7	279,21	0,55	80,18	278,75	0,05	78,60	278,63	0,02	78,63	278,73	0,03	78,61
	8	256,18	0,62	85,69	255,95	0,10	84,92	255,86	0,04	84,94	255,94	0,07	84,92
	Moyenne	**255,41**	**0,73**	**86,62**	**255,18**	**0,23**	**86,00**	**255,10**	**0,10**	**86,01**	**255,18**	**0,20**	**86,00**
50	1	499,10	1,10	90,03	499,18	0,39	89,66	499,09	0,20	89,86	499,12	0,36	89,73
	2	477,66	1,64	90,03	477,77	0,65	89,85	477,67	0,31	89,92	477,73	0,66	89,88
	3	524,15	0,79	89,82	523,69	0,22	88,49	523,80	0,12	89,15	523,76	0,23	88,79
	4	495,78	1,67	90,03	495,92	0,64	89,62	495,79	0,32	89,82	495,84	0,61	89,70
	5	521,84	0,75	85,34	521,47	0,20	83,95	521,57	0,10	84,68	521,59	0,24	84,44
	6	479,82	1,63	90,03	479,94	0,65	89,82	479,81	0,31	89,89	479,85	0,65	89,86
	7	547,07	0,57	78,61	546,24	0,08	76,32	546,60	0,04	77,42	546,42	0,11	76,87
	8	507,17	0,73	85,65	506,96	0,17	84,61	506,95	0,08	85,03	506,95	0,16	84,73
	Moyenne	**506,57**	**1,11**	**87,44**	**506,40**	**0,38**	**86,54**	**506,41**	**0,19**	**86,97**	**506,41**	**0,38**	**86,75**
75	1	755,75	1,41	88,70	755,92	0,63	88,23	755,79	0,36	88,63	755,67	0,58	88,33
	2	732,54	2,28	88,70	732,57	1,02	88,48	732,59	0,54	88,68	732,60	0,98	88,55
	3	793,78	0,95	88,57	793,38	0,36	87,15	793,62	0,20	88,05	793,40	0,35	87,26
	4	745,49	2,19	88,70	745,48	1,01	88,13	745,55	0,54	88,63	745,56	0,96	88,26
	5	792,52	0,92	86,09	792,29	0,34	84,60	792,53	0,18	85,82	792,25	0,44	84,97
	6	733,88	2,27	88,70	734,09	1,02	88,45	733,92	0,53	88,68	733,94	1,01	88,56
	7	833,79	0,65	79,93	831,83	0,13	76,83	833,08	0,06	78,79	832,73	0,28	78,00
	8	777,15	0,90	85,93	777,00	0,29	84,76	777,09	0,15	85,61	776,90	0,32	85,05
	Moyenne	**770,61**	**1,45**	**86,91**	**770,32**	**0,60**	**85,83**	**770,52**	**0,32**	**86,61**	**770,38**	**0,62**	**86,12**
100	1	1214,29	1,98	90,53	1214,36	0,96	89,38	1214,31	0,52	90,38	1214,25	1,10	89,64
	2	1173,87	3,03	90,53	1174,03	1,57	89,81	1173,94	0,76	90,43	1173,69	1,52	89,95
	3	1287,14	1,32	90,38	1286,93	0,59	88,02	1287,08	0,30	89,96	1286,75	0,95	88,67
	4	1201,19	3,10	90,53	1201,31	1,57	89,49	1201,22	0,77	90,38	1201,00	1,75	89,71
	5	1265,38	1,24	86,54	1264,02	0,51	84,05	1265,18	0,25	86,05	1264,07	0,80	84,62
	6	1176,61	3,09	90,53	1176,34	1,51	89,68	1176,67	0,75	90,43	1176,42	1,55	89,93
	7	1332,23	0,83	80,34	1330,02	0,17	76,38	1331,72	0,08	79,33	1330,63	0,55	77,76
	8	1242,86	1,21	86,94	1241,98	0,42	84,95	1242,59	0,21	86,32	1241,63	0,54	85,20
	Moyenne	**1236,70**	**1,98**	**88,29**	**1236,12**	**0,91**	**86,47**	**1236,59**	**0,46**	**87,91**	**1236,05**	**1,10**	**86,94**

6.5 Conclusion

Pour résoudre le NRP, une méta-heuristique à trois phases est proposée.

En commençant par une procédure efficace et prometteuse de fixation des variables, éliminant jusqu'à 90% des variables du problème, sans sacrifier la qualité de la solution. Ensuite un algorithme de propagation des plantes (PPA) simple et facile à implémenter est conçu pour résoudre le problème résultant RNRP. Enfin, les solutions élite obtenues par le PPA peuvent aider à réduire encore le RNRP et à obtenir un NRP très dispersé qui peut être résolu directement par un solveur MIP à usage général. L'application séquentielle des trois phases ci-dessus aboutit à une méthode efficace pour résoudre le NRP, dominant deux méthodes récemment publiées (Haddadi et Guessoum, 2018 ; Haddadi, 2018b). Avant de conclure, en soulignant trois points avantageux de notre contribution :

— L'heuristique générique de fixation de variables a prouvé encore une fois son importance et son efficacité.
— PPA confirme ses qualités de méta-heuristique performante.
— L'utilisation de solutions élite est très avantageuse.

Conclusion générale

On a étudié le problème de l'élaboration d'un planning pour la rotation des infirmières (NRP), c'est un problème d'optimisation combinatoire NP-dur.

Une introduction à l'optimisation combinatoire et à la théorie de la complexité est présenté dans le chapitre 1 qui sert à situer le contexte de ce mémoire. suivie d'une description du problème étudié dans le chapitre 2, où un état de l'art est proposé. On a ensuite présenté la panoplie des techniques utilisées dans notre étude au chapitre 3. Dans les trois chapitres suivants on a détaillé nos contributions.

Dans le chapitre 4 , on a proposé une méta-heuristique de recherche dans le voisinage à très grande échelle (VLSN) guider par une méthode d'optimisation de sous gradient. Cette approche est appréciée par une expérimentation numérique rigoureuse, avec des comparaisons à des méthodes existantes dans l'état de l'art, à l'issue desquelles notre méthode prouve sa supériorité. Ce travail a fait l'objet d'une publication dans la revue ≪American Journal of Mathematical and Management Sciences≫ publie par Taylor & Francis [83].

Dans Le chapitre 5 on a présenté une nouvelle méthode à deux phases. La première phase utilise une méthode de sous-gradient appliquée à la relaxation lagrangienne des contraintes de couverture. À partir des informations fournies, une heuristique de fixation des variables (VFH) réduit la taille du problème. La deuxième phase utilise un solveur MIP (Cplex) pour résoudre le NRP réduit (RNRP). Une heuristique gloutonne est conçue pour trouver une solution initiale. La méthode à deux phases proposée testée sur l'ensemble de données NSPLib, fournit des résultats intéressants. La comparaison montre que la nouvelle contribution est en concurrence avec quatre méthodes publiées récemment. Cette approche aussi a fait l'objet d'une publication dans la revue ≪ American Journal of Mathematical and Management Sciences ≫ [81].

Dans Le chapitre 6 on a suggéré une méta-heuristique de la propagation des plantes (PPA) pour résoudre un problème réduit générer à l'aide de l'heuristique de fixation des variables présentée dans le chapitre 5, le PPA génère des solutions élites qui seront utilisées pour diminuer la taille du problème qui sera enfin résolue par un solveur MIP (Cplex). l'efficacité et la compétitivité de l'approche proposée est justifiée par une expérimentation numérique rigoureuse.

Voici quelques axes pouvant faire l'objet de recherches futures :

1. Montrer que nos approches sont génériques, et peuvent être appliquées au problème général de l'optimisation combinatoire.
2. Il serait intéressant de tester nos approches avec d'autres bases de données et sous d'autres contraintes.
3. Les problèmes multi-objectives ne sont pas traités dans ce travail, et pourraient être incorporés pour les futurs travaux.

Bibliographie

[1] M Abdel-Basset, G Manogaran, L Abdel-Fatah, and S Mirjalili. An improved nature inspired meta-heuristic algorithm for 1-d bin packing problems. *Personal and Ubiquitous Computing*, 22(5-6) :1117–1132, 2018.

[2] R Agarwal. *Solving parallel machine scheduling problems with variable depth local search*. PhD thesis, School of Mathematics, University of Southampton, 2004.

[3] R K Ahuja, J B Orlin, and D Sharma. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91(1) :71–97, 2001.

[4] R K Ahuja, Ö Ergun, J B Orlin, and A P Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1) :75–102, 2002.

[5] R K Ahuja, J B Orlin, and D Sharma. A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem. *Operations Research Letters*, 31(3) :185–194, 2003.

[6] R K Ahuja, Ö Ergun, J B Orlin, and A P Punnen. Very large-scale neighborhood search : Theory, algorithms, and applications. *Handbook of Approximation Algorithms and Metaheuristics*, 10, 2007.

[7] R K Ahuja, K C Jha, J B Orlin, and D Sharma. Very large-scale neighborhood search for the quadratic assignment problem. *Informs journal on computing*, 19(4) :646–657, 2007.

[8] U Aickelin and K Dowsland. Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, 3(3) :139–153, 2000.

[9] U Aickelin and K A Dowsland. An indirect genetic algorithm for a nurse-scheduling problem. *Computers & Operations Research*, 31(5) :761–778, 2004.

[10] S Akyol and B Alatas. Plant intelligence based metaheuristic optimization algorithms. *Artificial Intelligence Review*, 47(4) :417–462, 2017.

[11] S K Amponsah, E Agyeman, and K G Okrah. Graph colouring, an approach to nurses scheduling, case study : Ejura district hospital, ashanti region, ghana. *American-Eurasian Journal of Scientific Research*, 6(1) :01–05, 2011.

[12] K Anwar, M Awadallah, A Khader, and M Al-Betar. Hyper-heuristic approach for solving nurse rostering problem. In *Computational Intelligence in Ensemble Learning (CIEL), 2014 IEEE Symposium on*, pages 1–6. IEEE, 2014.

[13] Y Z Arajy, S Abdullah, and S Kifah. Non-liner great deluge algorithm for handling nurse rostering problem. *International Journal of Applied Engineering Research*, 12(15) :4959–4966, 2017.

[14] S Asta, E Özcan, and T Curtois. A tensor based hyper-heuristic for nurse rostering. *Knowledge-based systems*, 98 :185–199, 2016.

[15] M A Awadallah, A L Bolaji, and M A Al-Betar. A hybrid artificial bee colony for a nurse rostering problem. *Applied Soft Computing*, 35 :726–739, 2015.

[16] M A Awadallah, M A Al-Betar, A T Khader, A L Bolaji, and M Alkoffash. Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem. *Neural Computing and Applications*, 28(3) :463–482, 2017.

[17] J Baeklund. Nurse rostering at a danish ward. *Annals of Operations Research*, 222(1) :107–123, dec 2013.

[18] R Bai and J Xie. Heuristic algorithms for simultaneously accepting and scheduling advertisements on broadcast television. *Journal of Information and Computer Science*, 1(4) :245–251, 2006.

[19] R Bai, E K Burke, G Kendall, J Li, and B McCollum. A hybrid evolutionary approach to the nurse rostering problem. *IEEE Transactions on Evolutionary Computation*, 14(4) :580–590, aug 2010.

[20] J F Bard and H W Purnomo. Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164(2) :510–534, 2005.

[21] J Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31(1) :85–93, 1987.

[22] J Beasley. Lagrangian relaxation. In Colin R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, chapter 6, pages 243–303. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[23] F Bellanti, G Carello, F Della-Croce, and R Tadei. A tabu search approach to a nurse rostering problem. In *Proceedings of the 4'h Metaheuristics Inter-national Conference, MIC*, 2001.

[24] M Bellare and S Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1) :97–119, 1994.

[25] B Bilgin, P De Causmaecker, B Rossie, and G V Berghe. The problem description and a solution method for the nurse rostering problem in belgian hospitals. In *Proceedings of the 22nd Annual Conference of the Belgian Operations Research Society (ORBEL 22)*, volume 22, pages 37–38, 2008.

[26] B Bilgin, P De Causmaecker, and G V Berghe. A hyperheuristic approach to belgian nurse rostering problems. In *Proceedings of the 4th Multidisciplinary International Conference on Scheduling : Theory and Applications*, pages 693–695, 2009.

[27] B Bilgin, P De Causmaecker, B Rossie, and G V Berghe. Local search neighbourhoods for dealing with a novel nurse rostering model. *Annals of Operations Research*, 194(1) :33–57, nov 2010.

[28] B Bilgin, P Demeester, M Mısır, W Vancroonenburg, G V Berghe, and T Wauters. A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT'10)*, 2010.

[29] N Boland, M Hewitt, D M Vu, and M Savelsbergh. Solving the traveling salesman problem with time windows through dynamically generated time-expanded networks. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 254–262. Springer, 2017.

[30] O Borzog-Haddad, M Solgi, and H A Loiciga. *Metaheuristic and evolutionary algorithms for engineering optimization*, volume 294. John Wiley & Sons, 2017.

[31] I Boussaid. *de métaheuristiques pour l'optimisation continue*. PhD thesis, Université Paris-Est ; Université des sciences et de la technologie Houari Boumédienne (Alger), 2013.

[32] A Brabazon, M O'Neill, and S McGarraghy. Plant-inspired algorithms. In *Natural Computing Algorithms*, pages 455–477. Springer, 2015.

[33] P Brucker, E K Burke, T Curtois, R Qu, and G V Berghe. A shift sequence based approach for nurse scheduling and a new benchmark dataset. *Journal of Heuristics*, 16(4) :559–573, nov 2008.

[34] P Brucker, R Qu, and E Burke. Personnel scheduling : Models and complexity. *European Journal of Operational Research*, 210(3) :467–473, 2011.

[35] E K Burke and T Curtois. New computational results for nurse rostering benchmark instances. Technical report, Technical report, 2011.

[36] E K Burke and T Curtois. New approaches to nurse rostering benchmark instances. *European Journal of Operational Research*, 237(1) :71–81, aug 2014.

[37] E K Burke, G Kendall, and E Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6) :451–470, 2003.

[38] E K Burke, P De Causmaecker, G V Berghe, and H P Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6) :441–499, nov 2004.

[39] E K Burke, P De Causmaecker, S Petrovic, and G V Berghe. Variable neighborhood search for nurse rostering problems. In *Metaheuristics : computer decision-making*, pages 153–172. Springer, 2004.

[40] E K Burke, T Curtois, R Qu, and G V Berghe. A scatter search for the nurse rostering problem. *School of Computer Science, University of Nottingham, Tech. Rep*, 2007.

[41] E K Burke, T Curtois, and G V Berghe. Problem model for nurse rostering benchmark instances. *ASAP, School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, UK*, 2008.

[42] E K Burke, T Curtois, R Qu, and G V Berghe. A scatter search methodology for the nurse rostering problem. *Journal of the Operational Research Society*, 61(11) :1667–1679, 2010.

[43] E K Burke, J Li, and R Qu. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research*, 203(2) :484–493, jun 2010.

[44] E K Burke, T Curtois, L F van Draat, J K van Ommeren, and G Post. Progress control in iterated local search for nurse rostering. *Journal of the Operational Research Society*, 62(2) :360–367, 2011.

[45] S Ceschia and A Schaerf. Solving the INRC-II nurse rostering problem by simulated annealing based on large neighborhoods. In *Proceedings of the 12th international conference on the practice and theory of automated timetabling*, pages 331–338, 2018.

[46] S Ceschia, N T T Dang, P De Causmaecker, S Haspeslagh, and A Schaerf. Second international nurse rostering competition (INRC-II)—problem description and rules—. *arXiv preprint arXiv :1501.04177*, 2015.

[47] I Charon, A Germa, and O Hudry. *Méthodes d'optimisation combinatoire*. Masson Paris, 1996.

[48] B Cheang, H Li, A Lim, and B Rodrigues. Nurse rostering problems—-a bibliographic survey. *European Journal of Operational Research*, 151(3) :447–460, 2003.

[49] J Chen and T Yeung. Development of a hybrid expert system for nurse shift scheduling. *International Journal of Industrial Ergonomics*, 9(4) :315–327, 1992.

[50] B M W Cheng, J Lee, and J Wu. A constraint-based nurse rostering system using a redundant modeling approach. In *Tools with Artificial Intelligence, 1996., Proceedings Eighth IEEE International Conference on*, pages 140–148. IEEE, 1996.

[51] M Cheraitia and S Haddadi. Simulated annealing for the uncapacitated exam scheduling problem. *International Journal of Metaheuristics*, 5(2) :156–170, 2016.

[52] M Cheraitia, S Haddadi, and A Salhi. Hybridising plant propagation and local search for uncapacitated exam scheduling problems. *International Journal of Services and Operations Management*, 32(4) :450–467, 2019.

[53] A A Constantino, D Landa-Silva, E L de Melo, C F X de Mendonça, D B Rizzato, and W Romão. A heuristic algorithm based on multi-assignment procedures for nurse scheduling. *Annals of Operations Research*, apr 2013.

[54] N T T Dang, S Ceschia, A Schaerf, P De Causmaecker, and S Haspeslagh. Solving the multi-stage nurse rostering problem. In E K Burke, L Di Gaspero, E Özcan, B McCollum, and A Schaerf, editors, *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling, 23-26 Aug., Udine, Italy*, pages 473–475, 2016.

[55] C Darwin. *On the Origin of Species by Means of Natural Selection.* John Murray, London, 1859.

[56] P De Causmaecker and G V Berghe. A categorisation of nurse rostering problems. *Journal of Scheduling*, 14(1) :3–16, 2011.

[57] V G Deineko and G J Woeginger. A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. *Mathematical programming*, 87(3) :519–542, 2000.

[58] S J M Den Hartog. On the complexity of nurse scheduling problems. Master's thesis, Faculty of Science - Utrecht University, The Netherlands, 2016.

[59] M Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* PhD thesis, Friedrich-Schiller-Universität, Jena, Germany, 2008.

[60] O Dominguez, D Guimarans, A A Juan, and I de la Nuez. A biased-randomised large neighbourhood search for the two-dimensional vehicle routing problem with backhauls. *European Journal of Operational Research*, 255(2) :442–462, dec 2016.

[61] J J Dongarra. Performance of various computers using standard linear equations software. Technical Report CS - 89 - 85, University of Manchester, June 2014.

[62] U Dorndorf and E Pesch. Fast clustering algorithms. *ORSA Journal on Computing*, 6(2) :141–153, 1994.

[63] F Dusberger and G R Raidl. A variable neighborhood search using very large neighborhood structures for the 3-staged 2-dimensional cutting stock problem. In *International Workshop on Hybrid Metaheuristics*, pages 85–99. Springer, 2014.

[64] Ö Ergun and J B Orlin. A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem. *Discrete Optimization*, 3(1) :78–85, 2006.

[65] Ö Ergun. *New neighborhoods search algorithms based on exponentially large neighborhoods*. PhD thesis, Massachussets Institute of Technology, 2001.

[66] A Ernst, H Jiang, M Krishnamoorthy, and D Sier. Staff scheduling and rostering : A review of applications, methods and models. *European journal of operational research*, 153(1) :3–27, 2004.

[67] M L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27 :1–18, 1981.

[68] M L Fisher. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15 :10–21, 1985.

[69] A S Fraenkel and D Lichtenstein. Computing a perfect strategy for nxn chess requires exponential time. *Journal of Combinatorial Theory*, A31 :199–214, 1981.

[70] V François, Y Arda, Y Crama, and G Laporte. Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, 255(2) :422–441, dec 2016.

[71] M R Garey and D S Johnson. *Computers and intractability*. wh freeman New York, 1979.

[72] R Geleijn, M van der Meer, Q Van der Post, and D van der Berg. The plant propagation algorithm on timetables : First results. *EVO* 2019*, page 2, 2019.

[73] A M Geoffrion. Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2 :82–114, 1974.

[74] A Ghoniem, T Flamand, and M Haouari. Optimization-based very large-scale neighborhood search for generalized assignment problems with location/allocation considerations. *INFORMS Journal on Computing*, 28(3) :575–588, 2016.

[75] S Gilroy. Plant tropisms. *Current Biology*, 18(7) :275–277, 2008.

[76] C Glass and R Knight. The nurse rostering problem : A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2) :379–389, 2010.

[77] F Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5) :533–549, 1986.

[78] F Glover. Scatter search and path relinking. *New ideas in optimization*, pages 297–316, 1999.

[79] P Grangier, M Gendreau, Lehuédé, F , and L M Rousseau. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, 254(1) :80–91, oct 2016.

[80] A. Gretsista and E. K. Burke. An iterated local search framework with adaptive operator selection for nurse rostering. In R Battiti, D E Kvasov, and Y D Sergeyev, editors, *Learning and Intelligent Optimization*, pages 93–108, Cham, 2017. Springer International Publishing.

[81] F Guessoum, S Haddadi, and E Gattal. Simple, yet fast and effective two-phase method for nurse rostering. *American Journal of Mathematical and Management Sciences*, pages 1–19, 2019.

[82] W J Gutjahr and M S Rauner. An ACO algorithm for a dynamic regional nurse-scheduling problem in austria. *Computers & Operations Research*, 34(3) :642–666, mar 2007.

[83] S Haddadi and F Guessoum. Hybridizing subgradient optimization and very large scale neighborhood search for nurse rostering. *American Journal of Mathematical and Management Sciences*, 37(4) :344–357, 2018.

[84] S Haddadi, M Cheraitia, and A Salhi. A two-phase heuristic for set covering. *International Journal of Mathematics in Operational Research*, 35(1) :61–78, 2018.

[85] S Haddadi. Simple lagrangian heuristic for the set covering problem. *European Journal of Operational Research*, 97(1) :200–204, 1997.

[86] S Haddadi. Benders decomposition for set covering problems. *Journal of Combinatorial Optimization*, 33(1) :60–80, 2017.

[87] S Haddadi. Three-phase method for nurse rostering. *International Journal of Management Science and Engineering Management*, 14(3) :193–205, 2019.

[88] S Haddadi. Variable-fixing then subgradient optimization guided very large scale neighborhood search for the generalized assignment problem. *4OR*, 17(3) :261–295, 2019.

[89] M Hadwan, M Ayob, N Sabar, and R Qu. A harmony search algorithm for nurse rostering problems. *Information Sciences*, 233 :126–140, 2013.

[90] J K Hao, P Galinier, and M Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle*, 13(2) :283–324, 1999.

[91] S Haspeslagh, T Messelis, G V Berghe, and P De Causmaecker. An efficient translation scheme for representing nurse rostering problems as satisfiability problems. In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, pages 303–310, 2013.

[92] S Haspeslagh, P De Causmaecker, A Schaerf, and M Stølevik. The first international nurse rostering competition 2010. *Annals of Operations Research*, 218(1) :221–236, 2014.

[93] F He and R Qu. A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research*, 39(12) :3331–3343, dec 2012.

[94] J He, P Flener, and J Pearson. Solution neighbourhoods for constraint-directed local search. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 74–79. ACM, 2012.

[95] M Held and R M Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6) :1138–1162, 1970.

[96] C Hicks and D Hennessy. A task-based approach to defining the role of the nurse practitioner : the views of uk acute and primary sector nurses. *Journal of Advanced Nursing*, 29(3) :666–673, 1999.

[97] S C Ho and W Szeto. A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem. *Transportation Research Part B : Methodological*, 95 :340–363, 2017.

[98] ILOG, Inc. ILOG CPLEX 12.6 : High-performance software for mathematical programming and optimization, 2013. See http://www.ilog.com/products/cplex/.

[99] G M Jaradat, A Al-Badareen, M Ayob, M Al-Smadi, I Al-Marashdeh, M Ash-Shuqran, and E Al-Odat. Hybrid elitist-ant system for nurse-rostering problem. *Journal of King Saud University-Computer and Information Sciences*, 2018.

[100] R G Jeroslow. There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research*, 21 :221–224, 1972.

[101] R C Jeroslow. There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research*, 21(1) :221–224, 1973.

[102] K C Jha. *Very large-scale neighborhood search heuristics for combinatorial optimization problems*. PhD thesis, University of Florida, 2004.

[103] H Jin, G Post, and E Van der Veen. ORTEC's contribution to the second international nurse rostering competition. In E K Burke, L Di Gaspero, E Özcan, B McCollum, and A Schaerf, editors, *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling, 23-26 Aug., Udine, Italy*, pages 499–501, 2016.

[104] S H Jin, H Y Yun, S J Jeong, and K S Kim. Hybrid and cooperative strategies using harmony search and artificial immune systems for solving the nurse rostering problem. *Sustainability*, 9(7) :1090, 2017.

[105] L Jourdan. *Métaheuristiques pour l'extraction de connaissances : Application à la génomique*. PhD thesis, Université des Sciences et Technologie de Lille-Lille I, 2003.

[106] A Kheiri, E Özcan, R Lewis, and J Thompson. A sequence-based selection hyper-heuristic - a case study in nurse rostering. In E K Burke, L Di Gaspero, E Özcan, B McCollum, and A Schaerf, editors, *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling, 23-26 Aug., Udine, Italy*, pages 503–505, 2016.

[107] S Kirkpatrick, C D Gelatt, M P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598) :671–680, 1983.

[108] K Komarudin, M A Guerry, T De-Feyter, and G V Berghe. The roster quality staffing problem – a methodology for improving the roster quality by modifying the personnel structure. *European Journal of Operational Research*, 230(3) :551–562, 2013.

[109] K Komarudin, M A Guerry, P Smet, T De-Feyter, and G V Berghe. A two-phase heuristic and a lexicographic rule for improving fairness in personnel rostering. In E Özcan, E K Burke, and B McCollum, editors, *Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling, 26-29 Aug., York, UK*, pages 292–308, 2014.

[110] J Kytöjoki, T Nuortio, O Bräysy, and M Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & operations research*, 34(9) :2743–2757, 2007.

[111] A Layeb. *Utilisation des Approches d'Optimisation Combinatoire pour la Vérification des Applications Temps Réel.* PhD thesis, Université Mentouri de Constantine, 2010.

[112] J Leeuwen. *Handbook of theoretical computer science*, volume 1. Elsevier, 1990.

[113] A Legrain, J Omer, and S Rosat. A rotation-based branch-and-price approach for the nurse scheduling problem. *Working paper, available at https ://hal.archivesouvertes.fr/hal-01545421*, 2017.

[114] H Li, A Lim, and B Rgues. A hybrid ai approach for nurse rostering problem. In *Proceedings of the 2003 ACM symposium on Applied computing*, pages 730–735. ACM, 2003.

[115] Y Li, Y Tao, and F Wang. A compromised large-scale neighborhood search heuristic for capacitated air cargo loading planning. *European Journal of Operational Research*, 199(2) :553–560, 2009.

[116] Y Li, H Chen, and C Prins. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. *European Journal of Operational Research*, 252(1) :27–38, 2016.

[117] X Liu, G Laporte, Y Chen, and R He. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Computers & Operations Research*, 2017.

[118] Z Liu, Z Liu, Z Zhu, Y Shen, and J Dong. Simulated annealing for a multi-level nurse rostering problem in hemodialysis service. *Applied Soft Computing*, 64 :148–160, 2018.

[119] A Lodi, M Monaci, and E Pietrobuoni. Partial enumeration algorithms for two-dimensional bin packing problem with guillotine constraints. *Discrete Applied Mathematics*, 217 :40–47, 2017.

[120] H R Lourenço, O C Martin, and T Stützle. Iterated local search : Framework and applications. In *Handbook of metaheuristics*, pages 129–168. Springer, 2019.

[121] Z Lü and J Hao. Adaptive neighborhood search for nurse rostering. *European Journal of Operational Research*, 218(3) :865–876, 2012.

[122] B Maenhout and M Vanhoucke. New computational results for the nurse scheduling problem : A scatter search algorithm. In *Evolutionary Computation in Combinatorial Optimization*, pages 159–170. Springer Nature, 2006.

[123] B Maenhout and M Vanhoucke. A branch-and-price procedure for nurse staffing incorporating roster preferences. In *3rd Multidisciplinary International Conference on Scheduling : Theory and Applications,, August 28–31.*, 2007.

[124] B Maenhout and M Vanhoucke. An electromagnetic meta-heuristic for the nurse scheduling problem. *Journal of Heuristics*, 13(4) :359–385, apr 2007.

[125] B Maenhout and M Vanhoucke. Comparison and hybridization of crossover operators for the nurse scheduling problem. *Annals of Operations Research*, 159(1) :333–353, 2008.

[126] M Mahi, Ö K Baykan, and H Kodaz. A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. *Applied Soft Computing*, 30 :484–490, 2015.

[127] M l Marmion. *Recherche locale et optimisation combinatoire : de l'analyse structurelle d'un problème à la conception d'algorithmes efficaces.* PhD thesis, Université des Sciences et Technologie de Lille-Lille I, 2011.

[128] S Martin, D Ouelhadj, P Smet, G Vanden Berghe, and E Özcan. Cooperative search for fair nurse rosters. *Expert Systems with Applications*, 40(16) :6674–6683, 2013.

[129] J Menana, S Demassey, and N Jussien. Modélisation et optimisation des préférences en planification de personnel. Technical report, Technical Report Research Report 11-01-INFO, École des Mines de Nantes, 2010 . . . , 2010.

[130] T Messelis and P De Causmaecker. An algorithm selection approach for nurse rostering. In *Proceedings of the 23rd Benelux Conference on Artificial Intelligence*, pages 160–166. Nevelland, 2011.

[131] A R Meyer and L J Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. *13th annual conference on switching and automata theory*, pages 125–129, 1972.

[132] C Meyers and J B Orlin. Very large-scale neighborhood search techniques in timetabling problems. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 24–39. Springer, 2006.

[133] F Mischek and N Musliu. Integer programming and heuristic approaches for a multi-stage nurse rostering problem. In E K Burke, L Di Gaspero, E Özcan, B McCollum, and A Schaerf, editors, *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling, 23-26 Aug., Udine, Italy*, pages 245–262", 2016.

[134] F Mischek and N Musliu. Integer programming model extensions for a multi-stage nurse rostering problem. *Annals of operations research*, 275(1) :123–143, 2019.

[135] M Moz and M V Pato. A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research*, 34(3) :667–691, 2007.

[136] S Nag. Adaptive plant propagation algorithm for solving economic load dispatch problem. *arXiv preprint arXiv :1708.07040*, 2017.

[137] G L Nemhauser. The age of optimization : Solving large-scale real-world problems. *Operations Research*, 42 :5–14, 1994.

[138] Y Nie, B Wang, and X Zhang. Hybrid harmony search algorithm for nurse rostering problem. In *Harmony Search Algorithm*, pages 109–120. Springer, 2016.

[139] M Okada and M Okada. Prolog-based system for nursing staff scheduling implemented on a personal computer. *Computers and Biomedical Research*, 21(1) :53–63, 1988.

[140] T Osogami and H Imai. Classification of various neighborhood operations for the nurse scheduling problem. In *Algorithms and Computation*, pages 72–83. Springer Nature, 2000.

[141] E Özcan. Memetic algorithms for nurse rostering. In *Computer and Information Sciences-ISCIS 2005*, pages 482–492. Springer, 2005.

[142] M Paauw and D Van Den Berg. Paintings, polygons and plant propagation. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, pages 84–97. Springer, 2019.

[143] C H Papadimitriou and K Steiglitz. *Combinatorial optimization : algorithms and complexity*. Courier Corporation, 1982.

[144] C H Papadimitriou. *The complexity of combinatorial optimization problems*. PhD thesis, Princeton University, 1976.

[145] C H Papadimitriou. *Computational Complexity*. Prentice-Hall, 1994.

[146] C H Papadimitriou. Np-completeness : A retrospective. In *In Proceedings of the 24th International Colloquium on Automata, Lan guages, and Programming (ICALP '97)*, pages 2–6. Springer, 1997.

[147] P M Pardalos and M G C Resende. Handbook of applied optimization. 2002.

[148] D Pisinger and S Ropke. Large neighborhood search. In *Handbook of Metaheuristics*, pages 399–419. Springer Nature, 2010.

[149] G Post and B Veltman. Harmonious personnel scheduling. In *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling. PATAT*, pages 557–559, 2004.

[150] M Posta, J Ferland, and P Michelon. An exact method with variable fixing for solving the generalized assignment problem. computational optimization and applications. *Computational Optimization and Applications*, 52(3) :629–644, 2012.

[151] P Putz. Subgradient optimization based lagrangian relaxation and relaxand-cut approaches for the bounded diameter minimum spanning tree problem. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, 2007.

[152] M Quiroz-Castellanos, L C Reyes, J T Jimenez, C Gómez, H J F Huacuja, and A C F Alvim. A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research*, 55 :52–64, 2015.

[153] E Rahimian, K Akartunali, and J Levine. A hybrid integer and constraint programming approach to solve nurse rostering problems. *Computers & Operations Research*, 82(Supplement C) :83–94, 2017.

[154] R Ramli and S Ahmad. Innovative neighbor generations in tabu search technique for a nurse rostering problem. *International Proceedings of Computer Science and Information Technology : Information and Electronics Engineering (ICIEE)*, 2011.

[155] G M Ribeiro and G Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & operations research*, 39(3) :728–735, 2012.

[156] C C Ribeiro and N Maculan. *Applications of combinatorial optimization*, volume 50. JC Baltzer, 1994.

[157] M Römer and T Mellouli. A direct milp approach based on state-expanded network flows and anticipation for multi-stage nurse rostering under uncertainty. In *Proceedings of the 11th international conference on the practice and theory of automated timetabling*, pages 549–551, 2016.

[158] S Ropke and D Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4) :455–472, 2006.

[159] Ö Şafak, Ö Çavuş, and M S Aktürk. Multi-stage airline scheduling problem with stochastic passenger demand and non-cruise times. *Transportation Research Part B : Methodological*, 114 :39–67, 2018.

[160] F Salassa and G V Berghe. A stepping horizon view on nurse rostering. In *Proceedings of the 9th international conference on the practice and theory of automated timetabling.*, pages 161–173, 2012.

[161] A Salhi and E S Fraga. Nature-inspired optimisation approaches and the new plant propagation algorithm. In *Proc. of the International Conference on Numerical Analysis and Optimization*, pages K2.1–K2.8. 2011.

[162] H G Santos, T A M Toffolo, R A M Gomes, and S Ribas. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, 239(1) :225–251, may 2014.

[163] B I Selamoglu and A Salhi. The plant propagation algorithm for discrete optimisation : The case of the travelling salesman problem. In *Nature-Inspired Computation in Engineering*, pages 43–61. Springer Nature, 2016.

[164] J F Shapiro. A survey of lagrangean techniques for discrete optimization. In *Annals of Discrete Mathematics*, volume 5, pages 113–138. Elsevier, 1979.

[165] P Smet, S Martin, D Ouelhadj, E Özcan, and G V Berghe. Investigation of fairness measures for nurse rostering. In *Proceedings of the 9th international conference on the practice and theory of automated timetabling*, pages 369–372, 2012.

[166] P Smet, P De Causmaecker, B Bilgin, and G V Berghe. Nurse rostering : a complex example of personnel scheduling with perspectives. In *Automated Scheduling and Planning*, pages 129–153. Springer, 2013.

[167] P Smet, B Bilgin, P De Causmaecker, and G V Berghe. Modelling and evaluation issues in nurse rostering. *Annals of Operations Research*, 218(1) :303–326, 2014.

[168] P Smet, P Brucker, P De Causmaecker, and G V Berghe. Polynomially solvable personnel rostering problems. *European Journal of Operational Research*, 249(1) :67–75, 2016.

[169] L D Smith and A Wiggins. A computer-based nurse scheduling system. *Computers & Operations Research*, 4(3) :195–212, 1977.

[170] I Solos, I Tassopoulos, and G Beligiannis. A generic two-phase stochastic variable neighborhood approach for effectively solving the nurse rostering problem. *Algorithms*, 6(2) :278–308, may 2013.

[171] L J Stockmeyer and A K Chandra. Provably difficult combinatorial games. *SIAM Journal on Computing*, 8 :151–174, 1979.

[172] L Stockmeyer. Classifying the computational complexity of problems. *The journal of symbolic logic*, 52(1) :1–43, 1987.

[173] M Sulaiman and A Salhi. A seed-based plant propagation algorithm : the feeding station model. *The Scientific World Journal*, 2015, 2015.

[174] M Sulaiman and A Salhi. A hybridisation of runner-based and seed-based plant propagation algorithms. In *Nature-inspired computation in engineering*, pages 195–215. Springer, 2016.

[175] M Sulaiman, A Salhi, B I Selamoglu, and O B Kirikchi. A plant propagation algorithm for constrained engineering optimisation problems. *Mathematical Problems in Engineering*, 2014 :10 pages, 2014.

[176] M Sulaiman, A Salhi, E S Fraga, Mashwani W.K, and M.M Rashidi. A novel plant propagation algorithm : Modifications and implementation. *Science International*, 28(1) :201–209, 2016.

[177] E G Talbi. *Metaheuristics : from design to implementation*, volume 74. John Wiley & Sons, 2009.

[178] L Tang and X Wang. Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. *The International Journal of Advanced Manufacturing Technology*, 29(11) :1246–1258, 2006.

[179] J Thornton and A Sattar. An integer programming-based nurse rostering system. In *Concurrency and Parallelism, Programming, Networking, and Security*, pages 357–358. Springer, 1996.

[180] N Todorovic and S Petrovic. Bee colony optimization algorithm for nurse rostering. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 43(2) :467–473, mar 2013.

[181] L Trilling, A Guinet, and D Le Magny. Planification de ressources mutualisées : le cas des infirmiers anesthésistes. In *Conférence JDMACS-JNMACS 2005*, pages 1–7, 2005.

[182] L Trilling, A Guinet, and D Le Magny. Nurse scheduling using integer linear programming and constraint programming. In *12th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2006*, volume 3, pages 651–656. Elsevier, 2006.

[183] C Tsai and S H A Li. A two-stage modeling with genetic algorithms for the nurse scheduling problem. *Expert Systems with Applications*, 36(5) :9506–9512, jul 2009.

[184] A Turing. On computable numbers with application to the entscheidungsproblem. In *Proc. London Math. Soc. Ser.*, pages 230–265 and 544–546, 1936.

[185] C Valouxis, C Gogos, G Goulas, P Alefragis, and E Housos. A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, 219(2) :425–433, jun 2012.

[186] M Vanhoucke and B Maenhout. Nsplib—a nurse scheduling problem library : a tool to evaluate (meta-) heuristic procedures. In *Operational research for health policy : making better decisions, proceedings of the 31st annual meeting of the working group on operations research applied to health services*, pages 151–165, 2007.

[187] USA Web page by The Clay Mathematics Institute of Cambridge, Massachusetts. Millennium prize problem. http://www.claymath.org/millennium, 2009. Accessed : 2016-09-30.

[188] T C Wong, M Xu, and K S Chin. A two-stage heuristic approach for nurse scheduling problem : A case study in an emergency department. *Computers & Operations Research*, 51 :99–110, 2014.

[189] M Yagiura, S Iwasaki, T Ibaraki, and F Glover. A very large-scale neighborhood search algorithm for the multi-resource generalized assignment problem. *Discrete Optimization*, 1(1) :87–98, 2004.

[190] Z Zheng and X Gong. Chemical reaction optimization for nurse rostering problem. In *Lecture Notes in Electrical Engineering*, pages 3275–3279. Springer Nature, dec 2013.

[191] X Zhong, J Zhang, and X Zhang. A two-stage heuristic algorithm for the nurse scheduling problem with fairness objective on weekend workload under different shift designs. *IISE Transactions on Healthcare Systems Engineering*, 7(4) :224–235, 2017.

[192] Y Zhou, Y Wang, X Chen, L Zhang, and K Wu. A novel path planning algorithm based on plant growth mechanism. *Soft Computing*, 21(2) :435–445, 2016.