HEC Montréal

Affiliée à l'Université de Montréal

Quatre essais sur la confection d'horaires appliquée au secteur de la santé

par

Julien Crowe

Service d'enseignement de méthodes quantitatives de gestion

Thèse présentée à la Faculté des études supérieures et postdoctorales, à HEC Montréal en vue de l'obtention du grade de PhD en administration option méthodes quantitatives de gestion

Décembre 2012

HEC Montréal

Affiliée à l'Université de Montréal

Service d'enseignement de méthodes quantitatives de gestion

Cette thèse intitulée :

Quatre essais sur la confection d'horaires appliquée au secteur de la santé

présentée par :

Julien Crowe

a été évaluée par un jury composé des personnes suivantes :

Sylvain Perron, HEC Montréal
président-rapporteur

Patrick Soriano, HEC Montréal
directeur de recherche

Gilles Caporossi, HEC Montréal
membre du jury

Greet Vanden Berghe, KAHO - KU Leuven
examinateur externe

Jean-François Cordeau, HEC Montréal
représentant du doyen

Résumé

Les services de santé peuvent être prodigués par de grands réseaux d'organisations qui opèrent en continu, tel que c'est le cas dans la province du Québec, au Canada, où les services de santé sont publics. Ces réseaux sont cependant gérés en silos, créant ainsi un environnement où le partage de ressources humaines et matérielles est peu répandu. Dans une perspective d'optimisation du réseau, ce contexte crée d'excellentes opportunités de recherche en flexibilité et portabilité d'approches de résolution de problèmes de confection d'horaires. Nous présentons donc dans cette thèse quatre essais sur la confection d'horaires de personnel appliquée au secteur des services de santé.

Le premier essai est une revue de la littérature en confection d'horaire de personnel. Elle documente plus de 160 publications et décrit leurs travaux par modèles, méthodes de résolution et applications. Bien que les services de santé y soient discutés, la revue inclut aussi d'autres applications avec leurs méthodes de résolution et leurs modèles puisqu'elles partagent plusieurs caractéristiques avec les services de santé.

Le deuxième essai présente SOFA (Scheduling Optimizer with a Flexible Approach), une méthode heuristique flexible et portable de décomposition séquentielle pour résoudre les problèmes de confection d'horaires de personnel en santé. Nous y présentons une classification des types de contraintes rencontrées en confection d'horaires qui est utilisée pour structurer la méthode de résolution. SOFA est une métaheuristique divisée en trois phases qui resolvent séquentiellement un problème de jour de travail, un problème simplifié d'affectation de quart de travail, et un problème complet d'affectation de quart de travail. Un mécanisme de brassage ainsi que deux mécanismes de rétroaction sont ajoutés comme extensions afin d'améliorer la qualité des solutions. La méthode de résolution est testée en deux variantes sur un cadre expérimental de 115 instances, et les résultats indiquent qu'une méthode flexible et portable peut produire de bonnes solutions sur des problèmes de confection d'horaires.

Le troisième essai présente CHAIR (Column generation Heuristic Approach for Indefinite Rostering), une heuristique flexible et portable inspirée de la génération de colonnes pour résoudre les problèmes de confection d'horaires de personnel en santé. Cette méthode est conçue afin d'analyser le potentiel d'une approche de décomposition fondamentalement différente de celle de SOFA : Elle est structurée avec un problème-maître et un ensemble de sous-problèmes. Le problème-maître est résolu par une métaheuristique avec plusieurs structures de voisinages pour la recherche locale, alors que les sous-problèmes sont résolus par une heuristique gloutonne. Plusieurs extensions sont ajoutées afin d'améliorer la qualité de l'approche, incluant une procédure de post-optimisation pour les sous-problèmes, et un mécanisme de brassage pour le problème-maître. Une analyse comparative des résultats de CHAIR et de SOFA permet de mieux comprendre les forces et faiblesses des deux approches.

Finalement, le quatrième essai présente les résultats d'une version améliorée de CHAIR appliquée spécifiquement aux problèmes de confection d'horaires d'infirmières. La méthode de résolution est testée sur un cadre expérimental théorique de 144 instances, et sur 10 instances de la littérature incluant plusieurs applications réelles. Les résultats obtenus sur le cadre expérimental théorique sont des horaires de bonne qualité et fournissent un aperçu intéressant de l'impact de certains types de contraintes. De plus, les résultats obtenus sur les instances tirées de la littérature sont comparés à ceux des méthodes de résolution de la même littérature et démontrent qu'une approche flexible et portable peut se mesurer à la plupart des algorithmes spécialisés.

Mots clés : Santé, Confection d'horaires, Infirmières, Heuristique, Décomposition, Flexibilité, Portabilité.

Abstract

Health care services can be provided by large networks of organizations operating around the clock, as it is the case in the province of Quebec, Canada where health care services are public. These networks employ numerous groups of employees which are however managed in silos, creating an environment where sharing resources such as software is not natural. This context creates however great research opportunities on flexibility and portability of staff scheduling approaches. Hence, in this thesis we present four essays on staff scheduling applied to health care.

The first essay is a literature survey on staff scheduling. It surveys more than 160 papers and describes their research by models, solution approaches and applications. While health care applications are discussed, the survey also includes other applications with their models and solution approaches as they share many characteristics.

The second essay presents SOFA (Scheduling Optimizer with a Flexible Approach), a sequential decomposition heuristic for flexible and portable staff scheduling in health Care. We present a classification of types of staff scheduling constraints which is used to structure the solution approach. SOFA is a metaheuristic divided into three phases solving sequentially a days off scheduling problem, a simplified shift scheduling problem, and a full shift scheduling problem. A shuffling mechanism and two backtracking mechanisms are implemented as extensions to improve solution quality. The solution approach is tested in two variants on an experimental framework of 115 instances, and results indicate that a flexible and portable approach can provide good solutions on staff scheduling problems.

The third essay presents CHAIR (Column generation Heuristic Approach for Indefinite Rostering), a heuristic inspired from column generation for flexible and portable staff scheduling in health care. This solution approach is designed in order to analyze the potential of a fundamentally different decomposition structure than that of SOFA : it is structured as a master problem with subproblems. The master problem is solved by a metaheuristic with multiple local search neighborhood structures, while the subproblems are solved by a greedy heuristic. A number of extensions are added to improve solution quality, including a post optimization procedure for subproblems, and a shuffling mechanism for the master problem. A comparative analysis of results obtained with CHAIR and SOFA shows the relative strengths and weaknesses of each approach.

Finally, the fourth essay presents results of an improved version of CHAIR applied specifically on nurse scheduling problems. The solution approach is tested on a nurse scheduling experimental framework including 144 instances, and on 10 instances from the literature, including many real life applications. Results obtained on the experimental framework show good solution quality and provide interesting insight on the impact of some types of constraints. Meanwhile, results on instances from the literature are benchmarked with solution approaches from the same literature and demonstrate that a flexible and portable approach can compete with most specialized algorithms.

Keywords : Health care, Nurse scheduling, Heuristic, Decomposition, Staff scheduling, Flexibility, Portability.

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

BM1 : First backtracking mechanism for SOFA

BM2 : Second backtracking mechanism for SOFA

C1 : Algorithm variant of CHAIR with the subproblem post optimization, the reshuffle mechanism and the limited tour pool

C2 : Algorithm variant of CHAIR with all extensions of C1 plus the extended subproblem

CHAIR : Column generation Heuristic Approach for Indefinite Rostering

MP : Master problem

RM : Reshuffling mechnism for SOFA

S1 : Algorithm variant of SOFA implementing both backtracking mechanisms

S2 : Algorithm variant of SOFA implementing all extensions of S1 plus the reshuffling mechanism

SOFA : Scheduling Optimizer with a Flexible Approach

SP : Subproblem

Remerciements

Je veux d'abord et avant tout à remercier mon directeur de thèse, M. Patrick Soriano, pour m'avoir offert l'opportunité de travailler sur ce sujet de thèse. J'ai grandement apprécié sa disponibilité, ses idées, ainsi que ses conseils tout au long du doctorat. Il va sans dire que sans son support financier, les travaux présentés dans cette thèse n'auraient pu être accomplis. Sur une autre note, nos lunchs "de travail" au restaurant Kam Shing vont bien me manquer, ainsi que les conférences en Europe !

Je tiens aussi à remercier M. Michel Gendreau pour son encadrement et son support dans le cadre de nos travaux de recherche. Michel a grandement contribué aux travaux par ses judicieux conseils, par ses efforts pour mettre en place un stage de recherche à l'université de Nottingham à l'automne 2011, et par son soutient financier. Au risque de me répéter, nos lunchs au restaurant Kam Shing vont encore une fois me manquer !

Je me dois aussi de remercier M. Edmund Burke pour sa contribution et son support financier sans lesquels mon passage à Nottingham n'aurait pu être possible. J'y ai beaucoup apprécié l'intérêt, le support et la collaboration de M. Tim Curtois qui a été déterminant pour la réalisation du dernier article.

Finalement, je désire aussi remercier M. Gilles Caporossi et M. Walter Rei pour leur participation au jury de la proposition de thèse, avec les conseils et avis importants qu'ils m'ont donnés pour améliorer le contenu des travaux de recherche. J'en profite par le fait même pour remercier l'implication de ceux qui se sont engagés à faire partie du jury de thèse.

Chapitre 1

Introduction

Depuis plusieurs années, les soins de santé sont devenus un enjeu important au Québec, au Canada et ailleurs dans le monde. Le vieillissement anticipé de la population dans les pays occidentaux ne fera qu'augmenter l'importance de cet enjeu dans les prochaines décennies. Pourtant, au Québec et ailleurs, la gestion des soins de santé fait déjà face à plusieurs problèmes : des listes d'attentes sur prise de rendez-vous allant de quelques semaines à plus d'un an, des files d'attente aux urgences de plusieurs heures, des retards fréquents de prestation de service sur rendez-vous, un niveau élevé d'insatisfaction du personnel à l'égard de leur emploi, un écart important entre les disponibilités offertes par les professionnels et les besoins des employeurs, et un budget de service qui ne croît pas nécessairement proportionnellement à la demande pour ne nommer que ceux-ci.

Dans un tel contexte, les sciences de la gestion ont le potentiel d'enrichir le réseau de la santé en le guidant vers plus d'efficience, un contrôle raisonnable des coûts et un service de qualité pour le patient. Les gestionnaires exécutifs trouveront du support sur les politiques de gestion à implanter dans les théories du management, les ressources humaines trouveront du support pour attirer les talents vers le réseau dans les approches de marketing, les gestionnaires à tous les niveaux pourront avoir accès à de l'information pertinente et ciblée pour supporter leur prise de décision grâce aux technologies de l'information, et la qualité du service offert pourra être améliorée grâce aux stratégie de gestion des opérations et de la chaîne logistique.

Les méthodes quantitatives de gestion fournissent quant à elles aussi un potentiel important de support aux gestionnaires de services de santé : techniques prévisionnelles d'intelligence d'affaires pour anticiper l'évolution des besoins, analyses statistiques pour comprendre les relations actuelles entre différents leviers du système de santé, et la recherche opérationnelle pour optimiser l'utilisation des ressources disponibles.

Au Québec par exemple, la plus grande part des dépenses publiques annuelles en santé de 30 milliards de dollars est imputée aux ressources humaines qui totalisent un nombre d'employés de près de 300 000 personnes. Cette taille critique fait de l'optimisation de l'utilisation des ressources humaines un terrain de recherche riche en opportunités, parmi lesquelles la confection d'horaires (« staff scheduling », « personnel scheduling » ou « personnel rostering ») en est une ayant un potentiel majeur aussi bien sur la qualité de service, que sur le niveau de productivité et la rétention du personnel.

Un grand nombre de recherches dans la littérature portent sur la confection d'horaires. Dans le domaine des services de santé, c'est sur les applications d'infirmière qu'on en recense le plus. La très grande majorité de ces recherches portent sur des groupes d'employés restreints (généralement quelques dizaines de personnes au maximum) d'une unité de soin en particulier, ce qui explique en partie pourquoi il en existe autant.

Ce type de contenu de la littérature s'explique lorsqu'on remarque que dans la pratique des services de soins de santé, les disparités entre les règles de confection d'horaires sont importantes entre groupes de professionnels, départements et établissements. Toutefois, avec un nombre de ressources humaines limitées, de budget limité, et de temps limité, il n'est que rarement envisageable d'avoir accès aux ressources nécessaires pour concevoir un système de confection d'horaire spécifique à chaque classe de professionnel de chaque unité de soin de chaque établissement. Les gestionnaires continuent donc à ce jour d'avoir recours à des processus de confection d'horaires en grande partie manuels.

Dans la présente thèse, nous proposons une alternative : le développement d'approches flexibles et portables de confection d'horaires pour les services de santé. L'objectif est l'accomplissement d'un système portable et donc rapidement configurable, basé sur une structure ayant la flexibilité requise pour s'adapter à différentes combinaisons de règles de confection d'horaires. Nous désirons donc proposer une approche au coût d'implantation minimal pour l'organisation et produisant des horaires de qualité.

Cette thèse est composée de quatre articles. L'auteur de cette thèse en est l'auteur principal de chacun, tant pour la recherche et le développement du contenu que pour la rédaction. Le premier est une revue exhaustive de la littérature en confection d'horaires de personnel qui n'est pas limitée au domaine de la santé. Elle décrit les différents modèles, méthodes de résolution et applications présents dans la littérature. Le deuxième article présente une approche de résolution flexible et portable de confection d'horaire pour le domaine de la santé appelé SOFA (Scheduling Optimization with a Flexible Approach). C'est une approche heuristique basée sur une décomposition séquentielle du problème de confection d'horaires, qui est testée sur un cadre expérimental à multiples dimensions afin d'évaluer ses performances et de comprendre les impacts de certains choix de gestion sur les horaires. Le troisième article présente une approche de résolution alternative appelée CHAIR (Column generation heuristic approach for indefinite rostering) basée sur une décomposition de type problème maître/sous-problème. Les résultats de CHAIR y sont comparés à ceux de SOFA afin de comprendre les forces et faiblesses de chacune des approches. Finalement, le quatrième article présente une version de CHAIR améliorée pour gérer les particularités des problèmes de confection d'horaires d'infirmières. Dans un premier temps, la méthode y est évaluée sur un cadre expérimental conçu spécifiquement pour tester ses performances en fonction de la réalité des horaires d'infirmières. Par la suite, elle est comparée aux meilleures approches connues sur un ensemble de dix problèmes d'horaire réels d'infirmière. Finalement, une conclusion clôturera cette thèse.

Chapitre 2

A Literature Survey of Staff Scheduling

2.1 Introduction

There are many important aspects regarding capacity management in service operations, one of which being human resources management, and more precisely personnel scheduling (also referred to as personnel rostering or staff scheduling). In the service operations context, it is straightforward to reach the conclusion that in order to have the required capacity available at the required time, you need to have the right number of employees available at the right time. To obtain this result, one must follow a sequence of decisions which includes (but is not limited to) : shift scheduling, days off scheduling, the combination of the former and the latter into a tour (also called a line of work), and the assignment of the tour to an employee.

The operations research literature regarding these decisions will be reviewed here. Furthermore, because of the very large number of papers published on this topic, and the fact that most envision a scope limited to one single very well-defined scheduling application, we believe there may be very interesting research opportunities in the area of flexible and portable personnel scheduling algorithms and models. Hence, we will provide special attention to the few published papers which focus on algorithmic and modeling flexibility and portability.

The following section will describe in details the scope and limitations of this review. Section 2.3 will then present an overview or the main models found in the literature, while section 2.4 will expose the different solution methods proposed to tackle them. Section 2.5 will discuss the large amount of different application areas in which the previously presented models and solution methods have been applied, and finally, section 2.6 will conclude this review.

2.2 Scope

Literature on the topic of personnel scheduling is very abundant in operations research. An exhaustive list of it, dating from 2004, can be found in Ernst & al. (2004 [83]) . Ernst & al. (2004 [84]) define personnel scheduling (or personnel rostering) as the process of constructing work timetables for staff so that an organization can satisfy the demand for its goods or services. In practice, it can be viewed as a set (and often a sequence) of subproblems. Ernst & al. (2004 [84]) defines it as the set of the following subproblems : Demand modeling, days off scheduling, shift scheduling, line of work construction, task assignment, and staff assignment. Break scheduling (at lunchtime for instance) is usually assumed to be included within shift scheduling, but could be considered as an independent subproblem. Indeed, in practice managers schedule shifts on a weekly/monthly basis, but frequently assign lunch breaks to employees on a daily basis. Furthermore, Ernst & al. (2004 [84]) specifies that line of work construction is usually called tour scheduling when dealing with flexible demands, and crew rostering when dealing with crew pairings. The latter will be excluded as our scope will be limited to tour scheduling. The reader interested in crew scheduling is referred to Gopalakrishnan & al. (2005 [94]), which reviews its applications in airlines.

Days off scheduling, shift scheduling, break scheduling, tour scheduling and staff assignment will be the focus of the present review. This choice is justified by the following : (1) all five subproblems are present in every standard personnel scheduling problem, (2) decisions taken within any of those five subproblems have an impact on the decisions that must be taken in all the subsequent ones, and (3) all five are either optimization or constraint satisfaction problems in practice. This last argument brings us to exclude demand modeling, which is generally considered as a statistical analysis problem that will be used as an input (deterministic or stochastic) into the subsequent subproblems. Task assignment will also be excluded since it does not arise frequently in standard personnel scheduling problems. Furthermore, personnel scheduling is by definition a tactical problem. Hence, it is a clearly distinct problem from manpower planning, which is a strategic one. It is also distinct from re-rostering (also called real-time rostering), which consists of the operational day-to-day (or even within day) decision of scheduling personnel at the last minute according to no-shows and unexpected fluctuations in demand. As such, manpower planning and re-rostering will also be excluded from this review. Figure 2.1 presents an aggregate diagram of the process of personnel rostering.

2.3 Models

The literature in personnel scheduling contains a large variety of models because of important differences between individual problems. The following is a non-exhaustive list of elements causing those differences :

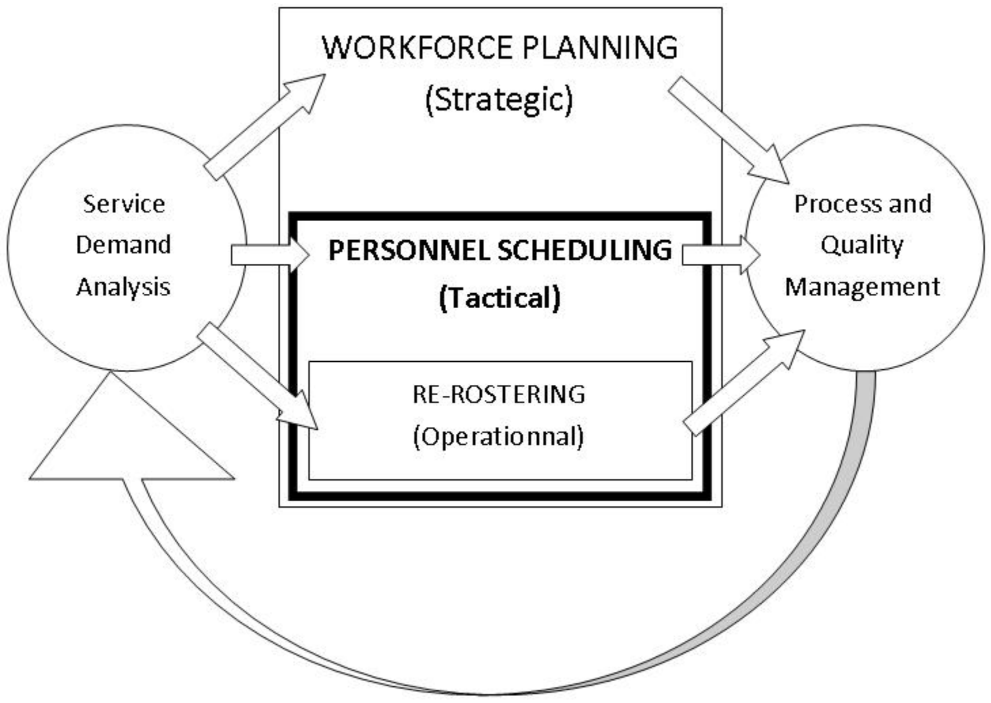


Figure 2.1 – Personnel scheduling processes

- Number of possible shifts per day,
- Presence of overlapping shifts,
- Number of service providing days per week,
- Objective of the model,
- Constraints on individual tours,
- Flexibility,
- Solution method.

All the above elements have impacts on the mathematical model used, and they will be discussed throughout the following paragraphs, along with a few important models. The present section is structured as follows : days off scheduling will be discussed first, followed by shift assignment and tour scheduling. Then, scheduling cycles will be analyzed, followed by objective functions and finally constraints.

2.3.1 Days Off Scheduling

Days-off scheduling is the process of assigning a sequence of work days and days off to individual schedules. This problem arises where a service is provided during a larger number of days than the number of shifts worked by an employee in a week. Hence, it usually arises in services running 6 or 7 days per week since full timers typically work 5 days. Baker (1976 [19]) proposes a basic days-off scheduling model presented between (2.1) and (2.3) where :

- x_j is the number of employees working on sequence j,
- $a_{ij} = 1$ if sequence j calls for work on day i, 0 otherwise,
- r_i is the service demand for day i,
- c_j is the cost of working sequence j.

In this model, the different work sequences are predefined in the $[a_{ij}]$ matrix. While this model is an adequate base for most problems, the allowable sequences for the $[a_{ij}]$ matrix may differ from one setting to another. For instance, some problems will require that all days off per week be consecutive ones, as reported in Alfares (1998 [7]). In the specific case where only one shift length must be covered each day, solving the days-off scheduling problem is the only step required for solving the tour scheduling problem. Otherwise, solving a days-off scheduling problem means that a shift assignment problem will have to be solved subsequently.

$$\min \sum_{j=1}^{J} c_j x_j \tag{2.1}$$

subject to :

$$\sum_{j=1}^{j} a_{ij} x_j \geq r_i \qquad \forall\ i \in I, \tag{2.2}$$

$$x_j \in \mathbb{Z}^+ \qquad \forall\ j \in J, \tag{2.3}$$

Papers on days-off scheduling include the following : [7], [8], [15], [24], [25], [26], [37], [38], [81], [82], [99], [130], [132], [136] and [160].

2.3.2 Shift Assignment

Shift assignment (also called shift scheduling) is the process of assigning a single work shift to each line of work. This can be implemented as a subproblem solved independently for each work day of a planning horizon. Furthermore, if :

1. A service is provided the same number of days as the number of shifts worked by each employee per week (usually 5 days) and,
2. An employee works the same shift everyday,

Then, solving the shift assignment problem is a shortcut to solving the tour scheduling problem. Baker (1976 [19]) presents a model for shift scheduling presented between (2.4) and (2.6) where :
– x_j is the number of employees working on shift j,
– $a_{ij} = 1$ if shift j is on duty during period i, 0 otherwise,
– r_i is the service demand for period i,
– c_j is the cost of working shift j.
This model is the same as the one proposed by Dantzig (1954 [69]), and it requires all shifts to be defined in matrix $[a_{ij}]$, which can make the problem rather large when many different types of shifts (defined by shift length, shift starting-time,

$$\min \sum_{j=1}^{J} c_j x_j \tag{2.4}$$

subject to :

$$\sum_{j=1}^{J} a_{ij} x_j \geq r_i \qquad \forall\ i \in I, \tag{2.5}$$

$$x_j \in \mathbb{Z}^+ \qquad \forall\ j \in J, \tag{2.6}$$

break time-window, etc.) are considered. Putting aside the definitions of variables and parameters, it is the same set covering model as presented in section 2.3.1 for days-off scheduling. A more complete goal programming (multi objective) approach to shift assignment is proposed by Thompson (1996 [153]) and presented between (2.7) and (2.11) where :

– e is the index for employees,
– E is the set of available employees,
– S_e is the set of shifts for which employee e is available (defined by the specific periods in which employee e is available for work and by the minimum and maximum lengths of shifts to which employee e can be assigned),
– $x_{en} = 1$ if employee e is assigned to shift n, 0 otherwise,
– $u_{pj} = 1$ if period p is understaffed by at least j employees, 0 otherwise,
– $o_{pj} = 1$ if period p is overstaffed by at least j employees, 0 otherwise,
– k_{pj} is the incremental monetary cost of increasing the understaffing in period p from j-1 to j employees, where $1 \leq k_{p1} \leq k_{p2} \leq ... \leq k_{pr_p}$,
– b_{pj} is the incremental monetary benefit of increasing the overstaffing in period p from j-1 to j employees, where $1 > b_{p1} \geq b_{p2} \geq ...$.

This model has the interesting characteristic of allowing understaffing, and making a trade-off between the cost of hiring employees and the cost of not meeting staffing targets. Many more extensions can be included in a shift assignment model; however, as they all are extensions that can also be included in the tour scheduling problem, they will be discussed in section 2.3.3.

Papers on shift assignment include : [31], [96], [97], [112], [128], [132], [134], [149], [151], [153], [154], [155], [164], [169] and [171].

$$\min Z = \sum_{e \in E} \sum_{n \in S_e} C_n x_{en} + \sum_{p \in P} (\sum_{j=1}^{r_p} k_{pj} u_{pj} - \sum_{j=1}^{\infty} b_{pj} o_{pj}) \tag{2.7}$$

subject to :

$$\sum_{e \in E} \sum_{n \in S_e} a_{np} x_{en} + \sum_{j=1}^{r_p} u_{pj} - \sum_{j=1}^{\infty} o_{pj} = r_p \qquad \forall\, p \in P, \tag{2.8}$$

$$\sum_{n \in S_e} x_{en} \leq 1 \qquad \forall e \in E, \tag{2.9}$$

$$x_{en} \in \{0,1\} \qquad \forall\, e \in E,\ n \in S_e, \tag{2.10}$$

$$u_{pj} \in \{0,1\} \qquad \forall\, p \in P,\ j \in J, \tag{2.11}$$

2.3.3 Explicit and Implicit Models in Tour Scheduling

Tour scheduling is the process of combining days-off and shift assignment solutions into a full schedule. In tour scheduling problems, two main types of models are encountered : explicit and implicit. In explicit models, also called set-covering and based on Dantzig (1954 [69]), the full tours (possible individual schedules) are explicitly represented in the models by a large number of constant parameters. These are usually binary and represent the presence (value of 1) or absence (value of 0) of a work shift in a given tour for a given period. Linked to the decision variables, they create a large number of service coverage constraints, which unfortunately makes the model very large and difficult to solve optimally when many tours are included in it. On the other hand, implicit models can be more complex to design but have the advantage of greatly reducing the size of the problem by representing shifts and/or breaks as variables.

The basic tour scheduling model, presented as a set covering model by Dantzig (1954 [69]), is presented between (2.12) and (2.14) where :
– x_j is the number of employees working on tour j,

– $a_{ij} = 1$ if tour j is on duty during period i, 0 otherwise,

– r_i is the service demand for period i,

– c_j is the cost of working tour j.

Apart from some variables definitions, this is the same basic set covering model presented in sections 2.3.1 for days-off scheduling and 2.3.2 for shift assignment. One of its notable characteristic is that it allows overstaffing, but not understaffing. This can be useful in situations where a minimum staff must be provided for each period. An extension of this model is the following, referred by Baker (1976 [19]) and initially proposed by Luce (1973 [122]) is presented between (2.15) and (2.18) where :

– c_j is the cost of assigning one employee to tour j,

– s_i is the shortage allowed at period i,

– t_i is the surplus allowed at period i.

This extension provides an interesting improvement over the previous model : (2.16) and (2.17) allow bounded variations around r_i, which becomes a target of service coverage instead of a strict minimum. One limitation of this approach is the dependence of the quality of the solution on how parameters s_i and t_i have been adjusted. If the bounds are too loose, the resulting coverage will likely be far from r_i, while if they are too tight, a feasible solution may not exist.

$$\min \sum_{j=1}^{J} c_j x_j \tag{2.12}$$

subject to :

$$\sum_{j=1}^{J} a_{ij} x_j \geq r_i \qquad \forall\, i \in I, \tag{2.13}$$

$$x_j \in \mathbb{Z}^+ \qquad \forall\, j \in J, \tag{2.14}$$

$$\min \sum_{j=1}^{J} c_j x_j \tag{2.15}$$

subject to :

$$\sum_{j=1}^{J} a_{ij} x_j \geq r_i - s_i \qquad \forall\, i \in I, \tag{2.16}$$

$$\sum_{j=1}^{J} a_{ij} x_j \leq r_i + t_i \qquad \forall\, i \in I, \tag{2.17}$$

$$x_j \in \mathbb{Z}^+ \qquad \forall\, j \in J, \tag{2.18}$$

As reported by Alfares (2004 [9]), Easton & al. (1996 [76]) proposes a goal programming (multi-objective) approach presented between (2.19) and (2.21) where :

- J_e is the set of feasible tours for employee category e,
- Z is the total labor cost,
- C_j is the cost of assigning one employee to tour $j \in J_e$,
- d_i^+, d_i^- are respectively the labor understaffing and overstaffing at period i,
- u_i, o_j are respectively the penalties for understaffing and overstaffing at period i.

Two main differences arise when comparing this last model with the previous ones. First, different categories of employees are considered here. This limits tours to which an employee may be assigned to, and hence allows a higher level of indivi-

$$\min Z = \sum_{e=1}^{E} \sum_{j \in J_e} C_j x_j + \sum_{i=1}^{I} u_i d_i^- - o_i d_i^+ \tag{2.19}$$

subject to :

$$\sum_{e=1}^{E} \sum_{j \in J_e} a_{ij} x_j + d_i^- - d_i^+ = r_i \qquad \forall\, i \in I, \tag{2.20}$$

$$x_j \in \mathbb{Z}^+ \qquad \forall\, j \in J, \tag{2.21}$$

dualization of assignments. Second, service coverage is now a soft constraint and its violations are penalized in the objective function. Hence, this goal program is a multi-objective approach in which the cost of scheduling employees must be balanced with the cost of reaching the service coverage target. Important characteristics of such an approach include that it guaranties a feasible solution and that it provides more flexibility.

Thompson (1995 [152]) presents another extension of the explicit tour scheduling model which includes the positive impact of staffing levels in its objective (for clarity purposes, we use the notation of Goodale & al. (2004 [91])) presented between (2.22 and (2.24) where :

- i is an index for planning periods,
- j is an index of work schedules,
- k is the number of additional workers in period i beyond the minimum acceptable level,
- T is the set of work schedules,
- x_j is the number of employees assigned to work schedule j,
- c_j is the cost of assigning an employee to work schedule j,
- $a_{ij} = 1$ if i is a work period of schedule j, 0 otherwise,
- r_i is the number of workers required at period i,
- $\tau_{ik} = 1$ if the number of employees working in period i equals or exceeds $m_i + k$, 0 otherwise,
- q_i is the number of employees in period i in excess of the minimum reasonable staff size, who, ignoring labor costs, contribute to increased NPV (net present value) profits,
- m_i is the minimum reasonable number of workers for period i,
- d_{ik} is the incremental improvement in NPV profit (ignoring labor costs) that occurs with the addition of the $(m_i + k)$th employee in period i (assuming

nonincreasing marginal NPV returns for each period - i.e., $d_{ik} \geq d_{i,k+1}$ for $i \in I$ and for all j).

An interesting characteristic of this model is that it doesn't treat scheduling as a cost, but rather as a source of profit to be maximized. However, maximization of profits remains a relatively rare objective in personnel scheduling.

With no exceptions, all models presented previously are explicit ones. We define explicit models as those including in their formulation a matrix describing all possible tours (set covering models). They are simple and easy to implement, but require to define beforehand the content of the matrix. Furthermore, when flexibility is required, the consequently large number of tours included in the models make them very time consuming to solve. In order to deal with these limitations, many implicit models have been proposed. We define implicit models as those defining shifts and tours by variables and constraints, in such a way that the formulation defines the structure of tours, but does not provide a list of possible ones. Such formulations are more time consuming to develop, but require much less time to solve. Jacobs & al. (1996 [106]) proposes an implicit tour scheduling model presented between (2.25) and (2.29) where :

- i is the index of planning periods in an operating day,
- j is the index of days in an operating week,
- k is the index of start-time bands,

$$\max \sum_{i \in I} \sum_{k=1}^{q_i} d_{ik} \tau_{ik} - \sum_{j \in T} c_j x_j \tag{2.22}$$

subject to :

$$\sum_{j \in T} a_{ij} x_j - \sum_{k=1}^{q_i} \tau_{ik} \geq m_i \qquad \forall\ i \in I, \tag{2.23}$$

$$\tau_{ik} \in \{0, 1\} \qquad \forall\ i \in I,\ 0 \leq k \leq q_i, \tag{2.24}$$

– l is the index of shift-starting times per band,
– m is the index of days-on patterns,
– D_{km} is the number of employees working in start-time band k and days-on pattern m,
– $\alpha_{ikl} = 1$ if planing period i is a work period in a shift included in start-time band k and beginning at period l, 0 otherwise,
– γ_{jkl} is the number of employees working a shift on day j, in start-time band k, beginning at planning period l,
– $\beta_{jm} = 1$ if day j is a work day in days-on pattern m, 0 otherwise.

Compared to explicit models, an implicit model such as the one here above greatly reduces the size of the problem to solve. Furthermore, by using variables which define both days-on and shift assignments, it also eliminate the need of explicitly selecting the tour schedules included within the master problem. Also, the concept of start-time bands allows a given tour to have different bounded shift start-times, which greatly increases the flexibility of the model. However, none of the previously presented models (explicit or implicit) consider breaks. The model proposed by Topaloglu & al. (2002 [161]) and presented between (2.30) and (2.34) does consider breaks, where :

– i is the index for days of week,
– k is the index of shift types,
– l is the index for time periods,
– L_i is the number of time periods to be scheduled on day i,
– D is the duration of a shift type,
– DS is the set of indices for shift types with duration D,
– DP is the set of indices for days-on patterns with shift of duration D,
– K_i is the number of shift types on day i,
– $A_{lk} = 1$ if shift pattern k calls for work during period l, 0 otherwise,

– BL_{ik} is the set of time periods in which an employee working shift k on day i may start his/her break,
– TL_{il} is the set of shifts for which period l is a break start time within their time windows on day i,
– D_{il} is the number of employees required during period l on day i,
– X_j is the number of employees assigned to days-on pattern j,
– X_{jk} is the number of employees assigned to shift type k on day i,
– B_{ikl} is the number of employees assigned to shift type k and taking their breaks in period l on day i.

When compared to previous models, an interesting element of this model is that breaks are included. Indeed, many service providers must take breaks into account in order to obtain a schedule providing good adequate coverage in application. When breaks are not considered, service coverage requirements can often be underestimated. Also, if few different shift start times are used, many employees will take their breaks simultaneously, which may cause critical gaps in coverage. Another comparable model including breaks can be found in Rekik & al. (2004 [142]).

Papers on tour scheduling include : [1], [3], [4], [5], [10], [11], [13], [14], [16], [17], [18], [20], [21], [22], [23], [27], [28], [29], [30], [32], [33], [34], [35], [36], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [57], [59], [61], [62], [63], [69], [70], [71], [72], [73], [74], [75], [80], [85], [86], [87], [88], [90], [91], [93], [95], [98], [100], [102], [103], [104], [105], [106], [107], [108], [110], [111], [113], [114], [116], [117], [118], [119], [120], [121], [123], [125], [126], [127], [132], [135], [137], [138], [141], [142], [143], [144], [147], [150], [155], [158], [161], [162], [165], [166], [167] and [172].

$$\min \sum_{k=1}^{K} \sum_{m=1}^{M} D_{km} \tag{2.25}$$

subject to :

$$\sum_{k=1}^{K} \sum_{l=1}^{L} \alpha_{ikl}\gamma_{jkl} \geq r_{ij} \qquad \forall\ i \in I,\ j \in J, \tag{2.26}$$

$$\sum_{l=1}^{L} \gamma_{jkl} - \sum_{m=1}^{M} \beta_{jm} D_{km} = 0 \qquad \forall\ j \in J,\ k \in K, \tag{2.27}$$

$$\gamma_{jkl} \in \mathbb{Z}^{+} \qquad \forall\ j \in J,\ k \in K,\ l \in L, \tag{2.28}$$

$$D_{km} \in \mathbb{Z}^{+} \qquad \forall\ k \in K,\ m \in M, \tag{2.29}$$

$$\min \sum_{j=1}^{J} X_j \tag{2.30}$$

Subject to :

$$\sum_{k=1}^{K_i} A_{lk} X_{ik} - \sum_{k \in TL_{il}} B_{ikl} \geq D_{il} \qquad \forall\ l \in L_i,\ i \in I, \tag{2.31}$$

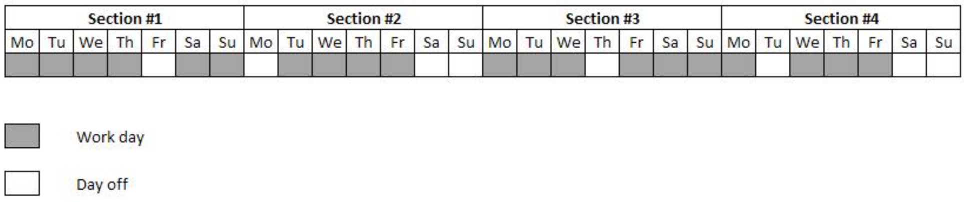
$$\sum_{j \in DP} A_{ij} X_j - \sum_{k \in DS} X_{ik} = 0 \qquad \forall\ i \in I,\ \forall D, \tag{2.32}$$

$$X_{ik} - \sum_{I \in BL_{ik}} B_{ikl} = 0 \qquad \forall\ k \in K_i,\ i \in I, \tag{2.33}$$

$$X_{ik}, X_j, B_{ikl} \in \mathbb{Z}^{+} \qquad \forall\ i \in I,\ j \in J,\ k \in K,\ l \in L, \tag{2.34}$$

2.3.4 Cyclic Schedules

Ernst & al. (2004b [84]) suggest that in a cyclic roster, employees of the same class perform exactly the same line of work (tour), but with a different starting day for the first duty. This case is illustrated in Figure 2.2 and table 2.I.



Section #1							Section #2							Section #3							Section #4						
Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
▨	▨	▨	▨	☐	▨	▨	☐	▨	▨	▨	▨	☐	☐	▨	▨	▨	☐	▨	▨	▨	▨	☐	▨	▨	▨	☐	☐

▨ Work day

☐ Day off

FIGURE 2.2 – Example of a tour for cyclic scheduling

Figure 2.2 presents the tour of a cyclic schedule, and table 2.I presents how this tour is assigned to each employee. In this case, the four week long tour is done by 4 employees. Each of them works the same shifts but at different times of the tour. For instance, employee 1 starts the tour at section 1, employee 2 at section 2, and so on. The main advantages of a cyclic schedule are that : it is normally easier to manage since it consists in reality of only one tour, and it is fair for all employees since everyone works the same shifts in the same order.
The rotating schedule is a special cyclic case where the service is provided around the clock, 7 days a week. This means that at least two different shift start times must be covered each day (f.ex. : two 12 hours shifts starting at midnight and noon). Hence, because the same tour is shared by all employees, all employees

	Week #1							**Week #2**							**Week #3**							**Week #4**						
	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
Employee #1	Section#1							Section#2							Section#3							Section#4						
Employee #2	Section#2							Section#3							Section#4							Section#1						
Employee #3	Section#3							Section#4							Section#1							Section#2						
Employee #4	Section#4							Section#1							Section#2							Section#3						

TABLE 2.I – Example of the Assignment of a Cyclic Tour to Employees

work all possible shifts. This situation increases the difficulty of solving the problem, partly because many ergonomic constraints must be taken into consideration (f.ex. : a night shift can only be worked after a night or afternoon shift, an afternoon shift can only be worked after an afternoon or day shift, a day shift can only be worked after another day shift, and finally night shift sequences must be followed by a given number of days off, etc.).

Papers on cyclic scheduling include : [7], [8], [22], [23], [24], [40], [61], [87], [125], [132] and [147]. More specifically, papers on rotating schedules include [113] and [114].

On the other hand, cyclic schedules are not adequate for situations where high fluctuations in demand are present or where employees are allowed to have individual preferences (f.ex. : length of shift, number of working hours, start times, etc.). In such cases, acyclic schedules must be used. These provide different tours for each employee, and hence allow a degree of individualization, as well as a larger flexibility to cope with high fluctuations in demand. However, they are usually larger than in the cyclic equivalent, and hence harder to solve. Furthermore, acyclic schedules may have individual tours in which shift starting times may vary. However, because they are not cyclic, and because their individual schedules do not necessarily contain a full rotation (day, evening and night shifts), they will not be considered as rotating schedules.

Papers on acyclic schedules include : [1], [3], [4], [5], [10], [11], [13], [14], [15], [16], [17], [18], [20], [21], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [57], [59], [62], [63], [69], [70], [71], [72], [73], [74], [75], [80], [81], [82], [85], [86], [88], [90], [91], [93], [95], [96], [97], [98], [99], [100], [102], [103], [104],

[105], [106], [107], [108], [110], [111], [112], [116], [117], [118], [119], [120], [121], [123], [126], [127], [128], [130], [134], [135], [136], [137], [138], [141], [142], [143], [144], [149], [150], [151], [153], [154], [155], [158], [160], [161], [162], [164], [165], [166], [167], [169], [171] and [172].

2.3.5 Objective Functions

In sections 2.3.1, 2.3.2 and 2.3.3, a few different objective functions have been presented. However, these remain rather basic ones, and literature on personnel scheduling presents a much larger variety of objectives, including very complex ones. Generally, two types of problems are encountered in the literature :

- Optimization problems,
- Constraint satisfaction problems.

Typically, previous literature was focused on optimization problems, while there has been an emergence of constraint satisfaction problems in recent years. This is perhaps due to the ever-rising importance of quality of service, as well as the shortage of qualified manpower in many industries, which logically leads to believe that minimising cost is no longer as important as providing a schedule that will be good enough both to deliver the required level of service and to keep your current staff. However, among those two types of problems, many variations exist in encountered objective functions. According to Alfares (2004 [9]) and Bechtold & al. (1991 [30]), the objective functions suggested in the tour scheduling literature include :

- Total labor hours scheduled,
- Total number of employees,
- Labor costs,
- Unscheduled labor costs,
- Customer service,
- Overstaffing,

– Understaffing,

– Number of schedules with consecutive days off,

– Number of different work schedules utilized,

– Net present value of profit,

– Employee satisfaction,

– Consistent employee workloads,

– Fair assignment of employees to schedules,

– Different combinations of the above elements.

Aggarwal (1982 [2]) also presents a list of objective functions, but this one is classified by application type. We will elaborate on this one in section 2.5. As a general rule, models in personnel scheduling have at least one of the following type of objective :

– Profit maximization,

– Cost minimization,

– Maximization of employee satisfaction.

In our opinion, all objective functions in personnel scheduling are combinations of elements included in the three above. Clearly, the type of objective used is related to the selected solution method. For instance, single objective functions will be well suited for pure linear programming, while multi-objective functions will be better suited for goal-programming or sophisticated heuristics.

2.3.6 Constraints

In sections 2.3.1, 2.3.2 and 2.3.3, typical basic constraints were presented. However, the literature on personnel scheduling offers a much larger variety of constraints, including very complex ones. According to Alfares (2004 [9]), the constraints in personnel scheduling concern the following aspects :

– Allowable shift start times,

– Minimum and maximum length of each shift,

– Frequency and duration of meal and rest breaks,
– Minimum rest period between shifts,
– Operating hours per day,
– Number of workdays per week,
– Limits on the number of consecutive workdays,
– Shift rotation.

Furthermore, through our review of the existing literature we have identified the following which were not included in Alfares (2004 [9]) :

– Service coverage,
– Required skills,
– Individual preferences,
– Ergonomic (worker's health) factors,
– Number of week-ends off,
– Fairness.

Aggarwal (1982 [2]) also presents a list of constraints, but this one is classified by application type. We will elaborate on it in section 2.5. Because of the large number of constraints in most realistic personnel scheduling problems, many are often modeled as soft constraints. This is due to the fact that a large combination of hard constraints would most likely make the problem unfeasible. This explains why sophisticated heuristics and goal programming approaches are so often encountered in the personnel scheduling literature : they are well suited for multiobjective approaches dealing with many conflicting constraint violations.

2.4 Solution Methods

The large variety in structures and characteristics of personnel scheduling problems have motivated researchers and practitioners to implement a large number of different types of solution approaches. Alfares (2004 [9]) classifies tour scheduling approaches in the following way :

- Manual solution,
- Integer programming,
- Implicit modeling,
- Decomposition,
- Goal programming,
- Working set generation,
- LP-based solution,
- Construction/improvement,
- Metaheuristics,
- Other methods.

This classification is quite detailed. However, it has the limitation of having some classes that are closely interrelated. For instance, implicit modeling is a subset of integer programming, and decompositions are frequently LP-based solution approaches. However, in our opinion this limitation is hardly avoidable when using a detailed classification. This may explain why some reviews such as Aggarwal (1982 [2]) have opted for limiting themselves to presenting a classification by application, which is also a pertinent choice in the context of personnel scheduling. To the best of our knowledge, the most detailed classification is the one proposed by Ernst & al. (2004 [83]), which is the following :

- Branch-and-bound,
- Branch-and-cut,
- Branch-and-price,
- Column generation,
- Constraint logic programming,
- Constructive heuristic,
- Dynamic programming,

- Enumeration,
- Evolution,
- Expert systems,
- Genetic algorithms,
- Goal programming,
- Integer programming,
- Iterated randomized construction,
- Lagrangian relaxation,
- Linear programming,
- Matching,
- Mathematical programming,
- Network flow,
- Other meta-Heuristic,
- Other methods,
- Queuing theory,
- Set covering,
- Set partitioning,
- Simple local search,
- Simulated annealing,
- Simulation,
- Tabu search.

The above classification includes a large number of different approaches. However, because the scope of Ernst & al. (2004 [83]) is much wider than ours, we believe this classification to be much too large for our needs. On the other hand, Ernst & al. (2004 [84]) has opted for using the following much simpler aggregated classification :

- Demand modeling,
- Artificial intelligence approaches,

– Constraint programming,

– Metaheuristics,

– Mathematical programming approaches.

Because of its simplicity, we believe this last classification provides less opportunities for a method to fit into more than one category. However, this classification is strongly unbalanced as AI approaches are much less common than mathematical programming and metaheuristic ones. Furthermore, since we have excluded demand modeling from the scope of our review, we will hence use the following classification which is inspired from the previous ones, but adapted to our scope and needs :

– Mathematical programming approaches,

– Heuristics,

– Constraint programming,

– Hybrid approaches,

– Other approaches.

Although unbalanced because the first two classes account for most of the literature, we believe this classification includes all solution methods proposed up to this day, without allowing many methods to be assignable to more than one class. Hence, the rest of this section will be structured accordingly.

2.4.1 Mathematical Programming Approaches

Mathematical programming approaches are very often used to solve personnel scheduling problems. The most basic version of them is pure linear programming, which can solve problems dealing with continuous variables. With present day computer performances, this approach can solve real-life problems in many areas. However, personnel scheduling problems require the use of integer variables. Hence, linear programming must be combined with other methods such as branch-and-bound, in order to obtain a feasible solution. Unfortunately, for a similar

number of variables, algorithms for problems with integer variables require much more computing time.
The current section will be structured according to the most common mathematical programming approaches used in personnel scheduling : Mixed-integer programming, column generation and other mathematical decomposition techniques, and finally other less common mathematical programing approaches. For each solution approach, papers considered particularly interesting with regard to our scope will be discussed.

Isken (2004 [104]) proposes a mixed-integer programming solution for a tour scheduling problem applied to health care, which is modeled implicitly in order to reduce its size. The approach was implemented using CPLEX 7.1 on a pentium III running at 1.0 GHz, with 128 MB of RAM. A total of 1800 problem instances were solved, which have a number of integer variables ranging from 896 to 9842, and a number of constraints ranging from 1004 to 3567. They represent a large array of different combinations of start-time bands, tour types, objective functions and bounds on part-timers. Also, 25 demand profiles are used, which all average 1600 hours per week, or 40 full-time equivalents. With a time limit of 14400 seconds, 1534 problems were solved optimally, among which 1483 within 600 seconds. Interesting findings of this paper include that start-time band width can have an important effect on the cost of the schedule, depending on the type of demand pattern. Also, on some instances, a larger start-time band width seems to compensate for stronger limitations on the number of part timers available. However, the strongest savings come from increased tour type flexibility. Overall, the results presented by Isken (2004 [104]) confirm the intuitive idea that more flexibility leads to cost savings, at least on the short term, and from a purely theoretical point of view. However, as it is the case for most problem structure analysis in personnel scheduling, these results can not automatically be assumed to be correct for different problem structures (and there are a lot of different problem structures in personnel scheduling !).

Jacobs & Bechtold (1993 [105]) proposes another interesting mixed-integer program to solve an implicit model of tour scheduling. Like Isken (2004 [104]), it also analyses the impact of a given set of schedule structure choices on the cost of the solution. However, some of the results presented by Jacobs & Bechtold (1993 [105]) contradict those of Isken (2004 [104]). For instance, Jacobs & Bechtold (1993 [105]) concludes that modifying consecutive days-off constraints has no impact on labor utilization, and almost no improvements come from allowing shift start times to vary across the working days. The first finding contradict the positive impact of tour-type flexibility found by Isken (2004 [104]), and the second one the positive impact of the width of start-time bands. The elements analyzed by Jacobs & Bechtold (1993 [105]) are the following : Tour length, break placement, start-time float, start time (level of discretization), shift length, days off, day length, labor requirement mean and labor requirement amplitude. This design produced a total of 896 instances, which were solved by the SAS/OR branch-and-bound mixed ILP module, on an Amdahl 5860 mainframe computer. With a time limit of 300 seconds, an optimal solution was obtained for 75% of the instances. The findings were that : (1) break placement flexibility could be extremely effective in improving labor utilization, (2) part-timers working fewer days per week can also support labor utilization improvements, (3) increasing the number of daily shift start times also has a positive effect. Furthermore, an intuitive finding is that relationships between flexibility alternatives, labor requirement characteristics and labor requirement distributions are found to be highly complex, and hence hard to predict. This is a fact that has unfortunately strongly limited the usefulness of comparative studies in personnel scheduling up to this day.

Thompson (1995 [151]) presents a large implicit model for shift scheduling, solved by integer programming and based on Bechtold & al. (1990 [31]). This shift scheduling approach considers both a large number of shift and break start times. A total of 588 tests were made, into which the varying parameters included employee requirements, shift types, length of planning interval, length of operating day and restrictions on acceptable shifts. Some of the tests were using periods as small as 15 minutes, which is one of the most detailed level of discretization found in the literature. The model is solved by the branch and bound module of SAS/OR on a 486DX33 personal computer. The problem instances solved had up to 15,885 possible shifts, being 55% more than previously solved by [31]. On average, [151]'s model was 36.5% quicker to solve than the former.

Topaloglu & al. (2002 [161]) proposes another implicit tour scheduling model solved by mixed integer programming, this time with flexible breaks. Among other things, this approach allows for different shift starting times throughout an individual tour. Also, different days-on patterns, shift length, and break start times are allowed. A total of 36 problems are solved, on a 800 MHz Pentium III with OPL studio and CPLEX 7.1. The largest instance had 5,361,741 variables and the average solution time was under 310 seconds for 30 instances. The authors note that for a break window of one hour and for 8 hours shifts on a day of work of 12 hours, the implicit formulation reduces the number of variables from 700,000 in the set covering model to 112. With a stopping criterion of 100,000 iterations, an optimal solution could be found for all instances with the implicit model, but only for 3 with the set covering model.

Finally, Brusco & al. (1998 [43]) proposes a procedure to eliminate redundant columns (tours) in set covering models for tour scheduling problems. The proposed approach eliminates tours that are not part of the optimal solution, because they begin or finish during periods with no service requirements. However, because of this last element, this approach can only be used in a limited number of situations.

The other common mathematical programming approach to personnel scheduling is column generation, and generally speaking, mathematical decomposition techniques. Indeed, while approaches like implicit modeling can efficiently reduce the size of problems, another way to accomplish a similar result is to decompose the problem. Jaumard & al. (1998 [108]) proposes such a column generation approach for nurse scheduling. Here, the master problem finds the best combination of individual schedules to create the full solution, and the subproblem produces an acceptable schedule for a given nurse. With some small variations, this appears to be the most common (and natural) way of decomposing personnel scheduling problems. Preliminary results based on data from a hospital show that, for a problem of 41 employee on a horizon of 6 weeks, a solution to a linearly relaxed problem is obtained in less than 40 minutes. A good integer solution however takes 16.5 hours with a partial branch and bound, and a gap of 0.86% from the lower bound. Eitzen & al. (2004 [80]) also proposes a column generation approach, which has a master problem optimizing the schedule, and a subproblem that generates individual tours to add to the master problem as required. Here, the results of the column generation approach are compared with those of two heuristics referred to as reduced columns subset and column expansion methods. The computational effort of the tests is limited in cpu to either 5000 nodes in the branch and bound tree or 4 hours in cpu time, whichever is reached first. Because not all tours are included in column expansion and reduced columns subsets, there are no guarantees that these approaches can obtain the optimal solution. In terms of solution quality, column generation produces better solutions than the two other methods. With the objective function being to reduce understaffed shifts, column generation provides solution costs more than 25% lower than both other methods. However, on CPLEX 6.6 with a 450 MHz running on Linux, column generation requires more computing time.

LP-based decomposition approaches other than column generation have also been proposed. Unlike column generation many of them do not guarantee finding an optimal solution. They are often sequential approaches, where a large problem is split into smaller ones, and then sequentially solved. For such methods, the objective is rather to obtain a solution within reasonable time than an optimal one. In fact, each sequential problem can potentially be solved optimally, but the original global problem will most likely not be. Jarrah & al. (1994 [107]) proposes such an approach, where a tour scheduling problem is handled by first solving a days off scheduling problem, and then 7 different shift assignment problems (one for each day of the week). Each day is divided into 48 half hour periods. Throughout all the different sets of instances for the shift assignment subproblem, the largest number of constraint is 947, and the largest number of variables is 1607. The branch and bound code is written in FORTRAN and runs on an IBM 3081-D machine. Results are obtained within times largely inferior to 1000 seconds for most instances. Bechtold & al. (1994 [27]) is another paper decomposing the tour scheduling problem in two sequential phases, being shift scheduling and days-off scheduling. In fact, handling a tour scheduling problem by splitting it into a day sequencing problem and a shift problem is a very natural way to solve it. Among sequential decomposition approaches, variations of this one clearly seem to be the most common.

Finally, some mathematical programming approaches are combined with heuristics to decrease solution time. For instance, Wan & al. (2007 [165]) proposes two different LP-based heuristics, including a column generation one. Tests were made on a Pentium IV 2.53 GHz with 512 MB of RAM and CPLEX 8.1. When compared with CPLEX, results obtained by the column generation heuristics were considered very interesting in terms of solution time and solution quality. Such

methods can however be considered as hybrid, such as those described in section 2.4.4.
A few other exact approaches exist, but they are much less frequently used for personnel scheduling. For instance, Elshafei & al. (2008 [81]) proposes a dynamic programming approach for days-off scheduling. However, like mathematical programming, all exact approaches have the limitation of requiring much longer solution times than a well conceived heuristic for a given problem.

Papers on exact solution methods include : [7], [8], [14], [16], [17], [18], [20], [21], [24], [26], [30], [31], [36], [38], [39], [42], [43], [45], [46], [47], [48], [49], [61], [69], [72], [80], [81], [86], [87], [90], [93], [97], [103], [104], [105], [106], [107], [108], [110], [113], [114], [116], [119], [121], [123], [125], [128], [135], [136], [137], [138], [142], [149], [151], [155], [158], [160], [161], [162], [164], [166], [167], [169] and [171].

Papers on LP-based heuristics include : [11], [16], [22], [28], [29], [30], [40], [48], [49], [74], [80], [87], [96], [107], [123], [132], [144], [150], [165] and [172].

2.4.2 Heuristics

Heuristics are the other common type of approaches frequently found in personnel scheduling literature. They are useful when exact approaches can't provide a good solution within reasonable time. There is a large number of different heuristics, but most of them are included within the following categories :
- LP-based heuristics,
- Constructive heuristics,
- Metaheuristics.

LP-based heuristics are procedures based on mathematical programming methodology but which are not solved completely in order to reduce computation times. They have been briefly discussed in the previous section and will be excluded from the present one. In the current section, we will focus our attention on constructive heuristics and metaheuristics.

Goodale & al. (2004 [91]) compare a set of heuristics for assigning individual employees to tour schedules, considering both individual costs and productivity. Five methods are compared : one random assignment, three simple heuristics, and one simulated annealing approach. The first heuristic (H-DOP) sorts individuals in descending order of estimated productivity. It assigns workers to longer tours first, and then to shorter tours. The second heuristic (H-AOC) sorts individuals in ascending order of estimated relative cost and assigns the workers to longer tours first also. The third heuristic (H-DOR) sorts individuals in descending order of their estimated productivity/cost ratio and assigns workers to longer tours first again. Finally, the last one is a simulated annealing approach (H-NPV) which minimizes the net profit value (NPV) of the entire schedule by finding a good combination of assignments. H-NPV turned out to be the most successful profit maximizing approach, closely followed by H-DOR. Respectively, they provided increases of 3.23% and 2.61% over H-RAN on the instance in which they performed best, while H-AOC provided worse results than H-RAN.

Baxter & al. (1988 [23]) present a heuristic that was built to produce schedules in a workplace where manpower needs fluctuate frequently, and hence where the manager spends too much time making new schedules on a regular basis. The heuristic was built to produce an array of feasible schedules from which the manager could subsequently chose the one he preferred. It is divided in four stages : the generation of a schedule allocating men to days for a period of one week, the allocation of men to shifts, the creation of an acceptable rotation of men from one week to the next, and finally assessing a set of potential arrangements produced by the first three phases to obtain the best offering to the workforce. Results of this simple heuristic have been satisfying and implemented, but not empirically compared.

Chiaramonte (2008 [62]) proposes a heuristic agent-based nurse rostering system. According to Chiaramonte (2008 [62]), typical mathematical model or heuristics are not well suited to handle individual preferences for rosters in an objective function. However, the agent based rostering system allows to isolate the preference from the cost minimization problem. Among other things, the concept of agent is used to simulate exchange of stints (sequences of shifts) between nurses, like it is frequently done manually in real life. The algorithm is tested on 20 instances with 2 work shifts per days on a horizon of 28 days, and with 20 nurses. The average runtime is 36 seconds. However, when compared to the manual solution used by a hospital, the nurses rated the performance of the agent based as being negative 33% of the time, versus 25% for the manual solution. An interesting element about this approach is that modeling a heuristic based on human behavior could potentially facilitate its implementation, as managers and workers may have more confidence in a schedule building process they understand.

There are also heuristic decompositions, which are usually sequential. Their logic is the same as mathematical programming decomposition methods : to split a problem, and then solve each resulting section individually. Afterwards, the solutions of each sections are combined to produce a good feasible final solution. For instance, McGinnis & al. (1978 [126]) proposes a two-phase heuristic approach for tour scheduling. The first phase selects the shifts that must be worked, and the second assigns shifts to individual employees in order to build tours. This heuristic is compared to a single-phase one for the same problem. Both are programmed in FORTRAN and solved on a Univac 1108. The instances tested counted around 200 employees, and the one-phase heuristic took an average one 3 minute to solve them, compared to 1 minute for the 2-phase one. Performance is measured in terms of idle manpower (understaffing is not allowed) and consecutive number of days off pairs. The one-phase heuristic leads to substantially better results in terms of idle manpower (15% less), but the 2-phase heuristic produces better results in terms of regular days off pairs (20% better). Of course, sequential decomposition heuristics have little advantage over single phase ones when the problem size is small. However, they become interesting when the problem to solve is so large that the search space can not be explored sufficiently with a single phase in order to guarantee a good solution.

Papers on greedy heuristics include : [13], [15], [20], [21], [23], [25], [26], [27], [30], [40], [44], [46], [47], [62], [70], [75], [82], [91], [93], [96], [98], [99], [100], [111], [112], [118], [126], [127], [130], [141], [147] and [154].

Metaheuristics are more sophisticated approaches than constructive heuristics. Their strength relies on their capacity to continue searching for a better solution even after finding a local optimum, something classical greedy construction or local search heuristics can not achieve. Furthermore, they have the capacity to do so within reasonable amounts of time. In fact, most recent papers using approximate approaches deal with metaheuristics, and the most frequently encountered in the personnel scheduling literature are the following :
– Tabu search,
– Genetic algorithms,
– Simulated annealing.

Metaheuristics usually function in the following manner : a simple constructive heuristic produces an initial solution, and afterwards the metaheuristic explores some solution space with the initial solution as a starting point. Often with applications in personnel scheduling, the solution space will be explored for instance by exchanging shifts between employees, changing the start time of a shift, or trading a shift for a day off.

More specifically, tabu search is an approach that explores the solution in the previously mentioned manner, and allows the search procedure to escape local optima by excluding them from the possible search space for some time, in order to prevent the search process from cycling over the same solutions continuously. These restrictions are kept in memory within a tabu list, hence the name tabu search. Dowsland (1998 [71]) proposes a tabu search for solving a nurse rostering problem. The proposed tabu search has the characteristic of using strategic oscillations to move around the boundaries of the feasible region, as well as chains of moves and the recognition of sub-optimal attributes in solutions. Basically, strategic oscillations is a method that allows the search mechanism to violate hard constraints, hence exiting the space of feasible solutions temporarily in order to re-enter it subsequently it in a different location. The standard neighborhood definition used here involves modifying the shift pattern for a single nurse. On the first test data set, the strategic oscillation tabu search was consistently able to find the optimal solution within 1000 moves. On a 60 MHz Pentium, executing 1000 moves took between 1 and 2.5 minutes. In comparison, a random descent algorithm, a simple tabu search and a simulated annealing could only obtain optimal solutions on 50% of the instances. Furthermore, on real data, all previously mentioned algorithms produced results with penalty costs almost twice as high as the strategic oscillation tabu search.
Bester & al. (2007 [37]) also propose a tabu search to solve a nurse rostering problem. Here, a total of five neighborhood moves are proposed, including swaps, modifications in values of variables and chain moves. The algorithm is implemented in visual basic 6.0. During a run of a little more than 120 iterations, the tabu search obtains a much better schedule than the randomly generated one, both in

terms of salary costs and dissatisfaction costs.

On the other hand, genetic algorithms explore the search space in a manner inspired from biological evolution : parent solutions are combined in order to produce a given number of offspring solutions. These offspring are created by combining the interesting sections of each parent solution, and in a small percentage of combinations, by adding a mutation factor to increase the exploration capacity. Easton & al. (1999 [73]) propose a distributed genetic algorithm for labor scheduling problems. The idea behind the concept of distribution is to have a set of different full schedule populations evolving in isolation, and to introduce a number of migrations between populations. Here, a chromosome is defined as a full schedule, and each element (gene) within it is an individual tour, which is a rather common way of defining a chromosome in personnel scheduling. Hence, the variable for each gene represents the number of employees assigned to that tour. Furthermore, the size of a chromosome depends on the number of feasible schedules, fitness is evaluated according to the objective function of cost minimization, and mating is done by classifying individuals according to fitness. Then, with a probability of being selected that is allocated according to their fitness level, individuals are selected and a cross-over is applied to them in order to create offspring, with a small probability of mutation. On the largest instances, more than 66 possible tours are available. The algorithm is programmed in FORTRAN and runs on an IBM RS/6000 550 workstation, with three subpopulations each of size 20. The approach is compared to a large set of others found in the literature, and the genetic algorithm outperforms all others by obtaining the least cost solution more than 66% of the time for general set covering problems, and 86% for stochastic goal programming. It also performs well for deterministic goal programming, but it does not outperform all other algorithms there. Furthermore, solution time is always on the higher half of the set of compared approaches.

Cai & al. (2000 [59]) also propose a genetic algorithm, this time for scheduling staff of mixed skills with multiple criteria. Here, each chromosome is coded as a set of all feasible schedules and the parents are selected by ranking. Then, two parents are selected (with a bias according to their ranking) and a cross-over is done to produce an offspring, with some probability of mutation. The offspring are then integrated in the next generation. The GA is programmed in C on a Sun Sparc station. Good feasible solutions are obtained within 10 minutes.

Finally, simulated annealing uses an analogy to the process of annealing steel in order to search for good solutions. It explores the search space using a single parameter representing a simulated temperature. As long as the temperature is high, the solution space is very easy to explore and constraint violations are not penalized much. Then, little by little, the temperature drops, and the solution space becomes more and more difficult to explore, until the temperature freezes on the best found solution. Brusco & al. (1993 [40]) proposes a simulated annealing approach to the cyclic staff-scheduling problem, which is compared to integer programming and linear-programming-based heuristics. A total of 72 test problems are used to compare each approach, with a maximum of 50 employees on a horizon of 1 week. Here, the authors provide a rather large definition of a neighborhood as being the dropping of a set of employee, and subsequently the rebuilding of the schedule back to feasibility. Integer programming was implemented on SAS/OR 6.06, and all heuristics were coded in FORTRAN 77, and executed on an Amdahl 5890. When parameterizing the simulated annealing heuristic, it was noted that neighborhood size had the largest impact on the quality of the solution. In average, the SA heuristic outperformed other approaches in terms of gap with respect to a lower bound, while run time were around 75 seconds, higher than most other approaches.

Brusco & al. (1993 [41]) also propose a simulated annealing approach for the solution of flexible labour scheduling problems. In this paper, many restrictions on schedule flexibility are eliminated, which results in problem environments containing more than 2 billion allowable tours. The neighborhood structure is defined in the same manner as in [40]. Tests are performed on a total of 36 different labour requirements distributions. On each instance, the obtained result is within a 2% gap from the lower bound. The SA heuristic obtained its best solution often within 10 seconds, and its allowable running time was 10 times smaller than that of the compared implicit model solved by branch and bound.

Other less frequent metaheuristics in personnel scheduling exist as well, such as the variable neighborhood (VNS) search used in Burke & al. (2003a [53]), where the search procedure can escape a local optimum by restarting the search at a new random point in the solution space through the use of a randomized control mechanism using a series of neighborhoods.

Papers on metaheuristics include : [3], [4], [5], [10], [18], [32], [33], [34], [35], [36], [37], [40], [41], [42], [50], [51], [52], [53], [54], [55], [57], [59], [71], [72], [73], [85], [88], [102], [103], [117], [120], [134], [143] and [153].

2.4.3 Constraint Programming

Constraint programming is much less frequent than the previously mentioned approaches in personnel scheduling. However, it is a very well suited approach for problems of the constraint satisfaction type, which arise quite frequently in personnel scheduling practice. Indeed, basic approaches of constraint logic programming do not include any objective functions to optimize, but simply constraints to satisfy. Hence, it is an adequate approach to problems where there is no natural objective to optimize, but many natural (hard) constraints to consider. It is also a well-suited approach to the philosophy of finding a good feasible schedule rather than an optimal one.

Auf'm Hofe (2000 [13]) presents a hierarchic constraint optimization approach. On top of a simple constraint optimization system, it combines an optimization function, as well as a set of constraints hierarchically arranged. The constraint hierarchy is used to classify constraints by level of importance : it allows the system to define mandatory constraints, important constraints, and less important ones. Here, there are hierarchies from level 0 to level 7. The system is based on a rostering system used in more than 60 German hospitals, but few details are given on how well it performs.

Abdennadher & al. (1999 [1]) propose a constraint logic programming system for nurse scheduling, based on a system called INTERDIP. Here, an objective function is also integrated in the system in order to evaluate and guide improvements of the solution after each iteration. Also, the problem is modeled as a partial constraint satisfaction problem, which allows the system to integrate the notion of soft constraints, not only hard ones. The system is said to produce satisfying schedules for 20 nurses within a few minutes, and is said to produce schedules equals or better to the ones manually made by an experienced head nurse.

Finally, Tsang & al. (2007 [163]) present an open source package named ZDC for rostering with constraint logic programming, which can be implemented with a different set of heuristics to explore the search space. A test is done with around 30 nurses for a horizon of two to three weeks on an Athlon 2500+ PC having 1 GB or RAM. Depending on the number of shifts to schedule (between 21 and 63), and the number of assistant nurses (8 to 12), solution time varies between 50 and 800 seconds.

Papers on constraint logic programming include : [1], [13], [63], [117] and [143].

2.4.4 Hybrid Approaches

Hybrid approaches combine aspects from at least two different methods. The mathematical programming based heuristics briefly presented at the end of section 2.4.1 can be considered as hybrid methods for instance. In this section, we will focus on papers about other types of hybrid approaches.

Rousseau & al. (2002 [143]) presents a hybrid approach combining constraint programming, local search techniques and genetic algorithms, to solve a physician rostering problem. First, a simple constraint programming model is used, subject to a set of distribution and pattern constraints. The idea is to use the constraint programming model to quickly generate a population of solutions. Then, these solutions are ranked according to a performance criteria such as the number of physicians who have a satisfying schedule. This population is then used as the starting point for an iterative process to improve the solution quality, based on partial reconstruction of existing solutions and combination of elements from different solutions. Two test runs are presented. First, for an instance of 23 physicians on a planning horizon of one month and six shifts per day, generating a feasible solution takes around 5 seconds, and a very good solution can take up to 15 minutes to produce, while with an efficient set of parameters, the optimal solution can be found in 5 minutes. The population has 10 individuals and the crossover is done with the best five. The second instance concerns 15 physicians over a planning horizon of 14 weeks, with 3 shifts per week-day and 1 per week-end, and takes around one hour and a half to run.

Burke & al. (2008 [51]) propose a hybrid method that combines an ordering heuristic and a variable neighborhood search for the nurse rostering problem. Here, the heuristic ordering procedure starts building the initial solution by sorting shifts in order of how likely they are to cause high penalties. Then, the most penalized shifts are assigned earlier in the schedule construction process, to the nurse that provides it with the least penalty for assignment. After, the VNS improves the solution of the ordering heuristic. Two neighborhood definitions are used : assigning a shift to a different nurse and swapping the nurses assigned to each of a pair of shifts. When a local optimum is reached, a fixed number of nurses who have the worst individual schedules are selected, and they are unassigned from all their shifts. The ordering heuristic is then used to rebuild those schedules, and hence serves as a restart mechanism. A total of 16 test instances varying between 12 and 30 nurses on a period of 13 weeks, are run on a P4 2.4 GHz. The results are compared to a commercial genetic algorithm, and the hybrid VNS is especially effective for instances under 20 nurses.

Dowsland & al. (2000 [72]) presents a hybrid tabu search and LP method for nurse scheduling also. Here, the tabu search is the main engine of the approach, and neighborhood moves involve changing the pattern of a single nurse. In order to avoid infeasibility problems during the solution process, a LP knapsack model is used for preprocessing in order to make available the correct number of nurses before starting the local search. Finally, a LP network flow model is used to make small adjustments on shifts allocations, as a post-processing tool. A total of 51 problems are tested, and the optimal solution is found for all problems with most of the runs.

Finally, hyperheuristics are another type of approach that can be considered as a hybrid. A hyperheuristic is an artificial intelligence mechanism consisting of a set of different heuristics ; through a learning process, it has the ability to determine which heuristic is the most likely to perform well on a problem including a given set of attributes. When a problem needs to be solved, the mechanism selects and uses the heuristics most likely to provide the best solutions for the problem. Burke & al. (2003b [57]) proposes a tabu-search hyperheuristic for timetabling and rostering.

Papers on hybrid approaches include : [13], [21], [32], [33], [34], [50], [72] and [117].

2.4.5 Other Approaches

There are a few more types of solution approaches in the literature on top of the ones presented previously. However, they are not encountered frequently. For instance, artificial intelligence techniques can be used, although they are usually combined with other approaches such as heuristics, and hence can be considered as hybrid methods. Beddoe & al. (2006 [32]), Beddoe & al. (2007 [33]) and Beddoe & al. (2009 [34]) present metaheuristic approaches including an artificial intelligence technique named case-based reasoning. With it, new problems are solved by considering the solutions to previous similar problems, and by keeping a history of hard constraint violations on many instances and how they were dealt with. This mechanism allows the metaheuristic to be guided towards good feasible solutions without explicitly defining soft constraint objectives. This is very practical for scheduling problems with many individual preferences, such as often encountered in nurse scheduling.

Other approaches include the use of a spreadsheet. Clearly, some spreadsheet models are linear or non-linear programs, but they can also simply be used for manual mathematical approaches. Ovchinnikov & al. (2008 [138]) presents a problem where between 15 and 20 residents in radiology must be scheduled. Here, in order to provide the chief resident with a tool that would mimic her manual approach to building schedules and hence not make her uncomfortable, both a calculator spreadsheet and and optimizer one on top of it were developed. Hence, the chief resident could use the linear program to produce a schedule, and try to apply modifications by hand to ensure the quality of the solution.

Laporte (1999 [113]) also discusses (among other things) some possible manual approaches, this time for rotating schedules. The author notes that even though many algorithms have been proposed for such schedules, they are often not flexible enough to produce a satisfying schedule. Hence, the author proposes that developing rotating schedules by hand can sometimes produce good results quicker. Throughout the paper, a series of simple arithmetic techniques are proposed, as well as tips. Finally, a case study for the Quebec ministry of transportation is presented.

Papers on other approaches include : [32], [33], [34], [95], [113], [117] and [127].

2.5 Applications

As personnel scheduling is an area of operations research motivated by real-life problems, much of the literature's content deals with either real or simulated applications. Most papers are specifically meant for one application, while a few others are described as more flexible and portable. Many classifications by application types have been proposed throughout past reviews ; the following, proposed by Ernst & al. (2004 [83]), is among the most detailed and is ordered by decreasing number of papers found in the literature :

1. Buses,
2. Nurse scheduling,
3. Airlines,
4. Railways,
5. Call centers,
6. General,
7. Manufacturing,
8. Mass transit,
9. Health care systems,
10. Civic services and utilities,
11. Venue management,
12. Protection and emergency services,
13. Other applications,
14. Transportation systems,
15. Hospitality and tourism,
16. Financial services,
17. Sales.

However, this classification considers a scope that is much larger than ours and includes elements such as crew scheduling and demand modeling problems. Ernst & al. (2004 [84]) uses the following somewhat more condensed classification :

- Transportation systems,
- Call centers,
- Health care systems,
- Protection and emergency services,
- Civic services and utilities,

– Venue management,

– Financial services,

– Hospitality and tourism,

– Retail,

– Manufacturing.

However, this last structure also considers a much larger scope than ours. Finally, the following is yet another classification, proposed this time by Aggarwal (1982 [2]) :

– Hospital health care systems,

– Emergency services,

– Transport systems,

– Street services,

– Telephone operations,

– Higher education systems,

– Library operations,

– Criminal justice systems,

– Electric power supply systems,

– Employee-to-customer individualized services.

Again, this last classification unfortunately includes many papers outside of our scope. Hence, because we wanted a structure both adapted and limited to our scope, and which would be representative of the most common applications encountered, we have therefore opted for the following more aggregated one :

– General applications,

– Health care services,

– Airport operations,

– Call centers,

– Postal services,

– Others.

This section will be structured according to this last classification. Because models and solution approaches have already been discussed, we will focus here on the characteristics of the applied problems, i.e. structure of the schedules, work regulations, coverage requirements, managerial objectives and other relevant environmental elements. Also, our focus will be on the more complex applications for each class.

2.5.1 General Applications

Here, we use the term general applications as being defined by the inclusion of the following two elements :

1. Purely academic papers studying a simulated generic problem,
2. Papers regarding flexible approaches that can (at least theoretically) solve a large number of different problems.

Because many papers in personnel scheduling are instance-specific, the first category includes the majority of the literature we will discuss here. Indeed, constraints, objective functions, size and complexity of the problem are all elements that can strongly vary from one application context to another. Hence, developing very flexible approaches is a hard task, and the strong limitations of current researches on this subject seem to have resulted in few technological transfers towards practitioners. On the other hand, a good number of instance-specific approaches have been implemented in practice. The following papers deal with theoretical generic problems.

Brusco & al. (1993 [41]) propose an approach to solve one of those generic problem. The characteristics dealt with here include that : the service is provided 16 hours per day and seven days per week, all employees work five days per week and are considered full-timers, shifts have a duration of 8 hours including a one hour break, breaks can start in five different time slots for each shift and can start at different times on different shifts of an individual tour, shifts can start at different times on an individual tour, days-off are not required to be consecutive, and the objective is to minimize the number of full time employees. Because the formulation used here doesn't allow understaffing, the best heuristic always produces 10 to 20% of excess labour.

Furthermore, some general application papers discuss characteristics that are not encountered in all personnel scheduling problems. Such is the case of Cai & al. (2000 [59]), which also discusses a generic problem of staff scheduling, but this time including mixed skills. The multi-objective approach considers three criteria : First, to minimize the total cost of manpower, second, to seek a schedule with maximum staff surplus, and third, to minimize the variations of staff surplus. A set of constraints ensures that workers are only used to cover labor requirements for skills they possess, and considers that some workers possess multiple skills. There is a total of 10 shifts per day, seven days a week.

In fact, most papers studying a generic instance focus on oversimplified problems that have little chances of being reproduced in real life without very large modifications and improvements. Like [59] with mixed skills, they frequently focus on researching one or a few basic aspects of personnel scheduling, such as flexible break assignments in [31] and [161], mixed workforce (part timers and full timers both in terms of shift length and number of works days) in [27] and [144], analyzing the impact of the number of scheduled employees on profits in [152], exploring the effect of variable productivity between employees in [158], dynamic service rates and learning curves within employees in [92], days-off scheduling with hierarchical skills in [82], [99] and [38], the latter with the addition of a minimum number of week-ends off for each worker in [136], multiskilled workforce for tour scheduling in

[80], designing shifts in [134], overlapping start-time bands in [106], improving the dispersion of surplus labor in [48], rotating schedules in [113] and [114], different work locations in [11], or non-continuously available employees in [153]. On the other hand, a few approaches pretending to be very flexible are developed based on generic problems ; they are the topic of the papers we will describe hereafter.

For instance, Thompson (1995 [151]) proposes one of those flexible approaches for shift scheduling. Here, the assumptions are the following : employees have identical skills and are continuously available for work, each shift receives at most one break, shifts and breaks may start at the beginning and finish at the end of any period, durations of shifts are consistent within each shift type, possible durations of work stretches before and after meal-breaks are also consistent within each shift type, and for each shift type, the cost is a linear function of the number of working periods in the shift (there are no shift premiums and meal breaks are unpaid). These assumptions are not very limitative and are quite realistic for many real life problems. The approach allows the user to define the possible shift lengths, the lengths of work stretches before and after meal-breaks, and the shift type - the attributes that will be the same for all shifts of this type (cost, length of meal-break, etc.). Results are presented with planning intervals as short as 15 minutes, and as much as 20 service operating hours each day.

Another interesting flexible approach is proposed by Jacobs & al. (1993 [105]), this time regarding tour scheduling. Here, the labor scheduling flexibility elements implemented are the following : days-off flexibility, shift-length flexibility, start-time flexibility, break-placement flexibility, start-time float and tour-length flexibility. One hour planning periods are used on a 7 days per week planning horizon, with however less than 24 hours of work per day. Furthermore, employees were assumed to be available for any tour and shifts were limited to 5 or 9 hours long, including a one hour break, while tours lasted either 2 or 5 days. Different generic instances were tested with different sets of labour requirements.

A very interesting table, providing good insight on flexibility of some proposed approaches by listing a large number of different attributes for each paper surveyed, is presented on page 166 and subsequents of Alfares (2004 [9]).

Papers on general applications include : [7], [8], [15], [17], [23], [25], [26], [27], [28], [29], [30], [31], [38], [39], [40], [41], [42], [48], [49], [59], [69], [73], [74], [75], [81], [82], [86], [91], [98], [99], [100], [103], [105], [107], [113], [114], [126], [132], [134], [135], [136], [137], [142], [144], [149], [150], [151], [153], [155], [158], [161] and [164].

2.5.2 Health Care Services

Health care is a very important area of application in personnel scheduling literature, among other things because many of its operations must run continuously 24 hours a day and 7 days a week, which increases the complexity of its related problems. Already in 1976, Fries (1976 [89]) compiled more than 30 papers on staffing in health care. At present time, the specific application on nurse scheduling largely outnumbers all the other ones in health care. Furthermore, nurse scheduling problems happen to have some characteristics rarely encountered in other problems. Hence, in this current section, we will isolate nurse applications from the other ones in health care.

Nurses Applications

Given the large number of papers dealing specifically with nurse scheduling, it is not surprising to find several review papers on the subject. Burke & al. (2004 [56]) consider a large number of papers and provides a list of approaches that were tested on real data and/or implemented in practice in hospitals. This is interesting considering that nurse rostering is a very practical application. As a matter of fact, Kellogg & al. (2007 [109]) also discuss the implementation of nurse scheduling approaches developed by researchers, and notes that only 30 % of systems discussed by research papers are implemented, while there is very little academic involvement in systems offered by third-party vendors. [56] suggests the following large number of elements as interesting future research areas :

- Multi-criteria reasoning,
- Flexibility and dynamic reasoning,
- robustness,
- Ease of use,
- Human/ computer interaction,
- Problem decomposition,
- Exloitation of problem specific information,
- Hybridisation,
- Inter-disciplinarity.

Also, a large number of appendixes can be found in Burke & al. (2004 [56]), classifying different papers on nurse rostering according to several criteria. Cheang & al. (2003 [60]) also surveys nurse rostering literature and presents the following list of commonly encountered constraints :

- Nurse workload (minimum/maximum),
- Consecutive same working shift (minimum/maximum/exact number),
- Consecutive working shift/days (minimum/maximum/exact number),
- Nurse skill levels and categories,
- Nurses' preferences or requirements,
- Free days (minimum/maximum/consecutive free days),
- Free time between working shifts (minimum),
- Shift types assignments (maximum shift type, requirements for each shift type),
- Holidays and vacations (predictable), e.g., bank holiday, annual leave,

– Working weekend, e.g., complete weekend,
– Constraints among groups/types of nurses, e.g., nurses not allowed to work together or nurses who must work together,
– Shift patterns,
– Historical record, e.g., previous assignments,
– Other requirements in a shorter or longer time period other than the planning time period, e.g., every day in a shift must be assigned,
– Constraints among shifts, e.g., only one shift be assigned to a person at a given time,
– Nurse requirements per skill category for each shift (minimum/maximum/exact number).

Burke & al. (2001 [50]) present an approach to solve a nurse rostering problem with different skill categories. Here, the wards consist of about 20 people, and the planning horizon is around 1 month. There is only one set of hard constraints, which are the labor requirements for each skill category. The other constraints are soft ones, and are the following :

– Minimum time between two assignments, depending on the type of duties involved,
– Maximum and minimum number of work hours during the planning period,
– Maximum number of assignments during the planning period,
– Working full weekends,
– Working according to a predefined pattern,
– Maximum number of assignments of each duty type during each week, and during the entire planning period,
– Restricting the order in which shifts and free days may and may not be scheduled (for instance never a night duty the day after a free day),
– Distributing the duty types uniformly over people with the same work regulations,

– Personal preferences such as days-off request, specific duty on a given day, illness leave, maternity leave, or temporary secondment to another ward.

Because of the complexity and large number of constraints in some nurse rostering problems, the use of soft constraints is often unavoidable as a large number of hard ones would most likely preclude any feasible solution. With such a number of soft constraints, the user sets the wanted penalty for each constraint violation, and the objective function then minimizes the weighted sum of all these penalties, making it a multi objective approach. This approach has been tested on four instances coming from real world Belgian hospitals. The resulting schedules are produced in times ranging anywhere from a few seconds to a few hours, which is clearly much faster than for the manual solution currently used. Furthermore, the resulting schedules are judged as being of a higher quality than their manual counterparts. [162] presents a nurse tour scheduling model with individual preferences considering :

– Schedule workload,
– Week-end and special days-off requirements,
– Shift type preferences,
– Consecutive-working-day limitations.

An interesting aspect of this approach is that it also considers the following flexibility alternatives :

– Start-time flexibility,
– Shift-length flexibility,
– Break-placement flexibility,
– Days-off flexibility,
– Start-time float,
– Shift-length float.

The goal programming approach used here allows the user to calibrate the respective importance of service coverage constraints versus employee preferences (both understaffing and overstaffing are allowed). Schedules cover a 7-day workweek, with one hour planning intervals, and shifts lengths of 8, 10 and 12 hours with a one hour break located within a 2 hour time window. Instances with two different objective functions are tested for wards of 11 nurses and optimal solutions are obtained.

Papers on nurse scheduling applications include : [1], [3], [4], [5], [13], [14], [18], [22], [32], [33], [34], [35], [36], [37], [50], [51], [52], [53], [54], [55], [57], [62], [63], [70], [71], [72], [88], [102], [108], [111], [117], [127], [130], [141], [147], [162], [166] and [167].

Other Healthcare Applications

Other personnel scheduling applications in health care are much less frequent than nurse rostering. However, most applications still come from within-hospital or hospital-based operations. For instance, Rousseau & al. (2002 [143]) proposes an approach for the scheduling of physicians, which is specifically structured to deal with constraints regarding individual tours. These constraints are split into two types : distribution and pattern. Distribution constraints deal with the number of shifts of a given type that a given physician can work during a predetermined set of days. They include preferences regarding if the week-end shifts should be worked in one block or separated, or limiting the number of night shifts that a physician can work. The pattern constraints deal with how work sequences in tours can be built. For instance, there could be a minimum of 16 hours between 2 shifts, no consecutive full week-ends allowed, or forcing a physician working a day shift to either work night shifts or be off for two days afterwards. Experiments were made on two instances, one with 23 physicians, a 1 month horizon and six shifts per day, and the other with 15 physicians, a 14 weeks horizon, three shifts per week-day and one per week-end day. Solutions require between 15 minutes and

1.5 hours of computational time, and tests were based on real data from hospitals of the Montreal area.

Also, Topaloglu (2009 [160]) proposes an approach for the scheduling of medical residents, which deals with a large number of constraints. Here, depending on the number of years of residency they have completed, residents are classified according to different levels of skills. The following is a list of the hard constraints considered :

- A senior and non-senior residents must be assigned to each weekday and weekend day shift.
- Shifts should be scheduled no more often than every third day for each resident.
- The number of weekend shift duties either remains the same or decreases within the same senior group as the duration of time in residency increases.
- The number of weekend shift duties of a resident in a senior group is less than or at most equal to that of a resident in one of the less senior groups.
- The difference between residents' total number of shift duties in consecutive senior groups should not be greater than two.
- The difference between the (m)th year and (m+2)th year senior residents' total number of shift duties should be at least greater than one.
- A resident's total number of shift duties is greater than or at most equal to that of a resident who has a higher seniority within the same group.
- Residents do not prefer working a shift on Friday because it is the last working day of the week and the day before the week-end holiday. For this reason, Friday shifts are allocated from the least ranked resident in each senior and non-senior group.
- Senior and non-senior residents are assigned to at most one Friday shift.
- Each non-senior resident should be assigned at least one week-end day shift if the number of weekend day shifts is greater than the number of non-senior residents.

- The first year residents to the unit should be preferred to the rotating non-senior residents if an extra shift duty is to be assigned. For this to happen, the rotating residents are already listed first in the non-senior group.

Even though this problem has a lot of hard constraints, it also has a large amount of soft ones. They are the following :

- Residents with the same number of years of seniority are gathered in different subgroups, each having approximatively the same number of days of seniority. Within a subgroup, residents should have equal number of weekend duties as well as equal number of total duties. The number of subgroups can be adjusted according to the scheduling environment.
- The total number of weekday and weekend day shifts between residents of consecutive senior groups can be equal to each other only if the less senior resident is assigned an extra weekend shift, otherwise the more senior resident should have been assigned at least one less duty in total.
- The difference between (m)th year and (m+2)th year residents' total number of duties should be at most three.
- Less senior residents within the same senior group can have at most one extra duty assigned compared to more senior ones.
- All non-senior residents should be allocated an equal number of duties. In case this equality cannot be provided, first the rotating residents and then the first years should be favored in the ascending duration of time in residency.
- Within the senior group, the duties on the middle days of a public holiday should be allocated to the less senior residents, while the beginning and ending days should be saved for the more senior ones.
- The total number of weekday and weekend shifts assigned to a resident should be less than or equal to the maximum number of weekday and weekend shifts specified according to his/her grade of seniority.
- A resident may or may not wish to work on specific days.

Furthermore, the objective functions are ordered as follows (by decreasing importance) :

- Equalize the number of weekend day shifts and total number of shifts assigned to the (m)th year residents who are in the same subgroup,
- Satisfy the soft constraints related to the shift numbers assigned to residents of different seniority levels,
- Satisfy the soft constraints related to the shift numbers assigned to senior residents within the same group and equalize the number of shift duties among the non-senior residents,
- Assign public holiday shift duties favoring the more senior residents for the beginning and ending days, allocating the less senior ones to the middle days,
- Do not assign more than the maximum number of allowed weekday and weekend day shifts for the residents,
- Satisfy the senior residents' requests and then the non-senior residents' requests.

Problems for wards with up to 40 residents based on real data have been considered, and solution times vary between 1 and 4000 seconds. The approach provides better results than the manual solutions previously used, and has been implemented in the pulmonary unit of a local hospital.

As can be seen from this above paper, some health care approaches require to consider a rather large number of constraints and objectives if their solutions are to be implemented in real life. Also, there is a trend in health care applications, mainly for nurses and physicians, to give more importance to the ergonomic qualities of a schedule and the individual preferences of the personnel, compared with past solution methods where service requirement was the only priority.

Finally, there are also a few cases of applications that are not hospital-based. One of them is presented by Ernst & al. (1999 [85]), which researches the rostering of ambulance officers within a single station. Here, a cyclic roster is made on a horizon of 48 weeks for 6 staff members, and includes the following possible assignments : Day shift (10h), night shift (14h), day off, leave, and management functions shifts. The schedule is built based on sequences of shifts called stints (the possible stints are built prior to the scheduling process). Work rules specify for instance that work stints must be followed by days-off stints. There are also transition rules, dictating which stint may follow which other, which can't, and which should preferably not. The quality of the roster is evaluated in terms of demand coverage (measured by undercoverage and overcoverage) and stint transition preferences, as well as with a measure of equity along the work sequence.

Papers on health care applications (excluding nurse scheduling) include : [85], [104], [138], [143] and [160].

2.5.3 Airports Operations

The remaining application areas we will discuss from hereon are far less important than the previous ones in terms of number of published papers ; they remain nevertheless interesting and relevant. Here, we define as airport operations the activities of airlines happening on the ground, mostly at customer service, luggage transport and handling or airplane refueling. Although less discussed in the literature than flight personnel scheduling (excluded from this review), they are nevertheless common subjects of research. A typical characteristic of these problems is that demand tends to fluctuate highly within a single day, sometimes with very short-lasting fluctuations that are difficult to cover. These fluctuations are caused by the arrival or departure of a plane. However, on the other hand, labour requirements are relatively easy to know with a good degree of certainty since the workload associated with the flight schedule can be known in advance. For instance, Hao & al. (2004 [95]) present an approach to solve an airport servicing staff rostering problem for an airline company. The characteristics of the problem are the following :

– Only full time staff members are to be scheduled,
– There are different types of duty shifts to be considered (early, middle, late, flex shift - a standby shift, and day off),
– The workforce volume fluctuates within a week, but not from a week to another,
– The appropriate number of personnel must be scheduled for each shift,
– The schedule is cyclic,
– Each staff member must take 2 days off per week (consecutive is better, but not compulsory),
– Full week-ends off should be schedules as much as possible,
– Each staff member is expected to work 42 hours per week, with some deviation allowed,
– An early or a flex shift must not follow a late-shift or a flex shift on two consecutive days,
– Different shifts should be evenly distributed across the roster.

The objective of this problem is divided between the following goals (classified in decreasing order of importance) :

1. Minimize the total amount of absolute deviation of working hours assigned to all staff members,
2. Maximize the total number of full weekends off.

Tests were made for instances varying between 40 and 500 employees, and took between one and 25 minutes to solve with the proposed neural network approach. Another paper for this type of application is presented by Alvarez-Valdes & al. (1999 [10]), which studies an application for labour scheduling at the refueling installations of a company operating in Spanish airports. The characteristics of the problem are the following :
– The installations provide continuous service 24 hours a day, 365 days per year,
– There are three types of periods : night, morning and evening,
– Labour requirements vary within days, across days and weeks,

- There is a total of 18 different shifts each day,
- There are full timers and part timers, as well as regular and discontinuous workers,
- Each worker has a set of admissible shifts,
- The shift assigned to a worker may vary from one day to the next, but the type of shift can only change after a day off,
- Each worker has a rest period of at least 36 consecutive hours each week,
- Each worker must receive at least one week-end off each four weeks,
- Full time employees must work an average of 35 hours per week,
- Regular workers can't work more than 223 days per year,
- Each regular worker is entitled to 30 days of holidays each year,
- The problem must be able to be solved for horizons ranging anywhere from one week to one year,
- Any feasible solution must respect labour requirements and worker's conditions,
- The problem is viewed more as a feasibility problem than an optimization one.

It takes 30 seconds for a one week schedule to be produced, 2 minutes for 4 weeks, and 30 minutes for one year. The solution approach, based on tabu search, has been implemented in a number of airports.

Papers on airport operations applications include : [10], [43], [44], [47], [61], [87], [95] and [125].

2.5.4 Call Centers

Personnel scheduling problems in call centers usually deal with a smaller number of work regulations constraints than the previous applications discussed. However, unlike in airport operations, demand can fluctuate strongly and unpredictably, which can complicate the decision-making process. Hence, the most complex problem to solve will frequently be the demand modeling, which is excluded from our scope. There are however a few exceptions. Lin & al. (2000 [119]) present the development of a workforce management system for a customer hotline service. The characteristics of the problem are the following :

- The center operates 24 hours a day,
- There are 8 shifts per day : 5 day shifts, one evening shift and 2 night shifts,
- Each employee works 22 days per month, minus the number of public holidays,
- Each shift has a meal break of a duration of one hour,
- Each officer should not work continuously for more than 6 days,
- At least 2 Sundays off and 1 Saturday off should be arranged for each staff member during one month,
- Each employee can request 2 specific days off during a month,
- Identical shifts on consecutive days is preferred by staff, as switching shift type everyday may cause confusion,
- There should be an acceptable number of off-duty periods between the end of a shift and the beginning of the following one,
- An officer should not be assigned more than 2 days off in a row (apart from the annual leave approved),
- At least one senior officer should be present in every daytime hour through specifying the following arrangements,
  - At least one senior officer should be assigned a given day shift everyday,
  - The senior officers of night shifts and evening shift should not be allowed off on the same day,
  - When the evening shift senior officer is off, at least two senior officers (one if Sunday) should be assigned on a given late day shift,
  - At least two junior officers must be on duty daily in the evening shift.
- At least 2 officers (including a senior) must be assigned to each night shift. On Sundays when traffic is the lightest, this may be relaxed to one officer assigned to one of the two shifts and two to the other one.

In the test instances presented, rosters are created for more than 70 employees by a decomposed approach, which takes into consideration both employee preferences and labour requirements. Another call center application of personnel scheduling is presented by Thompson (1997a [154]). The problem studied here is a shift assignment one, where the first objective is to minimize the number of unassigned shifts, and the second one is to satisfy the employee's personal choices of shift, in order of seniority if possible. The constraints ensure that : all employees get shifts, no employee gets more than his desired number of shifts, no more than one shift per day is assigned to any one employee, and a senior employee must work its desired shifts before a less senior one does. The heuristic developed was implemented after it was judged more performing than the previous manual method. Wilson & al. (1983 [171]) also proposes a heuristic, for scheduling telephone betting operators in horse racing. The cost of scheduling operators is not the same in the day as in the evening, and shift length may vary between 2h55 minutes and 6h10 minutes, including breaks. Here, the use of the simple proposed LP model has allowed savings of more than 350,000 $ per year, for 2 wards of over 200 operators each.

Papers on call center applications include : [16], [45], [46], [96], [97], [119], [154], [169] and [171].

2.5.5 Postal Services

Personnel scheduling problems applied to postal services are less frequent than any previously discussed application area. However, with many large mail processing centers operating continuously, it is another area where operations research approaches can provide important benefits. The modeling of problems here is akin to those for airport operations and call centers (2.5.3 and 2.5.4).

Bard & al. (2003 [21]) proposes an approach to solve a staff scheduling problem at the United States Postal Service mail processing and distribution centers. The characteristics of the problem are the following :

- There are three types of employees : Full time regulars, part time regulars and part time flexible (flexible employees are called when needed),
- Full timers shift last 8.5 hours, and include a half hour break, while part timers work anywhere between 4 and 8.5 hours per day (including a half hour lunch break when the shift is longer than 6 hours),
- The facility operates 24 hours a day, 7 days a week,
- A day is divided into 48 periods of half an hour each,
- There are 9 different possible start times for full time shifts,
- There are 12 different start times and 5 different lengths, which makes a total of 60 possible shifts for part timers,
- A worker must be given 2 days off per week, among which Saturdays, Sundays, and consecutive ones are preferable but not compulsory,
- Demand must be covered,
- The objective is to minimize the cost of the workforce,
- The schedule has a one week horizon,
- The number of part timers is limited.

With an approach that solves sequentially shift scheduling and days-off scheduling, respectively with integer programming and a heuristic, a solution with around 125 employees scheduled is obtained in around 45 minutes for this last problem. Wan & al. (2007 [165]) proposes another approach for a US postal service mail processing and distribution center, this time for staff scheduling with workstation group restrictions. The problem studied here has many similarities with Bard & al. (2003 [21]), but has shift lengths ranging from 4 to 12.5 hours, workstation group restrictions (akin to skills), and overtime constraints. An LP-based decomposition heuristic produces results within 2 or 3 minutes for instances of around

80 employees and four workstation groups, and in approximately 10 minutes for instances of around 230 employees and 8 workstation groups. Other papers on applications in postal services include Bard (2004 [20]) and Jarrah & Bard (1994 [107]), which both also study the case of the United States postal service facilities.

Papers on postal services applications include : [20], [21], [107] and [165].

2.5.6 Others

Many other application areas have been studied in personnel scheduling, but each one counting a small number of papers when focusing on our review scope. Many of them could fall under a category called retail sector, but because the reality of a fast-food restaurant is not the same as that of a bank, these remaining applications often have important differences between them. These papers include [128] which studies the scheduling of cashier at a supermarket store, [116] the scheduling of computer lab attendants, [118] the lockbox system personnel in a commercial bank, [112] the banking operations of check processing, [93] a newspaper publishing installation, [11] a set of petroleum stations, [110] police patrol cars, [123] a check proof and encoding facility, [120] a restaurant, [90] liquor stores, [80] a power station, and [121] a fast-food restaurant.

Papers on other applications include : [11], [24], [80], [90], [93], [106], [110], [112], [116], [118], [120], [121], [123], [128] and [172].

2.6 Conclusion

A large number of elements have an impact on the quality of a work schedule. Even though efforts have been made by researchers to find characteristics in solutions that empirically depend on constraint parametrization and choice of allowable schedule structures, no absolute rules have yet been found to know in advance what will have the most positive impact on the solution. Building a good personnel schedule remains partly a work of trial-and-error, and an art as much as of a science. This being said, the literature in personnel scheduling remains full of both interesting and useful ideas.

In this review, we have discussed models, solution methods, and applications. Regarding models, it seems that implicit modeling is a very promising approach, especially for more complex problems like tour scheduling. It allows a strong reduction of the number of variables, which increases the performance of integer programming. Furthermore, it doesn't require tours to be pre-selected and fed into the model, which in turn decreases the total time needed to obtain a solution. Regarding solution methods, a large number of them seem well suited to solve personnel scheduling problems. Here, the choice will most likely depend on the following : the size of the problem to solve, the time available to develop the method, the time available to solve instances, and of course the skills and preferences of the developer. However, for large problems, hybrid, sequential or iterative decomposition methods clearly seem very promising. Finally, regarding applications, it is obvious that researchers have proved how well real-life complex personnel scheduling problems could be handled with operations research methods. However, flexibility and portability of approaches from one problem to another (even within the same application area) remains a rarely discussed topic that we believe has much potential for future research.

Chapitre 3

A Sequential Decomposition Heuristic for Flexible and Portable Staff Scheduling in Health Care

3.1 Introduction

Providing health care services engenders important costs for stakeholders. For instance, according to the WHO (World Health Organization) [170], total expenditures on health represented 10.9% of the GDP while 18.1% of government expenditures were attributed to health in 2009 in Canada. In the province of Quebec only, around 45% of the annual public budget was allocated to the ministry of health care and social services in the financial year 2010-2011, out of which over 60% (or 20.5 billion dollars) were payroll expenses [131].

With such large financial stakes and limited public funds, optimizing the use of all available resources is crucial to maintaining cost effectiveness and a reasonable level of service accessibility. Furthermore, as an insufficient workforce must be reckoned with in health care, the quality of work conditions must be maximized simultaneously.

Individual applications in health care staff scheduling have been extensively dealt with by researchers, mostly on the topic of nurses as surveyed by Cheang et al. (2003) [60], Burke et al. (2004) [56] and Ernst et al. (2004) [83]. Also, emergency room physicians and ambulance officer scheduling applications have been studied respectively by Rousseau et al. (2002) [143] and Ernst et al. (1999) [85] among others. However, to the best of our knowledge a systemic methodology for dealing with families of related applications has not yet been proposed in health care. Such a methodology for staff scheduling is what we are proposing here, both for health care applications and others. As staff scheduling problems arising in the context of health care are among the most difficult to solve and vary significantly between each other in terms of problem structures, the context of health care applications has served as the main basis for designing our proposed approach.

In health care practice, tactical staffing decision-making is strongly decentralized : each profession (nurse, physician, ambulance officer, technician, etc.) is staffed independently within each unit (intensive care unit, emergency room, operating theaters, clinics, emergency medical service, etc.) with some unique set of objectives and constraints. In the context of an integrated health care system, it is too costly, time consuming and labor-intensive to develop one solution approach for each of these staff scheduling problems. In our opinion, this is a reason why OR research has resulted in a limited number of implementations in health care. To tackle this issue, we propose a more systemic methodology based on flexibility and portability, and the resulting solution approach named SOFA (Scheduling Optimizer with a Flexible Approach). We define flexibility as the capacity of a solution approach to tackle a large variety of instances, applications and problem structures, while we define portability as the capacity of a solution approach to be easily configured to solve a new instance, application or problem structure, quickly and with a limited amount of work. Based on these definitions, the objectives of this research are twofold :

1. Propose an efficient flexible and portable algorithm structure (SOFA) ;
2. Propose a framework for testing staff scheduling flexibility in order to provide benchmarking tools for past and future research.

This paper is structured as follows : the next section presents the literature review, followed by descriptions of the proposed solution approach and its extensions respectively in sections 3.3 and 3.4. The experimental framework developed to test flexibility and portability as well as the results obtained with the proposed approach are then discussed in section 3.5 and finally, section 3.6 concludes the paper.

3.2 Literature Review

A few of the solution approaches proposed in the literature to solve generic problems are flexible to some extent. For instance, Bechtold et al. (1991) [31] and Topaloglu et al. (2002) [161] research flexible break assignments. Meanwhile, flexibility of shift duration and of number of work days for individual tours is studied by Bechtold et al. (1994) [27] and Showalter et al. [144] (1988). Also, Isken (2004) proposes a generic staff scheduling implicit model for applications in health care and provides analysis on the impact of a limited number of factors such as start time bandwidth (the daily time interval within which shifts of a given employee may start). For a detailed survey of the staff scheduling literature, including but not limited to generic approaches and health care applications, we refer the reader to the literature survey of this thesis in section 2.

While some interesting generic solution approaches have been proposed in the literature, their scope generally does not consider the large variety of realistic staff scheduling characteristics often found in health care applications. For example, the kind of complex and conflicting sets of constraints often encountered in practice is rarely dealt with in the literature. Furthermore, to the best of our knowledge, the notion of portability of a solution approach on a variety of applications has not been discussed yet. Indeed, we believe in the importance for flexible approaches to require a limited configuration workload when tackling a new application. Hence, in this chapter we propose SOFA, an algorithm designed to contribute to research on flexible solution approaches for staff scheduling in health care and to introduce the notion of portability. The following section will describe the proposed structure of this algorithm.

3.3 Solution Approach

In order to maximize flexibility and portability, we have devoted much attention to the structure of the solution approach. Before presenting the proposed structure in more details, we will define some terminology which will be used further to discuss our staff scheduling problem : a *period* is a discretized unit of time, generally representing an hour in this paper ; a *planning horizon* is the continuous set of periods for which a schedule needs to be provided, generally representing two weeks in this paper ; a *work shift* is a continuous set of periods including a lunch break, during which a worker is available to provide service at any time except the lunch break ; a *work shift type* defines a work shift by duration (total number of periods from its beginning up until its termination), by duration of its lunch break and by lunch break time window ; a *lunch break* is a number of continuous periods inside a work shift during which the worker is not available to provide service ; a *tour* is a set of work shifts spread out through the planning horizon which represents the schedule of a single employee (also called an individual schedule) ; a *tour type* defines a tour by work shift type and total number of work shifts within the planning horizon ; and a *demand* represents the service requirements for periods of the planning horizon. Finally, a *coverage* represents how much service resources are provided during the periods of the planning horizon : *Undercoverage* is present when coverage is lower than demand for a given period, while *overcoverage* is present when coverage is higher than demand for a given period.

3.3.1 General Structure

A comprehensive analysis of the literature and of our consulting experiences in staff scheduling applied to health care has resulted in the following list of requirements for a flexible and portable approach : Able to solve the continuous tour scheduling problem ; include the fewest possible hard constraints ; allow the user to calibrate it for a new problem by modifying sets of constraints and parameters, but without altering its core structure or optimization procedures ; let the user define the bounds of possible individual schedules and shifts structures, without requiring him to list all the possible solutions.

These requirements are justified by the following factors : Some health care services such as emergency rooms need to operate continuously ; As all future problems to solve can not be fully anticipated during the development phase of the algorithm, most basic structural constraints should be soft in order to avoid unpredictable solution feasibility issues ; As a large number of different problems need to be solved, the different users should not have to dedicate a large amount of time to the calibration of the approach or its use will become inefficient and costly ; As these types of problems are complex to solve and sometimes have large feasible solution spaces, a user will not have the time nor an interest to dedicate time listing all the different possible individual schedules in order to solve the problem.

These elements have led us to build a continuous tour scheduling approach implementing only one hard constraint which limits the assignment of work shifts to one per day, uses mostly soft constraints to avoid infeasibility, and defines possible tour types by a combination of shift duration (up to 24 hours in duration), number of worked shifts, lunch break duration, and lunch break assignment time window. Furthermore, this approach allows the user to calibrate the bounds of all the different tour types wanted, the duration of the planning horizon, the duration of periods, and the penalty weight and right hand side values of the required constraints. As these constraints are always structured either on a period axis or on a tour axis, we propose the following classification by constraint types which is based on a classic matrix representation where tours are defined by lines and periods by columns.

- Vertical constraints :
  - Constraints on a single period :
    - Service demand coverage ;
    - Capacity (ex. : number of available vehicles or workspaces, budget per period, . . .).
- Horizontal constraints :
  - Constraints on sets of lines :
    - Management constraints (ex. : maximum number of tours of each type, budget by number of employees, . . .).
  - Constraints on a sets of shifts of a single tour :

    – Collective agreements, ergonomics, working conditions (ex. : maximum number of consecutive work days, minimum number of consecutive days off, . . .).
– Constraints on a single shift :
    – Collective agreements, ergonomics, working conditions (ex. limits on possible start-times for work shifts, . . .).

This classification of constraints will serve as a basic structure for our flexibility experimental framework presented in section 3.5. We have also designed the core of our solution approach in order to profit from this classification, by dividing the complete solution process including all constraint types into a number of phases which would each require only a limited subset of these constraint types. The proposed structure is the following sequential decomposition approach :

1. Solve the days off / days on problem ;
2. Solve a shift scheduling problem where work shifts of a given tour have the same start time ;
3. Solve a shift scheduling problem where work shifts may have different start times.

This approach will be discussed in detail in section 3.3. While instances of typical size for the full version of the staff scheduling problem can't be solved to optimality, most instances of each of our three subproblems could be solved with a commercial linear solver, save for the following elements :
– To maintain flexibility and avoid infeasibility, most constraints will be modeled as soft. However, linear programming tends to lose some efficiency when dealing with objective functions comprising many penalty terms ;
– Some complex constraints can be either difficult or unrealistic to model in a linear fashion ;

- Because of the large number of qualitative factors such as ergonomic quality, employee preferences and work conditions, staff scheduling in health care are typically constraint satisfaction problems rather than pure optimization ones ;
- Most health care managers often feel more comfortable with choosing the best schedule themselves among a set of proposed good schedules, rather than being provided with a single mathematically optimal solution by a computerized system ;
- As staff scheduling in practice is a process involving stakeholders with conflicting objectives (managers and unions), and as the production process of a new schedule is typically seen as an opportunity to analyze new work arrangements, the process is generally iterative, time consuming, and requires the production of a large number of scenarios. In this practical context, the algorithm needs to produce good solutions quickly for every scenario regardless of the size and complexity of the instances. Otherwise, deadlines for the scheduling process are not likely to be met.

The following paragraphs details phase 1 which solves the days off subproblem, phase 2 which solves a limited shift scheduling subproblem, and phase 3 which solves an extended shift scheduling subproblem. They will be followed by a description of extensions which will conclude this section.

3.3.2 Phase 1

Phase 1 is limited to solving the days off subproblem. Because it only considers whether a shift is assigned to a given day, phase 1 can quickly deal with all horizontal constraints on sets of lines and on sets of shifts. Meanwhile, as decisions related to these constraints are not questioned in subsequent phases, they can be excluded from the more time consuming phase 2 and 3. On the other hand, all constraints on individual shifts can be excluded from phase 1 as it does not deal with decisions regarding to which periods shifts are assigned. Globally, this makes phase 1 very efficient as the size of the problem it solves is limited even when the global approach deals with a very large problem.

Phase 1 is built as a variable neighborhood descent (VND). Because this phase is initialized with an empty initial solution, its local search is both a constructive and an improvement process executed by two local search structures : a line builder and an assignment swapper.

The line builder is the first activated local search structure of phase 1 and creates a days off pattern for each single line (individual schedule). Since its neighborhood is defined as the addition, replacement or deletion of an entire line, it makes phase 1 both a constructive and improvement heuristic. Its local search proceeds as follows : one complete work day assignment and one completely empty assignment are tested sequentially for each line. Complete work day assignments are built by randomly allocating a work assignment to an available day. The new solution value is calculated after each tested modification and the first improvement found to the current solution is immediately implemented and terminates a local search iteration. If no improvements are found, the current iteration terminates when each line has been visited. The algorithm for the first local search of phase 1 (the line builder) is succinctly presented on figure 3.1.

The second local search structure is an assignment swapper : it swaps days off and work days within a single line of work. For each line of work, it executes each possible swap between one day off and one work day. The local search terminates when all possible swaps have been tried, and the best swap is implemented at the end of the local search iteration only if it improves the current solution (best improvement). The algorithm of the second local search of phase 1 is presented

```
sol_A : current solution
v_A : value of sol_A

For each line l

    For each tour type t

        sol_B ← sol_A with a new random assignment for line l, tour type t
        v_B = solution value of sol_B

        If v_B < v_A
            sol_A ← sol_B
            v_A = v_B
            terminate local search
        End if

    Next tour type t

Next line l
```

Figure 3.1 – Algorithm of the First Local Search of Phase 1 (the line builder)

succinctly on figure 3.2.

Phase 1 always initiates the local search process with the line builder since it is the only neighborhood structure allowing the improvement of an empty (or partial) solution. Afterwards, the algorithm alternates between the two local search structures each time one of them cannot find any improvements during one full iteration. Phase 1 terminates after two full iterations without improvements (one with each neighborhood).

For flexibility purposes, the objective function of phase 1 is to minimize the sum of all costs arising from soft constraint violations. Except implicit constraints such as the number of work shifts for a given tour type, all constraints are implemen-

```
sol_A : current solution
v_A : value of sol_A
sol_C : empty solution
v_C : value of sol_C

For  each existing tour t

    For each possible swap s

        Swap a work day and a day off
        sol_B ← sol_A with swap s
        v_B = solution value of sol_B

        If  v_B < v_A and v_B < v_C
            sol_C ← sol_B
            v_C = v_B
        End if

    Next swap s

Next tour t

Implement the best solution between sol_A and sol_C
```

FIGURE 3.2 – Algorithm of the Second Local Search of Phase 1 (the assignment swapper)

ted as soft. Clearly, these constraints will vary from one problem to another, but typically they will include a subset of the following : service demand coverage (target or minimum level) ; upper or lower bounds on the number of consecutive work days ; lower bounds on the number of consecutive days off ; upper and lower bounds on the number of work days within a calendar week ; upper bound on the total number of lines of work ; upper bound on the total number of work hours ; lower bound on the number of complete week-ends off.

In days off problems, the duration of a period is one day (24 hours). As tour scheduling problems typically have periods of 12 hours or less, it is necessary to convert service demand values for the days off problem. This conversion is made by aggregating the demand of the initial tour scheduling problem per day. For instance, if the initial tour scheduling problem has a service demand discretized into 3 periods of eight hours per day, the demand of phase 1 for a complete day of the planning horizon will be the sum of the corresponding three values (one for each period of 8 hours) from the initial tour scheduling problem. An example of conversion is presented in table 3.I. On Monday for instance, the sum of the night demand (5), the day demand (10) and the evening demand (6) equals 21 which is the value used for the aggregated demand for Monday in phase 1.

Problem	Period	Mo	Tu	We	Th	Fr	Sa	Su
Tour scheduling	0 :00 am to 8 :00 am	5	6	7	7	6	5	4
	8 :00 am to 4 :00 pm	10	12	13	12	11	8	7
	4 :00 pm (day j) to 0 :00 am (day j+1)	6	7	7	6	5	4	5
Days on/Days off	0 :00 am (day j) to 0 :00 am (day j+1)	21	25	27	25	22	17	16

Table 3.I – Example of Service Demand Conversion Between the Tour Scheduling and the Days On/Days Off Problem for One Week from Monday to Sunday

The solution obtained with phase 1 is a set of lines of work where the days off found in all tours of the final solution are selected. Once the algorithm of phase 1 has reached its termination criterion, the best found solution is transferred to phase 2 which is described in the next subsection.

3.3.3 Phase 2

The solution of phase 1 provides the days off for the final solution to the staff scheduling problem. The objective of phase 2 is to assign a start time for shifts corresponding to work days provided by phase 1. Phase 2 provides solutions where all work shifts of a tour start at the same period. This single start time may however be located at any period of the day. The number of planning periods within a day is a user defined parameter which can theoretically vary between 1 and infinity. In practice, this number is generally either three when dealing with nurse scheduling problems where each period corresponds to one of the three usual 8 hours shifts of any day, or 24 when dealing with more complex problems where continuous demand variations create a need for hourly discretization and a large number of possible start times for shifts.

The initial solution of phase 2 is built by randomly assigning a single shift start-time to each line of work from the best solution obtained after phase 1. While phase 2 only solves a limited shift scheduling problem, combining its results with those of phase 1 provides a first solution to the tour scheduling problem. Excluding its initial solution mechanism, phase 2 is a single neighborhood local search heuristic.

The local search structure of phase 2 is a set of shift start-time movers in which the neighborhood of a solution is defined by modifying, simultaneously for all shifts of a given line, their start time to a value comprised between the first and last period of a work day. Hence, the local search sequentially evaluates the impact of moving the shift start time of each line to all possible periods. After all possible movements have been evaluated, an iteration is terminated by the implementation of the best move only if it improves the current solution (best improvement). The algorithm of the single local search procedure of phase 2 is presented succinctly in figure 3.3.

Because phase 2 has a single local search structure, its optimization process stops immediately after one iteration has been completed without producing any improvement to the current solution. Since the only set of variables in phase 2 manages start times for groups of work shifts, most constraints here deal with periods rather than lines of work. Typically, these constraints will include the following :

- Service demand (target or minimum level) ;
- Service capacity (an upper bound on the possible number of workers on duty at one given period) ;
- Limits on the number of work shifts starting or ending at given periods.

Service demand is expressed in full detail in phase 2 and no aggregation is needed (unlike during phase 1). The main objective of a solution to this subproblem is to provide a good draft of the final schedule. In fact, in many cases the solution provided by phase 2 will be very similar to the final one. A major advantage of the structure proposed for phase 2 is the small amount of time required to solve the problem, considering it has only one integer decision variable per line of work (the start time of all shifts). Furthermore, the set of constraints will typically be small as well since it will be limited to vertical constraints and horizontal constraints on individual shifts. When phase 2 terminates its optimization process, the best solution found is transferred to phase 3.

```
$sol\_A$ : current solution
$v\_A$ : value of $sol\_A$
$sol\_C$ : empty solution
$v\_C$ : value of $sol\_C$
$sol\_C$ leftarrow $sol\_A$
$v\_C = v\_A$

For  each existing tour $t$

    For each period of a day $p$

        Set start time $p$ for all shifts of tour $t$
        $sol\_B$ leftarrow $sol\_A$ with start time $p$ for tour $t$
        $v\_B$ = solution value of $sol\_B$

        If $v\_B < v\_C$
            $sol\_C \leftarrow sol\_B$
            $v\_C = v\_B$
        End if

    Next period $p$

Next tour $t$

Implement solution $sol\_C$
```

Figure 3.3 – Algorithm of the Single Local Search of Phase 2 (shift start time modifier)

3.3.4 Phase 3

The best solution found during phase 2 serves as the initial solution for phase 3, which provides a final solution to the tour scheduling problem. Phase 3 is an extension of phase 2 : it assigns specific and potentially different start times to individual shifts, providing the solution approach with the flexibility needed to further improve service coverage by implementing rotation of work shifts within tours if necessary.

Phase 3 is a single neighborhood search heuristic : its local search is an individual shift start time mover. The neighborhood of a solution is therefore defined by the move of the start-time of a single shift between the first and last period of a work day. The local search evaluates sequentially each possible starting period for all shifts of all tours. After having evaluated all possibilities, an iteration of the local search is terminated by the implementation of the best movement only if it improves the current solution (best improvement). The algorithm of the local search procedure of phase 3 is presented succinctly in figure 3.4.

Because phase 3 has a single local search structure, its optimization process stops immediately after one iteration has not produced any improvement to the current solution. Being an extension of phase 2, all constraints of phase 2 must also be present and satisfied in phase 3. The additional constraints in phase 3 will typically be the following :
– Limits on the variation of start times between consecutive work shifts ;
– Limits on the variations between all start times of one work line (start-time bandwidth constraints) ;
– Limits on the number of different start times within one tour.
Combined together, these constraints allow much flexibility for the structure of tours. They also provide the user with the possibility to either allow or block

```
$sol\_A$ : current solution
$v\_A$ : value of $sol\_A$
$sol\_C$ : empty solution
$v\_C$ : value of $sol\_C$
$sol\_C$ leftarrow $sol\_A$
$v\_C = v\_A$
For each existing tour $t$

    For each work shift $s$

        For each period of a day $p$

            Set start time $p$ for shift $s$ of tour $t$
            $sol\_B \leftarrow sol\_A$ with start time $p$ for tour $t$
            $v\_B$ = solution value of $sol\_B$

            If $v\_B < v\_C$
               $sol\_C \leftarrow sol\_B$
               $v\_C = v\_B$
            End if

        Next period $p$

    Next work shift $s$

Next tour $t$
Implement solution $sol\_C$
```

FIGURE 3.4 – Algorithm of the First Local Search of Phase 3 (individual shift start time modifier)

rotation of work shifts within a tour while maintaining a strong control over configuration of the rotation. The solution space of phase 3 is very large : Hence, even though phase 3 is an extension of phase 2, phase 2 remains necessary in order to quickly provide a good initial solution to phase 3.

In our opinion, the three phase algorithm structure we propose is well adapted to a flexible and portable context. Indeed, this sequential decomposition will allow the algorithm to exhibit quick solution times while having few limitations in terms of implementable problem characteristics. Furthermore, the use of randomness reduces the calibration workload when tackling new problems, making this solution approach simple and quick to parameterize. However, these qualities are meaningless without good performances in terms of solution quality. Indeed, sequential decomposition approaches generally have some weaknesses regarding the quality of the solutions they provide. In order to deal with these and make our approach more robust, we have developed a number of extensions presented in the next section.

3.4 Extensions

The most important limitation of a sequential decomposition approach is that decisions are taken in prior subproblems with limited information regarding their consequences on subsequent subproblems. Indeed, we have no guarantee that the best solution in phase 1 will lead to the best solutions in phases 2 and 3. Furthermore, because phase 2 serves as an initial solution for phase 3, the solution approach must be safeguarded against launching phase 3 in a local optima it can't escape. Hence, in order to mitigate the effects of these limitations, we have implemented one reshuffling and two backtracking mechanisms.

3.4.1 Reshuffle Mechanism

We propose a reshuffle mechanism (RM) somewhat inspired from the random restart of Variable Neighborhood Search (VNS). However, this mechanism is applied to the complete solution process instead of being executed within a phase. Once phase 3 has terminated, the best solution found is stored. We then obtain a new partial solution by erasing a proportion p of tours. The specific value of p is chosen randomly between predefined lower and upper bounds, and the tours erased are also selected in a random fashion. The solution process is then restarted at phase 1 with this partial solution. This mechanism is repeatedly executed until some termination criterion is satisfied, which is expressed as a number of complete solution processes executed without improving the best solution found. The reshuffling process can either be applied to the best known solution, or to the best solution of the current iteration.

We use randomness intensively for flexibility and portability purposes. Indeed, while using predefined strategies should likely produce better results on some instances, those same strategies may reduce the efficiency of the search on different problem structures. Furthermore, since our solution approach will deal with a variety of problem structures, a large set of predefined strategies would be required. The supplementary workload required to select the most adequate strategy for the current problem would increase solution times, which would reduce the number of applications for which our approach could be used. Hence, by using randomness, we protect our solution approach from risks related to the negative impacts of overspecialization in a flexible environment, and from risks related to creating obstacles to the solution process by using different conflicting exploration strategies.

3.4.2 Backtracking Mechanisms

A first limitation of RM is that it cannot guide a search as it relies solely on random selections. A second limitation is its significant consumption of CPU time as it requires the complete solution process to be executed several times. Hence, there is a need for both time efficient and guided optimization mechanisms.

In order to fill this need, we have developed two backtracking mechanisms. In a nutshell, they are an adaptation within phase 2 and 3 of the two local search structures used in phase 1 : they allow the algorithm to improve the solution of the days off subproblem while solving the shift scheduling subproblem. The main idea of the backtracking mechanisms is to profit from the information available in the current phase to improve the decisions made with less information during a previous phase.

The first backtracking mechanism is the tour builder (BM1). It is designed to be activated during phase 2 or phase 3 when the basic local search procedure of the current phase is unable to provide new improvements. The neighborhood of BM1 is very large : it consists of any solution obtained by adding, modifying or deleting one tour. In cases of addition or modification, BM1 sets the days off pattern of the tour (phase 1) while simultaneously defining the start time for its work shifts (phases 2 and 3). The local search implements modifications to the days off structure in the same manner as the first local search of phase 1 (the line builder), while it locates the best possible start time for all shifts of the current tour in the same fashion as the local search of phase 2 (the shift start time mover). Clearly, executing simultaneously a search in both solution spaces of phase 1 and 2 is time consuming. However, BM1 is a very powerful tool which can significantly remodel a solution and hence improve many past decisions. BM1 implements first improvements and its algorithm is summarized in figure 3.5.

The second backtracking mechanism is the day swapper (BM2), which is implemented in phases 2 and 3 with a structure based on the second local search of phase 1 (the assignment swapper). In fact, the differences between BM2 and the second local search of phase 1 are that the former can swap two work days (if the work shifts are not identical) and must update the start times when swaps are executed. Indeed, BM2 swaps both the work status of the day (work day versus day off) and the start time of the work shift. Unlike BM1 however, BM2 does not search the solution space for possible improvements of the start times. While BM2 has a limited impact on the solution structure in comparison with BM1, it is necessary in order to make precision improvements which BM1 cannot execute because it uses randomness to build the days off pattern. Unlike the second local search of phase 1, BM2 is structured to implement first improvements. The algorithm of BM2 is presented in figure 3.6.

The extensions presented in this section can combined in the solution approach. In fact, the combination of the backtracking mechanisms with the reshuffle allow the solution approach to profit from the improved guided search provided by the former and from the extensive diversification capabilities provided by the later. The results obtained with the proposed solution approach are presented in the next section.

3.5 Computational Experiments

In this section, we present experimental framework on which two variants of our solution approach were tested. This framework was developed specifically to test the flexibility of solution approaches on staff scheduling problems, and in our

```
sol_A : current solution
v_A_p2 : value of sol_A in phase 2
v_A_p3 : value of sol_A in phase 3

For  all lines l

    For  all tour types t

        For  all periods of a day p

            Assign a new random work day pattern for line l, tour type t
            Assign start time p for all work shifts
            sol_B ← sol_A with schedule l, t, p
            v_B_p2 = solution value of sol_B in phase 2
            v_B_p3 = solution value of sol_B in phase 3

            If  v_B_p3 < v_A_p3

                sol_A ← sol_B
                v_A_p2 = v_B_p2
                v_A_p3 = v_B_p3
                terminate local search

            End if
        Next period p
    Next tour type t
Next line l
```

FIGURE 3.5 – Algorithm of BM1

```
sol_A : current solution
v_A_p2 : value of sol_A in phase 2
v_A_p3 : value of sol_A in phase 3

For  all lines l

    For  all work/off day swap s

        Execute swap s on line l for all three phases
        sol_B ← sol_A with swap s, l
        v_B_p2 = solution value of sol_B in phase 2
        v_B_p3 = solution value of sol_B in phase 3

        If v_B_p3 < v_A_p3

          sol_A ← sol_B
          v_A_p2 = v_B_p2
          v_A_p3 = v_B_p3
          terminate local search

        End if

    Next swap s

Next line l
```

FIGURE 3.6 – Algorithm of BM2

opinion it constitutes an adequate reference basis for benchmarking past and future solution approaches proposed for staff scheduling problems. An analysis of the impact of different service demand requirements is then carried out, followed by an analysis on the impact of different sets of constraints. All programs were coded in C++. All experiments were carried out on an Intel®Core^{TM}Duo CPU running at 1.86 GHz with 1 Gb of RAM.

3.5.1 Experimental Framework

After extensive preliminary testing, we have set the lower and upper bound values for parameter p respectively to 0.10 and 0.30. This means that between 10 and 30% of tours will be erased from the solution when the RM is activated. Also, the stopping criterion of RM has been set to five iterations without improvement of the best solution found. For portability purposes, these parameters have been set to values yielding the best results on average in order to avoid the need to adjust them when dealing with a new problem instance.

Since the scope of our experimental framework is limited to the analysis of impact of types of constraints and service demand patterns, a number of instance characteristics will remain identical in all instances : A continuous planning horizon of 2 weeks (14 days), 24 work periods per day (one hour each), and 2 types of tours. The first tour type has 10 work days with shifts of 8 hours and a continuous break of 45 minutes within a time window opened on the 4 median periods of the shift, and the second type has 7 work days of 12 hours shifts with continuous breaks of 75 minutes also within a time window opened on the 4 median periods of the shift. Furthermore, a constraint forcing forward rotation of work shifts within a tour has been implemented in all problem instances, and its violations for each period of each shift are penalized by 30 points.

Demand Curves

Our experimental framework includes a total of 5 different service demand coverage curves specifically built for our tests. To allow an analysis on the impact of demand patterns on the cost of the solution, all five demands have an equal surface underneath their curves which means that the total theoretical number of service hours required is the same for all. These five service demand curves are presented in figure 3.7. They are respectively demands which represent a daily mild double peak (public transit, city tollbooths), a high early morning and low late afternoon peak (shipping personnel of a warehouse), a low early morning and high late afternoon peak (office cleaning personnel), a daily high peak (emergency services, retail stores, restaurants) and a uniform demand (factory personnel).

As each of our five initial demand curves averages a value of 1 on the vertical axis, we have multiplied the values of each point on each period of each curve by 30 in order to obtain a real-life representative size for our instances. This set of five service demand curves represents a good variety of shapes similar to those encountered in real-life applications (although their amplitudes may of course strongly vary in practice).

Constraint Sets

The proposed experimental framework includes instances with a variety of combinations of constraints from each class (see section 3.3.1 for our classification by type of constraint). This was necessary in order to thoroughly evaluate the level of flexibility of our approach, and to analyze the impact of different classes of constraints on schedules. While it is not possible to include all possible constraints of each class in our framework, it is reasonable to assume that constraints of a given class have similar impacts both on the structure of solutions and on the

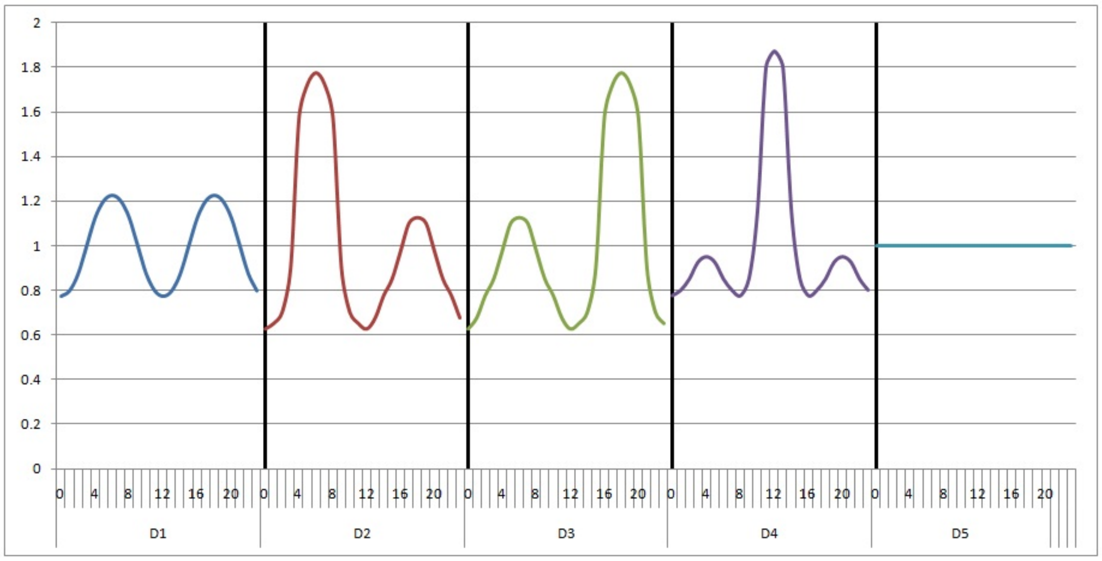


FIGURE 3.7 – Demand curves of D1, D2, D3, D4 and D5

performance of the solution approach. This last assumption is important because it is impossible to test all existing staff scheduling constraints in a single experimental framework.

The penalty for violations of each constraint is presented in figure 3.II, while the different combinations of constraint sets used in our experimental framework are presented in figure 3.III. When relevant, given sets of constraints have been tested with two different right hand side values : a looser bound (level 1) and a tighter one (level 2). In the class of vertical constraints, service demand target and service capacity constraint sets have been included. Service demand target constraints penalize both overcoverage and undercoverage by one point, but only undercoverage penalties for each period are squared in order to avoid solutions with large service level variations between periods. Meanwhile, capacity constraints model situations where levels of service coverage are limited by service capacity. In an emergency room for instance, it could model the number of beds, stretchers or triage desks. Likewise, it could model the number of available offices for physicians in a clinic and the size of the fleet of ambulances in an EMS. The first level for this constraint is expressed as a right hand side equal to 95% of the peak service demand for the instance solved, while the second level is expressed as 85% of this same peak. Furthermore, each service unit of violation is penalized by 10,000 points : this value was set very high in order to avoid violations, as service capacity is typically a hard constraint in practice.

Category		Constraint	Right hand side value		Penalty value per violation
			Level 1	Level 2	
Vertical		Target service demand	Service demand		Overcoverage : 1 (exp. 1.5)
					Undercoverage : 1 (squared)
		Capacity	0.95% of peak	0.85% of peak	10,000
Horizontal	Set of lines	Budget	130 employees	120 employees	10,000
		Maximum tours of each type	999, 50	999, 50	5,000
	Set of work shifts	Max consecutive work shifts	6, 4	5, 3	500
		Maximum consecutive days off	1, 2	2, 2	500
	Single shift	Limits on start times	Between 2 and 5 am		500
		Limits on end times			500

TABLE 3.II – Description of Constraints

In the general class of horizontal constraints, the first subclass includes constraints on sets of lines. This subclass includes constraints on the size of budget and on the maximum number of tours of each tour type. In our framework, constraints on the size of the budget have been expressed as a maximum number of employees, since all tour types have a comparable number of service hours. In other circumstances, it could have been expressed as the maximum total number of service hours. The right hand side value of this constraint is set at 130 employees for the first level, and 120 for the second. Each employee (tour) violating this constraint is penalized by 10,000 points : this value was again set very high since the number of available employees is typically a hard constraint. Meanwhile, constraints on the maximum number of tours of each tour type have been implemented to model situations where managers wish to force some diversity in the type of tours offered to their employees. There is only one level for this constraint in which no more than 50 of the second type of tours may be included in a solution. Each unit of violation for this constraint is penalized by 5000 points : this value was set very high in order to avoid violations, although lower than for budget constraints since diversifying tour types is less important than respecting a budget.

The second subclass of horizontal constraints is about sets of work shifts. It includes constraints on the maximum number of consecutive work days and constraints on the minimum number of consecutive days off. At the first level of constraints on the maximum number of consecutive work days, the value of the right hand side is 6 for the first tour type, and 4 for the second. At the second level, it is 5 for the first, and 3 for the second. Meanwhile for constraints on the minimum number of consecutive days off, the right-hand side value for the first level is respectively 1 and 2 for the first and second tour types, and 2 for all tour types on the second level. Each violation by one day for any constraint mentioned in this paragraph is penalized by 500 points since these constraints have a limited importance in comparison to others.

Finally, the last subclass of horizontal constraints is about single shifts. It includes constraints which forbid shifts to start at specified periods, and which forbid shifts to end at specified periods. There is only one level for both constraints, and they both forbid periods from 2 am to 5 am inclusively. Indeed, it is generally accepted in practice that work shifts should not begin or end in the middle of the night. Each shift violating either constraint is penalized by 500 points since their importance is comparable to that of constraints on sets of work shifts.

3.5.2 Experiments with Two Algorithm Variants

In this section we experiment with two algorithm variants : S1 which combines BM1 and BM2 in phases 2 and 3, and S2 which combines the former with RM. S1 is designed to obtain good solutions quickly and S2 to obtain the best possible solution values. The results obtained with S1 on the 23 constraint sets defined in section 3.5.1 and for each of the 5 demand curves are presented in table 3.IV, while those obtained with S2 appear in table 3.V.

Solution values for S1 are presented in table 3.IV, where solution times are in the left part of the table, solution values in the right part, each column is for one demand curve and each line for one constraint set. Most results were as expected on this table : Budget constraints (CS4, CS5, CS7, CS8, CS20 and CS21) have a strong negative impact on solution values, while constraints which could be qualified as ergonomic such as most horizontal constraints on sets of shifts and on single shifts (CS9 to CS23) generally have a much smaller impact on the solution cost. However, we have encountered a few isolated instances where S1 provided slightly higher solution values on looser problems. In our opinion, it is

Constraint set description			Set number
Service Demand	-		1
	Capacity 1		2
	Capacity 2		3
	Budget 1		4
	Budget 2		5
	Max tours of each type		6
	Max tours of each type	Budget 1	7
		Budget 2	8
	Max consecutive work shifts 1		9
	Max consecutive work shifts 2		10
	Min consecutive days off 1		11
	Min consecutive days off 2		12
	Max consecutive work shifts 1	Min consecutive days off 1	13
		Min consecutive days off 2	14
	Max consecutive work shifts 2	Min consecutive days off 1	15
		Min consecutive days off 2	16
	Limits on shift start times		17
	Limits on shift end times		18
	Limits on shift start times	Limits on shift end times	19
	All constraints 1		20
	All constraints 2		21
	All constraints 1 except capacity & budget		22
	All constraints 2 except capacity & budget		23

Table 3.III – Constraint Sets (CS) Reference Numbers with Their Corresponding Descriptions

Constraint	CPU (s)						Solution values					
set	D1	D2	D3	D4	D5	Mean	D1	D2	D3	D4	D5	Mean
1	330	505	441	343	285	380.80	143.43	790.32	791.76	333.47	74.58	426.71
2	470	612	464	402	251	439.80	676.53	981.07	981.52	656.25	2750.31	1209.14
3	510	468	496	432	247	430.60	3601.42	5701.34	5796.91	2636.75	11113.49	5769.98
4	203	337	329	208	142	243.80	694.51	2600.54	2613.71	1092.63	345.09	1469.30
5	132	251	265	159	142	189.80	4115.94	7541.12	7599.47	4889.48	3339.41	5497.08
6	350	362	394	375	297	355.60	73.80	845.06	808.04	355.11	58.01	428.00
7	185	272	238	227	178	220	865.45	2694.87	2631.83	1251.67	846.71	1658.11
8	166	191	220	168	150	179	4657.81	7592.92	7644.43	5121.79	4650.70	5933.53
9	337	441	472	407	328	397	199.77	832.82	831.42	336.16	83.03	456.64
10	367	567	555	445	264	439.60	247.80	840.51	894.20	307.55	88.10	475.63
11	391	568	647	477	354	487.40	156.01	805.60	805.77	312.76	70.53	430.13
12	368	549	628	442	383	474	238.54	900.01	893.22	467.96	69.12	513.77
13	500	659	618	546	395	543.60	154.15	867.50	879.77	389.73	63.28	470.89
14	490	649	747	523	329	547.60	324.19	1261.79	1337.85	686.22	83.09	738.63
15	539	682	637	499	447	560.80	85.92	1072.71	858.22	278.31	63.45	471.72
16	480	1052	1074	656	451	742.60	443.71	1949.24	2160.34	820.25	70.50	1088.81
17	276	430	394	471	259	366	412.57	5256.92	2187.93	403.72	97.43	1671.71
18	320	487	554	331	303	399	125.68	797.70	842.11	347.2	72.46	437.03
19	270	521	408	398	274	374.20	372.54	5283.46	2229.75	438.06	102.91	1685.34
20	444	404	521	473	495	467.40	2205.19	10128.68	4742.53	1765.83	2597.07	4287.86
21	691	699	625	599	504	623.60	6252.59	16183.90	10531.40	6323.85	12562.73	10370.89
22	486	554	554	611	479	536.80	893.82	5877.72	2176.26	815.11	102.57	1973.10
23	830	915	895	843	777	852	975.72	5950.93	2477.02	1006.66	77.22	2097.51
Mean	397.17	529.35	529.39	436.30	336.26	445.70	1213.79	3772.03	2726.76	1349.41	1712.25	2154.85

TABLE 3.IV – Solution Times (in seconds) and Values for S1 (algorithm variant implementing BM1, BM2 and excluding RM)

not a serious problem in this context since it only happens on a few instances with low solution costs, which means that the difference is quite small in terms of number of violations. Furthermore, such a behavior is to be expected of a flexible and portable solution approach where some local search procedures are executed randomly.

Furthermore, these results indicate that instances with capacity constraint (CS2, CS3, CS20 and CS21) have high solution values with D5. This situation is caused by the structure of our instances : capacity constraints are active when demand is within 5% of its peak value at level 1, and within 15% of its peak value at level 2. Hence, for variable curves (D1 to D4 inclusively), this constraint is active on a small number of periods only. However, it is constantly active on D5 (uniform demand) where all periods are peak periods. Hence, the best solution that can be found with D5 have a much larger number of violations of demand constraints. This makes sense when transposed to a real life context, since having a lower capacity than the maximum demand may be a good decision when the demand peak is steep and short in duration but certainly not the case anywhere when the demand peak is constant throughout the day.

The highest solution value is obtained when all constraints are activated at level 2 (CS21) for D2. This result is most likely explained by the fact that D2 has a very high but narrow peak during the night while CS21 includes limits on start and end times for shifts inclusively between 2 am and 5 am. Hence, the strong early morning variation of the curve becomes more difficult to cover because no shift can either start or finish before 6 am. As this situation is repeated every day for 14 days, it creates a large number of violations when combined with the impact of many more constraints.

As for the solution times reported in table 3.IV, they were in general surprisingly low for staff scheduling instances of such a large size. Indeed, the schedules built as solutions for the instances of the experimental framework generally include around 140 tours (less when budget or capacity constraints are activated). It is interesting to notice that instances with budget constraints were solved very fast, possibly because these constraints strongly reduce the size of the promising exploration areas of the solution space. On the other hand, some instances with constraints on sets of shifts seemed to require much more time to solve, possibly due to the large size of the promising solution space and the smaller improvements obtained with each local search iteration.

Solution values obtained with S2 are presented in table 3.V. These results are similar to those presented on table 3.IV, though without surprise S2 provides on average lower solution values than S1 by almost 10%. However, these better solution values come at a significant cost : solution times are almost 17 times larger with S2, a result that is due to the presence of the RM mechanism. Solution times

Constraint	CPU (s)						Solution values					
set	D1	D2	D3	D4	D5	Mean	D1	D2	D3	D4	D5	Mean
1	3482	6640	7581	3893	8352	5989.60	143.43	787.53	787.96	314.89	61.19	419
2	8167	7160	5780	7467	3899	6494.60	517.59	1020.46	985.66	456.11	2427.48	1081.46
3	14622	14985	10769	10790	3757	10984.60	3330.73	3673.37	3549.2	2369.65	11075.88	4799.77
4	3993	5094	3632	2831	3976	3905.20	596.69	2544.80	2575.07	1041.82	340.74	1419.82
5	2773	4366	5959	2874	2652	3724.80	4065.59	7542.93	7512.65	4868.34	3321.08	5462.12
6	6794	6748	7558	6842	4429	6474.20	66.81	791.04	805.66	303.37	57.48	404.87
7	3885	3138	4154	2941	4926	3808.80	855.25	2611.66	2562.89	1233.88	845.61	1621.86
8	4485	5102	2935	3356	5271	4229.80	4655.45	7574.08	7540.48	5060.29	4638.59	5893.78
9	6815	6988	6163	10963	9106	8007	129.12	797.69	843.05	339.35	59.86	433.81
10	4977	4824	5063	4892	5957	5142.60	172.11	790.64	829.23	351.62	62.23	441.17
11	15544	4891	5466	12069	11457	9885.40	78.17	780.16	783.01	278.68	57.54	395.51
12	15871	4845	5565	4428	5991	7340	97.04	852.27	813.14	344.05	65.26	434.35
13	9282	5252	13815	7522	8281	8830.40	81.21	806.32	795.73	274.05	57.12	402.89
14	5835	19444	10016	11648	3912	10171	253.99	976.29	1016.93	411.48	74.48	546.63
15	5807	5941	5007	8948	9396	7019.80	78.16	828.46	881.45	277.23	54.60	423.98
16	8735	15283	17760	9344	6041	11432.60	309.72	1363.30	1638.72	655.95	68.46	807.23
17	3374	13642	5126	7030	5002	6834.80	387.64	5222.28	2207.56	374.12	76.32	1653.58
18	5300	4376	4141	3284	9552	5330.60	121.68	792.40	838.42	405.47	65.90	444.77
19	3738	7178	3458	3755	4172	4460.20	364.89	5229.75	2201.70	405.50	61.61	1652.69
20	10060	7976	7686	5042	27455	11643.80	2067.00	10095.67	4597.50	1703.04	2475.12	4187.67
21	5591	12883	23073	5999	5563	10621.80	6252.59	16234.98	10200.13	6241.42	11620.12	10109.85
22	4976	11988	12352	6994	7539	8769.80	893.82	5873.46	2176.12	736.94	78.01	1951.67
23	6958	12820	15683	10433	5034	10185.60	933.93	5955.56	2321.05	834.64	85.40	2026.12
Mean	7002.78	8328.87	8206.17	6667.17	7031.30	7447.26	1150.11	3615.00	2541.88	1273.13	1640.44	2044.11

TABLE 3.V – Solution Times (in seconds) and Values for S2 (algorithm variant implementing all three extensions - RM, BM1 and BM2)

for S2 are also presented on table 3.V.

In table 3.VI, each cell contains the following ratio : The result obtained with S2 divided by the result obtained with S1 for the same problem instance. Hence, if the ratio is larger than 100%, S1 performed better, and if the ratio is smaller than 100%, S2 performed better, these situation being highlighted in bold type. This table demonstrates the superior performance of S2 in terms of solution value. Although the average performance improvement is smaller than 10%, S1 provided better solution values on very few instances. Again, the largest variations in terms of percentage were found on instances which resulted in small solution values with both algorithm variants, meaning that the difference between both solutions are in fact small in terms of number of violations. However, solution times presented in table 3.VI are almost 18 times larger on average with S2, which makes this algorithm variant less suitable for real life health care applications. Indeed, health care services are frequently unionized. In contexts where unions are consulted during the scheduling process, many iterations of scheduling scenarios have to

Constraint	CPU (s)						Solution values					
set	D1	D2	D3	D4	D5	Mean	D1	D2	D3	D4	D5	Mean
1	10.55	13.15	17.19	11.35	29.31	16.31	**100.00%**	**99.65%**	**99.52%**	**94.43%**	**82.05%**	**95.13%**
2	17.38	11.70	12.46	18.57	15.53	15.13	**76.51%**	104.02%	100.42%	**69.50%**	**88.26%**	**87.74%**
3	28.67	32.02	21.71	24.98	15.21	24.52	**92.48%**	**64.43%**	**61.23%**	**89.87%**	**99.66%**	**81.53%**
4	19.67	15.12	11.04	13.61	28.00	17.49	**85.92%**	**97.86%**	**98.52%**	**95.35%**	**98.74%**	**95.28%**
5	21.01	17.39	22.49	18.08	18.68	19.53	**98.78%**	100.02%	**98.86%**	**99.57%**	**99.45%**	**99.34%**
6	19.41	18.64	19.18	18.25	14.91	18.08	**90.53%**	**93.61%**	**99.71%**	**85.43%**	**99.09%**	**93.67%**
7	21.00	11.54	17.45	12.96	27.67	18.12	**98.82%**	**96.91%**	**97.38%**	**98.58%**	**99.87%**	**98.31%**
8	27.02	26.71	13.34	19.98	35.14	24.44	**99.95%**	**99.75%**	**98.64%**	**98.80%**	**99.74%**	**99.38%**
9	20.22	15.85	13.06	26.94	27.76	20.77	**64.63%**	**95.78%**	101.4%	100.95%	**72.09%**	**86.97%**
10	13.56	8.51	9.12	10.99	22.56	12.95	**69.46%**	**94.07%**	**92.73%**	114.33%	**70.64%**	**88.25%**
11	39.75	8.61	8.45	25.30	32.36	22.89	**50.11%**	**96.84%**	**97.18%**	**89.10%**	**81.58%**	**82.96%**
12	43.13	8.83	8.86	10.02	15.64	17.30	**40.68%**	**94.70%**	**91.03%**	**73.52%**	**94.42%**	**78.87%**
13	18.56	7.97	22.35	13.78	20.96	16.72	**52.68%**	**92.95%**	**90.45%**	**70.32%**	**90.27%**	**79.33%**
14	11.91	29.96	13.41	22.27	11.89	17.89	**78.35%**	**77.37%**	**76.01%**	**59.96%**	**89.64%**	**76.27%**
15	10.77	8.71	7.86	17.93	21.02	13.26	**90.97%**	**77.23%**	102.71%	**99.61%**	**86.05%**	**91.31%**
16	18.20	14.53	16.54	14.24	13.39	15.38	**69.80%**	**69.94%**	**75.85%**	**79.97%**	**97.11%**	**78.53%**
17	12.22	31.73	13.01	14.93	19.31	18.24	**93.96%**	**99.34%**	100.90%	**92.67%**	**78.33%**	**93.04%**
18	16.56	8.99	7.47	9.92	31.52	14.89	**96.82%**	**99.34%**	**99.56%**	116.78%	**90.95%**	100.69%
19	13.84	13.78	8.48	9.43	15.23	12.15	**97.95%**	**98.98%**	**98.74%**	**92.57%**	**59.87%**	**89.62%**
20	22.66	19.74	14.75	10.66	55.46	24.65	**93.73%**	**99.67%**	**96.94%**	**96.44%**	**95.30%**	**96.42%**
21	8.09	18.43	36.92	10.02	11.04	16.90	**100.00%**	100.32%	**96.85%**	**98.7%**	**92.50%**	**97.67%**
22	10.24	21.64	22.30	11.45	15.74	16.27	**100.00%**	**99.93%**	**99.99%**	**90.41%**	**76.06%**	**93.28%**
23	8.38	14.01	17.52	12.38	6.48	11.75	**95.72%**	100.08%	**93.70%**	**82.91%**	110.59%	**96.60%**
Mean	18.82	16.42	15.43	15.57	21.95	17.64	84.25	93.60	94.27	90.86	89.23	90.44

TABLE 3.VI – Ratios of Solution Times and Values for S2 Compared to S1

be done before a deal is reached. Typically, there is very little time available to produce the schedule at each iteration of the negotiations.

3.5.3 Analysis of the Impact of Service Demand

In order to understand the isolated impact of service demand and constraint sets, we will analyze them independently ; service demand will be discussed in the present section while constraint sets will be in the next. Since S1 provides both fast and very stable solution times as well as good quality solutions , it is the algorithm variant we have chosen to use in order to evaluate the impact of service coverage demand (in this section), and the impact of types of constraints (in section 3.5.4). These results are obtained from the experimental framework used for testing S1 and S2 in the previous sections.

In table 3.VII, the value in each cell is an average per service demand of the results obtained with all constraint sets of the experimental framework. This value is presented as a percentage computed from : the average of the corresponding instances

	D1	D2	D3	D4	D5
Solution Value	32.18%	100.00%	72.29%	35.77%	45.39%
Solution Time	75.02%	99.99%	100.00%	82.42%	63.52%
Solution value without capacity constraints	28.24%	100.00%	75.64%	36.56%	19.27%
Solution time without capacity constraints	69.71%	99.23%	100.00%	80.72%	61.94%

TABLE 3.VII – Analysis of the Impact of Service Demand Pattern on the Solution

in the experimental framework divided by the highest average for all demands. For example, the highest average for solution values of all constraint sets is for D2 (100%) while the lowest is for D1 (32.18%). This can be interpreted as follows : average values obtained for D1 are 32.18% of those found for D2. When looking at the first line, service coverage D1 has a lower cost than D5 (uniform) : this is caused by the instances with capacity constraints. Indeed, as capacity constraints are only active during the peak periods of demand, they are continuously active in the case of a uniform demand while rarely active on more variable curves (as previously explained). Without surprise, when instances with capacity constraints are excluded, the lowest average solution values are obtained on instances with uniform demand. These instances also happen to be the quickest to solve on average.

On the other hand, D2 and D3 are the most costly to cover (highest average solution values) most likely because their curves have the largest variations between bottoms and peaks. More specifically, while D2 and D3 are mirror versions of each other, D2 is more costly to cover (the cost increases by more than 30% on average). As explained in section 3.5.2 regarding why the CS21-D2 instance resulted in the most costly solution, a combination of two elements may explain this : in D2, service coverage peaks are reached during the night, and some instances of the framework have limits on shift starting and finishing times during the night (between 2 am and 5 am inclusively). When these two elements are combined in the same instance, they strongly increase the costs of potential solutions.

Meanwhile, D1 and D4 are not very costly to cover on average, though they are more costly than the uniform demand without capacity instances. This was expected for D1 as it has relatively small variations between bottoms and peaks of demand throughout the day, but it is somewhat more surprising for D4 which has the highest service demand peak (though not the largest difference between bottoms and peaks). In our opinion, this is explained by the very low variations of the D4 curve outside of the peak area.

Globally, if we isolate the impact of constraints, it is clear that service demand curves with smaller intra-day variations are less costly to cover. However, caution is necessary as capacity constraints for instance can amplify the coverage cost of the more flat demand curves in some specific circumstances.

3.5.4 Analysis of Impact of Types of Constraints

Table 3.VIII presents the results of the analysis by constraint sets where all values are presented as the following ratio : The average for instances sharing one constraint set from the experimental framework divided by the value for the constraint set with the highest average. For example, the value corresponding to constraint set 1 is the lowest while the value corresponding to constraint set 21 is the highest. Furthermore, the value of the former can be interpreted as being 4.11% of the value of the latter. In the column labeled solution value, all values are computed from averages of five instances each with a different service demand curve presented previously. The "solution times" column presents in each cell a value calculated in the same fashion as those in the solution value column, but for solution times. Finally, the last two columns present the same information respectively as the first and second one, but excluding instances with a uniform demand (D5) since they overweight the impact of the capacity constraint sets.

In table 3.VIII, better solution values are always found on instances with looser constraint sets, which in our opinion demonstrates the stability of the algorithm on a large variety of instances. Unsurprisingly, the lowest solution values are obtained with constraint set 1 (target demand only), while the highest are obtained with constraint set 21 (all constraints activated at level 2). In fact, the highest solution values are generally obtained when budget or capacity constraints are activated at level 2. When analyzing the differences in results obtained caused by capacity constraints when D5 is included, it is straightforward to understand that demand curves and constraints do interact with each other, and that specific combinations of these may have a strong impact on possible solutions. Also, it is interesting to notice how small the impact of some constraints on sets of shifts or individual shifts can be, while they may provide strong benefits for employee satisfaction (in our opinion, further studies on the relationship between solution costs and employee satisfaction should be made). Furthermore, while budget constraints have a strong impact on solution values as expected, instances with only budget constraints have the quickest solution times. On the other hand, instances with all constraints at level 2 (the tighter ones) have the longest solution times. This was expected since the solution space is more difficult to explore thoroughly in such contexts.

3.6 Conclusion

We have proposed a flexible and portable algorithm approach to solve staff scheduling problems in health care which could be extended to many other application areas with very limited adaptations. To the best of our knowledge, this is the first algorithm structure where both portability and flexibility are thoroughly

Constraint set	Solution value	Solution time	Solution value without D5	Solution time without D5
1	4.11%	44.69%	5.24%	46.48%
2	11.66%	51.62%	8.39%	55.93%
3	55.64%	50.54%	45.14%	54.72%
4	14.17%	28.62%	17.82%	30.92%
5	53.00%	22.28%	61.45%	23.17%
6	4.13%	41.74%	5.30%	42.52%
7	15.99%	25.82%	18.94%	26.47%
8	57.21%	21.01%	63.67%	21.39%
9	4.40%	46.60%	5.60%	47.57%
10	4.59%	51.60%	5.83%	55.53%
11	4.15%	57.21%	5.29%	59.80%
12	4.95%	55.63%	6.36%	57.05%
13	4.54%	63.80%	5.83%	66.70%
14	7.12%	64.27%	9.19%	69.16%
15	4.55%	65.82%	5.84%	67.67%
16	10.50%	87.16%	13.68%	93.65%
17	16.12%	42.96%	21.03%	45.10%
18	4.21%	46.83%	5.38%	48.58%
19	16.25%	43.92%	21.18%	45.85%
20	41.35%	54.86%	47.95%	52.89%
21	100.00%	73.19%	100.00%	75.05%
22	19.03%	63.00%	24.85%	63.31%
23	20.22%	100.00%	26.49%	100.00%

Table 3.VIII – Analysis of the Impact of Constraints on the Solution

considered. Indeed, the approach is adaptable to a large number of real-life staff scheduling instances without any other modifications needed than activating a few instance-specific constraints.

It is clear that a sequential phase structure has some weaknesses and can provide unstable results in some situations. However, when successfully combined with backtracking mechanisms, the stability of the algorithm can be highly increased in terms of solution value while maintaining the capacity to solve widely different problems. Furthermore, mechanisms such as a partial restart (reshuffle) can increase the quality of the solution, but with the consequence of requiring solution times that may not be suitable for some real life application instances. Indeed, quickness and stability of solution times are important for many real life applications in health care, for example in emergency departments where a schedule sometimes needs to be changed with very little delay of notice, or when schedules have to be produced in a context involving multiple shortly spaced rounds of negotiations between managers and unions.

We have also developed an experimental framework to evaluate staff scheduling flexibility. To the best of our knowledge, it is the first such framework built according to a classification by structure of constraints. In our opinion, this framework is comprehensive and includes a large number of constraints and demand curves, representing a reasonable number of realistic combinations. The results obtained with this framework now allows us to have more information on the impact of typical staff scheduling constraints, on some typical service demands, and on a good number of combinations of both. More importantly, we hope that our work will also set a basis for more thorough benchmarking between different flexible and portable staff scheduling solution approaches in the future.

Future research should include tests on real life applications and be focused on improving the algorithm to enlarge its implementation potential, among other things by implementing the notion of optimization of employee assignment, individual preferences, seniority, discrete flexible break allocation, competences, task assignments and integration with routing optimization for problems such as scheduling with home care nurses.

Chapitre 4

A Heuristic Inspired from Column Generation for Flexible and Portable Staff Scheduling in Health Care

4.1 Introduction

Nurse scheduling problems have been extensively researched in the literature, unlike physician and ambulance officers scheduling problems which are less frequently documented. Other applications in health care are even more scarce. While many different approaches have been proposed to solve these various problems, few flexible and portable solution approaches have been researched. As defined by [66], flexibility is the capacity of a solution approach to tackle a large variety of instances, applications and problem structures, and portability is the capacity of a solution approach to be easily configured to solve a new instance, application or problem structure, quickly and with a limited amount of work. To the best of our knowledge, the only solution approach built to tackle a wide variety of applications in health care by implementing these flexibility and portability concepts was SOFA proposed by [66]. In this paper, we propose an alternative flexible and portable approach to solving staff scheduling problems applied to health care. Hence, the objective and contributions of this paper are twofold :

1. Propose a very efficient flexible and portable algorithm structure (CHAIR - Column generation Heuristic Approach for Indefinite Rostering) able to tackle a variety of health care staff scheduling applications, including all those researched much less frequently than nurse scheduling ;
2. Benchmark CHAIR with the SOFA approach proposed by Crowe et al. (2011) [66] on the framework proposed in that same paper to test the flexibility and portability of staff scheduling approaches.

Crowe et Soriano (2011) [66] have demonstrated that SOFA, a sequential decomposition heuristic in three phases, was able to obtain interesting results on a large set of very different instances. We wish to investigate in this paper how CHAIR, a fundamentally different approach based on a master problem/subproblem structure, will perform in comparison to SOFA. This is in order to gain useful knowledge on the strengths and weaknesses of both approaches regarding various staff scheduling characteristics.

This paper is structured as follows : The next section presents the literature review, followed by descriptions of the general approach in section 4.3, the subproblem in section 4.5, the master problem in section 4.4, and the extensions of the solution approach in section 4.6. Our results are then discussed in section 4.7, and finally section 4.8 concludes this paper.

4.2 Literature Review

For a detailed survey of the staff scheduling applied to health care and other application areas, we refer the reader to chapter 2 of this thesis. From this survey, it is clear that flexible and portable solution approaches have not been discussed extensively in the literature. In this context, chapter 3 proposes a flexible and portable solution approach structured as a sequential heuristic called SOFA where phase 1 builds days off scheduling patterns, phase 2 produces a shift scheduling solution where all shifts of a same employee start at the same time, and phase 3 produces a more detailed shift scheduling solution where any shift can start at any time. A few solution improving mechanisms were also implemented, including backtracking mechanisms and a reshuffling procedure. Although it is a first step of research on flexible and portable solution approaches to staff scheduling, it has produced very encouraging results on a large variety of instances. Furthermore, an important contribution of that study was the introduction of a flexibility experimental framework for staff scheduling in order to provide a benchmarking basis for future research. The experimental tests executed in the present paper are based on the framework proposed in chapter 3.

While limited in terms of flexibility and portability in the current literature, other types of decomposition approaches such as column generation have proven to be able to provide good solution values on staff scheduling problems. However, column generation like all linear programming approaches is much more time consuming than heuristics on large problems. To deal with this drawback, Ikegami et al. [102] have proposed a heuristic approach partly inspired from the decomposition structure of column generation. To the best of our knowledge, this is the only published fully heuristic approach inspired from mathematical decomposition applied to staff scheduling. The proposed decomposition is called a subproblem-centric approach, and is applied to a nurse scheduling problem. In this approach, the scheduling problem is solved by iteratively minimizing a single nurse's tour while fixing the rest of the solution. The algorithm is tested on real life instances from Japanese hospitals and provided good results.

As previously mentioned in chapter 3, when dealing with health care problems both employees (or unions) and managers are generally involved in some kind of negotiation process regarding the production of the new work schedule. These negotiations make the process in practice iterative by nature, with a large number of scheduling scenarios iterations each produced with little time available. In this context where quick solution times will be required on large instances, the obvious approach is to use some type of decomposition technique : this is what was done with SOFA, where the problem was decomposed hierarchically into a days-off, a shift scheduling and an extended shift scheduling subproblem. In order to deal with the typical limitations of hierarchical decomposition approaches, two backtracking mechanisms were implemented in parallel with a reshuffling mechanism. While the solutions produced by this approach were visually appealing and appeared to be of good quality on all instances, we found no comparative basis to use in the literature. Furthermore, while the sequential decomposition of SOFA was interesting and proved to perform well, the master problem/subproblem approach proposed here with CHAIR was clearly worth testing considering the good track record of column generation approaches applied to staff scheduling. Hence, we believe there is an opportunity to gain valuable knowledge about the relative strengths and weaknesses of SOFA and CHAIR, two solution approaches structured fundamentally differently, by comparing their results on the same flexibility experimental framework.

In a context where both flexibility and portability are required, the structure of the decomposition is the basis for providing good overall performance. The decomposition approach proposed used for SOFA could be qualified as periodical : days of work are chosen first, followed by start times for work shifts. In this paper, we propose a somewhat perpendicular approach structured on lines (or tours) : a fully heuristic decomposition approach inspired from column generation. With CHAIR, the problem is decomposed into subproblems and a master problem : the SPs produce tours, and the MP selects which tours to include in the solution.

4.3 General Approach

We refer to the solution approach proposed here as CHAIR for Column generation Heuristic Approach with an Indefinite Range. It is a fully heuristic procedure which decomposes the original problem to be solved into one SP and one MP. To illustrate the principles underlying the proposed approach, we will first present a detailed problem description, followed by two general flexible staff scheduling models and finally by a description of the global algorithm. The SP and MP will respectively be discussed in sections 4.5 and 4.4.

4.3.1 Problem Description

Let us first recall a list of definitions of terms used in staff scheduling, as was introduced in Crowe et al. (2011) [66] : a *period* is a discretized unit of time, generally representing an hour in this paper ; a *planning horizon* is the continuous set of periods for which a schedule needs to be provided, generally representing two weeks in this paper ; a *work shift* is a continuous set of periods including a lunch break, during which a worker is available to provide service at any time except the lunch break ; a *work shift type* defines a work shift by duration (total number of periods from its beginning up until its termination), by duration of its lunch break and by lunch break time window ; a *lunch break* is a number of continuous periods inside a work shift during which the worker is not available to provide service ; a *tour* is a set of work shifts spread throughout the planning horizon which represents the schedule of a single employee (also called an individual schedule) ; a *tour type* defines a tour by work shift type and total number of work shifts on the planning horizon ; a *demand* represents the service requirements for periods of the planning horizon ; a *coverage* represents how much service load is provided during the periods of the planning horizon : *Undercoverage* is present when coverage is lower than demand for a given period, while *overcoverage* is present when coverage is higher than demand for a given period.

Since dealing with the flexible staff scheduling problem involves solving a wide variety of instance structures with a single solution approach, it is important to start with a model having very few limitations in terms of implementable schedule structures. Furthermore, the approach should allow the user to calibrate all the different tour types wanted, the duration of the planning horizon, the duration of periods, and the penalty weights and right hand side values of the required constraints. In order to potentially consider all staff scheduling constraints, we have proposed in Crowe et al. (2011) [66] the following classification by types of constraints :

- Vertical constraints :
  - Constraints on a single period :
    - Service demand coverage ;
    - Capacity (f.ex. : number of available vehicles or workspaces, budget per period, . . .).
- Horizontal constraints :
  - Constraints on sets of lines :

  - Management constraints (f.ex. : maximum number of tours of each type, budget by number of employees, . . .).
- Constraints on sets of shifts of a single tour :
  - Collective agreements, ergonomics, working conditions (f.ex. : maximum number of consecutive work days, minimum number of consecutive days off, . . .).
- Constraints on a single shift :
  - Collective agreements, ergonomics, working conditions (f.ex. limits on possible start-times for work shifts, . . .).

In this classification, vertical sets (or columns) represent periods while horizontal sets (or lines) represent tours. This representation is a perpendicular transposition of the usual mathematical design where columns are variables (tours) and lines are periods. This transposition is motivated by the following reason : managers typically work with graphic designs of schedules where columns are periods and lines are tours. Hence, in our opinion this representation is more suitable to a context inspired from real life applications. In a context of flexibility, we believe that the most interesting aspect of this classification is that it is built following a basic, natural and universal problem structure for staff scheduling : All characteristics of a schedule depend on periods of the planning horizon (vertical constraints), on the number of tours which can be scheduled (horizontal constraints on sets of lines), on the dynamics of days on and days off within a tour (horizontal constraints on sets of work shifts) and on the possible structure of individual work shifts (horizontal constraints on a single shift). The solution approach we propose in this paper is structured according to a decomposition by types of constraints based on this classification.

4.3.2 Flexible Staff Scheduling Models

In its basic form, a staff scheduling problem can be represented by a simple set covering model. However, the problem is more difficult to model when flexibility is required, since the model must then be structured in order to implement any type of constraints on tours and periods. A limited flexible staff scheduling model is presented here. It has some typical set covering constraints but includes an objective function structured differently.

In this model, $X_{ijk} = 1$ if tour i starts a work shift of tour type k at period j, 0 otherwise ; $Y_{ik} = 1$ if tour i has been assigned a type k, 0 otherwise ; n_k, the number of work shifts in a tour of type k ; d_j, service demand at period j ; p, the number of periods in one day ; q_j, the cost of one unit of undercoverage at period j ; r_j, the cost of one unit of overcoverage at period j ; o_j, the quantity of overcoverage at period j ; u_j, the quantity of undercoverage at period j ; s_{kl}, the service covering value of a shift of tour type k at its period l ; v, the exponential factor of the cost function for undercoverage variable u ; w, the exponential factor of the cost function for overcoverage variable o ; Z_{ij}, the coverage provided by tour i at period j ; L, the set of possible periods for the duration of a work shift.

In this flexible model limited to a set of basic constraints and demand coverage constraints, the objective function minimizes the sum of undercoverage and overcoverage with both penalty terms computed as exponential functions in order to balance coverage gaps throughout the planning horizon. Indeed, a linear cost function will consider equal the two following solution examples : the first with a perfect coverage on period 1 and an undercoverage of 10 shifts on period two, and the second with an undercoverage of 5 shifts on both periods. Meanwhile, an exponential function will result in the second solution being evaluated the best as coverage quality is more balanced. Regarding other constraints of the model, constraint set 4.2 forces each tour to have a number of shifts corresponding to

$$\min \sum_{j \in J} q_j (u_j)^v + \sum_{j \in J} r_j (o_j)^w \tag{4.1}$$

Subject to :

$$\sum_{j \in J} X_{ijk} = n_k Y_{ik} \qquad \forall\ i \in I,\ k \in K \tag{4.2}$$

$$Z_{i(j+l)} = s_{kl} X_{ijk} \qquad \forall\ i \in I, \forall\ j \in J, \forall\ k \in K, \forall\ l \in L \tag{4.3}$$

$$\sum_{j=j}^{j+p} \sum_{k \in K} X_{ijk} \leq 1 \qquad \forall\ j \in J \tag{4.4}$$

$$\sum_{k \in K} Y_{ik} \leq 1 \qquad \forall\ i \in I \tag{4.5}$$

$$\sum_{k \in K} X_{ijk} \leq 1 \qquad \forall\ i \in I,\ j \in J \tag{4.6}$$

$$u_j - o_j + \sum_{i \in I} Z_{ij} = d_j \qquad \forall\ j \in J \tag{4.7}$$

$$X_{ijk}, Y_{ik}, Z_{ij} \in \{0, 1\} \qquad \forall\ i \in I,\ j \in J,\ k \in K \tag{4.8}$$

its assigned tour type, constraint set 4.3 links variables X and Z, constraint set 4.4 ensures that no work shifts overlap each other, constraint set 4.5 ensures that each tour is related to no more than one tour type, constraint set 4.6 ensures that no more than one work shift starts at a given period of any given tour, and finally constraint set 4.7 ensures that the total coverage for each period equals demand when overcoverage and undercoverage are respectively subtracted and added.

This model is a faithful representation of the problem we are working on, although limited to a target demand coverage constraint for the sake of simplicity. While service coverage is the one constraint always present in every model, any other vertical or horizontal constraints could simply be added when needed. In this model, lunch breaks are considered and modeled as spread out on a predetermined time window because we assume that the real lunch break assignment decisions will be taken sometime during the work day by the manager. Indeed, the random nature of demand for health care services makes long term planing of lunch break assignments unrealistic in most practical situations. However, should other types of applications be considered, the assignment of lunch breaks to a fixed period within each work shift could be implemented in the model by simply adding another set of variables.

In the literature, most models use a linear minimum service coverage constraint. While linear cost functions for demand coverage typically perform well when the problem structure is such that demand can be covered tightly for all periods of the horizon, it is not efficient when demand can not be well covered on at least a small number of periods. This is due to the fact that the typical linear model does not distinguish between a solution where there is a large coverage gap on a few periods and one where the same total coverage gap is spread out on a larger number of periods. Furthermore, most models in the literature only penalize undercoverage.

In instances where the number of available units of service periods is greater than the demand, this can lead to a situation where the excess of coverage will be concentrated in a few periods instead of being spread out and balanced throughout the whole planning horizon. Hence, in order to ensure an adequate distribution of service coverage over all periods for all problems, we have implemented a demand coverage cost function with two dimensions :

- A coverage target where both undercoverage and overcoverage are penalized ;
- An exponential function in order to increase the impact of larger gaps on the solution value and force total gaps to be spread out over all periods.

A more generalized extended version of the previous model including all classes of constraints proposed by Crowe et al. (2011) [66] is modeled here. In this model : $V(X,Y,Z)$ is a function returning the total costs related to violations of vertical constraints ; $H_1(X,Y,Z)$ is a function returning the total costs related to violations of horizontal constraints on sets of tours ; $H_2(X,Y,Z)$ is a function returning the total costs related to violations of horizontal constraints on sets of work shifts ; $H_3(X,Y,Z)$ is a function returning the total costs related to violations of horizontal constraints on single work shifts ; C_V is the set of vertical constraints ; C_{H_1} is the set of horizontal constraints on sets of lines ; C_{H_2} is the set of horizontal constraints on sets of work shifts ; C_{H_3} is the set of horizontal constraints on single work shifts.

This more comprehensive flexible model is based on the constraint classification proposed by Crowe et al. [66] discussed in section 4.3.1. Here, the service coverage function of the first limited model is now included in the term $V(X,Y,Z)$; other constraints such as service capacity by period would also be included in this same function. $H_1(X,Y,Z)$ would include budget or management constraints limiting the number of employees working on a given tour type,

$$\min V(X,Y,Z) + H_1(X,Y,Z) + H_2(X,Y,Z) + H_3(X,Y,Z) \tag{4.9}$$

Subject to :

$$\sum_j X_{ijk} = n_k Y_{ik} \qquad \forall\ i \in I,\ k \in K \tag{4.10}$$

$$Z_{i(j+l)} = s_{kl} X_{ijk} \qquad \forall\ i \in I, \forall\ j \in J, \forall\ k \in K, \forall\ l \in L \tag{4.11}$$

$$\sum_{j=j}^{j+p} \sum_k X_{ijk} \leq 1 \qquad \forall\ j \in J \tag{4.12}$$

$$\sum_k Y_{ik} \leq 1 \qquad \forall\ i \in I \tag{4.13}$$

$$\sum_k X_{ijk} \leq 1 \qquad \forall\ i \in I,\ j \in J \tag{4.14}$$

$$C_V \tag{4.15}$$

$$C_{H_1} \tag{4.16}$$

$$C_{H_2} \tag{4.17}$$

$$C_{H_3} \tag{4.18}$$

$$X_{ijk}, Y_{ik}, Z_{ij} \in \{0,1\} \qquad \forall\ i \in I,\ j \in J,\ k \in K \tag{4.19}$$

$H_2(X, Y, Z)$ constraints such as the maximum number of consecutive work days, and $H_3(X, Y, Z)$ constraints such as the allowed shift start times.

4.3.3 Global Algorithm

While an important challenge in Crowe et al. (2011) [66] was to deal with how to make adjustments on suboptimal decisions made in prior hierarchical phases of the approach, an important challenge with CHAIR regards how to guide the construction of a tour in the SP when the objective of that tour is to improve the solution of the MP. In a mathematical approach such as column generation, this is done by using dual variables to compute reduced costs. However, there is no straightforward way of obtaining dual variables in a fully heuristic approach, and calculating them would require a substantial increase in solution times when starting from primal informations of a heuristic. We dealt with this problem in a fashion inspired from the approach proposed by Ikegami et al. (2003) [102] : Using only primal information, sets of constraints from the initial problem are either sent to the SP or MP, or implemented in both problems in order to serve as links between them, effectively ensuring that the SP produces tours which are interesting to include in the current solution of the MP. Based on the classification proposed by Crowe et al. [66], in the context of our research, these links should include all vertical constraints since they include information such as demand coverage and period capacity which dictate where shifts should be added or subtracted in the schedule in order to improve solution quality. Meanwhile, horizontal constraints on sets of lines should be implemented only in the MP, as it is the only problem where more than one individual schedule is involved at once. Finally, constraints on sets of work shifts and on individual work shifts will be considered only in the SP as these are the only problems where decisions are taken regarding the location of shifts within individual schedules.

The core of the algorithm combines and manages interactions between the two solution approaches respectively for the SP and MP. The MP algorithm will be detailed in section 4.4 and the SP algorithm in section 4.5. The structure of the global algorithm is presented on figure 4.1.

Basically, the global algorithm starts by activating the solution approach of the SP to obtain a given number of tours equal to the number of tour types defined by the user. This means that at each global iteration, the SP algorithm is executed k times in order to produce k tours if the user has defined k tour types. During each execution, the SP solution approach is constrained to produce one tour of a fixed tour type. At the next SP execution, it will produce one tour of the following tour type, and so on until one tour of each tour type has been produced. Then, these tours are all inserted in the tour pool used by the solution approach of the master problem. The tour pool includes all tours inserted in the current solution as well as all other tours that have previously been produced by the SP algorithm. The MP is then solved in order to improve the current schedule, and this process is repeated until solving the SP does not yield tours which can be used to improve the solution of the MP anymore. When this happens, the best known solution becomes the final solution and the solution process is terminated.

```
Initialize the tour pool
Do until the MP solution can't be improved
    For each tour type :
        Update the tour pool $\leftarrow$ Solve the subproblem
    Solve the master problem
Repeat
```

FIGURE 4.1 – Algorithm of the Core of the Solution Approach

4.4 Master Problem

The MP is solved by inserting tours provided by the SP in the schedule. In the MP, decisions taken by the solution approach are limited to the selection of which tours to include or exclude from the schedule. The solution approach for the MP is designed to be provided by the SP with one tour of each type for each global iteration. We have structured it in this fashion in order to increase the diversity of available tours. Indeed, the approach proposed by Crowe et al. [66] (2011) typically builds initial solutions in phase 1 by adding a much larger number of tours of the type providing more service hours than of any other types. As we have noticed that this behavior produces initial solutions lacking tour type diversity, we have structured CHAIR in order to avoid this limitation.

In the approach we propose, the set of tours provided by the SP are added to a list which includes all tours previously provided : We will refer to this list as the *tour pool*. In order to select which tours to select for the current solution, two local search structures are executed in loops until no improvements to the current solution can be found. At this point, the algorithm returns to the SP in order to generate tours which could potentially improve the solution at the next MP iteration. The local search structures of the MP will be discussed in section 4.4.2 once the MP model will have been presented. A description of the solution value calculation will conclude this section on the MP.

4.4.1 Master Problem Model

The size of this model is dramatically reduced compared to the size of the initial model of the full problem presented in section 4.3.2. Indeed, as no decisions are taken regarding the internal structure of tours, both X and Z variables have been removed from the MP model. Furthermore, as no decisions regarding tour types

$$\min V(Y) + H_1(Y) + H_2(Y) + H_3(Y) \tag{4.20}$$

Subject to :

$$Y_i \leq 1 \qquad \forall\, i \in I \tag{4.21}$$

$$C_V \tag{4.22}$$

$$C_{H_1} \tag{4.23}$$

$$Y_i \in \{0, 1\} \qquad \forall\, i \in I \tag{4.24}$$

must be made, all k indexes have also been removed. While vertical constraints among others require information about periods, index j may be removed and considered implicitly in the cost functions it impacts. Finally, while cost informations on H_2 and H_3 are required to compute correctly the overall cost of a solution, all constraints linked to those functions may be removed as no decisions regarding these will impact the structure of individual tours.

4.4.2 Heuristic Algorithm for Solving the Master Problem

In order to build the schedule, two local search structures have been implemented : the first one either includes or excludes a tour from the solution, and the second one swaps a tour of the solution with a tour excluded from it. All tours inserted or excluded during these local search procedures are taken from the tour pool. Furthermore, as tour selection variables are binary, a given tour may only be inserted once in the solution (there may however be identical tours defined by other variables).

For each iteration of the first local search, all inclusions and exclusions of tours in the pool are sequentially tested for implementation. After all possible moves have been tried, the best one is implemented only if it improves the solution. Meanwhile, for each iteration of the second local search, all swaps between included and excluded tours from the pool are tested. Here again, after all possible swaps have been tried, the best one is implemented only if it improves the solution in this case also. The algorithm of the MP is described on figure 4.2.

```
Do
    For each local search structure :
         Do :
               Execute current local search
         Repeat until no improvements are found
    Next local search structure
Repeat until no more improvements have been found with any local search
```

FIGURE 4.2 – Algorithm to Solve the Master Problem

4.4.3 Master Problem Solution Value Calculation

The classes of constraints for which penalties must be considered in the solution value of the MP are the following :

- Vertical constraints : all vertical constraints are considered
- Horizontal constraints :
  - Constraints on sets of lines : all constraints are considered ;
  - Constraints on sets of shifts : the cost is included, but no calculations are needed as each tour's individual cost has already been computed in the SP and memorized when it was transferred to the tour pool ;
  - Constraints on single shifts : Same case as constraints on sets of shifts.

While the cost from violations of all constraints must be considered in the MP in order to evaluate the true solution value, the calculation effort can be limited to violations of vertical constraints and of constraints on sets of lines. While the cost from violations of all other horizontal constraints must also be considered, it can simply be memorized each time a tour is produced by the SP as it will never need to be computed again. This effectively excludes calculations related to any single shift and sets of shifts constraint from the MP, substantially reducing its size and computation requirements. Moreover, the fact that the local search is limited to binary variables of tour selection is a key to making this solution approach structure efficient.

4.5 Subproblem

In a column generation approach, the SP provides variables with a negative reduced cost to the MP. More specifically, for a staff scheduling problem these variables typically define tours. Our heuristic approach is structured in the same fashion : the SP will also provide tours to the MP. However, unlike mathematical programming approaches, primal heuristics do not provide straightforward ways of obtaining reduced cost values.

As presented in section 4.3.1, Crowe et al. [66] have proposed a classification for types of constraints in which the two main categories were defined as vertical and horizontal. With our proposed solution approach architecture, horizontal constraints on sets of work shifts and on individual work shifts should be considered within the SP, while horizontal constraints on sets of lines in the MP. Vertical constraints should be considered in both in order to serve as links. To create these links, the current solution's information on the violations of vertical constraints will be transferred to the SP. With this information, the SP will then produce tours which increase the quality of the solution in terms of service coverage and other vertical constraints while limiting violations of the considered horizontal constraints. As most constraints are modeled as soft in order to avoid infeasibility and enforce flexibility, solving the SP will result in a trade-off between the horizontal constraint cost of the tour and the cost of adding this tour to the solution in terms of vertical constraints satisfaction. Our proposed SP solution approach is detailed in the subsequent paragraphs : First, the SP model is presented, followed by a description of its solution approach, and finally by a detailed description of the solution value computation method.

4.5.1 Subproblem Model

The definitions of variables and parameters in this model are the same as those used in the previous models. In our flexible approach, tour characteristics such as number of work days, duration of shifts, duration of breaks and so on are user defined. A user may define as many types of tours as he wishes, however a SP solution always provides only one tour of a single user-defined tour type. As explained in section 4.3.3, the SP is sequentially solved once for each tour type defined in the current instance. This allows the algorithm to produce one tour of each type during each global iteration. As solving the SP always produces only one tour of a predetermined type, the set of variables Y from the global flexible models have been completely removed. Furthermore, all constraints and related penalties regarding sets of tours (H_1) have been removed from this model. Indeed, considerations regarding the total number of tours are irrelevant as the SP always produces one tour of a predetermined type. Finally, both i and k indexes have also been removed because decisions taken for an instance of the SP do not concern more than one tour type. In the end, all these elements contribute to substantially reduce the size of the SP compared to the initial problem. Meanwhile, crucial information related to the quality of a tour both in terms of horizontal and vertical constraints remain.

$$\min V(X,Z) + H_2(X,Z) + H_3(X,Z) \tag{4.25}$$

Subject to :

$$\sum_j X_j = n \tag{4.26}$$

$$Z_{(j+l)} = s_l X_j \qquad \forall\, j \in J, \forall\, l \in L \tag{4.27}$$

$$\sum_{j=j}^{j+p} X_j \leq 1 \qquad \forall\, j \in J \tag{4.28}$$

$$C_V \tag{4.29}$$

$$C_{H_2} \tag{4.30}$$

$$C_{H_3} \tag{4.31}$$

$$X_j, Z_j \in \{0,1\} \qquad \forall\, i \in I,\ k \in K \tag{4.32}$$

4.5.2 Subproblem Algorithm

When solving an instance of the SP, the structure of the tour is built by the following greedy heuristic : sequentially, each shift is located to start at the period of the horizon where it will either improve the most or deteriorate the least the solution value. This iterative process is executed a number of times equal to the number of work shifts to be inserted in the tour. This number of work shifts is predetermined according to the tour type of the current instance of the SP. During this process, a shift can be assigned to start at any period as long as it respects the only hard constraint which prohibits shifts from overlapping one another ; all other constraints are soft and considered via penalties in the objective function of the SP. A description of the algorithm is presented on figure 4.3.

```
For each shift to assign :
    Add a shift starting at the period $j$ which minimizes the cost function
Next shift to assign
Return tour
```

FIGURE 4.3 – Algorithm to Solve the Subproblem

While such an algorithm structure seems simple when compared to more sophisticated metaheuristics found abundantly in the literature, we believe it is well suited to solve our SP. Indeed, it is clear that such a simple procedure should have a significant positive impact on solution times. As a matter of fact, a limited execution time is paramount considering that thousands of instances of the SP can be solved on typical problems. Furthermore, much of our work was focused on efficiently decomposing the global problem into simpler and easier to solve problem structures. Finally, it is important to keep in mind that we are working in a context of real life instances where constraint satisfaction has largely more meaning than pure mathematical optimality. Hence, our solution approach to the SP shares the same objective as our global approach, which is to produce good solutions quickly. Indeed, our preliminary tests have indicated that this is exactly what our greedy heuristic does for the SP.

4.5.3 Subproblem Solution Value Calculation

Using the classification presented in our previous paper, the costs to consider in the SP concern the following classes : all horizontal constraints limited to the current tour of the subproblem (constraints on sets of shifts and on single shifts are included, but constraints on sets of tours are excluded), and all vertical constraints. In a mathematical approach such as column generation, dual variables can be used to quickly obtain interesting variables with a negative reduced cost. As this approach is less suitable to our context, we will solve the subproblem by strictly using the following primal information :

- Vertical constraints : all vertical constraints are considered, but our calculations can be limited to variations of the current difference between left and right hand side values of constraints in the MP ;
- Horizontal constraints :
  - Constraints on sets of shifts : all included, only for the current SP tour ;
  - Constraints on single shifts : all included, only for the current SP tour.

While the impact of most types of constraints must be evaluated at the SP level, the solution time is strongly reduced by the fact that decision variables are for a single tour. For example, while full service coverage information must be considered at each SP decision, the calculation effort can be limited to modifications on the current SP tour. Indeed, as the schedule obtained by solving the MP is not modified when solving a SP instance, the impact of the current MP schedule on demand coverage can be calculated once before solving the SP, and memorized afterwards until the current SP instance has been solved.

4.6 Extensions

While the structure presented above is a solid flexible base, heuristics typically have limitations in terms of performance consistency between instances. Hence, in order to make our flexible and portable approach more robust, we have developed the following extensions : a post optimization procedure for the SP, an extended version of the SP, a shuffling mechanism, and a limited tour pool.

4.6.1 Subproblem Post Optimization

Firstly, we have extended our SP solution approach by implementing a post optimization procedure in order to improve the solutions found by the greedy algorithm. Two local search neighborhood structures are implemented in this extension : (1) the swap of two days, and (2) the modification of the start time of a work shift.

In the first local search structure, a neighborhood is defined as the set of feasible solutions obtained by swapping two non identical days of the current tour. Clearly, resulting solutions with shifts overlapping each other are prohibited since they are not feasible. An iteration of this local search analyses all possible swaps between two non identical days, and implements the best one if it improves the current solution value.

In the second local search structure, a neighborhood is defined as the set of feasible solutions obtained by moving the start time of a shift anywhere within the 24 hour period of its current calendar day (from midnight inclusively to midnight exclusively). Again, resulting solutions with shifts overlapping each other are prohibited to avoid infeasibility. An iteration of this local search analyses all possible movements of start times for all shifts, and implements the best one if it improves the current solution value.

The post optimization process is initiated with the first local search neighborhood. If no improvements to the current solution are found during one full iteration, the local search is then applied to the second neighborhood. When a full iteration with both neighborhoods has been executed without providing any improvement to the current solution, the post optimization process is terminated, and the resulting tour will be transferred to the MP.

4.6.2 Extended Subproblem

In the SP structure presented in section 4.5, the solution value calculation method assumes that new tours produced by solving the SP are added to the MP schedule. This is reasonable when the MP is in the constructive process of adding new tours to cover large remaining undercoverage gaps. However, once a solution with limited undercoverage has been found, it is likely that new tours provided by the SP will replace currently inserted tours instead. In this situation, the previously proposed subproblem solution value calculation function does not seem adequate since it does not consider the possibility of a tour swap. In fact, simulating the withdrawal of a tour from the current schedule may be a more suitable method for calculating the solution value in the SP. Furthermore, it is more similar to using dual information implicitly. Hence, we have extended our SP algorithm by implementing a mechanism which sequentially solves the SP for each possible tour withdrawal from the current MP schedule. While very time consuming, we expect this extended mechanism to substantially improve solution values.

When compared with the initial subproblem model presented in section 4.5, the main difference regards the insertion of horizontal constraints on sets of tours and the related cost function informations (H_1). Indeed, as the extended SP considers the possibility of withdrawing tours from the SP, considering C_{H_1} becomes necessary as not all withdrawals will have the same impacts on this set of constraints (for instance, constraints impacting tour types will be affected differently by withdrawals of tours of specific types). However, apart from the insertion of constraint set C_{H_1}, this model is identical to the one presented in section 4.5.

While this model should compensate for a limitation of the initial SP structure, it creates a new one : A large increase in solution times. In order to manage this situation, we have structured this extended SP to increase its solution speed.

$$\min V(X,Z) + H_1(X,Z) + H_2(X,Z) + H_3(X,Z) \tag{4.33}$$

Subject to :

$$\sum_j X_j = n \tag{4.34}$$

$$Z_{(j+l)} = s_l X_j \qquad \forall\ j \in J, \forall\ l \in L \tag{4.35}$$

$$\sum_{j=j}^{j+p} X_j \leq 1 \qquad \forall\ j \in J \tag{4.36}$$

$$C_V \tag{4.37}$$

$$C_{H_1} \tag{4.38}$$

$$C_{H_2} \tag{4.39}$$

$$C_{H_3} \tag{4.40}$$

$$X_j, Z_{ij} \in \{0,1\} \qquad \forall\ i \in I,\ k \in K \tag{4.41}$$

It is likely that the initial structure of the SP will provide the same solution as its extended version until the number of tours inserted in the MP schedule is large enough to bring the undercoverage to small values. Hence, we propose the following strategy : to use the initial SP structure until the undercoverage are small enough, and the extended version afterwards. Of course, the issue is to determine which value of undercoverage should be used as a trigger to activate the extension. Since we don't expect the SP to replace a tour until there is a period with less than one unit of undercoverage somewhere on the planning horizon, we have set this extension's activation trigger at the first period reaching an undercoverage smaller than one unit.

4.6.3 Shuffle Mechanism

Being a heuristic, our algorithm faces the risk of entering into a local optimum from which escaping will be difficult. This is an especially serious risk considering there are no moves implemented to allow temporary degradations of the solution value such as a tabu search would typically have. Indeed, as such mildest ascent mechanisms generally require specific calibrations for each instance in order to perform well, we believed it was not really suitable for a very flexible algorithm structure. Hence, we have developed a shuffling extension which excludes all tours from the current solution, providing the MP with an opportunity to rebuild a completely new schedule with a large set of tours already at its disposal in the tour pool.

4.6.4 Limited Tour Pool

Since it has access to all tours used in the previous solution, a limitation of the shuffle mechanism is that it might simply reconstruct a schedule identical to the previous solution. By tracking how many iterations have been executed since a given tour has been excluded from the solution, it is possible to specify how long to keep uninserted tours in memory. This allows a dynamic management reduction of the size of the problem which reduces solution times, but also provides an increase in the exploration potential of the shuffle mechanism by setting the maximum number of iterations for uninserted tours to a value smaller than the number of tours needed in the schedule. By doing so, the shuffle mechanism will run out of available tours from the tour pool before it has reconstructed a schedule with the same number of tours as the previous solution. Hence, in order to complete the new schedule, other tours will have to be provided by the SP from a partially reconstructed solution, which will create some potentially very useful diversification.

4.7 Results

We have tested two algorithm variants of CHAIR and in this section we will present how they perform on the flexibility benchmarking framework developed by Crowe et al. [66]. This first variant (C1) aims at providing good solutions fast, while the second variant (C2) aims instead at providing the best solution values. These results will be compared with those obtained with the SOFA algorithm variants presented in [66]. All programs were coded in C++. All experiments were carried out on an Intel®Core$^{\text{TM}}$Duo CPU running at 1.86 GHz with 1 Gb of RAM.

4.7.1 Configuration

The very limited number of parameters used increases the robustness of the present approach in terms of portability. In fact, the only parameters changing between instances in our tests are the values of penalties for constraint violations. The few other parameters of our approach will not be modified throughout the computational experiments discussed hereafter. They have been set after some extensive preliminary testing, and are the following : the size of the tour pool and the termination criteria of the reshuffle mechanism.

The size of the tour pool is limited by defining the number of iterations that a tour can be kept in the pool without having been used in a solution. For portability purposes, we set this value in the following fashion : a percentage of the number of tours needed to obtain a quantity of service coverage equal to the quantity of service demand required. This value is calculated by assuming that all tour types are used in equal proportion in the solution. While this assumption clearly introduces a significant bias in the context of a given instance or family of similar instances, it is however an efficient way of defining a portable parameter which does not have to be recalibrated on each new instance. We have set this parameter at 0.40 (40%). This means that if a problem requires 100 tours to obtain a theoretical complete coverage, tours excluded from the current solution will be kept in the tour pool during 40 iterations. The value of the termination criterion of the reshuffle mechanism has been set to five iterations without any improvement of the best known solution.

To enable the comparison with SOFA, all constraint violation penalties have been set to the same values as those selected by Crowe et al. [66] for the proposed flexibility benchmarking experimental framework. All implemented constraints and their respective penalties for violations are presented on figure 4.I. Meanwhile, the descriptions of sets of constraints activated for each instance set of the flexibility experimental framework are presented in table 4.II. Both tables were initially presented by Crowe et al. (2011) [66].

Category		Constraint	Right hand side value		Penalty value per violation
			Level 1	Level 2	
Vertical		Target service demand	Service demand		Overcoverage : 1 (exp. 1.5)
					Undercoverage : 1 (squared)
		Capacity	0.95% of peak	0.85% of peak	10,000
Horizontal	Set of lines	Budget	130 employees	120 employees	10,000
		Maximum tours of each type	999, 50	999, 50	5,000
	Set of work shifts	Max consecutive work shifts	6, 4	5, 3	500
		Maximum consecutive days off	1, 2	2, 2	500
	Single shift	Limits on start times	Between 2 and 5 am		500
		Limits on end times			500

TABLE 4.I – Description of Constraints

The framework proposed by [66] includes a total of 5 different service demand coverage curves specifically built for our tests. To allow an analysis on the impact of demand patterns on the cost of the solution, all five demands have an equal surface underneath their curves which means that the total theoretical number of service hours required is the same for all. These five service demand curves are presented in figure 4.4. They are respectively demands which represent a daily mild double peak (public transit, city tollbooths), a high early morning and low late afternoon peak (shipping personnel of a warehouse), a low early morning and high late afternoon peak (office cleaning personnel), a daily high peak (emergency services, retail stores, restaurants) and a uniform demand (factory personnel).

As each of the five initial demand curves averages a value of 1 on the vertical axis, the values of each point on each period of each curve have been multiplied by 30 in order to obtain a real-life representative size for our instances. This set of five service demand curves represent a good variety of shapes similar to those encountered in real-life applications (although their amplitudes may strongly vary in practice).

4.7.2 Testing Two CHAIR Variants

C1 was built to produce good results quickly and C2 to produce the best solution values. C1 combines the initial algorithm with the SP post optimization, the reshuffle mechanism and the limited tour pool, while C2 combines all the extensions of the former and the extended subproblem. The results of C1 and C2 on the framework proposed by [66] are respectively presented in table 4.III and 4.IV. Furthermore, table 4.V presents a comparison of results between C1 and C2. In this last table, the values in each cell are computed by dividing the corresponding value of C2 by that of C1. Hence, a value lower than 1 (or 100%) means that C2

Constraint set description			Set number
Service Demand	-		1
	Capacity 1		2
	Capacity 2		3
	Budget 1		4
	Budget 2		5
	Max tours of each types		6
	Max tours of each types	Budget 1	7
		Budget 2	8
	Max consecutive work shifts 1		9
	Max consecutive work shifts 2		10
	Min consecutive days off 1		11
	Min consecutive days off 2		12
	Max consecutive work shifts 1	Min consecutive days off 1	13
		Min consecutive days off 2	14
	Max consecutive work shifts 2	Min consecutive days off 1	15
		Min consecutive days off 2	16
	Limits on shift start times		17
	Limits on shift end times		18
	Limits on shift start times	Limits on shift end times	19
	All constraints 1		20
	All constraints 2		21
	All constraints 1 except capacity & budget		22
	All constraints 2 except capacity & budget		23

TABLE 4.II – Constraint Sets Reference Numbers with Their Corresponding Descriptions

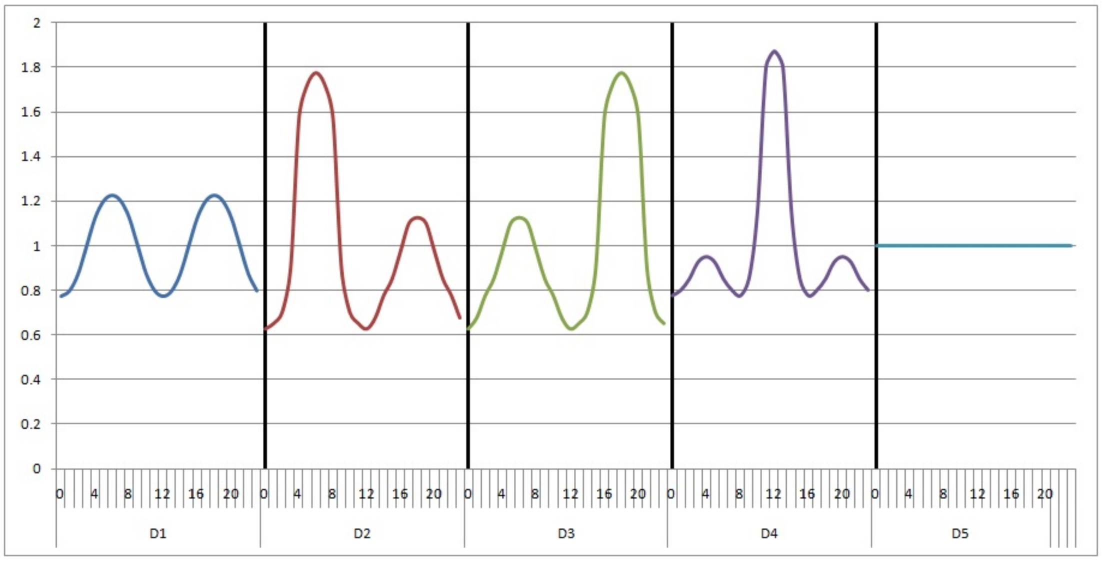


FIGURE 4.4 – Demand Curves of D1, D2, D3, D4 and D5

Constraint	CPU (s)						Solution values					
set	D1	D2	D3	D4	D5	Mean	D1	D2	D3	D4	D5	Mean
1	210	447	656	337	281	386.20	98.08	751.33	745.31	266.24	64.81	385.15
2	145	295	551	206	108	261	486.01	1056.59	1059.48	428.11	1835.45	973.13
3	116	186	376	142	278	219.60	3480.94	3847.84	3816.74	2368.79	8934.45	4489.75
4	187	288	406	192	239	262.40	599.74	2530.89	2528.63	1015.88	322.56	1399.54
5	263	264	332	152	150	232.20	4090.93	7523.72	7522.70	4823.55	3314.94	5455.17
6	165	362	412	184	162	257	86.22	754.55	755.67	369.83	67.12	406.68
7	132	447	235	136	258	241.60	884.75	2509.44	2523.30	1259.09	864.14	1608.14
8	135	250	174	107	106	154.40	4672.30	7519.54	7517.90	5114.47	4657.28	5896.30
9	243	336	392	297	247	303	101.44	750.09	751.56	276.20	68.79	389.62
10	193	439	375	199	329	307	105.09	752.53	753.28	275.61	75.24	392.35
11	100	372	479	262	191	280.80	96.16	760.62	749.93	276.17	95.11	395.60
12	209	476	700	367	312	412.80	98.08	751.33	745.31	266.24	64.81	385.15
13	264	449	550	449	272	396.80	101.29	756.65	757.43	279.78	85.23	396.08
14	244	466	472	277	347	361.20	152.01	833.38	830.42	341.17	109.85	453.37
15	327	457	683	212	229	381.60	107.36	757.67	754.06	282.03	95.47	399.32
16	175	538	532	231	302	355.60	156.36	843.56	833.41	342.69	113.32	457.87
17	225	631	988	648	180	534.40	360.67	5421.57	2191.57	380.03	111.42	1693.05
18	172	401	474	211	192	290	104.91	752.09	809.40	484.24	109.58	454.04
19	239	629	300	293	321	356.40	360.67	5421.57	2179.87	398.89	94.67	1691.13
20	153	191	369	155	219	217.40	2052.35	10342.50	4590.04	1867.89	2841.18	4338.79
21	130	165	188	125	100	141.60	6356.93	16437.62	9935.64	6538.47	11697.02	10193.14
22	210	424	636	222	213	341	952.69	6065.95	2204.59	889.03	177.64	2057.98
23	236	411	596	240	230	342.60	996.47	6166.11	2255.39	956.49	219.47	2018.61
Mean	194.48	388	472.87	245.39	228.96	305.94	1152.24	3622.05	2470.07	1282.65	1566.07	2018.61

TABLE 4.III – Solution Times (in seconds) and Values for C1

Constraint	CPU (s)						Solution values					
set	D1	D2	D3	D4	D5	Mean	D1	D2	D3	D4	D5	Mean
1	2083	15257	17646	2337	1589	7782.40	74.75	751.45	749.52	255.95	41.63	374.66
2	709	8584	7604	2011	84	3798.40	470.39	1061.64	1085.78	408.59	1835.45	972.37
3	235	2557	2443	1102	7314	2730.20	3458.96	3747.72	3768.05	2366.19	8934.45	4455.07
4	7326	11598	7279	619	2808	5926.00	596.90	2518.33	2527.30	1007.44	323.63	1394.72
5	6001	3366	2315	1467	1462	2922.20	4090.93	7529.89	7545.47	4823.55	3314.94	5460.96
6	1861	17507	17848	2354	1740	8262.00	84.95	748.63	751.52	369.83	61.42	403.27
7	117	11599	2650	374	5614	4070.80	884.75	2519.13	2543.26	1259.09	864.14	1614.07
8	555	2026	2244	96	95	1003.20	4672.3	7520.00	7529.71	5114.47	4657.28	5898.75
9	1798	7794	18678	2373	1431	6414.80	79.00	752.81	754.32	261.83	62.86	382.16
10	1972	17248	15774	2380	2196	7914.00	66.93	749.57	752.47	260.94	70.88	380.16
11	2026	16912	18670	2985	2143	8547.20	73.68	751.62	752.43	266.90	68.61	382.64
12	2458	11686	17150	3141	2339	7354.80	106.04	818.05	823.24	304.87	90.99	428.64
13	1937	11562	13754	2678	2252	6436.60	84.99	763.05	755.72	267.88	76.29	389.59
14	2114	16486	15608	2681	2145	7806.80	101.94	824.79	820.07	283.51	83.96	422.85
15	2269	11344	19218	2813	2093	7547.40	90.93	765.65	756.93	261.93	68.22	388.73
16	2403	16435	15259	3108	2523	7945.60	103.27	837.60	821.32	299.47	102.25	432.78
17	5191	20437	16810	2899	1863	9440.00	366.78	5414.89	2208.76	375.16	83.00	1689.72
18	2048	12356	8736	3715	2017	5774.40	80.98	754.49	813.49	432.14	81.86	432.59
19	6014	21089	18155	2745	2244	10049.40	366.78	5414.89	2182.23	415.91	83.82	1692.73
20	667	1288	13134	698	3784	3914.20	2052.35	10348.14	4594.86	1867.89	2841.18	4340.88
21	219	598	1500	185	83	517.00	6356.93	16447.50	9945.27	6538.47	11697.02	10197.04
22	3081	8073	16527	3657	2058	6679.20	938.64	6057.84	2203.47	889.03	186.65	2055.13
23	3838	9326	11761	3384	2543	6170.40	995.63	6154.28	2311.05	921.02	194.85	2115.37
Mean	2474.87	11092.52	12207.09	2165.30	2279.13	6043.78	1139.08	3619.65	2478.01	1271.83	1557.63	2013.26

Table 4.IV – Solution Times (in seconds) and Values of C2

performs better, while a higher value means C1 performs better.

Our main interest in this section is to compare the performances of C1 with those of C2. As expected, C2 provides in average slightly better solution values than C1. On a total of 115 instances, C2 produces better results on 61 instances, while C1 produces better results on 35 instances ; both algorithm variants provided the exact same solution values on 19 instances. The average solution value for C1 is 2018.61 compared to 2013.26 for C2, while standard deviations are respectively 2880.47 and 2884.15. Although the large standard deviations are simply an indicator of the large variations in the problem structures of our different instances, the very small difference between both averages would suggest that both algorithm variants are rather equivalent in terms of quality of solution value. However, comparing the results of corresponding individual instances in terms of solution values suggests some important differences between the performance of both algorithm variants. We have constructed table 4.V to provide a better insight on those dif-

Constraint	CPU (x)						Solution values (%)					
set	D1	D2	D3	D4	D5	Mean	D1	D2	D3	D4	D5	Mean
1	9.92	34.13	26.9	6.93	5.65	16.71	**76.21%**	100.02%	100.56%	**96.14%**	**64.23%**	**87.43%**
2	4.89	29.10	13.8	9.76	**0.78**	11.67	**96.79%**	100.48%	102.48%	**95.44%**	**100.00%**	**99.04%**
3	2.03	13.75	6.50	7.76	26.31	11.27	**99.37%**	**97.40%**	**98.72%**	**99.89%**	**100.00%**	**99.08%**
4	39.18	40.27	17.93	3.22	11.75	22.47	**99.53%**	**99.50%**	**99.95%**	**99.17%**	100.33%	**99.70%**
5	22.82	12.75	6.97	9.65	9.75	12.39	**100.00%**	100.08%	100.30%	**100.00%**	**100.00%**	100.08%
6	11.28	48.36	43.32	12.79	10.74	25.30	**98.53%**	**99.22%**	**99.45%**	**100.00%**	**91.51%**	**97.74%**
7	**0.89**	25.95	11.28	2.75	21.76	12.53	**100.00%**	100.39%	100.79%	**100.00%**	**100.00%**	100.24%
8	4.11	8.10	12.90	**0.90**	**0.90**	5.38	**100.00%**	100.01%	100.16%	**100.00%**	**100.00%**	100.03%
9	7.40	23.2	47.65	7.99	5.79	18.41	**77.88%**	100.36%	100.37%	**94.80%**	**91.38%**	**92.96%**
10	10.22	39.29	42.06	11.96	6.67	22.04	**63.69%**	**99.61%**	**99.89%**	**94.68%**	**94.21%**	**90.42%**
11	20.26	45.46	38.98	11.39	11.22	25.46	**76.62%**	**98.82%**	100.33%	**96.64%**	**72.14%**	**88.91%**
12	11.76	24.55	24.50	8.56	7.50	15.37	108.12%	108.88%	110.46%	114.51%	140.40%	116.47%
13	7.34	25.75	25.01	5.96	8.28	14.47	**83.91%**	100.85%	**99.77%**	**95.75%**	**89.51%**	**93.96%**
14	8.66	35.38	33.07	9.68	6.18	18.59	**67.06%**	**98.97%**	**98.75%**	**83.10%**	**76.43%**	**84.86%**
15	6.94	24.82	28.14	13.27	9.14	16.46	**84.70%**	101.05%	100.38%	**92.87%**	**71.46%**	**90.09%**
16	13.73	30.55	28.68	13.45	8.35	18.95	**66.05%**	**99.29%**	**98.55%**	**87.39%**	**90.23%**	**88.30%**
17	23.07	32.39	17.01	4.47	10.35	17.46	101.69%	**99.88%**	100.78%	**98.72%**	**74.49%**	**95.11%**
18	11.91	30.81	18.43	17.61	10.51	17.85	**77.19%**	100.32%	100.51%	**89.24%**	**74.70%**	**88.39%**
19	25.16	33.53	60.52	9.37	6.99	27.11	101.69%	**99.88%**	100.11%	104.27%	**88.54%**	**98.90%**
20	4.36	6.74	35.59	4.50	17.28	13.69	**100.00%**	100.05%	100.11%	**100.00%**	**100.00%**	100.03%
21	1.68	3.62	7.98	1.48	**0.83**	3.12	**100.00%**	100.06%	100.10%	**100.00%**	**100.00%**	100.03%
22	14.67	19.04	25.99	16.47	9.66	17.17	**98.53%**	**99.87%**	**99.95%**	**100.00%**	105.07%	100.68%
23	16.26	22.69	19.73	14.10	11.06	16.77	**99.92%**	**99.81%**	102.47%	**96.29%**	**88.78%**	**97.45%**
Mean	12.11	26.53	25.78	8.87	9.45	16.55	**90.33%**	100.21%	100.65%	**97.34%**	**91.89%**	**96.08%**

Table 4.V – Solution Times (in seconds) and Values of C2 Divided by C1

ferences.

Indeed, the average comparative value obtained from table 4.V is 96.08%. This value may be defined as an indicator of how much better C2 performs compared to C1 and hence can be translated as meaning that the former provides results which are on average 4.08% better in this context of minimization. Meanwhile, the corresponding standard deviation is 10.56%, which implies the existence of at least a few important gaps between the performances of each algorithm variant. At this point, the important question regards which instances are responsible for most variations of performance.

Since our algorithm is fully heuristic, has a quadratic objective function, requires small execution times relatively to problem sizes and prioritizes portability and flexibility, a number of instances might have solutions with significant optimality gaps. While optimal solutions (or reliable lower bounds) are unknown for any of our instances and of little interest to us in this context of constraint satisfaction, the reader may notice a few isolated instances such as constraint set 10 with service demand D1 or constraint set 12 with service demand D5 revealing large differences in solution values between both algorithm variants. An interesting fact seems to be that larger differences in solution values between algorithm variants happen with smaller solution values, which can be interpreted as meaning that the difference in terms of number of violations remains relatively limited.

While extracting individual results provides little insight on general performance (these last two examples combined represent less than 2% of all instances), averages by columns (service demands) and lines (constraint sets) appear to be much more revealing. Indeed, the average values of table 4.V show that C2 performs largely better on service demand 1 and 5 (averages for each column are respectively 90.32% and 91.89%), slightly better for service demand 4 (97.34%) and comparable performances for service demands 2 and 3 (respectively 100.21% and 100.65%). A noteworthy fact about these last statistics is that from our experience, these service demand coverage curves can be ordered from the most to least difficult to cover by the following sequence : 1, 5, 4, 3, 2. While experience obviously provides no statistical proof, it nevertheless seems to indicate that the performance differences between both algorithm variants is reduced when instances include more difficult service demands to cover. Another way to present this is that C2 provides better results when service demands are less constraining to cover.

This last insight seems to be confirmed when analyzing results by constraint sets. Indeed, experience with these instances has led us to observe that the inclusion of at least one type of constraint on sets of lines (maximum number of tours of each type or budget limitations) or of capacity constraints create difficult instances to solve. Indeed, the average values for instances including one of these three difficult constraints in table 4.V is 99.46% (this includes constraint sets 2, 3, 4, 5, 6, 7, 8, 20, 21, 22 and 23), while the average for all other instances is 92.98%. Again, this can be interpreted as a strong indicator that C2 generally provides larger improvements on less tightly constrained instances.

A joint analysis of tables 4.III, 4.IV and 4.V indicates that large differences in terms of percentage of solution values between algorithm variants always occur with small differences in number of penalty for violations. We have defined a large difference as a value under 85% or over 115% in table 4.V. Indeed, basing ourselves on table 4.IV for calculations, the average empirical value of instances with differences larger than +/- 15% is 91.66, which is almost 22 times smaller than the average solution value for the full table 4.IV. Hence, these cases can be interpreted as differences limited to a small number of minor violations. While we find solution value results quite convincing and stable in terms of overall performance, we are surprised by the larger improvements provided by C2 on less constrained instances. Indeed, we had initially developed the extended SP mechanism because we expected the initial SP algorithm to face challenges when capacity or budget constraints are included. This expectation was based on the fact that the initial SP algorithm always assumes new tours will be added to the current schedule when creating them. Intuitively, this led us to believe that it could perform poorly when reaching a point in the solution searching process where budget or capacity constraints forbid adding new tours, forcing instead the replacement of current ones to obtain improvements to the current solution. Hence, while the extended SP does improve solution values on average, it does not perform in the way we had initially built it for.

When analyzing the differences in values between solutions obtained with C1 and C2, we have computed that the average absolute difference is 9.90 penalty points for instances with tighter constraint sets and larger solution values (all instances of constraint sets 2, 3, 4, 5, 6, 7, 8, 20, 21, 22 and 23) and 17.86 for all other instances which have smaller solution values. When considering negative values in our calculations, these averages become respectively an improvement of 2.71 and of 7.79 points provided by C2. One possible way to explain these counter-intuitive results is the lack of performing local searches which can modify single shift structure inside the MP. Clearly, such a structure (like backtracking local searches 3 and 4 which were excluded after preliminary testings) goes against the basic philosophy of this approach. However, it is possible that dividing the construction of individual tours and the selection of tours is a less efficient strategy when facing budget or capacity constraints as they both limit sums of tours, which are more difficult to consider in the SP. More insight on this matter is available in section 4.7.3 where CHAIR is compared with SOFA [66].

On the other hand, analyzing solution times is very straightforward. Indeed, a quick look at table 4.V will suffice for the reader to notice the large differences between C1 and C2 on this matter. Out of 115 instances, only 5 have smaller solution times with C2. In addition, solution times are on average almost 20 times smaller with C1. Although clear solution time patterns are difficult to find when data is aggregated by constraint set, there is once again a clear pattern when aggregated by service demand : the increase in solution times are much higher for service demand 2 and 3 when using C2 compared to C1. This leads us to another possible explanation for the limited improvement provided by C2 on more constrained instances : these instances may generally include a large number of rather flat areas in the solution space, with a large number of local optima with only slight differences in solution value between them. This would indeed explain higher solution times as more searching is required to progress towards a local optima, while escaping an area of the solution space would be a difficult task.

4.7.3 Comparison with SOFA

In Crowe et al. [66], S1 and S2 of SOFA were the two algorithm variants tested. In this paper, we have tested C1 and C2 of CHAIR. For both algorithms, one algorithm variant was designed to provide the best solution values, and one algorithm variant was designed to provide a good solution time/solution value ratio. In order to make a fair comparison, S1 of SOFA and C1 of CHAIR will be compared together as they each provide the best solution time/solution ratio for their respective algorithm, while S2 of SOFA and C2 of CHAIR will be compared with each other as they each provide the best overall solution values out of all other algorithm variants of their respective algorithm.

Best Solution Time/Solution Value Ratio

In this section, S1 of SOFA will simply be referred to as SOFA and C1 of CHAIR will simply be referred to as CHAIR. Table 4.VI presents a comparison of solution values and solution times obtained by both algorithms. In each cell, values are obtained by dividing the result obtained with CHAIR by the one obtained with SOFA. If the value is lower than 100%, CHAIR performs better and if the value is over 100%, SOFA performs better.

In terms of solution values, results are very convincing and clearly indicate that CHAIR performs better. Indeed, CHAIR outperforms SOFA on 68.70% of instances. On average, solution values obtained with CHAIR are 6.40% lower. However, CHAIR is outperformed on some specific instances, mainly on those with constraint sets 20 to 23 inclusively, as well as on instances with the fifth service demand (uniform). On average, solution values are 19.01% higher for instances with the fifth service demand, while they are 13.53% higher on average for instances with constraint sets 20 to 23. However, when excluding results for the fifth service demand, solution values are only 0.67% higher on instances for constraint sets 20

Constraint	CPU (%)						Solution values (%)					
set	D1	D2	D3	D4	D5	Mean	D1	D2	D3	D4	D5	Mean
1	**63.64%**	**88.51%**	148.75%	**98.25%**	**98.60%**	**99.55%**	**68.38%**	**95.07%**	**94.13%**	**79.84%**	**86.90%**	**84.86%**
2	**30.85%**	**48.20%**	118.75%	**51.24%**	**43.03%**	**58.41%**	**71.84%**	107.70%	107.94%	**65.24%**	**66.74%**	**83.89%**
3	**22.75%**	**39.74%**	**75.81%**	**32.87%**	112.55%	**56.74%**	**96.65%**	**67.49%**	**65.84%**	**89.84%**	**80.39%**	**80.04%**
4	**92.12%**	**85.46%**	123.40%	**92.31%**	168.31%	112.32%	**86.35%**	**97.32%**	**96.74%**	**92.98%**	**93.47%**	**93.37%**
5	199.24%	105.18%	125.28%	**95.60%**	105.63%	126.19%	**99.39%**	**99.77%**	**98.99%**	**98.65%**	**99.27%**	**99.21%**
6	**47.14%**	**100.00%**	104.57%	**49.07%**	**54.55%**	**71.07%**	116.83%	**89.29%**	**93.52%**	104.15%	115.70%	103.90%
7	**71.35%**	164.34%	**98.74%**	**59.91%**	144.94%	107.86%	102.23%	**93.12%**	**95.88%**	100.59%	102.06%	**98.78%**
8	**81.33%**	130.89%	**79.09%**	**63.69%**	**70.67%**	**85.13%**	100.31%	**99.03%**	**98.34%**	**99.86%**	100.14%	**99.54%**
9	**72.11%**	**76.19%**	**83.05%**	**72.97%**	**75.30%**	**75.92%**	**50.78%**	**90.07%**	**90.39%**	**82.16%**	**82.85%**	**79.25%**
10	**52.59%**	**77.43%**	**67.57%**	**44.72%**	124.62%	**73.39%**	**42.41%**	**89.53%**	**84.24%**	**89.61%**	**85.40%**	**78.24%**
11	**25.58%**	**65.49%**	**74.03%**	**54.93%**	**53.95%**	**54.80%**	**61.64%**	**94.42%**	**93.07%**	**88.30%**	134.85%	**94.46%**
12	**56.79%**	**86.70%**	111.46%	**83.03%**	**81.46%**	**83.89%**	**41.12%**	**83.48%**	**83.44%**	**56.89%**	**93.76%**	**71.74%**
13	**52.80%**	**68.13%**	**89.00%**	**82.23%**	**68.86%**	**72.20%**	**65.71%**	**87.22%**	**86.09%**	**71.79%**	134.69%	**89.10%**
14	**49.80%**	**71.80%**	**63.19%**	**52.96%**	105.47%	**68.64%**	**46.89%**	**66.05%**	**62.07%**	**49.72%**	132.21%	**71.39%**
15	**60.67%**	**67.01%**	107.22%	**42.48%**	**51.23%**	**65.72%**	124.95%	**70.63%**	**87.86%**	101.34%	150.46%	107.05%
16	**36.46%**	**51.14%**	**49.53%**	**35.21%**	**66.96%**	**47.86%**	**35.24%**	**43.28%**	**38.58%**	**41.78%**	160.74%	**63.92%**
17	**81.52%**	146.74%	250.76%	137.58%	**69.50%**	137.22%	**87.42%**	103.13%	100.17%	**94.13%**	114.36%	**99.84%**
18	**53.75%**	**82.34%**	**85.56%**	**63.75%**	**63.37%**	**69.75%**	**83.47%**	**94.28%**	**96.12%**	139.47%	151.23%	112.91%
19	**88.52%**	120.73%	**73.53%**	**73.62%**	117.15%	**94.71%**	**96.81%**	102.61%	**97.76%**	**91.06%**	**91.99%**	**96.05%**
20	**34.46%**	**47.28%**	**70.83%**	**32.77%**	**44.24%**	**45.92%**	**93.07%**	102.11%	**96.78%**	105.78%	109.40%	101.43%
21	**18.81%**	**23.61%**	**30.08%**	**20.87%**	**19.84%**	**22.64%**	101.67%	101.57%	**94.34%**	103.39%	**93.11%**	**98.82%**
22	**43.21%**	**76.53%**	114.80%	**36.33%**	**44.47%**	**63.07%**	106.59%	103.20%	101.30%	109.07%	173.19%	118.67%
23	**28.43%**	**44.92%**	**66.59%**	**28.47%**	**29.60%**	**39.60%**	102.13%	103.62%	**91.05%**	**95.02%**	284.21%	135.21%
Mean	**59.30%**	**81.23%**	**96.16%**	**61.08%**	**78.88%**	**75.33%**	**81.82%**	**90.61%**	**89.33%**	**89.16%**	119.01%	**93.99%**

TABLE 4.VI – Ratios of Solution Times and Values Obtained Dividing the Results of C1 of CHAIR by Those of S1 of SOFA

to 23. This clearly indicates us that SOFA significantly outperforms CHAIR on instances with the fifth service demand. An analysis of empirical values demonstrates that differences in terms of number of penalty violations are in fact rather small for these specific instances : They are of 53.37 penalty points of violations on average for instances where SOFA performs better. In fact, the few times SOFA outperforms CHAIR seem to be mostly limited to instances with small solution values. Meanwhile, CHAIR tends to outperform SOFA on instances with large numbers of penalized violations, even when the instance includes the fifth service demand. For example, this is the case for the instance combining constraint set 3 with the fifth service demand.

In terms of solution times, results are straightforward. On average, solution times are 32.75% lower with CHAIR. The only exception seems to be for instances with constraint set 5 (tight budget constraints) and constraint set 17 (limits on possible start time for shifts) where solution times are on the contrary respectively 26.19%

and 37.22% longer with CHAIR. Overall, solution times are quicker with CHAIR on 80% of the instances.

Best Overall Solution Values

In this section, S2 of SOFA will simply be referred to as SOFA and C2 of CHAIR will simply be referred to as CHAIR. Table 4.VII presents a comparison of solution values and solution times obtained by both algorithms. In each cell, values are obtained by dividing the result obtained by chair with the one obtained by SOFA. If the value is lower than 100% , CHAIR performs better and if the value is higher than 100%, SOFA performs better.

The performance differences in terms of solution values is very small between SOFA and CHAIR. Indeed, CHAIR provides solution values 0.47% better than SOFA in average, which is clearly not a very significant difference in this context. While a certain number of large performance differences can be noticed on some instances, 63.48% of instances have values bounded between 90% and 110% while 76.52% of instances have values bounded between 80% and 120% on table 4.VII. Considering that our main research objective is flexibility and that we are dealing with a large variety of instances with very different problem structures, we believe one can interpret these numbers as meaning that both algorithms perform in a similar fashion and are competitive on most instances. However, CHAIR performs better on instances with service demands 1, 2, 3 and 4 and constraint sets 1 to 19 inclusively, while SOFA performs better on service demand 5 and constraint sets 20 to 23 inclusively. In fact, this behavior is very similar to the one we had already noticed with the fast versions of these algorithms in section 4.7.3. Indeed, the solution value is 6.15% better on average with CHAIR on all instances built with demands 1 to 4. The performances of this same algorithm become 8.93% better for instances built with the same four first demand when limited to constraint

Constraint	CPU (%)						Solution values (%)					
set	D1	D2	D3	D4	D5	Mean	D1	D2	D3	D4	D5	Mean
1	**59.82%**	229.77%	232.77%	**60.03%**	**19.03%**	120.28%	**52.12%**	**95.42%**	**95.12%**	**81.28%**	**68.03%**	**78.39%**
2	**8.68%**	119.89%	131.56%	**26.93%**	**2.15%**	**57.84%**	**90.88%**	104.04%	110.16%	**89.58%**	**75.61%**	**94.05%**
3	**1.61%**	**17.06%**	**22.69%**	**10.21%**	194.68%	**49.25%**	103.85%	102.02%	106.17%	**99.85%**	**80.67%**	**98.51%**
4	183.47%	227.68%	200.41%	**21.87%**	**70.62%**	140.81%	100.04%	**98.96%**	**98.14%**	**96.70%**	**94.98%**	**94.76%**
5	216.41%	**77.10%**	**38.85%**	**51.04%**	**55.13%**	**87.71%**	100.62%	**99.83%**	100.44%	**99.08%**	**99.82%**	**99.96%**
6	**27.39%**	259.44%	236.15%	**34.41%**	**39.29%**	119.31%	127.15%	**94.64%**	**93.28%**	121.91%	106.85%	108.77%
7	**3.01%**	369.63%	**63.79%**	**12.72%**	113.97%	112.62%	103.45%	**96.46%**	**99.23%**	102.04%	102.19%	100.67%
8	**12.37%**	**39.71%**	**76.46%**	**2.86%**	**1.80%**	**26.64%**	100.36%	**99.29%**	**99.86%**	101.07%	100.40%	100.20%
9	**26.38%**	111.53%	303.07%	**21.65%**	**15.71%**	**95.67%**	**61.18%**	**94.37%**	**89.48%**	**77.16%**	105.01%	**85.44%**
10	**39.62%**	357.55%	311.55%	**48.65%**	**36.86%**	158.85%	**38.89%**	**94.81%**	**90.74%**	**74.21%**	113.90%	**82.51%**
11	**13.03%**	345.78%	341.57%	**24.73%**	**18.70%**	148.76%	**94.26%**	**96.34%**	**96.09%**	**95.77%**	119.24%	100.34%
12	**15.49%**	241.20%	308.18%	**70.93%**	**39.04%**	134.97%	109.27%	**95.98%**	101.24%	**88.61%**	139.43%	106.91%
13	**20.87%**	220.14%	**99.56%**	**35.60%**	**27.19%**	**80.67%**	104.65%	**94.63%**	**94.97%**	**97.75%**	133.56%	105.11%
14	**36.23%**	**84.79%**	155.83%	**23.02%**	**54.83%**	**70.94%**	**40.14%**	**84.48%**	**80.64%**	**68.90%**	112.73%	**77.38%**
15	**39.07%**	190.94%	383.82%	**31.44%**	**22.28%**	133.51%	116.34%	**92.42%**	**85.87%**	**94.48%**	124.95%	102.81%
16	**27.51%**	107.54%	**85.92%**	**33.26%**	**41.76%**	**59.20%**	**33.34%**	**61.44%**	**50.12%**	**45.65%**	149.36%	**67.98%**
17	153.85%	149.81%	327.94%	**41.24%**	**37.25%**	142.02%	**94.62%**	103.69%	100.05%	100.28%	108.75%	101.48%
18	**38.64%**	282.36%	210.96%	113.12%	**21.12%**	133.24%	**66.55%**	**95.22%**	**97.03%**	106.58%	124.22%	**97.92%**
19	160.89%	293.80%	525.01%	**73.10%**	**53.79%**	221.32%	100.52%	103.54%	**99.12%**	102.57%	136.05%	108.36%
20	**6.63%**	**16.15%**	170.88%	**13.84%**	**13.78%**	**44.26%**	**99.29%**	102.50%	**99.94%**	109.68%	114.79%	105.24%
21	**3.92%**	**4.64%**	**6.50%**	**3.08%**	**1.49%**	**3.93%**	101.67%	161.25%	**61.26%**	104.76%	100.66%	105.92%
22	**61.92%**	**67.34%**	133.80%	**52.29%**	**27.30%**	**68.53%**	105.01%	103.14%	101.26%	120.64%	239.26%	133.86%
23	**55.16%**	**72.75%**	**74.99%**	**32.44%**	**50.52%**	**57.17%**	106.61%	103.34%	**99.57%**	110.35%	228.16%	129.61%
Mean	**52.69%**	168.98%	193.14%	**36.45%**	**41.66%**	**98.59%**	**89.17%**	**99.04%**	**93.47%**	**95.17%**	120.81%	**99.53%**

TABLE 4.VII – Ratios of Solution Times and Values Obtained Dividing the Results of C2 of CHAIR by Those of S2 of SOFA

sets 1 to 19 inclusively. Meanwhile, solution values are 20.81% better on average with SOFA for instances with the fifth service demand, and 18.66% better on average with SOFA for all instances with constraint sets 20 to 23 inclusively. In fact, the compared behavior of both algorithms is similar to what had been observed in section 4.7.3. Indeed, it seems that SOFA performs better on instances with tighter solution spaces and less natural demand curves, while CHAIR performs better on instances with looser bounds and more natural service demand curves.

In terms of solution times, results are also similar : CHAIR outperforms SOFA only by a slight 1.43% on average. A very interesting fact to notice is that solution times are 2.4 times quicker in average with CHAIR on instances with the fifth service demand, which would indicate that this algorithm does not manage to fully explore the solution space on those instances. Likewise, solution times are 2.3 times faster on average with CHAIR on instances with constraint sets 20 to 23, leading to the same observations. On the other hand, solution times are 55% lower with SOFA on instances with service demands 1 to 3 and constraint sets

1 to 19. Interestingly, solution times are much quicker with CHAIR for instances with the fourth service demand even though CHAIR also obtains better solution values on average on these instances.

Overall, CHAIR performs slightly better than SOFA both in terms of solution value and solution time. However, it is relevant to note that this better performance comes mostly from the less severely constrained instances. Indeed, constraints seem to explain a larger share of performance variations than demand functions do.

4.8 Conclusion

In this paper we have demonstrated the flexibility and portability potential of a fully heuristic approach inspired from column generation to solve a wide variety of staff scheduling problems. Our approach is divided into a subproblem (SP) which builds individual tours and a master problem (MP) which selects which tours are inserted in the schedule. In its basic structure, this algorithm solves the SP with a greedy heuristic and the MP with a combination of two local search procedures. In order to improve solution quality, a number of extensions have also been implemented : a post-optimization local search procedure for the SP, an extended SP local search algorithm, an accelerated version of the latter, a shuffle mechanism, a tour pool to use jointly with the shuffle mechanism and two backtracking local search structures. After a significant set of preliminary tests, two versions of the algorithm were found the most promising : the first one includes the post optimization mechanism for the SP combined with the shuffle mechanism and the tour pool, and the second one includes all the extensions of the first plus the accelerated extended SP structure. The first one was chosen for the very good compromise between solution quality and time it provides, while the second one was chosen because it provided the best overall performances in terms of solution quality. Both versions were tested on a large experimental framework of 115 instances structured specifically to evaluate the level of flexibility of a staff scheduling algorithm. Results were very promising and were obtained without any calibration of the algorithm between instances in order to demonstrate its portability. Meanwhile, the overall quality of the solutions and the limited variations in terms of solution quality relatively to this context of wide variations in the structure of instances demonstrated in our opinion the strong flexibility of the algorithm.

Both selected versions of the algorithm were each compared with a similar version of SOFA. When the objective was to obtain the best compromise between solution quality and solution times, CHAIR performed significantly better than SOFA. On the other hand, CHAIR performed only slightly better than SOFA on average when the objective was to simply obtain the best solution value. Interestingly, results from tests on the flexibility experimental framework seemed to indicate that CHAIR performed better on instances with more natural demand structures and looser solution search spaces, while SOFA performed better on instances with more angular demand structures and tighter solution search spaces.

While we believe we have demonstrated the potential of using heuristics inspired from column generation to obtain good solutions on a large variety on instances without requiring any calibrations, there is clearly in our opinion many opportunities left for more work on this topic. Indeed, our work has been focused on demand shape variations and schedule structure variations : assignment of tours to specific employees and individual tailorship of tours have been excluded from the scope of this paper. In our opinion, future research on flexibility and portability of staff scheduling algorithms should include important factors such as seniority, preferences, availability, competences and task assignment. As the structure of our algorithm is based on the assembly of individual tours at the SP level, we believe there is much potential for increasing the scope of staff scheduling problems it can solve in the future.

Chapitre 5

A Flexible and Portable Approach Applied to Nurse Scheduling

5.1 Introduction

Nurses play an important role in the patient care process. Yet, in places such as the province of Quebec, nursing departments are frequently understaffed for varying reasons : budget cuts, too few graduates, too many retirees, uncompetitive working conditions, service capacity limitations, etc. To deal with this, managers have little choice but to do better with less.

In this context, operations research has much to offer to nurse chiefs and health care service managers. Not surprisingly, nurse scheduling is the application with the largest number of published papers in staff scheduling according to Ernst et al. (2004) [83]. However, few of these papers share methodologies or solution approaches. Hence, we are interested in knowing how a shared methodology would perform in comparison with the stand-alone ones. Indeed, as suggested by Crowe and Soriano [66], practitioners and academics alike might have much to gain by having access to an effective flexible and portable methodology for nurse scheduling. This provides us with an opportunity to assess the performances of CHAIR (Column generation Heuristic Approach for Indefinite Rostering), a flexible and portable solution approach proposed by Crowe and Soriano (2012) [67], on a set of nurse scheduling problems.

CHAIR is a fully heuristic approach inspired from column generation : A subproblem builds a set of tours, and a master problem selects which tours are included in the schedule (the solution). This approach aims to provide good solutions on a large variety of staff scheduling problems, while limiting to a minimum the configuration efforts required to implement it on a different scheduling problem.

Our objective is to demonstrate that a flexible and portable solution approach can provide good results on nurse scheduling problems, and is an option worth considering for organizations dealing with many different nurse scheduling problems (e.g. a large general hospital) but having limited time and financial resources to develop a uniquely tailored approach for each one of them. Likewise, we believe it is an option worth considering for researchers collaborating with many nurse teams but lacking resources to provide a tailored approach for each of them.

CHAIR has already been successfully tested on a large set of theoretical staff scheduling instances inspired from the health care sector, as well as successfully benchmarked on 2 real-life health care scheduling problems from Montreal. In this paper, we plan to test it even further with 2 sets of nurse scheduling instances : a set of 144 theoretical instances created by the authors of this paper, and a set of 10 real-life instances from all over the world proposed both by academics and practitioners. The first set of instances will be used to test the impact of a number of selected types of constraints of interest on the behavior of the solution approach while understanding their impact on the structure of solutions, and the second set of instances will be used to benchmark CHAIR with the best solution approaches known for each problem.

This paper is structured as follows. Firstly, section 5.2 will shortly review the nurse scheduling literature. This review will be followed by a description of the algorithmic approach in section 5.3, and then in section 5.4 by a a detailed description of the test instances and the results achieved. Finally, section 5.5 will conclude this paper.

5.2 Literature Review

Literature in nurse scheduling is abundant. Among all problems tackled in the literature, the objectives are generally to minimize violations of demand coverage, minimize violations of work conditions, or a combination of both. Minimizing cost is rarely an issue for two reasons : firstly, the priority for health care service providers is generally to maximize the use of the allocated budget, as many are non-profit or public organizations, and secondly, the typically understaffed context creates more concern about filling vacancies than about reducing overcapacity.

While objective functions of different models are similar, a large variety of constraints are found in the literature. As reviewed by Crowe et al. (2001) [65], Cheang et al. (2003) [60] propose a comprehensive survey of the nurse rostering literature in which the most typical constraints are categorized as follows :

- Nurse workload (minimum/maximum) ;
- Consecutive same working shift (minimum/maximum/exact number) ;
- Consecutive working shift/days (minimum/maximum/exact number) ;
- Nurse skill levels and categories ;
- Nurses' preferences or requirements ;

- Free days (minimum/maximum/consecutive free days) ;
- Free time between working shifts (minimum) ;
- Shift types assignments (maximum shift type, requirements for each shift type) ;
- Holidays and vacations (predictable), e.g., bank holiday, annual leave ;
- Working weekend, e.g., complete weekend ;
- Constraints among groups/types of nurses, e.g., nurses not allowed to work together or nurses who must work together ;
- Shift patterns ;
- Historical record, e.g., previous assignments ;
- Other requirements in a shorter or longer time period other than the planning time period, e.g., every day in a shift must be assigned ;
- Constraints among shifts, e.g., only one shift be assigned to a person at a given time ;
- Nurse requirements per skill category for each shift (minimum/maximum/exact number).

While nurse scheduling problems rarely include all the aforementioned categories of constraints simultaneously, they frequently include enough to make them difficult to solve. More details about the nurse scheduling literature can be found in chapter 2. Meanwhile, the literature listed in the following paragraphs contain problems and solution approaches which are discussed later in this paper when benchmarking CHAIR on nurse scheduling problems. These problems and the benchmark test structure will be discussed further in section 5.4.

Azaiez et al. (2005) [14] propose a binary goal programming model for nurse scheduling. The model aims to solve a problem from a Saudi Arabia hospital and to replace a currently manual scheduling procedure. While all the labor constraints of the local hospital are taken into account, the authors also use a survey on nurses of a few different countries to gain an understanding of nurse preferences.

The results from this survey are used to add some work quality soft constraints to the model.

Ikegami et al. (2003) [102] propose a heuristic solution approach based on tabu search with a structure sharing similarities with column generation to solve a nurse scheduling problem from a Japanese hospital. This problem is particularly complex due to large numbers of workload balance constraints and employee skills which must be considered.

Li et al. (2003) [117] propose a hybrid AI approach to solve a class of nurse rostering problems which they classify as over-constrained. The approach is divided into two phases : the first is a relaxed problem including all hard constraints and a limited number of preference constraints solved by a forward checking algorithm, while the second is a local descent using tabu search on the full problem. On top of personal preferences, the problem is also constrained by a total of 8 work rules on how shifts can be assigned.

Millar et al. (1998) [129] solve some nurse scheduling problems by network programming. They use stints of shifts in their model, meaning that the schedules are built by assembling preexisting sequences of work shifts.

Finally, Burke et al. (2010) [58] propose a hybrid model of integer programming and VNS (variable neighborhood search) to solve highly-constrained nurse rostering problems. The integer programming is used to solve a partial problem in a first phase including all hard constraints and a subset of soft constraints. The VNS is then used in a second phase to improve the solution by considering all soft constraints excluded from the first phase. This approach is used to solve a nurse scheduling problem from an intensive care unit in the Netherlands. A total of 10 hard constraints and 7 soft ones are included.

Clearly, all the above listed problems differ from each other in many ways, which makes it challenging to solve them all with a single solution approach. Nevertheless, we will assess the capacity of our approach to solve all these instances in section 5.4.2.

Interestingly, Burke et al. (2004) [56] suggests the following elements as valuable future research areas for nurse scheduling : multi-criteria reasoning, flexibility and dynamic reasoning, robustness, ease of use, human/computer interaction, problem decomposition, exploitation of problem specific information, hybridization, and inter-disciplinarity. In fact, our research will deal with a number of them : multi-criteria reasoning, flexibility and dynamic reasoning, robustness and problem decomposition. Furthermore, our research is clearly about inter-disciplinarity as the solution approach used was initially built with a much larger application spectrum in mind than nurse scheduling, as discussed by Crowe et al. (2011) [67]. This solution approach is CHAIR, which is described in the following section.

5.3 Algorithmic Approach

The solution approach used to solve nurse scheduling problems in this paper is CHAIR (Column generation Heuristic Approach for Indefinite Rostering) proposed by Crowe et al. (2011) [67]. It is a fully heuristic approach based on a decomposition method inspired from column generation : a subproblem is solved to identify variables which are likely to improve the current solution of the master problem, and these variables are fed to the master problem in order to find a good solution to the full problem. As CHAIR is an algorithm built to solve staff scheduling problems, solving the subproblem produces tours, and these tours are used in the master problem to create a good schedule.

The present section will provide the reader with a general description of nurse scheduling problems, followed by a presentation of the solution approach. Since this paper focuses on presenting and discussing the results obtained by CHAIR on nurse scheduling problems, the reader is referred to Crowe et al. (2011) [67] from which section 5.3.2 to 5.3.5 are based for a more detailed description of the solution approach.

5.3.1 Problem Description

Nurse scheduling problems are solved by assigning a set of work shifts to a set of nurses in order to create a feasible work schedule. The objective can be to maximize demand coverage, maximize employee satisfaction, minimize work constraint violations, or any combination of these. Constraints typically include work preferences (length of shift, start time of shift, number of days off), work rules (typically described in work contracts or collective agreements), varying skills between nurses and service demand coverage.

The algorithm approach presented in the following paragraphs is structured according to a classification by constraint types proposed by Crowe et al. (2011) [66]. To the best of our knowledge, this classification can include any constraint found in nurse scheduling contexts. It is organized as follows :

- Vertical constraints :
  - Constraints on a single period :
    - Service demand coverage ;
    - Capacity (f.ex. : number of available vehicles or workspaces, budget per period, . . .).

- Horizontal constraints :
  - Constraints on sets of lines :
    - Management constraints (f.ex. : maximum number of tours of each type, budget by number of employees, . . .).
  - Constraints on sets of shifts of a single tour :
    - Collective agreements, ergonomics, working conditions (f.ex. : maximum number of consecutive work days, minimum number of consecutive days off, . . .).
  - Constraints on a single shift :
    - Collective agreements, ergonomics, working conditions (f.ex. : limits on possible start-times for work shifts, . . .).

As CHAIR is structured on the above classification, which includes a wide specter of possible types of constraints, it provides us with the ability to tackle a large number of nurse scheduling problem characteristics. It is required in order to deal with the large sets of problems presented in section 5.4.

5.3.2 Global Algorithm

Since CHAIR is structured in a fashion inspired from column generation, selecting a good strategy for constructing tours in the subproblem (SP) that will improve the solution of the master problem (MP) is critical. In a mathematical approach such as column generation, this is done by using dual variables to compute reduced costs. However, there is no straightforward way of obtaining dual variables in a fully heuristic approach, and calculating them would require a substantial increase in solution times when starting from primal informations of a heuristic. We dealt with this problem in a fashion inspired from the approach proposed by Ikegami et al. (2003) [102] : using only primal information, sets of constraints from the initial problem are either sent to the SP or MP, or implemented in both problems in order to serve as links between them, effectively ensuring that the

SP produces tours which are interesting to include in the current solution of the MP. Based on the classification proposed by Crowe et al. [66], in the context of our research, these links should include all vertical constraints since they include information such as demand coverage and period capacity which dictate where shifts should be added or subtracted in the schedule in order to improve solution quality. Meanwhile, horizontal constraints on sets of lines should be implemented only in the MP, as it is the only problem where more than one individual schedule is involved at once. Finally, constraints on sets of work shifts and on individual work shifts will be considered only in the SP as these are the only problems where decisions are taken regarding the location of shifts within individual schedules.

The core of the algorithm manages interactions between the two solution approaches respectively for the SP and MP ; the SP algorithm in section 5.3.3 and the MP algorithm will be discussed in section 5.3.4. The structure of the core algorithm is succinctly presented in figure 5.1.

The first step executed by the algorithm is to execute the solution approach of the SP in order to obtain a number of tours equal to the number of user-defined tour types. Basically, for each global iteration the SP algorithm is executed a number of times equal to the number of tour types. A tour type is defined by the com-

```
Initialize the tour pool
Do until the MP solution cannot be improved
    For each tour type :
          Update the tour pool ← Solve the subproblem
    Move to the next tour type
    Solve the master problem
Repeat
```

FIGURE 5.1 – Core Algorithm of the Solution Approach

bination of a single duration for work shifts (the number of periods between the beginning and the end of a work shift) and of a number of work shifts to schedule in the tour. This approach by tour type was a very efficient way to provide the flexibility needed to deal with the problems tackled by Crowe et al. (2011) [67]. Indeed, these problems included large numbers of possible start times for work shifts, meaning that solution approaches where all possible work shifts are predefined were inadequate and fastidious to use. However, this tour type approach has a limitation in such contexts : only one shift duration can be used in a tour. While in our experience this suits most real-life health care applications, it does exclude a small number of nurse scheduling ones.

Thankfully, nurse scheduling problems have much simpler work shift structures than problems tackled by Crowe et al. (2011) [67]. Furthermore, the notions of shifts and periods are typically not distinct in nurse scheduling. Hence, by modeling a planning horizon where a time period always corresponds to a shift, all work shift durations can be equal to one period even though some shift durations may differ in reality. Hence, this modeling approach eliminates the main limitation of the tour type approach proposed by Crowe et al. (2011) [67] by allowing shifts of different durations to be assigned to the same tour.

In the CHAIR algorithm structure, each SP execution produces a single tour of a fixed tour type (for the reasons discussed in the previous paragraph, a tour type in this paper is defined only by the number of shifts worked on the planning horizon). Other tours type will be produced one by one during the following iterations, until one tour of each type has been produced. Afterwards, these new tours are all inserted in the tour pool of the MP. The tour pool includes all tours inserted in the current solution as well as all other tours that have previously been produced by solving the SP. Once an iteration of the SP has been executed for each tour type, the MP is solved in order to improve the current schedule, and this process is repeated until solving the SP does not generate any new tours which can improve the current MP solution. When this happens, the best known solution becomes the final solution and the solution process is terminated.

5.3.3 Subproblem

As initially proposed by Crowe et al. (2011) [67], the SP was solved by a greedy heuristic which sequentially assigns shift to the best improving period of the planning horizon. As a robust post-optimization procedure, discussed in section 5.3.5, has been added, this greedy procedure has been replaced by a random assignment. While this may seem like a downgrade, preliminary testing on real-life instances indicated that this actually improved solution quality when the algorithm was allowed to run for a large number of iterations. Clearly, this can only be explained by the increased diversification provided by randomness. This new simple mechanism is as flexible as the previous one in terms of flexibility. Furthermore, as it was the case with the previous mechanism, all SP constraints are soft except the number of shifts to schedule on a tour. Hence, the objective function is to minimize the sum of penalties from constraint violations.

In this model which returns a single tour as a solution by minimizing the sum of violation penalties from constraints of the vertical type $V(X, Z)$, of the horizontal on sets of shifts type $H_2(X, Z)$, and on the horizontal on a single shift type $H_3(X, Z)$, $X_j = 1$ if a work shift is assigned at period j, 0 otherwise ; n is the number of work shifts to assign ; s_l is the service covering value of a shift for period l ; Z_j is the coverage provided at period j ; J is the set of periods of the planning horizon ; and L is the set of possible periods for the duration of a work shift (always 1 for nurse scheduling problems, as mentioned in section 5.3.2).

$$\min V(X,Z) + H_2(X,Z) + H_3(X,Z) \tag{5.1}$$

Subject to :

$$\sum_j X_j = n \tag{5.2}$$

$$Z_{(j+l)} = s_l X_j \qquad \forall\, j \in J, \forall\, l \in L \tag{5.3}$$

$$\sum_{j=j}^{j+p} X_j \leq 1 \qquad \forall\, j \in J \tag{5.4}$$

$$C_V \tag{5.5}$$

$$C_{H_2} \tag{5.6}$$

$$C_{H_3} \tag{5.7}$$

$$X_j, Z_j \in \{0,1\} \qquad \forall\, i \in I,\ k \in K \tag{5.8}$$

In this flexible approach, tour characteristics such as the number of work days, the duration of shifts, the duration of breaks and so on are user defined. However, as periods represent shifts with this modeling approach, breaks will not be considered. A user may define as many types of tours as he wishes, even though a SP solution always provides only one tour of a single user-defined type. As explained in section 5.3.2, the SP is sequentially solved a number of times equal to the number of tour types defined in the current instance. This allows the algorithm to produce one tour of each type during each global iteration. As solving the SP always produces only one tour of a predetermined type, all tour types indexes have been removed from the model. Furthermore, all constraints and related penalties regarding sets of tours (H_1) have been removed as well. Indeed, considerations regarding the total number of tours are irrelevant as the SP always produces one tour of a predetermined type. Hence, the sets of constraints left to consider are :

the vertical sets of linking constraints $V(X, Z)$, the horizontal constraints on sets of shifts $H_2(X, Z)$, and the horizontal constraints on single shifts $H_3(X, Z)$. Finally, both employees and tour types indexes have also been removed because decisions taken for an instance of the SP do not concern more than one employee and one tour type. In the end, all these elements contribute to substantially reduce the computing effort required to produce tours when compared to what it would have been in a single full nurse scheduling problem. Meanwhile, crucial informations related to the quality of a tour both in terms of horizontal and vertical constraints remain.

5.3.4 Master Problem

The MP is solved by including into or excluding from the solution tours priorly obtained when solving the SP. Decisions taken in the MP are strictly limited to inclusions and exclusions of tours from the schedule, effectively limiting the size of the problem. The solution approach for the MP is designed to be provided by the SP with one tour of each type for each global iteration. We have structured it in this fashion in order to increase the diversity of available tours and ensure the approach does not initially favor a tour type with a larger number of service hours when building the initial solution. While the number of employees to schedule is predetermined in the tests results presented in this paper (CHAIR has dealt with soft constraints on the number of employees as presented by Crowe et al. (2011) [67]), all other MP constraints are soft in order to ensure flexibility. For this reason, the MP objective function is to minimize the sum of all penalties from constraint violations.

In the approach we propose, the set of tours provided by the SP is added to a list which includes all tours previously provided : we will refer to this list as the *tour pool*. In order to select which tours to include in the current solution, two local search structures are executed in loops until no improvement to the current solution can be found. At this point, the algorithm returns to the SP in order to generate tours which could potentially improve the solution at the next MP iteration. The neighborhood of the first local search is defined as any move involving one inclusion or one exclusion of a tour from the solution. The neighborhood of the second local search is defined as any move involving the swap of a tour in the current solution with an inactive tour of the tour pool. In our opinion, these flexible local search structures provide the algorithm with the capacity to deal with almost any nurse scheduling problem.

The complexity of this model for selecting tours to include in the schedule is dramatically reduced compared to what it would have been in a single full nurse scheduling problem. Indeed, as no decisions are taken regarding the internal structure of tours, work shift variables have been removed from the MP model. Furthermore, as no decisions regarding tour types must be made, all corresponding indexes can be excluded. Even though vertical constraints (among others) require information about periods, these period indexes may still be removed and considered implicitly in the cost functions they impact. Finally, while cost informations about constraints on sets of shifts and on single shifts are required to compute

$$\min V(Y) + H_1(Y) + H_2(Y) + H_3(Y) \tag{5.9}$$

Subject to :

$$Y_i \leq 1 \qquad \forall\, i \in I \tag{5.10}$$

$$C_V \tag{5.11}$$

$$C_{H_1} \tag{5.12}$$

$$Y_i \in \{0,1\} \qquad \forall\, i \in I \tag{5.13}$$

correctly the overall cost of a solution, all constraints linked to those functions may be removed as no decisions regarding them will impact the structure of individual tours. The remaining information left necessary in the model outside of the objective function is the decision regarding the inclusion (1) or exclusion (0) of a tour Y_i, vertical linking constraints C_V and horizontal constraints on sets of tours C_{H_1}.

5.3.5 Extensions

While the structure presented previously is a solid flexible base in our opinion, heuristics typically have limitations in terms of performance consistency between instances. Hence, in order to make our flexible and portable approach more robust, a number of extensions proposed by Crowe et al. (2011) [67] have been kept : a post optimization procedure for the SP, an extended version of the SP, a shuffling mechanism, and a limited tour pool.

The post optimization procedure is a variable neighborhood search without any restart mechanism. As initially proposed by Crowe et al. (2011) [67], it includes two local search structures : the neighborhood of the first one is defined as any possible swaps between 2 days of the schedule, while the second one is defined as any possible deletion or insertion of a shift.

An extended version of the SP has also been included for the following reason : the initial subproblem cost mechanism assumed that a tour would be added to the current MP solution. This assumption holds when the algorithm is building an initial solution. However, when a full initial solution has already been built, it is more likely that a tour will be swapped instead. Hence, the extended version of the subproblem also evaluates the cost impact for each possible swaps between the current SP tour and any given tour currently included in the MP solution.

The shuffling mechanism allows the global algorithm to restart from an empty solution with a tour pool containing tours from past iterations. It is activated when the SP is not able to provide new tours to improve the current MP solution anymore. This mechanism allows robust diversification, and makes the logic of the global solution approach very similar to a variable neighborhood search.

Finally, the limited tour pool is a mechanism which deletes tours which have not been included in the MP solution for a long time. This mechanism allows intensification while somewhat reducing the computing effort.

5.3.6 CHAIR Improvements for Nurse Scheduling

Crowe et al. (2011) [67] tested CHAIR on a set of instances where individual assignment of tours was not considered. The assumption was that all personnel were sharing the same work rules and had no individual preferences for work shifts and days off. This assumption is reasonable in many health care applications, but not in nurse scheduling where individual preferences and skills must frequently be taken into account. Hence, although no modifications have been made to the core structure of the algorithm proposed by Crowe et al. (2011) [67], we have improved the approach by providing it with the capacity to assign tours to individuals. Once this improvement has been made, dealing with preferences was simple as they can be modeled as constraints on sets of shifts or on individual shifts from the classification proposed by Crowe et al. (2011) [66]. Meanwhile, the same reasoning applies to skills which can be modeled as vertical constraints related to demand coverage.

The other important limitation of CHAIR as proposed by Crowe et al. (2011) [67] is that all durations of shifts are assumed to be identical for a single tour. This limitation has been explained in detail in section 5.3.2. As explained then, by adopting a modeling approach where shifts and periods are the same, work shifts of different durations may now be assigned on the same tour without modifying the core solution approach.

Furthermore, during preliminary testings we have noticed that CHAIR was sometimes struggling to produce good tours when constraints with quadratic penalty costs were used. These types of constraints had the impact of limiting the capacity of the subproblem local search mechanisms to explore solution space thoroughly. In order to deal with this, we have added a diversification mechanism to the SP which temporally lowers all violation costs for constraints on tours and increases the weight of vertical constraints.

Finally, to improve the diversification capacities of the MP, we have implemented a weight adjustment mechanism. This mechanisms is activated for a small number of iterations once the MP has found a solution which its local search procedures can not improve anymore. At that point, the mechanism lowers the violation cost of the most violated constraint and increases the cost of the least violated one. This mechanism is only activated for a small number of iterations to ensure the quality of the current solution is not strongly reduced.

5.4 Results

Crowe et al. (2011) [67] have tested the CHAIR solution approach on an experimental framework consisting of instances with different combinations of constraints and service demands from health care applications in general. In this paper, we first test the CHAIR solution approach on an experimental framework specific to nurse scheduling. Afterwards, we benchmark it on a set of problems from the nurse scheduling literature. All programs were coded in C++. All experiments were carried out on an Intel®Core$^{\text{TM}}$i3-2350M CPU running at 2.30 GHz with 6 Gb of RAM.

5.4.1 Nurse Scheduling Experimental Framework

The objective of this experimental framework is twofold : firstly to test CHAIR on a combination of important characteristics specific to nurse scheduling, and then to evaluate their impact on the cost of schedules. The three characteristics tested are contracts, preferences and skills.

We define contracts as characteristics dictating the availability of personnel on a stable (or long term) basis. In this paper, we consider full time regular personnel, part time regular personnel, full time temporary personnel and part time temporary personnel. A full timer works 20 shifts on a horizon of 28 days (an average of 5 per week), while a part timer works 10 shifts on 28 days. A regular employee always works on a single predetermined shift type (day, evening or night) with no rotations, while a temporary employee may work on any shift type, and may work on different shift types within a single schedule. The six combinations of contract types tested in the framework are presented on table 5.I.

Preferences include characteristics dictating the availability of personnel on a varying (short term) basis. They are days off requests which may correspond to a special request (a single day) or a vacation (a sequence of days). They can either be soft constraints or hard constraints. Sequences are only modeled as hard constraints since vacations must not be violated once they have been granted by

Contracts	Permanent full timers	Permanent part timers	Temporary full timers	Temporary part timers
1	No	No	Yes	No
2	No	No	Yes	Yes
3	Yes	No	No	No
4	Yes	No	Yes	No
5	Yes	Yes	No	No
6	Yes	Yes	Yes	No

Table 5.I – The Six Contract Type Combinations of the Experimental Framework

the chief nurse. In all instances with preferences, a number of days off equal to 5% of the sum of all work shifts has been allocated to employees. The four combinations of preference types tested in the framework are presented in Table 5.II.

Finally, skills determine the abilities of each nurse to execute tasks or take charge of responsibilities. Multiple skills describe contexts where each nurse may have a unique combination of skills, while hierarchic skills describe problems where higher ups have all skills of underlings and more. There are three different skills a nurse can have in the multiple skills instances, and three levels in the hierarchic skills instances (assistant nurse, full nurse, coordinator nurse). As skills are always related to service demand, skill sets have been coupled to two different demand structures : a uniform one and a varying one where demand is reduced on evening shifts, night shifts and week-end shifts. The six combinations of skill types tested in the framework are presented in Table 5.III.

Preferences	Single, Soft	Single, Hard	Sequence, Hard
1	No	No	No
2	Yes	No	No
3	No	Yes	No
4	No	No	Yes

Table 5.II – The Four Preference Type Combinations for the Experimental Framework

Set	Skills		Service Demand	
	Multiple	Hierarchic	Uniform	Varying
1	No	No	Yes	No
2	No	No	No	Yes
3	Yes	No	Yes	No
4	Yes	No	No	Yes
5	No	Yes	Yes	No
6	No	Yes	No	Yes

Table 5.III – The Six Skill Type Combinations for the Experimental Framework

Besides contracts, skills and preferences, all instances of the framework share the same following characteristics. The planning horizon is 28 days. On each day, a nurse can be assigned to a day shift, an evening shift, a night shift or a day off (no assignment). Employees must have at least one full week-end off, must not work more than 6 consecutive work shifts, must not have any isolated work days or days off, and must have a clockwise forward rotation of work shifts (for temporary employees as their contracts allow rotation). If all nurses are full timers, there is a total of 21 nurses to schedule, and when there are part timers, a total of 28 must be scheduled with 14 part timers. Most constraints are soft, and each violation is penalized by 5 points. Exceptions include service demand where violations are penalized by 1 point each, and hard preferences for which violations are penalized by 500 points. The number of work shift per tour (20 for full timers and 10 for part timers) is modeled as a hard constraint. When permanent nurses are scheduled, the fixed shift type of their contracts are allocated proportionally to the demand of each shift.

Skills, (S)	CPU (s)							Solution values						
Pref (P)	C1	C2	C3	C4	C5	C6	Mean	C1	C2	C3	C4	C5	C6	Mean
S1, P1	226	313	201	236	257	318	258.50	14	2	10	14	8	3	8.50
S1, P2	220	340	248	380	434	407	338.17	36	16	36	37	23	18	27.67
S1, P3	284	355	200	272	345	332	298.00	47	21	45	45	26	24	34.67
S1, P4	276	406	203	243	375	505	334.67	55	33	53	55	38	33	44.50
S1 mean	251.50	353.50	213.00	282.75	352.75	390.50	307.33	38.00	18.00	36.00	37.75	23.75	19.50	28.83
S2, P1	274	301	183	253	362	342	285.83	56	56	56	56	56	56	56.00
S2, P2	238	452	187	250	279	251	276.17	80	71	81	81	71	71	75.83
S2, P3	305	313	141	252	338	359	284.67	91	70	91	91	76	76	82.50
S2, P4	266	354	200	237	285	346	281.33	96	86	97	96	85	83	90.50
S2 mean	270.75	355.00	177.75	248.00	316.00	324.50	282.00	80.75	70.75	81.25	81.00	72.00	71.50	76.21
S3, P1	384	278	336	358	419	490	377.50	14	16	56	20	50	20	29.33
S3, P2	422	536	288	379	382	753	460.00	34	31	82	51	60	34	48.67
S3, P3	315	571	292	542	460	607	464.50	43	36	92	57	70	40	56.33
S3, P4	434	429	239	430	286	433	375.17	51	46	97	62	80	48	64.00
S3 mean	388.75	453.50	288.75	427.25	386.75	570.75	419.29	35.50	32.25	81.75	47.50	65.00	35.50	49.58
S4, P1	260	170	358	221	424	862	382.50	0	0	36	8	20	1	10.83
S4, P2	347	409	258	522	494	557	431.17	25	15	48	27	26	15	26.00
S4, P3	309	565	229	206	494	505	384.67	35	20	71	45	40	20	38.50
S4, P4	227	356	304	294	390	576	357.83	41	30	77	46	52	30	46.00
S4 mean	285.75	375.00	287.25	310.75	450.50	625.00	389.04	25.25	16.25	58.00	31.50	34.50	16.50	30.33
S5, P1	347	372	410	406	409	898	473.67	14	7	93	41	103	42	50.00
S5, P2	578	531	463	459	552	634	536.17	40	18	123	68	117	58	70.67
S5, P3	356	602	313	325	416	570	430.33	48	21	131	78	121	63	77.00
S5, P4	459	408	302	328	563	799	476.50	54	32	137	84	132	72	85.17
S5 mean	435.00	478.25	372.00	379.50	485.00	725.25	479.17	39.00	19.50	121.00	67.75	118.25	58.75	70.71
S6, P1	561	703	282	509	395	640	515.00	8	5	52	43	63	50	36.83
S6, P2	547	676	292	378	790	690	562.17	31	18	80	69	78	65	56.83
S6, P3	358	630	307	394	727	924	556.67	44	22	91	79	83	70	64.83
S6, P4	590	646	274	373	422	813	519.67	50	34	98	85	97	80	74.00
S6 mean	514.00	663.75	288.75	413.50	583.50	766.75	538.38	33.25	19.75	80.25	69.00	80.25	66.25	58.13
P1 mean	342.00	356.17	295.00	330.50	377.67	591.67	382.17	17.67	14.33	50.50	30.33	50.00	28.67	31.92
P2 mean	392.00	490.67	289.33	394.67	488.50	548.67	433.97	41.00	28.17	75.00	55.50	62.50	43.50	50.94
P3 mean	321.17	506.00	247.00	331.83	463.33	549.50	403.14	51.33	31.67	86.83	65.83	69.33	48.83	58.97
P4 mean	375.33	433.17	253.67	317.50	386.83	578.67	390.86	57.83	43.50	93.17	71.33	80.67	57.67	67.36
Mean	357.63	446.50	271.25	343.63	429.08	567.13	402.53	41.96	29.42	76.38	55.75	65.63	44.67	52.30

TABLE 5.IV – Nurse Scheduling Experimental Framework Best Solutions Obtained with CHAIR

Skills, (S)	CPU (s)						Solution values					
Pref (P)	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6
S1, P1	268.80	397.20	279.40	331.00	417.80	469.40	15.20	3.00	10.60	15.40	9.20	5.00
S1, P2	372.20	543.00	403.00	423.80	644.80	547.00	38.80	19.00	36.80	38.60	23.80	19.60
S1, P3	400.00	488.40	353.00	372.80	424.00	424.20	51.00	24.20	46.80	48.20	28.40	26.60
S1, P4	349.40	554.40	280.00	349.40	549.40	779.80	57.40	34.20	54.60	55.60	39.80	33.60
S2, P1	320.00	377.00	274.60	311.20	494.40	478.20	56.80	56.00	56.60	56.20	56.00	56.00
S2, P2	427.80	531.60	281.80	287.00	426.20	457.20	81.80	71.20	81.20	81.00	71.40	72.00
S2, P3	372.20	548.00	234.60	347.40	409.80	413.80	91.60	74.80	94.80	91.00	78.40	76.00
S2, P4	313.60	474.20	262.20	288.20	402.00	405.00	97.80	86.00	97.80	96.40	87.00	85.40
S3, P1	544.20	539.00	446.60	548.00	547.20	664.00	17.20	19.80	57.40	23.20	51.00	22.20
S3, P2	606.40	630.40	508.80	653.40	484.40	987.20	36.20	31.60	83.20	52.60	65.40	35.80
S3, P3	704.00	701.60	415.00	666.80	687.20	867.80	46.20	37.00	92.80	60.00	70.00	41.20
S3, P4	648.60	546.60	280.60	605.20	712.60	911.40	54.20	46.60	97.20	67.20	80.40	51.00
S4, P1	406.40	221.20	443.00	535.00	595.20	1 055.20	0.40	0.00	36.20	9.60	20.80	2.60
S4, P2	474.40	678.60	376.00	673.80	811.60	771.00	25.00	16.00	59.60	33.60	35.20	18.20
S4, P3	427.80	640.40	360.40	401.00	975.20	787.40	36.00	20.00	72.00	47.20	41.20	24.20
S4, P4	566.80	628.40	353.60	483.40	657.20	815.60	41.20	30.00	77.80	50.80	53.00	32.40
S5, P1	447.00	574.20	601.60	798.20	533.80	1 213.80	22.40	9.00	96.40	43.60	103.80	44.40
S5, P2	686.20	862.60	588.20	595.60	688.20	1 066.20	43.60	20.60	124.80	71.80	118.20	62.00
S5, P3	626.40	764.80	374.20	506.00	530.00	803.40	50.00	25.60	134.40	80.20	122.60	65.40
S5, P4	776.00	960.40	515.80	678.40	749.00	1 280.40	57.40	36.20	140.00	87.40	133.20	75.40
S6, P1	747.40	998.20	467.60	757.00	620.80	1 158.20	10.00	5.80	53.60	44.60	64.00	51.40
S6, P2	773.60	855.20	427.40	529.80	922.40	1 100.80	36.00	21.20	80.40	76.40	78.80	67.40
S6, P3	716.40	1 315.60	412.00	588.20	1 138.20	1 089.20	63.80	26.40	92.60	80.60	83.40	71.20
S6, P4	848.00	994.00	456.60	605.40	799.00	894.20	54.80	39.60	99.20	88.20	98.40	81.80

TABLE 5.V – Nurse Scheduling Experimental Framework Average Solution Time and Value Obtained with CHAIR

Skills, (S)	CPU (s)						Solution values					
Pref (P)	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6
S1, P1	32.02	70.09	60.91	78.83	112.88	138.24	0.84	1.73	0.89	1.67	0.84	1.87
S1, P2	105.20	118.97	91.05	61.82	146.12	135.04	1.79	2.74	1.79	1.52	1.30	1.52
S1, P3	108.60	82.28	99.30	89.20	65.61	54.09	3.39	2.17	1.92	2.28	1.67	2.70
S1, P4	78.78	91.00	78.38	88.36	182.71	231.83	2.30	1.64	1.14	0.89	1.30	0.55
S2, P1	42.99	42.99	83.00	49.99	143.13	98.06	0.84	0.00	0.89	0.45	0.00	0.00
S2, P2	115.71	63.18	90.99	54.16	123.49	160.43	1.30	0.45	0.45	0.00	0.89	2.24
S2, P3	40.75	161.24	89.15	65.84	64.07	56.96	0.89	2.68	2.68	0.00	2.88	0.00
S2, P4	44.98	76.18	65.23	32.75	90.47	61.29	1.48	0.00	0.84	0.55	1.22	1.34
S3, P1	133.17	212.82	92.41	157.11	93.38	164.85	2.86	7.40	1.14	2.86	1.00	1.79
S3, P2	172.82	86.04	211.94	279.56	81.91	325.00	1.92	1.34	2.68	1.14	3.91	1.79
S3, P3	280.67	120.96	125.59	99.83	322.42	255.28	3.96	0.71	0.84	2.24	0.00	1.10
S3, P4	193.52	103.49	40.43	212.86	251.97	376.68	2.95	0.55	0.45	3.90	0.89	2.35
S4, P1	103.39	73.65	82.44	193.82	140.82	247.42	0.55	0.00	0.45	3.05	0.45	1.14
S4, P2	100.44	226.44	100.42	147.99	389.90	288.03	0.00	2.24	6.50	4.10	5.26	3.96
S4, P3	108.93	49.85	92.13	150.41	505.10	319.91	1.73	0.00	0.71	2.28	1.10	3.35
S4, P4	264.82	185.37	52.20	133.28	154.45	160.84	0.45	0.00	0.45	3.27	0.71	1.52
S5, P1	111.64	178.33	231.74	262.70	106.91	311.05	5.22	1.58	3.13	2.97	0.84	2.07
S5, P2	104.10	253.76	113.00	179.41	165.65	283.38	2.30	2.07	1.10	2.77	0.84	4.36
S5, P3	190.90	187.04	83.66	170.19	83.29	164.83	1.58	5.22	2.30	1.48	1.52	1.95
S5, P4	203.55	544.94	176.59	228.81	151.95	431.55	2.88	4.71	2.24	2.30	1.10	3.78
S6, P1	230.31	300.76	156.74	294.04	233.44	348.63	1.58	1.30	1.14	1.95	1.41	0.89
S6, P2	247.63	219.24	125.62	131.63	102.10	349.34	3.87	3.03	0.89	7.70	0.84	2.79
S6, P3	370.02	647.22	86.52	168.16	456.76	150.85	18.61	4.39	1.52	2.07	0.55	1.30
S6, P4	210.73	352.80	127.43	231.33	403.80	59.03	3.83	6.58	0.84	3.96	1.52	1.79

TABLE 5.VI – Nurse Scheduling Experimental Framework Standard Deviation for Solution Times and Values Obtained with CHAIR

Skills, (S)	CPU (s)						Solution values					
Pref (P)	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6
S1, P1	11.91%	17.65%	21.80%	23.81%	27.02%	29.45%	5.50%	57.74%	8.44%	10.87%	9.09%	37.42%
S1, P2	28.26%	21.91%	22.59%	14.59%	22.66%	24.69%	4.61%	14.41%	4.86%	3.93%	5.48%	7.74%
S1, P3	27.15%	16.85%	28.13%	23.93%	15.47%	12.75%	6.65%	8.96%	4.11%	4.73%	5.89%	10.16%
S1, P4	22.55%	16.41%	27.99%	25.29%	33.26%	29.73%	4.01%	4.80%	2.09%	1.61%	3.28%	1.63%
S2, P1	13.44%	11.40%	30.23%	16.06%	28.95%	20.51%	1.47%	0.00%	1.58%	0.80%	0.00%	0.00%
S2, P2	27.05%	11.89%	32.29%	18.87%	28.97%	35.09%	1.59%	0.63%	0.55%	0.00%	1.25%	3.11%
S2, P3	10.95%	29.42%	38.00%	18.95%	15.63%	13.77%	0.98%	3.59%	2.83%	0.00%	3.67%	0.00%
S2, P4	14.34%	16.06%	24.88%	11.36%	22.50%	15.13%	1.52%	0.00%	0.86%	0.57%	1.41%	1.57%
S3, P1	24.47%	39.48%	20.69%	28.67%	17.07%	24.83%	16.65%	37.35%	1.99%	12.34%	1.96%	8.06%
S3, P2	28.50%	13.65%	41.65%	42.79%	16.91%	32.92%	5.31%	4.25%	3.23%	2.17%	5.98%	5.00%
S3, P3	39.87%	17.24%	30.26%	14.97%	46.92%	29.42%	8.58%	1.91%	0.90%	3.73%	0.00%	2.66%
S3, P4	29.84%	18.93%	14.41%	35.17%	35.36%	41.33%	5.44%	1.18%	0.46%	5.80%	1.11%	4.60%
S4, P1	25.44%	33.30%	18.61%	36.23%	23.66%	23.45%	136.93%	-	1.24%	31.77%	2.15%	43.85%
S4, P2	21.17%	33.37%	26.71%	21.96%	48.04%	37.36%	0.00%	13.98%	10.91%	12.20%	14.95%	21.77%
S4, P3	25.46%	7.78%	25.56%	37.51%	51.79%	40.63%	4.81%	0.00%	0.98%	4.83%	2.66%	13.83%
S4, P4	46.72%	29.50%	14.76%	27.57%	23.50%	19.72%	1.09%	0.00%	0.57%	6.44%	1.33%	4.68%
S5, P1	24.98%	31.06%	38.52%	32.91%	20.03%	25.63%	23.33%	17.57%	3.25%	6.80%	0.81%	4.67%
S5, P2	15.17%	29.42%	19.21%	30.12%	24.07%	26.58%	5.28%	10.07%	0.88%	3.86%	0.71%	7.03%
S5, P3	30.48%	24.46%	22.36%	33.63%	15.71%	20.52%	3.16%	20.41%	1.71%	1.85%	1.24%	2.98%
S5, P4	26.23%	56.74%	34.24%	33.73%	20.29%	33.70%	5.02%	13.02%	1.60%	2.63%	0.82%	5.02%
S6, P1	30.82%	30.13%	33.52%	38.84%	37.60%	30.10%	15.81%	22.48%	2.13%	4.37%	2.21%	1.74%
S6, P2	32.01%	25.64%	29.39%	24.85%	11.07%	31.73%	10.76%	14.31%	1.11%	10.08%	1.06%	4.14%
S6, P3	51.65%	49.20%	21.00%	28.59%	40.13%	13.85%	29.16%	16.64%	1.64%	2.57%	0.66%	1.83%
S6, P4	24.85%	35.49%	27.91%	38.21%	50.54%	6.60%	7.00%	16.62%	0.84%	4.49%	1.54%	2.19%

TABLE 5.VII – Nurse Scheduling Experimental Framework Coefficient of Variation (standard deviation divided by average) for Solution Times and Values Obtained with CHAIR

For each instance of the experimental framework, a total of five runs with five different random seeds were made. Table 5.IV presents the value of the best solution with its time for each instance, while table 5.V presents the average solution values and times, table 5.VI presents the standard deviations and table 5.VII presents the coefficients of variation (standard deviation divided by average) as a percentage.

On table 5.IV, the best solution value and the corresponding solution time are presented for each instance. On this table, each line is a combination of a skill set (S#) and a preference set (P#), while each column is a contract set (C#). The last column of each half table contains the average value of the half line, and the last line of each column contains the average of its column. Tables 5.V through 5.VII have the same structure but no lines or columns for averages.

As expected from table 5.IV, on average solution values for preference set 1 are smaller than for set 2, set 2 smaller than set 3, and set 3 smaller than set 4. Of course, instances without preferences resulted in a reduced number of soft constraint violations. Furthermore, dealing with hard preferences (3 and 4) turned out to increase the total number of soft constraint violations. Likewise, respecting the days off sequences (4) was more costly than the isolated days off requests (2 and 3). The fact that standard deviations for solution values is similar for each set of preferences suggests that each turned out to be impacted in a similar way by combinations with skills and contracts. Finally, preferences do not appear to have an important impact on solution times.

Results of solution values for skills are more difficult to interpret as there was no significance in creating a relationship between a type of skill and a level of tightness in the problem. The same can be said about the service demand structures, where the difficulty of covering week-ends on the uniform demand was counterbalanced by the difficulty of covering day shifts on weekdays in the varying demand structure (both demand structures require the same total number of nurse-shifts for each type of skill). Of course, the distribution of fixed work shift types by contracts for permanent workers also had an impact on the coverage. Actually, the questions for which we wanted answers by including skills were twofold : first, how well can the algorithm deal with different types of skills, and then what is the impact of dealing with different types of skills on computing workloads (solution times). Clearly, the results obtained show that the algorithm can deal with multiple and hierarchic of skills, as well as provide very reasonable solutions. Regarding solution times, the computing effort was reduced without skills, which was expected. Indeed, skills require to deal with supplementary sets of service demand constraints, which increase the time required to execute a similar number of iterations. However, it is interesting to notice that hierarchic skills turned out to require more solution time in average than multiple skills. This might be explained by the fact that some hierarchic skills were possessed by a limited number of coordinator nurses, hence making the problem tighter and the scheduling tasks more complex to solve.

Finally, as expected regarding contracts, instances with contract set #2 provided the best solution values on average, while those with contract set #3 corresponded to solutions with the largest number of violations. From lowest to highest solution cost, the order is as follows : 2, 1, 6, 4, 5, 3. Surprisingly, the average for 1 and 6 is very similar, meaning that a mix of permanent part timers and temporary full timers can offset the cost impact of permanent full timers on demand coverage quality. This is a very interesting finding as it means that organizations might manage to provide some better working conditions to a limited group of employees without much impact on total cost. Other results were as expected, and we can deduce that the cost of contracts from lowest to highest is as follows : temporary part timers, temporary full timers, regular part timers and regular full timers. In terms of solution times, 1, 2, 4 and 5 were comparable, while 3 was shorter and 6 longer. This might be explained by the fact that there is a more limited solution space to explore with contract set #3 (rigid regular full timers only) compared to contract 6 (both flexible and rigid contracts, both part time and full time).

Overall, solution quality was good in our opinion. The resulting schedules usually had a limited number of violations and seemed adequate for implementation. Clearly, while solution times remain reasonable, they are high for a heuristic solving nurse scheduling problems. This is explained by the fact that the algorithm was built to deal with much more complex problem structures for periods and shifts, requiring the implementation of mechanisms which are not the most time efficient to deal with nurse scheduling. Likewise, the flexible local search mechanisms implemented allow the algorithm to execute adequate local descents in any nurse scheduling problems, but these descent mechanisms can not compete in time efficiency with those crafted for a specific nurse scheduling environment. In other words, longer solution times are the price to pay for saving the time and effort required to build a new solution approach for every new problem to solve. In this case, we believe the benefits to largely outweighs the costs.

Finally, tables 5.VI and 5.VII show that CHAIR is quite robust in terms of solution values when launched with different random seeds. In fact, table 5.VI shows that the average standard deviation is around two. While some coefficients of variation may appear high on instances with low average solution values, the only large penalty points variation found between different random seeds is for the S6-P3-C1 instance (standard deviation of 18.61 with a coefficient of variation of 29.16% for solution values). A few other noticeable variations are found on the following instances : S3-P1-C2, S4-P2-C3, S5-P1-C1, S5-P3-C2, S6-P2-C4 and S6-C4-P2. Standard deviations for all other instances are smaller than 5 points, which is very satisfactory in this context of constraint satisfaction. On the other hand, solution times vary significantly between random seeds, but that is to be expected since CHAIR implements a shuffling mechanism and a stop criteria based on the number of iterations without improvements.

5.4.2 Literature and Practice Benchmark

The series of tests executed in the experimental framework presented in the previous section provide an insight on the impact of different decisions on the performance of the solution approach. However, this framework leaves two questions unanswered : how does CHAIR perform on real life instances, and how does CHAIR perform in comparison with other solution approaches. To answer these questions, we have assembled a set of 10 problems both from the literature and practice in nurse scheduling. These problems, their best known solutions and the solution approaches used to obtain them are all available on the employee scheduling benchmark data sets of the ASAP research group [68].

In table 5.VIII, the Azaiez problem [14] is about scheduling 13 employees on a 4-week planning horizon. For any given day of the horizon, the possible assignments are a day shift, a night shift or a day off. The problem includes multiple skills (there are 2, and an employee can have both) and demand is a hard constraint with a minimum level. Other constraints include minimum and maximum numbers of days off for each employee, a maximum number of days on, limits on the number of shifts of each type, a maximum number of consecutive work days, no isolated days off or days of work, and a limit on the number of days worked during week-ends. The objective is to minimize the sum of penalties for constraint

Problem	CPU (s)		Solution values	
	CHAIR	Z*	CHAIR	Z*
Azaiez [14]	109	600	0	0
Gpost [140]	723	-	15	5
Ikegami-2-shift [102]	1207	13	10	0
LLR [117]	296	10	371	301
Millar 2 shift [129]	2	1	0	0
MUSA [133]	1	-	175	175
ORTEC01 [58]	2299	105	1132	270
Ozkarahan [139]	1	-	0	0
SINTEF [146]	78	-	0	0
WHPP [168]	1302	-	16	5

TABLE 5.VIII – Nurse Scheduling Experimental Framework Results Obtained with CHAIR

violations.

The Gpost problem [140] is about scheduling 8 employees on a four week long planning horizon. A day shift, a night shift and a day off can be scheduled every day. Demand is a hard target constraint with upper and lower bounds equal for each period. Other constraints include limits on the number of shifts worked, on the number of consecutive work shifts, on the number of days off, forbidding broken week-ends, limits on the number of week-ends worked, bounds on the number of consecutive days off, and workload balance between the different weeks of the planning horizon. The objective function is to minimize the sum of penalties for soft constraint violations.

The Ikegami-2-shift problem [102] involves 28 employees on a planning horizon of 30 days. There is a total of 8 different skills, and three possible shifts assignments for every day plus days offs ; Demand is a target soft constraint. Other constraints include limits on the number of work shifts per week, limits on the number of worked week-ends, limits on the number of shifts assigned of each type, and limits on the number of consecutive work shifts. The problem also includes some personal assignment requests. The objective function is to minimize the sum of penalties for soft constraint violations.

The LLR problem [117] is about scheduling 27 employees on a one-week horizon. Everyday, three different shifts can be assigned, plus a day off. Demand is a target hard constraint where the upper and lower bounds are equal for each period. Other constraints include a minimum number of night shifts, a maximum number of work shifts, clockwise shift rotation, and a set of forbidden days off sequences. Finally, the problem includes some personal assignment requests. The objective function is to minimize the sum of penalties for soft constraint violations.

The Millar 2 shift problem [129] is about scheduling 8 employees on a 2-week horizon. A day shift and a night shift can be assigned everyday, plus a day off. Demand is a target hard constraint, where the lower and upper bounds are equal. Other constraints include limits on the number of days off, on the number of shifts of each type, on the number of week-ends off, and on the limit of consecutive work shifts or days off. The objective function is to minimize the sum of penalties for soft constraint violations.

The MUSA problem [133] is about scheduling 11 employees on a planning horizon of 2 weeks. There are three different exclusive skills and some individual days off requests must be considered. Demand is a lower bound soft constraint. Other constraints include limits on the number of work shifts, the number of days off, and some forbidden shift sequences. The objective function is to minimize the sum of penalties for soft constraint violations.

The ORTEC01 problem [58] is about scheduling 16 employees on a 31-day planning horizon. It includes shift off requests, there are 4 different work shifts types which can be assigned every day, and days off also. Demand is a target hard constraint where lower and upper bounds are equal for any given period. Other constraints include many different bounds on the number of work shifts of each type, workload balance between weeks, forbidden isolated work days or days off, and forbidden broken week-ends. The objective function is to minimize the sum of penalties for soft constraint violations.

The Ozkarahan problem [139] is about scheduling a total of 14 employees on a 7-day planning horizon. There are 2 possible work shifts which can be assigned every day, plus days off. The problem includes some individual requests on types of shift. There are two different skills, and demand is a lower bound hard constraint. Other constraints include bounds on the number of shifts of each type, and on week-ends off. The objective function is to minimize the sum of penalties from soft constraint violations.

The SINTEF problem [146] is about scheduling 24 employees on a horizon of three weeks. The problem includes individual requests for days off. There is a total of 5 shift types which can be assigned every day, plus days off. Demand is a target hard constraint. Other constraints include limits on consecutive work shifts of each type, and limits on the maximum number of hours worked. The objective function is to minimize the sum of penalties for soft constraint violations.

Finally, the WHPP problem [168] is about scheduling 30 employees on a planning horizon of 2 weeks. There are three possible shifts which can be assigned every day : a day shift, an evening shift and a night shift. Also, a day off can be assigned. Demand is a target soft constraint. Other constraints include bounds on the number of consecutive work days of each type, clockwise rotation of shifts, no isolated work days of days off, no broken week-ends and bounds on the number of week-ends worked. The objective function is to minimize the sum of penalties for soft constraint violations.

For each of the above mentioned problems, Table 5.VIII presents a comparison between CHAIR and the solution approach providing the best known solution. Essentially, we are benchmarking our solution approach with a total of 10 others, each on the single problem where they provide the best known solution.

In such a context, we did not expect our algorithm to outperform the benchmarking set. We did however expect it to provide good solutions for all problems. In this regard, we were very satisfied with the performance of the algorithm, with the exception of the solution obtained on the ORTEC01 instance. For all other instances, CHAIR provided good work schedules, and solution values generally close or equal to the best known.

In fact, in half of the instances, our solution approach has found the optimal solution (Azaiez, Millar 2 shift, MUSA, Ozkarahan and SINTEF). On three other problems, the solutions obtained with CHAIR were close to the optimal ones in terms of number of violations (Gpost, LLR and WHPP). When dealing with the ikegami-2-shift problem, we obtained a solution with 10 points for violations of demand over the optimal solution. However, because of the large number of skills, the large number of employees and the long planning horizon, this actually meant a solution which is very good in terms of demand coverage quality. Hence, in our opinion, our algorithm performed well and achieved its objectives on 9 instances out of 10.

In fact, ORTEC01 was the only problem where we did not find a satisfactory solution. Firstly, it is a problem with a target demand hard constraint where both the lower and upper bounds are equal. From information gathered during tests with the other 9 instances of Table 5.VIII, it is clear that CHAIR's search descent mechanisms are less effective with tight target demand hard constraints. A few reasons can explain this behavior. As it builds schedules in a column generation type of approach, reaching a perfect equilibrium between service demand and offer is challenging when compared with the type of approach which starts from an initial solution built to have the exact right number of employees and shifts already scheduled. Also, our solution approach was initially built with the flexibility required to produce schedules with a large number of employees and overlapping shifts, which are a type of problems where a perfect equality between service demand and offer is generally impossible. Indeed, the use of an initial solution mechanism providing a number of shifts perfectly equal to demand is not appropriate with the objectives of flexibility and portability for which CHAIR was built.

Another challenge with the ORTEC01 problem is the large number of soft constraints with quadratic cost structures. Indeed, we have used the exact same cost structure as proposed on the employee scheduling benchmark data sets page [68], which included a large number of quadratic penalty functions. However, from information gathered with the other 9 problems of Table 5.VIII, it appears that our solution approach is less efficient for solution space exploration when dealing with a heavy quadratic cost structure. Hence, we believe the fact that ORTEC01 implements both tight target demand hard constraints and many quadratic penalty functions explains why our algorithm was not able to produce a satisfactory solution on this difficult problem.

While the solution values obtained with our approach are comparable with the best known on 9 out of 10 problems, the solution times are generally higher on instances where that information was available. This is simply due to the fact that in order to be flexible and portable, the solution approach must sacrifice some time efficiency when dealing with certain problem structures that it was not specifically tailored to take advantage of.

5.5 Conclusion

In this paper, we have improved the CHAIR flexible and portable solution approach. While it was initially built to solve staff scheduling problems of health care applications in general, we have tested it thoroughly on nurse scheduling problems. In order to do so, we have proposed a nurse scheduling experimental framework based on three typical characteristics of these problems : contracts, preferences and skills. The objective of this framework was to understand the impact of each one of these characteristics both on the cost of the solution and on the performance of the algorithm. CHAIR was tested on this framework and provided interesting insights on the impacts and relationships of different scheduling decisions. We also benchmarked CHAIR against a set of 10 solution approaches, each one on a different problem where it had found the best known solution.

Overall, the results obtained were very satisfying and demonstrated that it is possible to obtain good implementable schedules on widely different problems without having to tailor a unique solution approach for each one. We believe this is a very valuable demonstration both for researchers and practitioners, as it could potentially open a whole new area of research on flexibility and portability, and result in large time and financial savings for practitioners. In our opinion, this also creates a great opportunity to increase collaborations between researchers and practitioners.

Chapitre 6

Conclusion

Le contenu d'un ensemble de quatre articles a été présenté dans cette thèse. Premièrement, une revue de littérature a décrit et catégorisé par type d'applications et par méthode de résolution plusieurs recherches publiées à ce jour sur la confection d'horaire de personnel. Deuxièmement, une méthode de résolution flexible et portable en confection d'horaires appliquée au domaine de la santé appelée SOFA (Schedule Optimization with a Flexible Approach) a été proposée et testée sur un cadre expérimental exhaustif. Troisièmement, une méthode de résolution alternative appelée CHAIR (Column generation Heuristic Approach for Indefinite Rostering) a été présentée et comparée à SOFA sur le même cadre expérimental mentionné précédemment. Finalement, une version de CHAIR améliorée afin d'être en mesure de gérer les spécificités des problèmes de confection d'horaires d'infirmières a été proposée et testée sur deux ensembles d'instances : un cadre expérimental exhaustif spécifique à la confection d'horaires d'infirmière, et un ensemble de dix instances réelles pour lesquelles les meilleurs solutions et les meilleurs algorithmes sont documentés sur le site web du groupe de recherche ASAP de l'université de Nottingham [68].

Avec la revue de littérature, le lecteur a pu remarquer l'étendue des recherches publiées en confection d'horaires de personnel. Bien que certaines applications telles la confection d'horaire d'infirmières aient été favorisées jusqu'à aujourd'hui en termes de nombre de publications, l'ensemble des publications en confection d'horaires de personnel couvre un large éventail de méthodes de résolutions et de domaines d'applications. Toutefois, les méthodes de résolutions flexibles et portables pouvant résoudre un grand nombre de problèmes tout en requérant un effort limité de configuration ont fait l'objet de peu de recherches publiées dans la littérature. Le reste de cette thèse a donc porté sur le développement de telles méthodes de résolution appliquées au domaine de la santé. Ce domaine d'application a été choisi pour ses caractéristiques telles que son grand volume de service et sa structure en unités multiples qui en font un terrain fertile pour l'implantation de ce type de méthodes.

La première méthode de résolution flexible et portable appliquée au domaine de la santé que nous avons proposée est SOFA, une méta-heuristique qui résout les problèmes par une décomposition séquentielle en trois étapes. Cette méthode est testée sur un cadre expérimental de 115 instances qui comprend 5 différentes demandes de service et 23 différents ensembles de contraintes. Ce cadre, ainsi que SOFA, sont structurés à partir d'une classification des contraintes de confection d'horaires de personnel présentée dans ce même article. Les résultats obtenus sont jugés satisfaisants en termes de qualité d'horaires, et permettent d'avoir un premier aperçu de l'impact de certaines contraintes sur le comportement de l'algorithme et sur la qualité générale de l'horaire résultant. La limitation principale de cet article est l'absence de base objective pour comparer les performances de SOFA.

Afin de combler cette limitation d'une part, et d'autre part d'évaluer le potentiel d'une approche fondamentalement différente, nous proposons dans le deuxième article une méthode alternative appelée CHAIR. Cette dernière est entièrement heuristique et construite selon une décomposition en sous-problème/problème-maître inspirée de la génération de colonne. Elle est testée sur le cadre expérimental proposé dans l'article précédent, et produit des horaires satisfaisants sur ces instances. Ces tests permettent par ailleurs de comparer objectivement les résultats obtenus par SOFA et CHAIR sur le cadre expérimental. Il en ressort que CHAIR performe légèrement mieux en moyenne, surtout sur des problèmes moins sévèrement contraints. De plus, bien que la performance des deux algorithmes soit comparable sur la majorité des instances, certains écarts de performances sont très important sur un petit nombre d'instances. Ces comportements extrêmes sont généralement causés par une sous-performance de SOFA.

CHAIR a été sélectionnée pour être utilisée dans les recherches du quatrième article, entre autre pour ses performances légèrement supérieures, ainsi que pour sa structure algorithmique plus adéquate à l'affectation d'horaires individualisés. Dans ce dernier article, CHAIR est améliorée afin de pouvoir résoudre des problèmes de confection d'horaires d'infirmière, une problématique importante dans la littérature en confection d'horaire. La principale amélioration apportée ici à CHAIR est l'ajout d'une gestion des préférences individuelles, une caractéristique exclue des articles précédents. Suite aux améliorations, CHAIR est d'abord testée sur un nouveau cadre expérimental de 144 instances spécifiquement conçu pour les problèmes de confection d'horaires d'infirmières, où les trois dimensions testées sont les habiletés, les préférences, et les conditions contractuelles. Les résultats obtenus ont permis de comprendre l'impact de ces trois différentes dimensions sur la structure des horaires, et de valider que la version améliorée de CHAIR produisait des horaires satisfaisant sur l'ensemble des instances. Ensuite, CHAIR a été testée sur un ensemble de 10 instances réelles documentées sur le site web de l'ASAP [68]. CHAIR y est comparé au meilleur algorithme connu sur chaque instance, et produit des résultats très satisfaisants considérant sa nature flexible et portable. En effet, CHAIR obtient la meilleure solution connue sur 50% des instances, et ne produit qu'une seule solution jugée insatisfaisante.

Dans l'ensemble, nous espérons que les travaux présentés dans cette thèse contribuerons positivement au développement de la recherche en méthodes de résolution flexibles et portables. Nous sommes très satisfaits des résultats obtenus durant les recherches dont les résultats ont été présenté dans cette thèse, bien que nous soyons conscients d'un certain nombre de limitations de SOFA et de CHAIR. Entre autre, la réduction des temps de calcul sera un élément utile à améliorer durant les travaux futurs, aussi bien pour CHAIR que pour SOFA. Plus spéciquement pour SOFA, améliorer les mécanismes de rétroaction et tester l'intégration de mécanismes de prévision sera une priorité afin d'augmenter la cohésion des décisions de chaque phase, sans toutefois nuire à l'avantage en temps de calcul procuré par la décomposition séquentielle. En ce qui concerne CHAIR, la priorité sera plutôt d'investiguer les pistes d'amélioration du mécanisme liant le sous-problème et le problème-maître. Finalement, élargir le spectre d'applications sur lequel les deux approches sont utilisées permettra de démontrer leur potentiel pour aller bien au delà du secteur de la santé.

En terminant, nous sommes convaincus que les méthodes flexibles et portables représentent un intérêt autant pour la recherche que pour les praticiens. Nous souhaitons d'ailleurs que ces méthodes puissent continuer d'évoluer à l'avenir vers une rédution des temps de calcul, une amélioration de la qualité des solutions, et un élargissement des spectres d'applications.

Bibliographie

[1] S. Abdennadher & H. Schlenker, *Nurse Scheduling using Constraint Logic Programming*, Eleventh Annual Conference on Innovative Applications of Artificial Intelligence, *http ://www.pms.informatik.uni-muenchen.de/publikationen*, 1999.

[2] S.C. Aggarwal, *A focussed review of scheduling in services*, European Journal of Operational Research, 9, pp. 114-121, 1982.

n

[3] U. Aickelin, E.K. Burke & J. Li, *An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering*, Journal of the Operational Research Society, 58, pp. 1574-1585, 2007.

[4] U. Aickelin & K.A. Dowsland, *Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem*, Journal of Scheduling, 3, pp. 139-153, 2000.

[5] U. Aickelin & P. White, *Building Better Nurse Scheduling Algorithms*, Annals of Operations Research, 128, pp. 159-177, 2004.

[6] Z. Aksin, M. Armony & V. Mehrotra, *The Modern Call Center : A Multi-Disciplinary Perspective on Operations Management Research*, Production and Operations Management, 16, pp. 665-688, 2007.

[7] H.K. Alfares, *An efficient two-phase algorithm for cyclic days-off scheduling*, Computers & Operations Research, 25, pp. 913-923, 1998.

[8] H.K. Alfares, *Dual-based optimization of cyclic three-day workweek scheduling*, Asia-Pacific Journal of Operational Research, 17, pp. 137-148, 2000.

[9] H.K. Alfares, *Survey, Categorization, and Comparison of Recent Tour Scheduling Literature*, Annals of Operations research, 127, pp. 145-175, 2004.

[10] R. Alvarez-valdes, E. Crespo & J.M. Tamarit, *Labour scheduling at an airport refuelling installation*, Journal of the Operational Research Society, 50, pp. 211-218, 1999.

[11] S.M. Al-Yakoob & H.D. Sherali, *A column generation approach for an employee scheduling problem with multiple shifts and work locations*, Journal of the Operational Research Society, 59, pp. 34-43, 2008.

[12] J.P. Arabeyre, J. Fearnley, F.C. Steiger & W. Teather, *The Airline Crew Scheduling Problem : A Survey*, Transportation Science, 3, pp. 140-163, 1969.

[13] H.M. Auf'm Hofe, *Solving Rostering Tasks as Constraint Optimization*, Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Burke and Erben, 2000.

[14] M.N. Azaiez & S.S. Al Sharif, *A 0-1 goal programming model for nurse scheduling*, Computers & Operations Research, 32, pp. 491-507, 2005.

[15] C.S. Azmat & M. Widmer, *A case study of single shift planning and scheduling under annualized hours : A simple three-step approach*, European Journal of Operational Research, 153, pp. 148-175, 2004.

[16] J. Bailey, *Integrated days off and shift personnel scheduling*, Computers & Industrial Engineering, 9, pp. 395-404, 1985.

[17] J. Bailey & J. Field, *Personnel Scheduling with Flexshift Models*, Journal of Operations Management, 5, pp. 327-338, 1985.

[18] R.N. Bailey, K.M. Garner & M.F. Hobbs, *Using simulated annealing and genetic algorithms to solve staff-scheduling problems*, Asia-Pacific Journal of Operational Research, 14, pp. 27-43, 1997.

[19] K.R. Baker, *Workforce Allocation in Cyclical Scheduling Problems : A Survey*, Operational Research Quarterly, 27, pp. 155-167, 1976.

[20] J.F. Bard, *Staff scheduling in high volume service facilities with downgrading*, IIE Transactions, 36, pp. 985-997, 2004.

[21] J.F. Bard, C. Binici & A.H. deSilva, *Staff scheduling at the United States Postal Service*, Computers & Operations Research, 39, pp. 745-771, 2003.

[22] J.F. Bard & H.W. Purnomo, *Cyclic preference scheduling of nurses using a Lagrangian-based heuristic*, Journal of Scheduling, 10, pp. 5-23, 2007.

[23] J. Baxter & M. Mosby, *Generating Acceptable Shift-working Schedules*, Journal of the Operational Research Society, 39, pp. 537-542, 1988.

[24] N. Beaumont, *Using mixed integer programming to design employee rosters*, Journal of the Operational Research Society, 48, pp. 585-590, 1997.

[25] S.E. Bechtold, *Work Force Scheduling for Arbitrary Cyclic Demands*, Journal of Operations Management, 1, pp. 205-214, 1981.

[26] S.E. Bechtold, *Implicit optimal and heuristic labor staffing in a multiobjective multilocation environment*, Decision Sciences, 19, pp. 353-372, 1988.

[27] S.E. Bechtold & M.J. Brusco, A microcomputer-based heuristic for tour scheduling of a mixed workforce, Computers & Operations Research, 21, pp. 1001-1009, 1994.

[28] S.E. Bechtold & M.J. Brusco, *Working set generation methods for labour tour scheduling*, European Journal of Operational Research, 74, pp. 540-551, 1994.

[29] S.E. Bechtold & M.J. Brusco, *Microcomputer-based working set generation methods for personnel scheduling*, International Journal of Operations & Production Management, 15, pp. 63-74, 1995.

[30] S.E. Bechtold, M.J. Brusco & M.J. Showalter, *A Comparative Evaluation of Labor Tour Scheduling Methods*, Decision Sciences, 22, pp. 683-699, 1991.

[31] S.E. Bechtold & L.W. Jacobs, *Implicit Modeling of Flexible Break Assignments in Optimal Shift Scheduling*, Management Science, 36, pp. 1339-1351, 1990.

[32] G.R. Beddoe & S. Petrovic, *Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering*, European Journal of Operational Research, 175, pp. 649-671, 2006.

[33] G. Beddoe & S. Petrovic, *Enhancing case-based reasoning for personnel rostering with selected tabu search concepts*, Journal of the Operational Research Society, 58, pp. 1586-1598, 2007.

[34] G. Beddoe, S. Petrovic & J. Li, *A hybrid metaheuristic case-based reasoning system for nurse rostering*, Journal of Scheduling, 12, pp. 99-119, 2009.

[35] F. Bellanti, G. Carello, F. Della Croce & R. Tadei, *A greedy-based neighborhood search approach to a nurse rostering problem*, European Journal of Operational Research, 153, pp. 28-40, 2004.

[36] I. Berrada, J.A. Ferland & P. Michelon, *A Multi-objective Approach to Nurse Scheduling with both Hard and Soft Constraints*, Socio-Economic Planning Sciences, 30, pp. 183-193, 1996.

[37] M.J. Bester, I. Nieuwoudt & J.H. Van Vuuren, *Finding good nurse duty schedules : a case study*, Journal of Scheduling, 10, pp. 387-405, 2007.

[38] A. Billionnet, *Integer programming to schedule a hierarchical workforce with variable demands*, European Journal of Operational Research, 114, pp. 105-114, 1999.

[39] M.J. Brusco, *Solving personnel tour scheduling problems using the dual all-integer cutting plane*, IIE Transactions, 30, pp. 835-844, 1998.

[40] M.J. Brusco & L.W. Jacobs, *A Simulated Annealing Approach to the Cyclic Staff-Scheduling Problem*, Naval Research Logistics, 40, pp. 69-84, 1993.

[41] M.J. Brusco & L.W. Jacobs, *A Simulated Annealing Approach to the Solution of Flexible Labour Scheduling Problems*, Journal of the Operational Research Society, 44, pp. 1191-1200, 1993.

[42] M.J. Brusco & L.W. Jacobs, *Cost analysis of alternative formulations for personnel scheduling in continuously operating organizations*, European Journal of Operational Research, 86, pp. 249-261, 1995.

[43] M.J. Brusco & L.W. Jacobs, *Eliminating redundant columns in continuous tour scheduling problems*, European Journal of Operational Research, 111, pp. 518-525, 1998.

[44] M.J. Brusco & L.W. Jacobs, *Personnel Tour Scheduling When Starting-Time Restrictions Are Present*, Management Science, 44, pp.534-547, 1998.

[45] M.J. Brusco & L.W. Jacobs, *Optimal Models for Meal-Break and Start-Time Flexibility in Continuous Tour Scheduling*, Management Science, 46, pp. 1630-1641, 2000.

[46] M.J. Brusco & L.W. Jacobs, *Starting-time decisions in labor tour scheduling : An experimental analysis and case study*, European Journal of Operational Research, 131, pp. 459-475, 2001.

[47] M.J. Brusco, L.W. Jacobs, R.J. Bongiorno, D.V. Lyons & B. Tang, *Improving Personnel Scheduling at Airline Stations*, Operations research, 43, pp.741-751, 1995.

[48] M.J. Brusco & T.R. Johns, *Improving the dispersion of surplus labor in personnel scheduling solutions*, Computers & Industrial Engineering, 28, pp. 745-754, 1995.

[49] M.J. Brusco & T.R. Johns, *A sequential integer programming method for discontinuous labor tour scheduling*, European Journal of Operational Research, 95, pp. 537-548, 1996.

[50] E.K. Burke, P. Cowling, P. De Causmaecker & G. Vanden Berghe, *A Memetic Approach to the Nurse Rostering Problem*, Applied Intelligence, 15, pp. 199-214, 2001.

[51] E.K. Burke, T. Curtois, G. Post, R. Qu & B. Veltman, *A hybrid heuristic ordering and variable neighborhood search for the nurse rostering problem*, European Journal of Operational Research, 188, pp. 330-441, 2008.

[52] E.K. Burke, P. De Causmaecker, S. Petrovic & G. Vanden Berghe, *Fitness Evaluation for Nurse Scheduling Problems*, Proceedings of Congress on Evolutionary Computation, pp. 1139-1146, 2001.

[53] E. Burke, P. De Causmaecker, S. Petrovic & G. Vanden Berghe, *Variable Neighborhood Search for Nurse Rostering Problems*, Chapter 7, Metaheuristics : Computer Decision-Making, pp. 153-172, 2003.

[54] E.K. Burke, P. De Causmaecker, S. Petrovic & G. Vanden Berghe, *Metaheuristics for handling time interval coverage constraints in nurse scheduling*, Applied Artificial Intelligence, 20, pp. 743-766, 2006.

[55] E. Burke, P. De Causmaecker & G. Vanden Berghe, *A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem*, Lecture Notes in Artificial Intelligence, 1585, pp. 187-194, 1998.

[56] E.K. Burke, P. De Causmaecker, G. Vanden Berghe & H. Van Landeghem, *The State of the Art of Nurse Rostering*, Journal of Scheduling, 7, pp. 441-499, 2004.

[57] E.K. Burke, G. Kendall & E. Soubeiga, *A Tabu-Search Hyperheuristic for Timetabling and Rostering*, Journal of Heuristics, 9, pp.451-470, 2003.

[58] E.K. Burke, J. Li and R. Qu, *A Hybrid Model of Integer Programming and Variable Neighbourhood Search for Highly-Constrained Nurse Rostering Problems*, Technical Report NOTTCS-TR-2007-2, School of Computer Science and IT, University of Nottingham, 2006.

[59] X. Cai & K.N. Li, *A genetic algorithm for scheduling staff or mixed skills under multi-criteria*, European Journal of Operational Research, 125, pp. 359-369, 2000.

[60] B. Cheang, H. Li, A. Lim , & B. Rodrigues, *Nurse rostering problems - a bibliographic survey*, European Journal of Operational Research, 151, pp. 447-460, 2003.

[61] K.L. Chew, *Cyclic Schedule for Apron Services*, Journal of the Operational Research Society, 42, pp. 1061-1069, 1991.

[62] M.V. Chiaramonte, L.M. Chiaramonte, *An Agent-based nurse rostering system under minimal staffing conditions*, International Journal of Production Economics, 114, pp. 697-713, 2008.

[63] A.H.W. Chun, S.H.C. Chan, G.P.S. Lam, F.M.F. Tsang, J. Wong & D.W.M. Yeung, *Nurse rostering at the hospital authority of Hong Kong*, AAAI/IAAI, pp. 951-956, 2000.

[64] J. Crowe, *Une Approche Hybride pour la Confection Automatisée d'Horaires de Paramédics*, Masters Thesis, HEC Montréal, 2007.

[65] J. Crowe & P. Soriano, *Staff Scheduling Literature Review*, Thesis working paper, 2011.

[66] J. Crowe & P. Soriano, *Personnel Scheduling Optimization with a Flexible and Portable Heuristic Approach Based on Sequential Decomposition*, Thesis working paper, 2011.

[67] J. Crowe & P. Soriano, *Personnel Scheduling Optimization with a Flexible and Portable Heuristic Approach Inspired from Column Generation*, Thesis working paper, 2011.

[68] T. Curtois, Employee Scheduling Benchmark Data sets, ASAP, University of Nottingham, http ://www.cs.nott.ac.uk/ tec/NRP/.

[69] G.B. Dantzig, *A Comment on Edie's "Traffic Delays at Toll Booths"*, Journal of the Operational Research Society of America, 2, pp. 339-341, 1954.

[70] P. De Causmaecker & G. Vanden Berghe, *Relaxation of Coverage Constraints in Hospital Personnel Rostering*, Practice and Theory of Automated Timetabling, Fourth International Conference, Lecture Notes in Computer Science, 2740, pp. 129-147, 2003.

[71] K.A. Dowsland, *Nurse scheduling with tabu search and strategic oscillation*, European Journal of Operational Research, 106, pp. 393-407, 1998.

[72] K.A. Dowsland & J.M. Thompson, *Solving a nurse scheduling problem with knapsacks, networks and tabu search*, Journal of the Operational Research Society, 51, pp. 825-833, 2000.

[73] F.F. Easton & N. Mansour, A distributed genetic algorithm for deterministic and stochastic labor scheduling problems, European Journal of Operational Research, 118, pp. 505-523, 1999.

[74] F.F. Easton & D.F. Rossin, *Sufficient Working Subsets for the Tour Scheduling Problem*, Management Science, 37, pp. 1441-1451, 1991.

[75] F.F. Easton & D.F. Rossin, *Equivalent Alternate Solutions for the Tour Scheduling Problem*, Decision Sciences, 22, pp. 985-1007, 1991.

[76] F.F. Easton & D.F. Rossin, *A Stochastic Goal Program for Employee Scheduling*, Decision Sciences, 27, pp. 541-568, 1996.

[77] F.F. Easton, D.F. Rossin & W.S. Borders, *Analysis of alternative scheduling policies for hospital nurses*, Production and operations management, 1, pp. 159-174, 1992.

[78] L.C. Edie, *Traffic Delays at Toll Booths*, Journal of the Operations Research Society of America, 2, pp. 107-138, 1954.

[79] J.S. Edwards, *A Survey of Manpower Planning Models and their Application*, Journal of the Operational Research Society, 34, pp. 1031-1040, 1983.

[80] G. Eitzen, D. Panton & G. Mills, *Multi-Skilled Workforce Optimisation*, Annals of Operations Research, 127, pp. 359-372, 2004.

[81] M. Elshafei & H.F. Alfares, *A dynamic programming algorithm for days-off scheduling with sequence dependent labor costs*, Journal of Scheduling, 11, pp. 85-93, 2008.

[82] H. Emmons & R.N. Burns, *Off-day scheduling with hierarchical worker categories*, Operations Research, 39, pp. 484-495, 1991.

[83] A.T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens & D. Sier, *An Annotated Bibliography of Personnel Scheduling and Rostering*, Annals of Operations Research, 127, pp. 21-144, 2004.

[84] A.T. Ernst, H. Jiang, M. Krishnamoorthy & D. Sier, *Staff scheduling and rostering : A review of applications, methods and models*, European Journal of Operational Research, 153, pp. 3-27, 2004.

[85] A. Ernst, P. Hourigan, M. Krishnamoorthy, G. Mills, H. Nott & D. Sier, *Rostering Ambulance Officers*, Proceedings of the 15th National Conference of the Australian Society for Operations Research, pp. 470-481, 1999.

[86] P. Eveborn & M.Rönnqvist, *Scheduler - A System for Staff Planning*, Annals of Operations Research, 128, pp. 21-45, 2004.

[87] G. Felici & C. Gentile, *A Polyhedral Approach for the Staff Rostering Problem*, Management Science, 50, pp. 381-393, 2004.

[88] J.A. Ferland, I. Berrada, I. Nabli, B. Ahiod, P. Michelon, V. Gascon & E. Gagné, *Generalized Assignment Type Goal Programming Problem : Application to Nurse Scheduling*, Journal of Heuristics, 7, pp. 391-413, 2001.

[89] B.E. Fries, *Bibliography of Operations Research in Health-Care Systems*, Operations Research, 24, pp. 801-814, 1976.

[90] B. Gendron, *Scheduling Employees in Quebec's Liquor Stores with Integer Programming*, Interfaces, 35, pp. 402-410, 2005.

[91] J.C. Goodale & G.M. Thompson, *A Comparison of Heuristics for Assigning Individual Eployees to Labour Tour Schedules*, Annals of Operations Research, 128, pp. 47-63, 2004.

[92] J.C. Goodale & E. Tunc, *Tour scheduling with dynamic service rates*, International Journal of Service Industry Management, 9, pp. 226-247, 1998.

[93] M. Gopalakrishnan, S. Gopalakrishnan & D. Miller, *A Decision Support System for Scheduling Personnel in a Newspaper Publishing Environment*, Interfaces, 23, pp. 104-115, 1993.

[94] B. Gopalakrishnan & E.L. Johnson, *Airline Crew Scheduling : Sate-of-the-Art*, Annals of Operations Research, 140, pp. 305-337, 2005.

[95] G. Hao, K.K. Lai & M. Tan, *A Neural Network Application in Personnel Scheduling*, Annals of Operations Research, 128, pp. 65-90, 2004.

[96] W.B. Henderson & W. Berry, *Heuristic methods for telephone operator shift scheduling : An experimental analysis*, Management Science, 22, pp. 1372-1380, 1976.

[97] W.B. Henderson & W.L. Berry, *Determining optimal shift schedules for telephone traffic exchange operators*, Decision Sciences, 8, pp. 239-255, 1977.

[98] R. Hung, *A Three-Day Workweek Multiple-Shift Scheduling Model*, The Journal of the Operational Research Society, 44, pp. 141-146, 1993.

[99] R. Hung, *Single-shift off-day scheduling of a hierarchical workforce with variable demands*, European Journal of Operational Research, 78, pp. 49-57, 1994.

[100] R. Hung, *A Multiple-Shift Workforce Scheduling Model Under the 4-Day Workweek With Weekday and Weekend Labour Demands*, Journal of the Operational Research Society, 45, pp. 1088-1092, 2009.

[101] D. Hur, V.A. Mabert & K.M. Bretthauer, *Real-Time Work Schedule Adjustment Decisions : An Investigation and Evaluation*, 13, pp. 322-339, 2004.

[102] A. Ikegami, A. Niwa, *A subproblem-centric model and approach to the nurse scheduling problem*, Mathematical Programming, B 97, pp. 517-541, 2003.

[103] A. Ingolfsson, A. Haque & A. Umnikov, *Accounting for time-varying queueing effects in workforce scheduling*, European Journal of Operational Research, 139, pp. 585-597, 2002.

[104] M.W. Isken, *An Implicit Tour Scheduling Model with Applications in Healthcare*, Annals of Operations Research, 128, pp.91-109, 2004.

[105] L.W. Jacobs & S.E. Bechtold *Labor Utilization Effects of Labor Scheduling Flexibility Alternatives in a Tour Scheduling Environment*, Decision Sciences, pp. 148-166, 1993.

[106] L.W. Jacobs & M.J. Brusco, *Overlapping Start-time Bands in Implicit Tour Scheduling*, Management Science, 42, pp. 1247-1259, 1996.

[107] A.I.Z. Jarrah, J.F. Bard & A.H. deSilva, *Solving Large-scale Tour Scheduling Problems*, Management Science, pp. 1124-1144, 40, 1994.

[108] B. Jaumard, F. Semet & T. Vovor, *A generalized linear programming model for nurse scheduling*, European Journal of Operational Research, 107, pp. 1-18, 1998.

[109] D. Kellogg & S. Walczak, *Nurse Scheduling : From Academia to Implementation or Not ?*, Interfaces, 37, pp. 355-369, 2007.

[110] P.J. Kolesar, K.L. Rider, T.B. Crabill & W.E. Walker, *A Queuing-Linear Programming Approach to Scheduling Police Patrol Cars*, Operations Research, 23, pp. 1045-1062, 1975.

[111] M.M. Kostreva & K.S.B. Jennings, *Nurse scheduling on a microcomputer*, Computers & Operations Research, 18, pp. 731-739, 1991.

[112] L.J. Krajewski, L.P. Ritzman & P. McKenzie, *Shift Scheduling in Banking Operations : A Case Application*, Interfaces, 10, pp. 1-6, 1980.

[113] G. Laporte, *The Art and Science of Designing Rotating Schedules*, The Journal of the Operational Research Society, 50, pp. 1011-1017, 1999.

[114] G. Laporte, Y. Nobert & J. Biron, *Rotating schedules*, European Journal of Operational Research, 4, pp. 24-30, 1980.

[115] H.C. Lau, *On The Complexity of Manpower Shift Scheduling*, Computers & Operations Research, 23, pp. 93-102, 1996.

[116] J. Lauer, L.W. Jacobs, M.J. Brusco & S.E. Bechtold, *An Interactive, Optimization-based Decision Support System for Scheduling Part-time Computer Lab Attendants*, Omega International Journal of Management Science, 22, pp. 613-626, 1994.

[117] H. Li, A. Lim & B. Rodrigues, *A Hybrid AI Approach for Nurse Rostering Problem*, Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 730-735, 2003.

[118] C. Li, E.P. Robinson, Jr. & V.A. Mabert, *An Evaluation of Tour Scheduling Heuristics with Differences in Employee Productivity and Cost*, Decision Sciences, 22, pp. 700-718, 1991.

[119] C.K.Y. Lin, K.F. Lai & S.L. Hung, *Development of a workforce management system for a customer hotline service*, Computers & Operations Research, 27, pp. 987-1004, 2000.

[120] J.A. Litchfield, A. Ingolfsson & K.J. Cheng, *Rostering for a restaurant*, INFOR, 41, pp. 287-300, 2003.

[121] R.R. Love, Jr. & J.M. Hoey, *Management Science Improves Fast-Food Operations*, Interfaces, 20, pp. 21-29, 1990.

[122] B.J. Luce, *A Shift Scheduling Algorithm. Paper presented at the ORSA National Meeting in San Diego*, 1973.

[123] V.A. Mabert & C.A. Watts, *A simulation analysis of tour-shift construction procedures*, Management Science, 28, pp. 520-532, 1982.

[124] A.J. Mason, *Elastic Constraint Branching, the Wedelin/Carmen Lagrangian Heuristic and Integer Programming for Personnel Scheduling*, Annals of Operations Research, 108, pp. 239-276, 2001.

[125] A.J. Mason, D.M. Ryan & D.M. Panton, *Integrated Simulation, Heuristic and Optimisation Approaches to Staff Scheduling*, Operations Research, 46, pp. 161-175, 1998.

[126] L.F. McGinnis, W.D. Culver & R.H. Deane, *One- and two-phase heuristics for workforce scheduling*, computers & industrial engineering, 2, pp. 7-15, 1978.

[127] J.D. Megaeth, *Successful hospital personnel scheduling*, Interfaces, 8, pp. 55-59, 1978.

[128] E. Melachrinoudis & M. Olafsson, *A microcomputer cashier scheduling system for supermarket stores*, International Journal of Physical Distribution & Logistics management, 25, pp. 34-50, 1995.

[129] Millar H. H. & M. Kiragu, *Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming*, European Journal of Operational Research, 104, pp. 582-592, 1998.

[130] H.E. Miller, W.P. Pierskalla & G.J. Rath, *Nurse Scheduling Using Mathematical Programming*, Operations Research, 24, pp. 857-870, 1976.

[131] Ministère de la santé et des services sociaux (MSSS), Comptes de la santé 2009-2010 à 2011-2012, http ://msssa4.msss.gouv.qc.ca/fr/document/publication.nsf/liste ?OpenView, 2012.

[132] J.G. Morris & M.J. Showalter, *Simple approaches to shift, days-off and tour scheduling problems*, Management Science, 29, pp. 942-950, 1983.

[133] A. Musa and U. Saxena, *Scheduling nurses using goal-programming techniques*, IIE transactions, 16, pp. 216-221, 1984.

[134] N. Musliu, A. Schaerf & W. Slany, *Local search for shift design*, European Journal of Operational Research, 153, pp. 51-64, 2004.

[135] R. Narasimhan, *An algorithm for multiple shift scheduling of hierarchical workforce on four-day or three-day workweeks*, INFOR, 38, pp. 14-32, 2000.

[136] R. Narasimhan, *An algorithm for single shift scheduling of hierarchical workforce*, European Journal of Operational Research, 96, pp. 113-121, 1996.

[137] H. Ni & H. Abeledo, *A branch-and-price approach for large-scale employee tour scheduling problems*, Annals of Operations Research, 155, pp. 167-176, 2007.

[138] A. Ovchinnikov & J. Milner, *Spreadsheet Model Helps to Assign Medical Residents at the University of Vermont's College of Medicine*, Interfaces, 38, pp. 311-323, 2008.

[139] I. Ozkarahan, *The Zero-One Goal Programming Model of a Flexible Nurse Scheduling Support System*, Proceedings of International Industrial Engineering Conference, May, pp.436-441, 1989.

[140] G. Post, Employee scheduling instance provided to the employee scheduling benchmark data sets, http ://wwwhome.math.utwente.nl/ postgf/.

[141] S.U. Randhawa & D. Sitompul, *A heuristic-based computerized nurse scheduling system*, Computers & Operations Research, 20, pp.837-844, 1993.

[142] M. Rekik, J.F. Cordeau & F. Soumis, *Using Benders Decomposition to Implicitly Model Tour Scheduling*, Annals of Operations Research, 128, pp. 111-133, 2004.

[143] L.-M. Rousseau, M. Gendreau & G. Pesant, *A General Approach to the Physician Rostering Problems*, Annals of Operations Research, 115, pp. 193-205, 2002.

[144] M.J. Showalter & V.A. Mabert, *An Evaluation of a Full-/Part-time Tour Scheduling Methodology*, International Journal of Operations and Production Management, 8, pp. 54-71, 1988.

[145] S.P. Siferd & W.C. Benton, *Workforce staffing and scheduling : Hospital nursing specific models*, European Journal of Operational Research, 60, pp. 233-246, 1992.

[146] Instance provided by SINTEF, http ://www.sintef.no/.

[147] L.D. Smith, *The application of an interactive algorithm to develop cyclical rotational schedules for nursing personnel*, INFOR, 14, pp. 53-70, 1976.

[148] A.R. Smith & J. Bartholomew, *Manpower Planning in the United Kingdom : An Historical Review*, Journal of the Operational Research Society, 39, pp. 235-248, 1988.

[149] G. Thompson, *Improving the Utilization of Front-Line Service Delivery System Personnel*, Decision Sciences, 23, pp. 1072-1097, 1992.

[150] G.M. Thompson, *Representing Employee Requirements in Labour Tour Scheduling*, International Journal of Management Science, 21, pp. 657-671, 1993.

[151] G.M. Thompson, *Improved Implicit Optimal Modeling of the Labor Shift Scheduling Problem*, Management Science, 41, pp. 595-607, 1995.

[152] G.M. Thompson, *Labor scheduling using NPV estimates of the marginal benefit of additional labor capacity*, Journal of Operation Management, 13, pp. 67-86, 1995.

[153] G.M. Thompson, *A simulated-annealing heuristic for shift scheduling using non-continuously available employees*, Computers & Operations Research, 23, pp. 275-288, 1996.

[154] G.M. Thompson, *Assigning Telephone Operators to Shifts at New Brunswick Telephone Company*, Interfaces, 27, pp. 1-11, 1997.

[155] G.M. Thompson, *Labor Staffing and Scheduling Models for Controlling Service Levels*, Naval Research Logistics, 44, pp. 719-740, 1997.

[156] G.M. Thompson, *Labor Scheduling, Part 3 : Developing a Workforce Schedule*, Cornell Hotel and Restaurant Administration Quarterly, 40, pp. 86-96, 1999.

[157] G.M. Thompson, *Labor Scheduling, Part 4 : Controlling Workforce Schedules in Real Time*, Cornell Hotel and Restaurant Administration Quarterly, 40, pp. 85-96, 1999.

[158] G.M. Thompson & J.C. Goodale, *Variable employee productivity in workforce scheduling*, European Journal of Operational Research, 170, pp. 376-390, 2006.

[159] J.M. Tien & A. Kamiyama, *On Manpower Scheduling Algorithms*, SIAM Review, 24, pp. 275-287, 1982.

[160] S. Topaloglu, *A shift scheduling model for employees with different seniority levels and an application in healthcare*, European Journal of Operational Research, 198, pp. 943-957, 2009.

[161] S. Topaloglu & I. Ozkarahan, *Implicit optimal tour scheduling with flexible break assignments*, Computers & Industrial Engineering, 44, pp. 75-89, 2002.

[162] S. Topaloglu & I. Ozkarahan, *An Implicit Goal Programming Model for the Tour Scheduling Problem Considering the Employee Work Preferences*, Annals of Operations Research, 128, pp. 135-158, 2004.

[163] E. Tsang, J. Ford, P. Mills, R. Bradwell, R. Williams & P. Scott, *Towards a practical engineering tool for rostering*, Annals of Operations Research, 155, pp. 257-277, 2007.

[164] Y. Van Den Berg & D.M. Panton, *Personnel Shift Assignment : Existence Conditions and Network Models*, Networks, 24, pp. 385-394, 1994.

[165] L. Wan & J.F. Bard, *Weekly staff scheduling with workstation group restrictions*, Journal of the Operational Research Society, 58, pp. 1030-1046, 2007.

[166] D.M. Warner, *Scheduling Nursing Personnel According to Nursing Preference : A Mathematical Programming Approach*, Operations Research, 24, pp. 842-856, 1976.

[167] D.M. Warner & J. Prawda, *A mathematical programming model for scheduling nursing personnel in a hospital*, Management Science, 19, pp. 411-422, 1972.

[168] G. Weil, K. Heus, P. Francois, and M. Poujade, *Constraint programming for nurse scheduling*, IEEE Engineering in Medicine and Biology Magazine, 14, pp. 417-422, 1995.

[169] R.J. Willis & S.B. Huxford, *Staffing Rosters with Breaks – A Case Study*, The Journal of the Operational Research Society, 42, pp. 727-731, 1991.

[170] World Health Organization (WHO), *Statistics by country of the Global Health Observatory Data Repository*, http ://apps.who.int/ghodata/ ?vid=5900&theme=country, 2011.

[171] E.J.G. Wilson & R.J. Willis, *Scheduling of Telephone Betting Operators - A Case Study*, Journal of the Operational Research Society, 34, pp. 991-998, 1983.

[172] S. Zolfaghari, A. El-Bouri & B. Namiranian, *Heuristics for Large Scale Labour Scheduling Problems in Retail Sector*, INFOR, 45, pp. 111-122, 2007.