UNIVERSITÉ DE MONTRÉAL

GRAMMAR-BASED DECOMPOSITION METHODS FOR MULTI-ACTIVITY TOUR SCHEDULING

MARÍA ISABEL RESTREPO RUIZ
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
DÉCEMBRE 2015

© María Isabel Restrepo Ruiz, 2015.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

GRAMMAR-BASED DECOMPOSITION METHODS FOR MULTI-ACTIVITY TOUR SCHEDULING

présentée par : RESTREPO RUIZ María Isabel
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. DESAULNIERS Guy, Ph. D., président
M. ROUSSEAU Louis-Martin, Ph. D., membre et directeur de recherche
M. GENDRON Bernard, Ph. D., membre et codirecteur de recherche
M. PESANT Gilles, Ph. D., membre
Mme REKIK Monia, Ph. D., membre externe

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisors Louis-Martin Rousseau and Bernard Gendron for the continuous support during my Ph.D. I have been amazingly fortunate to have them as advisors since they gave me the freedom to explore on my own, and at the same time the knowledge, guidance and motivation to recover when my steps faltered.

Besides my supervisors, I would like to thank Prof. Guy Desaulniers, Prof. Gilles Pesant, Prof. Monia Rekik and Prof. Michel Gamache for accepting to be part of the Jury. Also for their comments that helped me improve the quality of this thesis.

The four years I spent in Montreal during my Ph.D. would not have been so enriching and amazing but for the people I met there. Thank you to all the friends I made at CIRRELT, it was very nice to feel that in some way I had a big multicultural family there.

Last but not the least, I would like to thank the members of my family, Simon and two special friends from Colombia (Juan Guillermo and David) who were always there for me during these years, I have really appreciated to have someone to talk to all times.

RÉSUMÉ

Les problèmes de planification d'horaires du personnel consistent à sélectionner un ensemble de quarts de travail en respectant certaines règles, et à assigner un certain nombre d'employés à chaque quart de travail, de sorte à satisfaire la demande de personnel. Ces problèmes sont généralement classés en trois catégories: planification des quarts de travail, planification des jours de repos et planification des patrons de travail. La première catégorie a pour but d'assigner les périodes de travail et de repos aux quarts de travail, et à sélectionner un ensemble de ces quarts de travail pour satisfaire les besoins en personnel. En planification des quarts de travail, l'horizon de planification est en général d'une journée, divisée en périodes de longueurs égales. La deuxième catégorie vise à sélectionner les jours de repos de chaque employé sur un horizon de planification d'au moins une semaine. Cette sélection est généralement contrainte par les préférences des employés ou par des conventions collectives. La dernière catégorie comprend les problèmes qui découlent de l'intégration des problèmes de planification des quarts de travail et de planification des jours de repos. Dans la version continue du problème de planification des patrons de travail, les quarts de travail peuvent s'étendre sur deux jours. Par contre, dans la version discontinue les quarts de travail doivent couvrir une seule journée.

Différentes extensions du problème de planification d'horaires du personnel apparaissent lorsque des applications réelles sont considérées. Par exemple, dans les problèmes de planification des quarts multi-activités (MASSP) ou planification des patrons de travail multi-activités (MATSP), en plus de la définition des périodes de travail et de repos, des activités de travail différentes doivent être attribuées aux quarts de travail. Dans un contexte de multi-activités, les caractéristiques spécifiques liées aux règles de travail, aux conventions collectives, aux compétences des employés et à leurs préférences définissent un ensemble de règles à respecter pour construire les horaires des employés.

D'autre part, le MASSP et le MATSP peuvent être soit personnalisés soit anonymes. Dans le premier cas, les employés ont des compétences et des préférences différentes, alors qu'elles sont identiques dans le second. Le problème peut également être stochastique, dans ce cas les besoins en employés (la demande) est incertaine. Dans cette thèse, nous aborderons trois catégories de MATSP : 1) MATSP discontinu, personnalisé, avec demande déterministe; 2) MATSP discontinu, anonyme avec demande déterministe; 3) MATSP discontinu, anonyme, avec demande stochastique. Pour résoudre ces problèmes, nous proposons différentes techniques de modélisation et de résolution qui sont principalement basées sur les méthodes de

décomposition et les langages formels.

Notre première contribution réside dans la conception de deux méthodes de type branch-and-price (B&P) pour aborder l'intégration de deux problèmes : le MASSP personnalisé et le problème de planification des patrons de travail discontinu. Chaque algorithme B&P repose sur une formulation mathématique différente. La première formulation (formulation basée sur les jours) est une extension naturelle du MASSP personnalisé, ou les colonnes correspondent aux quarts de travail multi-activités, et les patrons de travail sont assemblés dans le problème maitre en utilisant des contraintes supplémentaires. Dans la seconde formulation (formulation basée sur les patrons de travail), le sous-problèmes consistent à construire les quarts de travail multi-activités ainsi qu'à choisir les jours de repos. Par conséquent, dans cette formulation, les colonnes correspondent aux patrons de travail multi-activités. Dans les deux formulations, l'utilisation de grammaires nous permet de modéliser toutes les règles de travail pour la composition des quarts de travail, et de déduire des structures de graphes spéciales permettant de trouver les quarts de travail avec un coût réduit négatif.

Une comparaison expérimentale et théorique de la qualité des bornes obtenues par la relaxation linéaire de chacune des formulations est réalisée. Les résultats montrent que la formulation basée sur les patrons de travail est meilleure (relativement aux bornes obtenues par la relaxation linéaire) que la formulation basée sur les jours. De plus, nos expériences montrent d'une part que les approches de modélisation proposées peuvent traiter une grande variété de règles sur les quarts de travail et sur les patrons de travail, et d'autre part que les méthodes implémentées peuvent résoudre efficacement des versions réalistes du problème. Les approches proposées sont clairement pertinentes en pratique, cependant des problèmes liés à la taille du modèle apparaissent lorsque le nombre d'activités et d'employés augmentent.

La seconde contribution est une approche qui combine la décomposition de Benders et la génération de colonnes pour résoudre le problème intégrant le MASSP anonyme et la planification des patrons de travail discontinue. Afin de résoudre les problèmes de croissance des modèles et de symétrie associés aux nombres d'activités et d'employés, une autre façon de modéliser le MATSP est présentée. Les quarts de travail multi-activités sont implicitement générés par un modèle de programmation en nombres entiers basé sur une grammaire, alors que les patrons de travail sont explicitement composés via la génération de colonnes.

Comme les sous-problèmes de l'approche de Benders sont des MIP qui n'ont pas la propriété d'intégralité, nous présentons une stratégie qui combine la génération de coupes de Benders classiques avec des coupes de Benders entières afin de garantir la convergence de la méthode. Les résultats expérimentaux montrent : 1) la capacité de notre approche à résoudre des cas pratiques impliquant un grand nombre d'employés et d'activités de travail; 2) que l'approche

combinant la décomposition de Benders et la génération de colonnes à de meilleures performances que la méthode B&P pour le MATSP discontinu anonyme.

Notre dernière contribution présente une approche de programmation stochastique en deux étapes pour résoudre le MATSP stochastique, discontinu, et anonyme. Les décisions de la première étape correspondent à l'affectation des employés aux patrons de travail. Les décisions de la deuxième étape (actions de recours) sont associés à la répartition des activités de travail et des pauses dans les quarts de travail. Une heuristique de type *multi-cut L-shaped* est présentée. Les expériences montrent que les performances de la méthode dépendent du profil de la demande, et que l'utilisation du modèle stochastique permet de réduire les coûts, en comparaison avec l'espérance de la solution moyenne.

ABSTRACT

Personnel scheduling problems consist in constructing a set of feasible shift schedules and assigning them to the company staff to satisfy a given demand for staff requirements. These problems are typically classified into three main categories: shift scheduling, days-off scheduling and tour scheduling. The first category deals with the specification of work and rest periods to assign to shifts, as well as the selection of a set of those shifts to satisfy the demand for staff requirements. In shift scheduling, the planning horizon is usually one day divided into time periods of equal length. The second category involves the selection of days-off over a planning horizon of at least one week. Such selection is usually restricted by employee preferences or workplace agreements. The last category includes problems that arise from the integration of shift scheduling and days-off scheduling. The continuous version of the tour scheduling problem appears when shifts are allowed to span from one day to another. The discontinuous version arises when shifts span only one working day.

Different extensions of classical personnel scheduling problems appear when real applications are considered. For instance, when more than one work activity has to be scheduled, the multi-activity shift scheduling (MASSP) and the multi-activity tour scheduling (MATSP) problems appear. In both extensions not only the specification of work and rest periods is necessary, but also the assignment of work activities to the shifts. In a multi-activity context, specific characteristics related to work rules, workplace agreements, and employee skills and preferences define the rules to build the schedule of employees.

The MASSP and the MATSP can further be distinguished as personalized and anonymous problems. In the former, employee skills and preferences are different. In the latter, employee skills and preferences are identical. Additionally, if employee requirements (demand) is uncertain, the stochastic version of the problems appears. In this thesis we address three categories of the MATSP: 1) the discontinuous MATSP when employees have different skills and demand is deterministic; 2) the discontinuous MATSP when employees are identical and demand is deterministic; 3) the discontinuous MATSP when employees are identical and demand is stochastic. To address these problems we propose different modeling approaches and solution techniques which are mainly based on decomposition methods and formal languages.

Our first contribution lies in the proposal of two branch-and-price (B&P) methods to address the integration of two problems: the personalized MASSP and the discontinuous tour scheduling problem. Each B&P algorithm is based on a different mathematical formulation. The first formulation (daily-based formulation) arises as a natural extension of the person-

alized MASSP, where columns correspond to multi-activity shifts and tours are assembled into the master problem by means of extra constraints. The second formulation (tour-based formulation) aims to include, in the subproblem level, the construction of multi-activity shifts and the assembling of days-off. Therefore, in this formulation the set of columns correspond to multi-activity tours. In both formulations, the use of grammars allows us to model all the work rules for the composition of shifts and to derive specialized graph structures used to find the shifts with negative reduced cost.

An experimental and theoretical comparison on the quality of the LP relaxation bounds achieved by each formulation is made. The results show that the tour-based formulation is strong in terms of its LP relaxation bound, when compared with the daily-based formulation. Additionally, computational experiments suggest that the modeling approaches proposed can handle a wide variety of rules over shifts and tours and that the solution methods implemented efficiently solve realistic versions of the problem. However, while the practical relevance of the approaches is clear, convergence and scalability issues arise when the number of work activities and employees increases.

As a second contribution we present an approach that combines Benders decomposition and column generation to solve the integration of the anonymous MASSP and the discontinuous tour scheduling problem. The aim of the approach is to present an alternative way to model the MATSP in order to solve the scalability and symmetry issues associated with the number of work activities and employees. While multi-activity daily shifts are implicitly generated with a grammar-based integer programming model, tour patterns are explicitly composed via column generation.

Because Benders subproblems are MIP programs that do not possess the integrality property, we present an alternative algorithmic strategy that combines the generation of classical Benders cuts with integer Benders cuts to guarantee the convergence of the method. Experimental results show: 1) the capability of our approach to solve practical instances involving a large number of employees and work activities; 2) the combined Benders decomposition and column generation approach outperforms a B&P method that solves the anonymous discontinuous MATSP.

Our last contribution consists in the introduction of a two-stage stochastic programming approach to solve the discontinuous stochastic MATSP for employees with identical skills. The problem is formulated as a two-stage stochastic programming model. First-stage decisions correspond to the assignment of employees to weekly tours. Second-stage decisions (recourse actions) are related to the allocation of work activities and breaks to daily shifts. A heuristic multi-cut L-shaped method is presented as a solution approach. Computational

results show that the performance of the method depends on the demand profile used and that the use of the stochastic model helps to prevent additional costs, when compared with the expected-value problem solutions.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

RÉSUMÉ iv

ABSTRACT vii

TABLE OF CONTENTS x

LIST OF TABLES xiii

LIST OF FIGURES xiv

LIST OF SYMBOLS AND ABBREVIATIONS xv

LIST OF APPENDICES xvi

CHAPTER 1 INTRODUCTION 1
- 1.1 Definitions and Basic Concepts 1
  - 1.1.1 Categorization of Personnel Scheduling Problems 1
  - 1.1.2 Specific Characteristics in Personnel Sheduling Problems 2
  - 1.1.3 Grammars for Multi-activity Shift Scheduling 2
- 1.2 Problem Statement 6
  - 1.2.1 Work Rules for Daily Shift Definition 6
  - 1.2.2 Work Rules for Tour Definition 7
  - 1.2.3 Cost Factors in the Objective Function 7
  - 1.2.4 Nature of Demand 8
- 1.3 Research Objectives 8
- 1.4 Thesis Outline 8

CHAPTER 2 LITERATURE REVIEW 10
- 2.1 Shift Scheduling 10
- 2.2 Tour Scheduling 14
- 2.3 Multi-Activity Shift Scheduling 19
- 2.4 Stochastic Personnel Scheduling Problems 21

CHAPTER 3 GENERAL ORGANIZATION OF THE DOCUMENT 24

CHAPTER 4 ARTICLE 1: BRANCH-AND-PRICE FOR PERSONALIZED MULTI-ACTIVITY TOUR SCHEDULING . . . 25
4.1 Introduction . . . 28
4.2 Background and Related Research . . . 29
4.2.1 **Shift Scheduling** . . . 29
4.2.2 **Multi-Activity Shift Scheduling** . . . 30
4.2.3 **Tour Scheduling** . . . 31
4.2.4 Grammars and Shift Scheduling Problems . . . 34
4.3 Problem Definition and Formulations . . . 37
4.3.1 Problem Definition and Notation . . . 37
4.3.2 The Daily-Based Formulation . . . 38
4.3.3 The Tour-Based Formulation . . . 39
4.3.4 Comparison Between the Two Formulations . . . 40
4.4 B&P for the Daily-Based Formulation . . . 43
4.4.1 Restricted Master Problem . . . 43
4.4.2 Pricing Subproblems for Daily Shift Generation . . . 43
4.4.3 Branching Rule . . . 44
4.5 B&P for the Tour-Based Formulation . . . 44
4.5.1 Restricted Master Problem . . . 45
4.5.2 Pricing Subproblems for Tour Generation . . . 45
4.5.3 Branching Rule . . . 48
4.6 Computational Experiments . . . 48
4.6.1 Results on Random Instances . . . 48
4.6.2 Results on Instances From Brunner and Bard [17] . . . 57
4.7 Concluding Remarks . . . 59

CHAPTER 5 ARTICLE 2: COMBINING BENDERS DECOMPOSITION AND COLUMN GENERATION FOR MULTI-ACTIVITY TOUR SCHEDULING . . . 60
5.1 Introduction . . . 62
5.2 Background Material . . . 65
5.2.1 Shift and Tour Scheduling . . . 65
5.2.2 Multi-Activity Shift and Tour Scheduling . . . 66
5.2.3 Grammars . . . 67
5.3 Grammar-Based Model . . . 69
5.4 Benders Decomposition/Column Generation Algorithm . . . 73
5.4.1 Benders Decomposition . . . 73

5.4.2 Column Generation 78
5.4.3 Overall Algorithm 81
5.5 Computational Experiments 85
5.5.1 Results on MATSP Instances 85
5.5.2 Results on MASSP Instances 90
5.6 Concluding Remarks 93

CHAPTER 6 ARTICLE 3: A TWO-STAGE STOCHASTIC PROGRAMMING APPROACH FOR MULTI-ACTIVITY TOUR SCHEDULING 94
6.1 Introduction 95
6.2 Background Material 97
6.2.1 Literature Review on Multi-Activity Shift and Tour Scheduling . . . 97
6.2.2 Literature Review on Stochastic Shift and Tour Scheduling 98
6.2.3 Grammars for Multi-activity Shift Scheduling 100
6.3 Two-Stage Stochastic Problem 103
6.4 Heuristic Multi-cut L-shaped Method 107
6.5 Computational Experiments 114
6.5.1 Instances Generation 114
6.5.2 Problem Definition and Grammar 116
6.5.3 Computational Results 117
6.6 Concluding Remarks 119

CHAPTER 7 GENERAL DISCUSSION 121

CHAPTER 8 CONCLUSION 123
8.1 Advancement of Knowledge 123
8.2 Limits and Constraints 125
8.3 Recommendations 126

REFERENCES 127

APPENDICES 134

LIST OF TABLES

Table 4.1 Average number of nodes and arcs in the graphs for instances with flexible and restricted start times. 51
Table 4.2 Computational effort to solve the LP relaxation at the root node, for instances with flexible start times. 52
Table 4.3 Computational effort to solve the LP relaxation at the root node, for instances with restricted start times. 53
Table 4.4 Computational effort to find an upper bound at the root node, for instances with flexible start times. 54
Table 4.5 Computational effort to find an upper bound at the root node, for instances with restricted start times. 54
Table 4.6 Computational effort in the B&P algorithm for instances with flexible start times. 56
Table 4.7 Computational effort in the B&P algorithm for instances with restricted start times. 56
Table 4.8 Description of Brunner and Bard [17] problem's scenarios. 57
Table 4.9 Computational effort at the root node to solve the integer problem. . 58
Table 5.1 Size of MATSP instances 87
Table 5.2 Results on MATSP instances with smooth demand shape 88
Table 5.3 Results on MATSP instances with erratic demand shape 89
Table 5.4 Size of MASSP instances 91
Table 5.5 Results on MASSP instances 92
Table 6.1 Results on stochastic weekly instances with unimodal demand shape. 118
Table 6.2 Results on stochastic weekly instances with level demand shape. . . . 118
Table A.1 Employee requirements, shifts and tours structures. 134

LIST OF FIGURES

Figure 1.1 DAG Γ on words of length four and two work activities. 5
Figure 4.1 DAG Γ on words of length five and two work activities. 36
Figure 4.2 Parse trees derived from DAG Γ on words of length five and two work activities. 36
Figure 4.3 $G^e(\mathcal{N}, \mathcal{A})$ over a planning horizon of seven days. 47
Figure 5.1 DAG Γ on words of length five and two work activities 70
Figure 5.2 Weekly tours composed of and-nodes (shift shells) from Γ 73
Figure 6.1 DAG Γ_1 on words of length four and two work activities. 102
Figure 6.2 Weekly tours composed of shifts from Γ_1. 104
Figure 6.3 Flow chart for the multi-cut L-shaped method. 111
Figure 6.4 Deterministic demand profiles. 115
Figure A.1 Optimal solution for the LP relaxation of the example using the Daily-based formulation. 135

LIST OF SYMBOLS AND ABBREVIATIONS

SSP Shift scheduling problem

MASSP Multi-activity shift scheduling problem

MATSP Multi-activity tour scheduling problem

SMATSP Stochastic multi-activity tour scheduling problem

EO Extraordinary overlap

RMP Restricted master problem

B&B Branch-and-bound

B&P Branch-and-price

LP Linear programming

IP Integer programming

MIP Mixed integer programming

MILP Mixed integer linear programming

CP Constraint programming

CG Column generation

BD Benders decomposition

DFA Deterministic finite automaton

DAG Directed acyclic graph

SPPRC Shortest path problem with resource constraints

LIST OF APPENDICES

APPENDIX A EXAMPLE FOR THE COMPARISON BETWEEN THE TWO FORMULATIONS 134

APPENDIX B LABEL SETTING ALGORITHM TO SOLVE THE SPPRC 136

CHAPTER 1 INTRODUCTION

Personnel scheduling problems consist in constructing a set of feasible shift schedules and assigning them to the company staff to satisfy a given demand for employee requirements. These problems arise in diverse organizations such as hospitals, airline companies, retail stores, call centers and banks, where they have become an important task, due to the necessity to achieve a better level of service, to reduce staff costs and to increase employee satisfaction.

Realistic applications of personnel scheduling problems for companies that operate outside the standard 8-hour shift, 5-days per week schedule and that face wide fluctuations on demand for services become challenging due to several factors including union contracts, work rules for the composition of the schedules, multiple work activities, employee preferences, employee skills, stochastic nature of the demand, among others. When those factors are considered simultaneously, the personnel scheduling becomes a complex large-scale problem which requires a huge investment of time to solve. If the objective is to construct an optimal or near-optimal schedule in a reasonable amount of time, it is almost impossible to do it by hand, making necessary the use of mathematical models, specialized algorithms and different solution techniques.

The structure of this chapter is as follows. Section 1.1 introduces the basic elements and definitions of the problem addressed in the thesis. The problem statement is presented in Section 1.2. Section 1.3 defines the objectives and contribution of the thesis. Finally, Section 1.4 outlines the chapters of the thesis.

1.1 Definitions and Basic Concepts

Some concepts and definitions will be introduced before stating the problem to be addressed in the thesis. First, Section 1.1.1 gives an introduction on the different stages (problems) faced in personnel scheduling. Then, Section 1.1.2 describes the specific characteristics regarding the type of problem addressed in the thesis. Finally, Section 1.1.3 gives an introduction on the use of grammars for multi-activity shift scheduling.

1.1.1 Categorization of Personnel Scheduling Problems

According to Baker [8], three main categories of problems can be distinguished in personnel scheduling:

- *Shift scheduling problem (SSP)*: This category deals with the specification of work and rest time periods to assign to shifts, as well as with the selection of a set of those shifts to satisfy the demand for employee requirements. In the SSP, the *planning horizon* is usually one day divided into *time periods* of equal length.

- *Days-off scheduling*: This category involves the selection of employee *working days* and *days-off* over a planning horizon of at least one week.

- *Tour scheduling problem*: This category arises as an integration of days-off scheduling and shift scheduling. The aim of the tour scheduling problem is to specify the time periods of the day and the days of the week in which employees must work.

When more than one *work activity* has to be scheduled, the SSP and the tour scheduling problem become the *Multi-activity shift scheduling problem* (MASSP) and the *Multi-activity tour scheduling problem* (MATSP), respectively. In these problems not only the specification of working days, work time and rest time is necessary, but also the allocation of work activities to the shifts.

1.1.2 Specific Characteristics in Personnel Sheduling Problems

- *Continuity and discontinuity*: In a tour scheduling problem, continuity is present when shifts are allowed to span from one day to another. On the contrary, when shifts must cover exactly one working day, the discontinuous version of the problem appears.

- *Different and identical skills*: In a multi-activity environment, when employees have different qualifications (skills), preferences and availabilities, the *personalized* version of the MASSP and the MATSP appears. In the personalized problem, the shifts and tours must be designed according to the skills of each employee. On the contrary, when employees have identical skills the anonymous versions of the MASSP and the MATSP arise. In the anonymous problem, the set of feasible shifts and tours are the same for all the employees.

1.1.3 Grammars for Multi-activity Shift Scheduling

In shift scheduling, a *context-free grammar* (CFG) can be defined as a finite set of work rules that are used to generate valid sequences of work for a given day $d \in D$ in a multi-day planning horizon. A CFG consists of a tuple $G_d = \langle \Sigma_d, N_d, S_d, P_d \rangle$, where:

- Σ_d represents an alphabet of characters called the *terminal symbols* for day d, which consists of work activities, breaks, lunch breaks, and rest. Terminal symbols will be represented by lower case letters.
- N_d is a finite set of *non-terminal symbols* for day d. Non-terminal symbols will be represented by upper case letters.
- $S_d \in N_d$ is the *starting symbol* for day d.
- P_d is a set of *productions* for day d, represented as $A \rightarrow \alpha$, where $A \in N_d$ is a non-terminal symbol and α is a sequence of terminal and non-terminal symbols. The work rules used to generate shifts are represented by the set of productions. The productions of a grammar can be used to generate new symbol sequences until only terminal symbols are part of the sequence.

A *parse tree* is a tree where each inner-node is labeled with a non-terminal symbol N_d and each leaf is labeled with a terminal symbol Σ_d. A grammar recognizes a sequence if and only if there exists a parse tree where the leaves, when listed from left to right, reproduce the sequence.

A *DAG* Γ_d is a *directed acyclic graph* that embeds all parse trees associated with words, including rest and work stretches, for day d of a given length n recognized by a grammar. The DAG Γ_d has an and/or structure where the and-nodes represent productions (work rules) from P_d and the or-nodes represent non-terminals from N_d and letters from Σ_d. An and-node is true if all of its children are true. An or-node is true if one of its children is true. The root node is true if the grammar accepts the sequence encoded by the leaves. In Γ_d, O^{π}_{dil} denotes the or-node associated with $\pi \in N_d \cup \Sigma_d$, i.e., with non-terminals from N_d or letters from Σ_d, that generates a subsequence at position i of length l for day d. Note that if $\pi \in \Sigma_d$, the node is a leaf and l is equal to one. On the contrary, if $\pi \in N_d$, the node represents a non-terminal symbol and $l > 1$. $A^{\Pi,k}_{dil}$ is the kth and-node representing production $\Pi \in P_d$ that generates a subsequence from position i of length l at day d. There are as many $A^{\Pi,k}_{dil}$ nodes as there are ways of using P_d to generate a sequence of length l from position i. In Γ_d, the root node is described by O^{S}_{d1n} and its children by $A^{\Pi,k}_{d1n} \in ch(O^{S}_{d1n})$. The children of or-node O^{π}_{dil} are represented by $ch(O^{\pi}_{dil})$ and the parents by $par(O^{\pi}_{dil})$. Similarly, the children of and-node $A^{\Pi,k}_{dil}$ are represented by $ch(A^{\Pi,k}_{dil})$ and the parents by $par(A^{\Pi,k}_{dil})$. The sets of or-nodes, and-nodes and leaves in Γ_d are denoted by O_d, A_d and L_d, respectively. The DAG Γ_d is built by a procedure proposed in Quimper and Walsh [71].

Grammar G_1 and Figure 1.1 present an example on the use of context-free grammars for multi-activity shift scheduling. Two activities, w_1 and w_2, must be scheduled, shifts have

a length of 4 time periods and should contain exactly one break, b, of one time period that can be placed anywhere during the shift except at the first or the last time period. For clarity, we do not include the subscript of the day in the notation of grammar G_1 and nodes from Γ. The grammar that defines the set of feasible shifts on this example follows:

$G_1 = (\Sigma = (w_1, w_2, b), N = (S, X, W, B), P, S)$,
where productions P are: $S \to XW$, $X \to WB$, $W \to WW|w_1|w_2$, $B \to b$,
and symbol $|$ specifies the choice of production.

In the previous example, productions $W \to w_1$, $W \to w_2$ and $B \to b$ generate the terminal symbols associated with working on activity 1, working on activity 2, or having a break, respectively. Production $W \to WW$ generates two non-terminal symbols, W, meaning that the shift will include a working subsequence. Production $X \to WB$ means that the shift will include working time and then it will be followed by a break. Finally, production $S \to XW$ generates a sequence of length four (the daily shift), which includes working time followed by a break to finish with more working time.

Figure 1.1 represents the DAG Γ associated with G_1. Observe that there are 16 parse trees (different shifts) embedded in Γ. As an illustration, we present a dotted-parse tree that generates shift $w_1bw_1w_2$, and a dashed-line parse tree that generates shift $w_2w_2bw_1$.

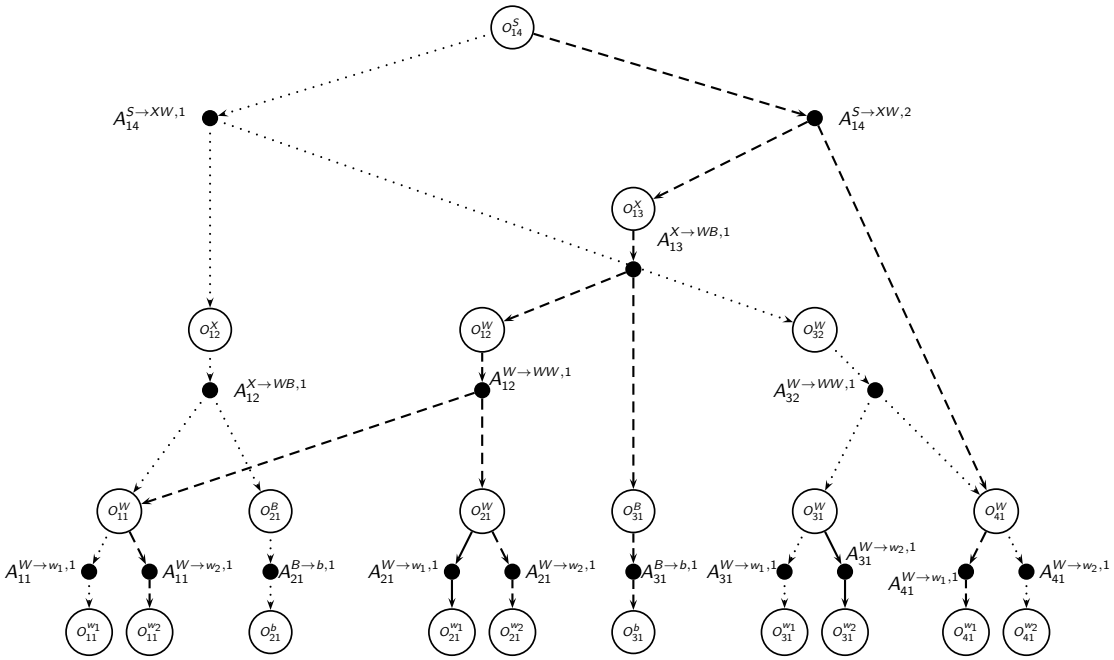


Figure 1.1 DAG Γ on words of length four and two work activities.

Note that the children of the root node ($\{A^{S\to XW,1}_{14},\ A^{S\to XW,2}_{14}\} \in ch(O^S_{1n})$) can be seen as shift "shells" because they do not consider the allocation of specific work activities to the shifts, only the shift start time and its length. Hence, and-nodes $A^{\Pi,k}_{d1n}$ are characterized by their start time $t^{\Pi,k}_{d1n}$, working length $w^{\Pi,k}_{d1n}$, and length including breaks $l^{\Pi,k}_{d1n}$. In Γ_1 and-node $A^{S\to XW,1}_{14}$ generates shift $wbww$, while and-node $A^{S\to XW,2}_{14}$ generates shift $wwbw$. Both shifts have a working length of three time periods, a total length of four time periods and both start at time period one ($i = 1$).

The productions P of a grammar G can be enriched to include more constraints in the composition of words and thus the construction of the associated DAG. These constraints normally appear when there are restrictions on the minimum and maximum length of a work activity and/or break inside of the shift, or when work activities and breaks must be allocated within a time window. The notation $A^{[tws,twe]}_{[lmin,lmax]} \xrightarrow{c} \alpha$ indicates that the subsequence generated from production A should be spanned within the positions $[tws, twe]$, has a length between $[lmin, lmax]$ and if it is used, has a cost of c.

The reader is referred to Sellmann [80] and Côté et al. [30] for a more extensive review on context-free grammars.

1.2 Problem Statement

This section presents the description of the elements that characterize the personnel scheduling problems addressed in this thesis. Those elements include the definition of work rules for the composition of daily shifts and tours, a discussion of the cost factors that might affect the objective function and a description of the nature of employee requirements.

1.2.1 Work Rules for Daily Shift Definition

- *Shift start times*: Under the presence of shift start times, daily shifts can only begin in a subset of time periods (*start time band*). This rule might be due to the existence of work regulations or because the aim is to try to simplify the complexity of the problem. On the other hand, when total flexibility is allowed, shifts can begin at any time of the day allowing enough time to complete their duration.

- *Number of work activities and breaks allocated per shift*: These rules are present when there is a minimum and a maximum number of activities and breaks that can be scheduled per shift. Generally, the number of breaks that must be scheduled into the shift depends on their type (*meal break*, *rest break* or breaks for the *transition* between work activities).

- *Shift, break and work activity length*: In the case of shift length, these rules are present when the length of the shift must fall between a minimum and a maximum number of time periods. For instance, when overtime is not allowed employees must work a maximum number of daily hours depending on workplace agreements. Similarly, the rules related to break length restrict the number of time periods allocated for breaks into the shift. The break length is typically associated with its type (i.e., a lunch break is normally longer than a rest break). Finally, when multiple work activities are considered, each activity has its own rules related to their length into the shift. In this case, the activity length should fall between a minimum and a maximum number of time periods into the shift.

- *Position of breaks*: These rules include the conditions to allocate breaks into the shifts. As an illustration, shifts should not start or end with a rest break, and meal breaks should not be scheduled before a minimum number of consecutive time periods.

- *Transition between activities*: These rules are useful to specify the allowed and forbidden work activity changes into the shifts, as well as the conditions between those changes.

For example, if there is a change from activity A to activity B, there must be a break of one time period between the two activities.

1.2.2 Work Rules for Tour Definition

- *Tour length*: In the same way as for the shift length, these rules are included in the problem when the length of the tour should fall between a minimum and a maximum number of time periods. The tour length is typically restricted by workplace agreements and employee regulations.

- *Rest time between shift changes*: These rules specify the number of rest time periods that must be allocated between consecutive shifts. Normally, the length of the rest time ranges between a minimum and a maximum number of time periods.

- *Days-off*: The rules for the allocation of days-off specify the required number of rest days over a week and the consecutiveness of those days. For instance, an employee must have, at each week, at least two days-off in a row.

- *Early morning and night shifts*: These rules are present when workplace agreements and labor unions restrict the number of shifts that cover early morning and night hours.

1.2.3 Cost Factors in the Objective Function

- *Total cost of the schedule*: When each work schedule has an associated cost, the objective of the personnel scheduling problem might be to select a set of minimum-cost schedules such as a given demand for employee requirements is guaranteed.

- *Total number of employees required*: In an anonymous context, the objective of the personnel scheduling problem might be to minimize the total number of employees required, as long as the demand for employee requirements is met.

- *Penalty for undercovering and overcovering of demand*: *Undercovering* occurs when the number of employees scheduled at a given time period is less than the required number of employees at that time period. On the contrary, *overcovering* occurs when the number of employees scheduled at a given time period is larger than the required number of employees at that time period. In personnel scheduling problems, a penalty cost might be included in the objective function when undercovering and overcovering take place in the schedule.

- *Penalty for transitions between work activities*: In a multi-activity context, a penalty might be added to the cost of the schedule when transitions between work activities occur into the shifts.

1.2.4 Nature of Demand

- *Deterministic and stochastic demand*: Deterministic demand occurs when the number of employees required at each time period of the planning horizon is fully known at the time of planning. On the contrary, employee requirements are uncertain when the quantity of employees required at each time period is a random variable.

1.3 Research Objectives

The primary objective of this thesis is to investigate the integration of the discontinuous tour scheduling problem and the MASSP from a mathematical programming point of view. In order to do that, a number of hybrid mathematical optimization models and specialized solution methods will be developed and tested through extensive computational experiments for large-scale instances. The secondary objective of the thesis is to propose an efficient solution method to solve the integrated problem when employee requirements are uncertain.

The objective is to include as much flexibility as possible in the definition of the problem. Such flexibility will be considered in the daily shift composition, in the tour construction, in the objective function and in the nature of the demand. In the objective function we will include several cost components as the cost of undercovering and overcovering of demand, the cost for the transition between activities and the cost of allocation of work activities to the employees. Stochasticity will be included considering the probability distribution of the employee requirements. Regarding the daily shift construction, we will consider flexible start times, flexible activity and break allocation and flexible activity, break and shift lengths. Finally, for tour construction we will consider flexible assignment of days-off to the employees, as well as the work rules to guarantee the feasibility of tours. Those rules include the constraints for the tour length and for the minimum rest time between consecutive shifts.

1.4 Thesis Outline

The remain of the thesis is divided into seven chapters. Chapter 2 presents the literature review on shift scheduling and tour scheduling problems, as well as the current work on multi-activity shift scheduling and stochastic personnel scheduling problems. Chapter 3 introduces

the general organization of the document. Chapters 4 - 6 present the main body of the thesis composed by three articles submitted to peer review scientific journals. Specifically, Chapter 4 describes a new approach to solve the personalized multi-activity tour scheduling problem. Chapter 5 presents an efficient approach that allows to solve practical instances for the anonymous discontinuous multi-activity tour scheduling problem. Chapter 6 addresses a variant of the anonymous discontinuous multi-activity tour scheduling problem when demand for services is uncertain. A general discussion of the work developed in the thesis in connection with the gaps found in the literature is presented in Chapter 7. Chapter 8 concludes the thesis with an analysis of the results obtained and with a discussion on the limits and recommendations regarding the work presented.

CHAPTER 2 LITERATURE REVIEW

This chapter presents the literature review on the models and methods to solve the SSP, the tour scheduling problem and the MASSP. Some key references on personnel scheduling problems under stochastic demand are also presented.

2.1 Shift Scheduling

The SSP was first introduced by Edie [41] in the context of toll booth operators. Although the author discusses the constraints related to the scheduling of employees, no mathematical model was presented. Later, Dantzig [34] proposes the first integer programming model to solve the SSP. The model is based on a *set covering* formulation where Ω and I represent the sets of feasible shifts and time periods, respectively. Parameter δ_{ij} takes the value of 1 if time period i is a working period in shift j, and assumes value 0 otherwise. The labor requirements at time period i and the labor cost of shift j are denoted by d_i and c_j, respectively. Finally, variable x_j represents the number of employees assigned to shift j. The set covering model for the SSP, denoted as (P1), is as follows:

$$(P1)\min \sum_{j\in\Omega} c_j x_j \tag{2.1}$$

$$\sum_{j\in\Omega} \delta_{ij} x_j \geq d_i,\ \forall\, i \in I, \tag{2.2}$$

$$x_j \geq 0 \text{ and integer},\ \forall\, j \in \Omega. \tag{2.3}$$

The objective of model (P1), (2.1), is to minimize the total labor cost. Constraints (2.2) ensure that demand per time period i are met. Finally, constraints (2.3) define the non-negativity and integrality of the decision variables x_j.

In model (P1), it is assumed that the set of feasible shifts are previously enumerated and that there is a different variable for each shift, even if their start and end times are the same (*explicit model*). In (P1), the incorporation of flexibility (e.g., use of flexible break assignments, different shift lengths or multiple shift start times) may increase the number of variables required, making the complete enumeration of shifts and the solution of the problem intractable. To circumvent this problem, Moondra [60] proposes a method to implicitly represent shifts. The model allows to reduce the number of variables, while considering flexibility

related to multiple shift lengths and start times. Later, Bechtold and Jacobs [12] present an *implicit formulation* (P2) that considers break flexibility in a SSP. In (P2), shifts are grouped into *shift types* according to their start time, length and break window. The computational experiments show that model (P2) has several advantages in terms of computational time and memory requirements when compared with model (P1). The equivalence between (P1) and (P2) was subsequently shown in Bechtold and Jacobs [13].

The formulation of the implicit model (P2) is as follows:

$$(P2)\min \sum_{j\in\Omega} c_j x_j \tag{2.4}$$

$$\sum_{j\in\Omega} \delta_{ij} x_j - \sum_{k\in K} \rho_{ik} b_k \geq d_i,\ \forall\, i \in I, \tag{2.5}$$

$$\sum_{j\in\Omega} x_j - \sum_{k\in K} b_k = 0, \tag{2.6}$$

$$\sum_{k_s\leq k'\leq k} b_{k'} - \sum_{j\in\Omega(k_s,k)} x_j \geq 0,\ \forall\, k \in K^e\backslash\{k_e\}, \tag{2.7}$$

$$\sum_{k\leq k'\leq k_e} b_{k'} - \sum_{j\in\Omega(k,k_e)} x_j \geq 0,\ \forall\, k \in K^s\backslash\{k_s\}, \tag{2.8}$$

$$x_j \geq 0 \text{ and integer},\ \forall\, j \in \Omega, \tag{2.9}$$

$$b_k \geq 0 \text{ and integer},\ \forall\, k \in K. \tag{2.10}$$

Where Ω, I and K are the sets of shift types, time periods and break types, respectively. P_j represents the break window of shift type j, $j \in \Omega(d, f)$ is the set of all shifts types such that their corresponding break windows P_j are completely contained in $[d, f]$, and $\mathcal{K}^s$, $\mathcal{K}^e$ are subsets of $\mathcal{K}$.

$$K^s = \bigcup_{j\in\Omega}\{\min\ P_j\}, \min\ P_j = \min\{k|k \in P_j\}$$
$$K^e = \bigcup_{j\in\Omega}\{\max\ P_j\}, \max\ P_j = \max\{k|k \in P_j\}$$

Parameter δ_{ij} takes the value of 1 if time period i is covered by shift j, regardless if it is a working period or a break period, and assumes value 0 otherwise. Parameter ρ_{ik} takes the value of 1 if break k is allocated in time period i, and assumes value 0 otherwise. The labor requirements at time period i and the labor cost of shift j are denoted by d_i and c_j,

respectively. Finally, variables x_j and b_k represent the number of employees assigned to shift j and break k, respectively.

Model (P2) assumes that every employee receives exactly one break and that the duration of breaks is identical for all shifts. The model is also limited to the discontinuous case and extraordinary overlap (EO) does not exist. EO occurs when there exist two shifts such that the break window for one shift begins strictly earlier and ends strictly later than the break window for the other shift. The objective of the model (P2), (2.4), is to minimize the total labor cost. Constraints (2.5) ensure that labor requirements per time period are met. Constraint (2.6) guarantees that every employee is assigned to exactly one break. Constraints (2.7)-(2.8) are the forward and backward constraints, respectively, and they ensure that the implicit shifts from (P2) are equivalent to the explicit shifts from formulation (P1). Finally, constraints (2.9)-(2.10) set the non-negativity and integrality of variables x_j and b_k.

Thompson [81] combines the works of Moondra [60] and Bechtold and Jacobs [12] to implicitly model meal breaks, but also to schedule rest breaks and allow the use of overtime. Aykin [5] presents an extension of model (2.4)-(2.10) to address the SSP with multiple rests, meal breaks and break windows. The model introduces a new set of integer variables that denote the number of employees assigned to a shift and starting their breaks in different time periods. Computational results show that including break flexibility helps to reduce the workforce requirements. More recently, Aykin [6] compares his previous work with an extension of model (P2). After running several computational experiments including different demand patterns, cyclical and acyclical versions of the problem and different break windows, the author shows that the percentage of problems solved to optimally with the extension of (P2) is substantially higher than those solved with the original formulation (P2).

Addou and Soumis [1] present an approach to implicitly model the SSP without the hypothesis of no extraordinary overlap. Although the model includes a minimal set of extra constraints, computational results show that adding those constraints does not cause any significant increase in the solution times of the model, when compared with (P2).

Rekik et al. [74] extend previous works on implicit modeling of break placement by proposing two implicit models that match shifts with admissible breaks. The authors show that a general reformulation of the forward and backward constraints helps to reduce the density of the constraint matrix without increasing the number of constraints, when compared to the classic forward and backward formulation. Computational experiments show that including multiple breaks and work stretch durations, considerably reduces the workforce size when compared with other formulations. In their following work, Rekik et al. [73] extend the ideas presented in Rekik et al. [74] to consider a multi-day planning horizon. Two solu-

tion approaches are presented: a local branching strategy and a time windowing approach. Computational experiments show that although the local branching approach yields better solutions on small instances, the windowing approach is more likely to find good feasible solutions when more difficult instances are solved.

Because the incorporation of flexibility in the composition of shifts might cause a considerable increase in the number of variables, column generation (CG) has been recently proposed as an alternative to solve complex SSP. In this method, a reduced set of shifts ($\widetilde{\Omega} \subseteq \Omega$) is introduced. Additionally, two different problems are defined: a *restricted master problem* and a *pricing subproblem*. The restricted master problem is typically the linear relaxation of problem (P1) over the reduce set of shifts $j \in \widetilde{\Omega}$. The pricing subproblem is responsible for the generation of new columns (shifts). The formulation of the restricted master problem, denoted as (RMP), is as follows:

$$(RMP) \min \sum_{j \in \widetilde{\Omega}} c_j x_j \tag{2.11}$$

$$\sum_{j \in \widetilde{\Omega}} \delta_{ij} x_j \geq d_i, \; \forall \, i \in \mathcal{I}, \tag{2.12}$$

$$x_j \geq 0, \; \forall \, j \in \widetilde{\Omega}. \tag{2.13}$$

Given a primal $\mathbf{x}$ and a dual $\boldsymbol{\pi}$ solution of (RMP), the reduced cost of any column $\boldsymbol{\delta}_j$ is given by:

$$\bar{c}_j = c_j - \boldsymbol{\pi} \cdot \boldsymbol{\delta}_j \tag{2.14}$$

The pricing subproblem is in charge of finding the column with the least reduced cost. If $\bar{c}_j$ is negative the new column deserves to enter into the RMP. If not, the solution $\mathbf{x}$ to the RMP optimally solves the linear relaxation of (P1).

Two strategies can be adopted in order to reach integrality in the problem. The first strategy is a heuristic approach where model (2.11)-(2.13) is solved by forcing the integrality constraints on $\mathbf{x}$ variables. The integer solution may not be optimal because only a reduced set of shifts ($\widetilde{\Omega} \subseteq \Omega$) has been considered. On the other hand, if the objective is to obtain an exact integer solution to the original problem (P1), the CG procedure should be embedded into a branch-and-price (B&P) framework. In that vein, Mehrotra et al. [59] work with B&P and exploit the advantages of the set covering formulation to solve the SSP with multiple rest breaks, one meal break, and break windows. The authors incorporate upper and lower

bounds on the number of employees needed at each time period and include restrictions on number of employees who can be on a break at any time period. Specialized branching rules are developed to demonstrate that the B&P method is not only superior in terms of computation effort, but also very competitive with alternative methods for the SSP.

Recently, constraint programming (CP) is presented as a new alternative for modeling the SSP where work rules involve non trivial relationships between variables. As an illustration, Pesant [66] takes advantage of *regular languages* to introduce a new global constraint (regular-constraint) that represents substructures commonly found in the SSP. In the approach, the constraint is defined over a *deterministic finite automaton* (DFA), which helps to recognize if the sequence of values taken by the variables belongs to the regular language.

In order to implicitly express a large set of rules and represent all possible patterns for an employee timetabling problem, Côté et al. [29] develop a different version of the regular-constraint by using a 0-1 mixed integer programming model. The approach is tested by using work rules from a real-world timetabling problem. A significant decrease in computational time is found when the model is compared with the classical mixed integer programming (MIP) formulation for the given problem.

The reader is referred to Ernst et al. [43, 44] and Van den Bergh et al. [82] for a comprehensive review and classification of more than 1000 papers about personnel scheduling and rostering.

2.2 Tour Scheduling

Since the introduction of Dantzig's model [34] for the SSP, a large number of research have been conducted in order to consider more realistic and complex versions of the problem. One example of such extensions is the work of Morris and Showalter [61] who, based on Dantzig's set covering model, present the first integer programming formulation to solve the tour scheduling problem over a one-week planning horizon. In their work, the authors combine linear programming (LP) and heuristic methods to solve the tour scheduling problem with a two-phase solution method. In the first phase, daily schedules are generated and sent to the second phase, where weekly tours are composed.

Taking advantage of the discontinuity of the problem, Brusco and Johns [23] propose a model based on a set covering formulation to solve the tour scheduling problem for full-time and part-time employees. A sequential integer programming heuristic is developed in order to solve several instances with different demand characteristics related to the mean and smoothness. Computational results show that the model is able to find a higher percentage of optimal solutions than other heuristic algorithms reported in the literature.

Bard et al. [9] work with a pure integer programming (IP) model to solve the continuous tour scheduling model for full-time and part-time postal service employees over a one-week planning horizon. The model is divided into three components. The first component involves shift scheduling, the second component includes the specification of days-off and the third component allocates lunch breaks to the shifts. Real-world instances including several scenarios with different allocation of days-off, different start times and changes in the ratio between full-time and part-time employees are solved as part of the computational experiments.

Alfares [3] proposes an exact algorithm that uses LP and primal-dual relations to minimize one of two criterias: the total number of employees or the total staffing cost. To assure integer solutions efficiently, a set of lower bounds on the workforce size are introduced as additional constraints in the model. Although model flexibility is introduced in terms of days-off, weekly patterns are previously enumerated.

Trying to introduce flexibility without increasing dramatically the size of the problem, Bailey [7] was the first to present a formulation to implicitly model the start time flexibility in a tour scheduling problem. A construction heuristic is used to assign shifts under a limited staff size constraint while a rounding heuristic is used to achieve integrality in variables. Although the model requires few variables when compared to the set covering formulation, only 8-hour shifts are allowed and no breaks are considered.

Based on the work presented in Bechtold and Jacobs [12], Jarrah et al. [50] propose an implicit model to solve the discontinuous tour scheduling problem. The method starts with an IP formulation to introduce a set of aggregate variables and related cuts. When variables are aggregated, the problem decomposes into seven SSP. A transportation model and a post-processor are used to assign breaks to shifts and shifts to tours, respectively. The method is tested on a real-world problem with several scenarios including flexibility in the start times and break allocation.

Jacobs and Brusco [49] present a compact implicit model that considers the suggestions of Jarrah et al. [50] about start time bands. In order to demonstrate the importance of bandwidth flexibility, the work is tested on a real-world problem in which toll both operators are scheduled. The results show a significant reduction in the required workforce size when compared with a method where the start times are fixed.

Attempting to develop more realistic models, Brusco and Jacobs [20] incorporate both start time and meal break flexibility in an IP model to solve the continuous tour scheduling problem. Computational results on a real-world problem show that the effect of the scheduling policies in the optimal workforce size can vary significantly depending on the level of other policies. In a recent work, Brusco and Jacobs [21] develop a large experimental study to

prove that only a small subset of start times is needed to find the optimal workforce size in tour scheduling problems. The model used for the study is based on an implicit formulation over a one-week planning horizon where breaks are not considered.

To solve the tour scheduling problem in a call center, Çezik et al. [26] propose a model with two components: a daily shift generator and a days-off generator. In the former component, seven daily shift schedules are generated through implicit modeling. In the latter component, the weekly requirements are met via network flows. The model is able to solve the discontinuous version of a tour scheduling problem when the start times of any two consecutive shifts do not differ by more than a specified bound. Computational experiments show that high-quality integer solutions are found when a heuristic branch-and-bound (B&B) is used (the values of most of the variables are fixed before starting with enumeration).

CG and decomposition techniques have also been used to solve tour scheduling problems. While CG is designed to solve problems in which there is a large number of variables, Benders decomposition (BD) takes advantage of special block structures to decompose the problem into smaller ones, easier to solve. Consider the following problem:

$$(P3)\ \min \mathbf{c}^T\mathbf{x} + \mathbf{f}^T\mathbf{y} \tag{2.15}$$

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = b, \tag{2.16}$$

$$\mathbf{x} \in \mathbb{R}_+^{n1}, \mathbf{y} \in Y \subseteq \mathbb{R}_+^{n2}. \tag{2.17}$$

Where $\mathbf{c} \in \mathbb{R}^{n1}$, $\mathbf{f} \in \mathbb{R}^{n2}$, $\mathbf{b} \in \mathbb{R}^{m1}$, Y is a polyhedron, $\mathbf{A} \in \mathbb{R}^{n1xm1}$ and $\mathbf{B} \in \mathbb{R}^{n2xm1}$. Suppose that y variables are complicating variables, in the sense that the problem becomes significantly easier to solve if y variables are fixed. BD partitions problem (P3) into two problems: 1) a *Benders master problem* and 2) a *Benders subproblem*. The Benders master problem includes the complicating y variables, while the Benders subproblem contains the x variables. Observe that problem (2.15)-(2.17) can be written in terms of the y variables as follows:

$$\min \mathbf{f}^T\mathbf{y} + Q(\mathbf{y}) \tag{2.18}$$

$$\mathbf{y} \in Y. \tag{2.19}$$

Where $Q(\mathbf{y})$ is defined to be the optimal value of:

$$Q(\mathbf{y}) = \min \mathbf{c}^T\mathbf{x} \tag{2.20}$$

$$\mathbf{A}\mathbf{x} = b - \mathbf{B}\mathbf{y}, \tag{2.21}$$

$$\mathbf{x} \in \mathbb{R}^{n1}_{+}. \tag{2.22}$$

Define $\mathbf{\Pi} = \{\pi \in \mathbb{R}^{m1} | \mathbf{A}^T\pi \leq \mathbf{c}\}$ and let $\mathbf{\Lambda}$ be the sets of extreme points associated with $\mathbf{\Pi}$ and let $\mathbf{\Phi}$ define the set of extreme rays of cone $C = \{\pi \in \mathbb{R}^{m1} | \mathbf{A}^T\pi \leq 0\}$. Applying an outer linearization to function $Q(\mathbf{y})$ it is possible to reformulate the original problem as the following equivalent Benders master problem:

$$\min \mathbf{f}^T\mathbf{y} + q \tag{2.23}$$

$$(b - \mathbf{B}\mathbf{y})^T\lambda \leq q \quad \forall \lambda \in \Lambda, \tag{2.24}$$

$$(b - \mathbf{B}\mathbf{y})^T\phi \leq 0 \quad \forall \phi \in \Phi, \tag{2.25}$$

$$\mathbf{y} \in Y,\ q \text{ free}. \tag{2.26}$$

Constraints (2.24) are called optimality cuts since they define the objective value associated with feasible values of y. Constraints (2.25) are called feasibility cuts since they eliminate values of y for which function $Q(\mathbf{y})$ is unbounded. Optimality cuts and feasibility cuts do not have to be exhaustively enumerated, since only a subset of them will be active at the optimal solution of the problem. Hence, an iterative algorithm [46] can be used to generate only the subset of cuts that are sufficient to identify an optimal solution of the original problem.

An illustration on the use of BD to solve the tour scheduling problem is presented in Rekik et al. [72]. The authors use BD to prove: 1) the correctness of the forward and backward constraints introduced by Bechtold and Jacobs [12]; 2) in the presence of break window or start time extraordinary overlap, forward and backward constraints are not sufficient, but Benders decomposition can be used to solve the problem. After conducting an extensive analysis, the authors conclude that the proposed model considerably decreases the number of variables, at the cost of a small increase in the number of constraints.

Ni and Abeledo [62] present a CG approach to solve the continuous tour scheduling problem. Two different pricing subproblems are considered: one in which weekly tours are decomposed into daily shifts and another one in which weekly tours are decomposed by start times. To solve the problem to optimality, new branching rules are implemented. Computational results show that in comparison with an implicit model, the proposed CG method shows a better

performance when evaluated on large-scale instances.

Because of achieving integrality for large-scale instances is a difficult task, Al-Yakoob and Sherali [2] propose a heuristic CG to schedule, over a one week planning horizon, different categories of employees. The model includes employee preferences for work centers (that can be seen as work activities), for shift types and for the allocation of days-off. Although shift types are previously enumerated, the method is able to generate, in a reasonable amount of time, employee schedules for up to 90 work centers and 336 employees.

Combining implicit and explicit shift definitions and developing an exact B&P algorithm, Brunner and Bard [17] solve a discontinuous tour scheduling problem over a one week planning horizon for postal service employees that are divided into 2 groups: full-time and part-time employees. A constrained-based formulation with shifts explicitly enumerated is used to solve the problem for the full-time employees, while a constraint-based model that exploits the idea of implicit shift construction, is used to solve the problem for the part-time employees. The authors analyze the flexibility impact on workforce size, costs and utilization rate by considering several scenarios in the computational experiments.

Brunner and Stolletz [18] present a stabilized B&P algorithm to solve a discontinuous tour scheduling problem that includes flexibility regarding labor regulations and assignment of lunch breaks. The master problem is based on a set covering formulation. Computational experiments show that the convergence of the model is faster when the stabilization technique is used.

Rather than focusing on traditional LP methods, some authors have developed different approaches to handle the tour scheduling problem. In particular, Loucks and Jacobs [56] formulate the weekly tour scheduling problem as an 0-1 integer goal programming model where the primary objective is to minimize the overstaffing. Although flexibility in terms of shift length is included in the model, meal breaks are not scheduled. A heuristic approach composed by a construction phase and an improvement phase, is adopted as a solution method. Computational results show that the approach is able to provide good solutions in a reasonable amount of time.

In a recent work, Brusco and Johns [24] introduce an integrated approach to overcome the lack of integration between start time selection and tour construction presented in Brusco et al. [22] and Brusco and Jacobs [19]. While tabu search is used for the shift start time selection, a cutting plane method is used for tour construction. Results on instances with and without consistency in demand patterns show that the method has a good performance in terms of computational time.

The reader is referred to Alfares [4] for a complete revision, according to modeling and solution techniques, of more than 100 papers about tour scheduling problems.

2.3 Multi-Activity Shift Scheduling

Although shift scheduling problems have been extensively studied in the literature, only recently the problem in a multi-activity context has begun draw some attention. One of the first attempts to solve the problem is presented in Ritzman et al. [78]. The authors develop a heuristic approach as an integration of a construction method and a simulation component to solve a multi-day, multi-activity SSP in a post office. Although the authors tackled the multi-activity context, they do not consider breaks nor rules related to switching between activities.

More recently, Demassey et al. [36] present a CG algorithm base on CP as a way to model complex regulation constraints that are present in large-scale instances for the MASSP. The authors introduce the *cost-regular* constraint to find shifts with negative reduced cost. Computational experiments on several real-world instances from a retail store, allow to obtain lower bounds that can be used as benchmarks for future work. The previous approach is extended in Demassey et al. [37] with a method that is able to send to the restricted master problem several shifts with negative reduced cost. The approach is tested on a complex real-world instance for the anonymous MASSP as well as on generated benchmark instances. Although the method is efficient to solve the LP relaxation of the problem, when integrality constraints are imposed integer solutions are found only for small instances (less than three work activities).

Quimper and Rousseau [69] solve the MASSP, by using specialized graph structures that are derived from formal languages and solved via large neighborhood search. Computational results show that the method achieves near-optimal solutions for instances with one work activity and that it is able to scale well for instances with up to ten work activities.

Extending the ideas presented in their previous work [29], Côté et al. [28] suggest two approaches for the solution of the MASSP: the use of an automata to derive a network flow model and the use of context-free grammars to obtain MIP models in which an and/or graph structure is used. After running several instances, the results show that the new formulations lead to faster computation times when compared to compact assignment MIP formulations. Nevertheless, when all work rules are encoded in a complete grammar the computational time tends to deteriorate because of symmetry issues.

Trying to solve the scalability issues of [29, 28] and using context-free grammars to model

the work rules for the construction of shifts, Côté et al. [30] propose an implicit formulation to solve the anonymous MASSP. The authors address model symmetry by using integer variables which express variable sums (Orbital shrinking, Fischetti and Liberti [45]). Their work is compared with other modeling approaches reported in the literature for mono-activity and multi-activity instances. In the mono-activity case the authors show that solution times of their model are comparable and sometimes superior to the ones reported in the literature. In the multi-activity case they find that their model can solve to optimality instances with up to ten work activities.

Côté et al. [31] present a grammar-based CG approach to solve the personalized version of the problem introduced in [30]. A classical set covering formulation is used for the master problem. The pricing subproblems are formulated using grammars and solved with a dynamic programming algorithm. A B&P algorithm with different branching rules is presented. Computational results show that: 1) the proposed algorithm exhibits superior performance on some classes of instances, when compared with the approaches presented in Demassey et al. [37], Côté et al. [28, 30] and Lequy et al. [55]; 2) although the expressiveness of grammars enables to encode a large set of work rules over shifts, some limitations are present regarding shift total length over long planning horizons (e.g., one week).

Boyer et al. [16] extends the work in Côté et al. [31], where besides considering multiple work activities, it also includes multiple tasks. An activity can be interrupted and can be assigned to several employees at the same time. On the contrary, tasks must be executed without interruption by a fixed number of employees. An extensive study of branching strategies is made, showing that the method is able to find, in a reasonable amount of time, the solution for all test instances with an optimality gap lower than 5%.

Instead of working with explicit and implicit models or with CP, some authors have proposed different methods based on heuristics or decomposition techniques to solve the MASSP. With that intention, Detienne et al. [38] present three different methods to solve the personalized version of the problem over a one week planning horizon: a Lagrangian lower bound, a heuristic based on a cut generation process and an exact method based on BD. The models do not consider work rules for the assignment of work activities to shifts and weekly patterns are previously defined. Computational experiments on several real-world and randomly generated instances show that: 1) the quality of the bounds found with the Lagrangian method were practically equal to the bounds of the theoretic continuous relaxation; 2) fast good-quality solutions were found with the heuristic method; 3) the performance of the exact methods was not the best in terms of computational time, when compared with the heuristic methods.

Restrepo et al. [77] develop a heuristic CG approach to solve the anonymous MASSP. The

pricing subproblem is modeled as a shortest path problem with resource constraints, where most of the work rules for shift composition are tackled while building the graph. The method is able to find near-optimal solutions for large-scale real-world instances with up to 16 work activities. Although a multi-day planning horizon was considered, the allocation of days-off was not included in the formulation of the problem.

Dahmen and Rekik [33] propose a heuristic based on tabu search and B&B to solve the personalized MASSP over a multiple-day planning horizon. Although the constraints related to the minimum and maximum number of working hours per week are considered, days-off are previously assigned to the employees.

When sequences of work, rest days, shift types, and breaks are already fixed, the MASSP becomes an activity assignment problem where the objective is to assign work activities to the shifts such that the demand of activities is satisfied over the planning horizon. In an attempt to address this problem, Lequy et al. [55] present three integer programming models. The first model is based on a multi-commodity network flow problem, the second model is based on an integer block model, while the third model corresponds to a CG model. In the method, shifts and breaks are fixed and previously assigned to the employees. After developing different heuristic solution methods for solving instances with up to 100 shifts per day and 15 work activities over a one week planning horizon, the authors conclude that the heuristic CG method embedded into a rolling horizon procedure provides the best results in general and that is able to find high-quality solutions in a reasonable time.

Finally, Elahipanah et al. [42] propose a two-phase heuristic model for the activity and task assignment problem. The first-phase uses an approximate MIP model to set temporary shift patterns and to allocate tasks. The second-phase uses a heuristic CG to fix the shifts and to allocate the work activities to them. Flexibility options regarding the composition of shifts are considered in the computational experiments. The authors test their method with randomly generated instances with up to 6 work activities over a one week planning horizon, finding good solutions in a reasonable amount of time.

2.4 Stochastic Personnel Scheduling Problems

Different models and solution approaches have been proposed in the literature to deal with stochastic employee requirements in personnel scheduling problems. As an illustration, Easton and Rossin [40] and Easton and Mansour [39] develop heuristic methods that aim to tackle problems where demand for employees is uncertain. Easton and Rossin [40] propose a tabu search method to solve a stochastic goal programming model that integrates and

optimizes labor requirements and employee scheduling. Easton and Mansour [39] present a genetic algorithm to solve the SSP in which the recourse decisions are related to the under-covering and overcovering of demand. Although both approaches aim to solve problems over a one week planning horizon, employee patterns are previously defined and only a small set of stochastic scenarios is considered. In a similar way, Bard et al. [10] propose a heuristic two-stage model that addresses the tour scheduling problem over a one week planning horizon. First-stage variables are related to the number of full-time and part-time employees hired, while second-stage decisions correspond to the allocation of the employees to specific shifts during the week. Computational experiments on real instances that consider three stochastic scenarios (high, medium and low demand) show that significant savings are likely when the recourse problem is used.

Some studies that use CG and decomposition approaches have been recently proposed as alternatives to solve workforce planning problems when employee requirements are uncertain. In his Ph.D. thesis, Wang [83] uses stochastic optimization models to solve two problems: a generalized assignment problem with uncertain resource capacity and unknown processing times, and a SSP with unknown demand. The SSP includes full-time and part-time employees as well as overtime and temporary workers to recover feasibility when demand is higher than expected. The author proposes a CG model embedded in a B&P procedure to solve the corresponding stochastic integer programming models. Computational experiments show that although the model fails to converge in several hours for large instances, the stochastic model provides some advantages over its deterministic counterpart.

More recently, Pacqueau and Soumis [63] propose a heuristic two-stage stochastic programming model to solve the SSP. The proposed model is based on a decomposition of Aykin's [5] implicit model, where first-stage variables are associated with the allocation of full-time shifts to the employees and recourse decisions correspond to hiring part-time employees, using overtime for full-time shifts, the allocation of breaks and the allowance of understaffing.

Punnakitikashem et al. [68] introduce a stochastic nurse scheduling problem that aims to minimize staffing costs and excess workload. The authors present a Benders decomposition approach, a Lagrangian relaxation with a Benders decomposition approach and a nested Benders decomposition approach as solution methods. Computational results suggest that simultaneously considering nurse staffing and assignment is more desirable than doing them sequentially.

Kim and Mehrotra [53] present an integrated staffing and scheduling approach applied to nurse management when demand is uncertain. In this method, the problem is formulated as a two-stage stochastic integer program, where daily shifts and weekly patterns are previously

enumerated. First-stage decisions correspond to the number of employees assigned to daily shifts and to weekly patterns, while second-stage decisions correspond to: 1) the possibility of adding or canceling daily shifts for every working pattern; 2) allowing undercovering or overcovering of employee requirements. A set of valid mixed-integer rounding inequalities that describe the convex hull of feasible solutions in the second-stage problem are included. Consequently the integrality of the second-stage decision variables can be relaxed. Computational experiments show that the use of the stochastic model prevents the hospital from being overstaffed.

An L-shaped method is presented in Robbins and Harrison [79] to solve a combined server-sizing and staff scheduling problem for call centers in which a service level agreement must be satisfied. First-stage decisions correspond to the employee staffing, while second-stage decisions correspond to the computation of a telephone service shortfall. Computational results show that ignoring variability is a costly decision, since the value of the stochastic solution for the model is substantially high.

Very limited literature is available on stochastic workforce planning for employees that have the skills to work in different work activities, different tasks or different unit departments. Zhu and Sherali [85] address a workforce planning problem for employees with multiple skills between service centers. A two-stage model under demand fluctuations is presented, where first-stage decisions correspond to personnel recruiting and allocation of employees to multiple locations, while second-stage decisions consists in reassigning the workforce among the locations.

The scheduling of cross-trained workers in a multi-department service environment with random demand is addressed in Campbell [25]. The author presents a two-stage model decomposable by days and by scenarios, where first-stage decisions are related to the scheduling of days-off and second-stage decisions correspond to the allocation of available employees at the beginning of each day. In the approach, days-off are previously defined and only a small number of scenarios is considered (10 in total).

Parisio and Jones [65] present a two-stage stochastic model for a multi-skill tour scheduling problem in retail outlets where first-stage variables are associated with the assignment of employees to weekly schedules, while recourse decisions correspond to the allocation of overtime and to the undercovering and overcovering of demand. Although multiple work activities are included in the problem, the authors assume that employees can only work in one activity per daily shift.

The reader is referred to Defraeye and Van Nieuwenhuyse [35] for a state-of-the-art literature review on staffing and scheduling problems that account for nonstationary demand.

CHAPTER 3 GENERAL ORGANIZATION OF THE DOCUMENT

The literature review on personnel scheduling problems reveals that no method has been proposed to integrate the tour scheduling and the multi-activity shift scheduling problems. In particular, no rules for the allocation and transition between work activities and for the composition of employee patterns were ever simultaneously taken into account in the existing research. The literature review on multi-activity shift scheduling shows that some authors have addressed these problems over planning horizons longer than one day, but only in situations where either weekly patterns were previously defined or rules concerning total tour length and days-off were not considered. The present thesis aims to address these gaps by proposing models and methods that integrate the discontinuous tour scheduling problem and the MASSP.

The thesis is divided into three parts corresponding to three different articles. Each article integrates the development of new ideas with the use of previous material to solve the MASSP. In particular, this work presents hybrid approaches that combine the use of formal languages and decomposition methods to solve the large-scale discontinuous MATSP when demand is deterministic and stochastic. Chapter 4 presents two B&P algorithms to solve the personalized MATSP in a discontinuous environment. Because practical problems might include a large number of employees, Chapter 5 presents an approach that aims to solve the scalability and symmetry issues associated with the employee dimension. This approach combines BD and CG to solve the discontinuous MATSP when employees have the same skills. Chapter 6 presents an extension of the work from Chapter 5 that aims to address the stochastic version of the MATSP. The three articles form a logical sequence because they first propose and develop some ideas to solve the integration of the tour scheduling problem with the MASSP. Then, those ideas are adapted and extended to more practical and realistic contexts.

Chapter 7 presents a general discussion about the methodological aspects of the thesis and its results, linked with the gaps found in the literature review from Chapter 2. Chapter 8 is divided into three parts. The first part presents a discussion on the contributions achieved by the thesis in the area of personnel scheduling problems. The second part presents an analysis of the limitations and constraints of the work developed. The third part discusses some recommendations and possible future work.

CHAPTER 4 ARTICLE 1: BRANCH-AND-PRICE FOR PERSONALIZED MULTI-ACTIVITY TOUR SCHEDULING

In this chapter of the thesis, we present an approach that integrates the discontinuous tour scheduling problem with the MASSP for employees with different skills. We introduce two formulations for the problem: the *Daily-based formulation* and the *Tour-based formulation*. The Daily-based formulation is an extension of the work presented in Côté et al. [31], where columns correspond to daily shifts and tours are assembled in the master problem by means of extra constraints. In the Tour-based formulation columns correspond to employee tours. We show that the Tour-based formulation is stronger in terms of its LP relaxation bound, when compared with the Daily-based formulation.

We implemented a B&P algorithm for each formulation. In both approaches the master problem is modeled as a generalized set partitioning problem. The columns for the Daily-based formulation are modeled with context-free grammars. In particular, since the planning horizon is at least one week and the personalized version of the problem is being considered, we build a grammar for each employee and each day. The work rules for the composition of shifts and the employee availability and skills are included in the construction of each grammar. A directed acyclic graph (DAG) with an and/or structure is then derived from each grammar. Additionally, each DAG contains all the possible shifts that a given employee can perform at a given day. New columns (shifts) with negative reduced cost are found by using a dynamic programming algorithm over each DAG. New columns (tours) for the Tour-based formulation are found with an exact two-phase procedure. In the first phase, daily shifts are composed in the same way as for the daily-shift formulation. In the second phase, daily shifts are assembled into tours by first building a directed acyclic graph ($G^e(\mathcal{N}, \mathcal{A})$), where the set of nodes correspond to daily shifts and the set of arcs represent feasible connections between shifts (two shifts are connected if the rules for the allocation of days-off and the rules for the rest time between shifts are met). Second, a label setting algorithm for the resource-constrained shortest-path problem over graph $G^e(\mathcal{N}, \mathcal{A})$ is used to find negative reduced cost paths (columns). In order to find integer solutions for the problem, we propose a branching rule that preserves the structure of the pricing subproblems.

Finally, we evaluate and compare the two methods on randomly generated instances for the mono-activity and the multi-activity context. The comparison allows us to show that the Daily-based formulation works better when fast-heuristic integer solutions are needed, but when exact solutions are required, the Tour-based formulation exhibits a better performance.

Additionally, we compared the Tour-based formulation on a mono-activity problem presented in Brunner and Bard [17]. The experiments suggested that the solution times and quality of our formulation are comparable with the solution times and quality reported by Brunner and Bard [17].

The next article was accepted for publication in *INFORMS Journal on Computing* in October 2015.

Branch-and-Price for Personalized Multi-Activity Tour Scheduling

María I. Restrepo
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, Canada

Bernard Gendron
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département d'informatique et de recherche opérationelle, Université de Montréal, Montréal, Canada

Louis-Martin Rousseau
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, Canada

Abstract. This paper presents a branch-and-price approach to solve personalized tour scheduling problems in a multi-activity context. Two formulations are considered. In the first formulation, columns correspond to daily shifts that are modeled with context-free grammars and tours are assembled in the master problem by means of extra constraints. In the second formulation, columns correspond to tours that are built in a two-phase procedure. The first phase involves the composition of daily shifts, while the second phase assembles those shifts to generate tours using a shortest path problem with resource constraints. Both formulations are flexible enough to allow different start times, lengths and days-off patterns, as well as multiple breaks, continuity and discontinuity in labor requirements. We present computational experiments on problems dealing with up to five work activities and one week planning horizon. The results show that the second formulation is stronger in terms of its lower bound and that it is able to find high-quality solutions for all instances with an integrality gap lower than 1%.

Keywords. Multi-activity tour scheduling problem, Branch-and-price, Context-free grammars, Shortest path problem with resource constraints.

4.1 Introduction

Personnel scheduling problems consist in constructing a set of feasible shift schedules and assigning them to the company staff to satisfy a given demand for staff requirements. These problems arise in diverse organizations such as hospitals, airline companies, retail stores, call centers and banks, where they have become important tasks, due to the necessity to achieve a better level of service, to reduce staff costs and to increase employee satisfaction.

According to Baker [8], three main categories of problems can be distinguished in personnel scheduling: *Shift scheduling* (SSP), *Days-off scheduling* and *Tour scheduling*. The first category deals with the specification of work and rest time periods to assign to shifts, as well as the selection of a set of those shifts to satisfy the demand for staff requirements. In shift scheduling, the *planning horizon* is usually one day divided into *time periods* of equal length. The second category involves the selection of *days-off* over a planning horizon of at least one week. Such selection is usually restricted by some *employee preferences* or *workplace agreements*. The last category includes problems that arise from the integration of shift scheduling and days-off scheduling; therefore, the aim of tour scheduling problems is to specify the time periods of the day and the days of the week in which employees must work.

Complex extensions of classical personnel scheduling problems appear when real applications are considered. For instance, when more than one activity has to be scheduled, the *Multi-activity shift scheduling* and the *Multi-activity tour scheduling* problems arise. In both extensions not only the specification of work and rest time periods is necessary, but also the assignment of activities to the shifts. In a multi-activity context, specific characteristics related to *work rules*, *workplace agreements*, and *employee skills* and *preferences* define the rules to build employee schedules.

The problem considered in this paper is the *personalized multi-activity tour scheduling problem* (MATSP). In the MATSP, a tour can be seen as a schedule over a planning horizon of at least one week, where for every time period it must be specified if the employee is working on an activity, having a break or resting. Aside from multiple activities, undercovering and overcovering of demand are considered. The MATSP is flexible enough to be easily adapted depending on different work rules and scenarios. The number of feasible tours grows fast with the number of activities, the number of employees and the length of the time horizon, making the complete enumeration of tours impractical. Therefore, we propose, to the best of our knowledge, the first exact method to solve personalized multi-activity tour scheduling problems. The approach is based on a column generation procedure embedded into a branch-and-price (B&P) method. Two formulations for the MATSP are considered, each giving rise

to a different B&P algorithm. First, the *Daily-based* formulation consists of an extension of a multi-activity shift scheduling problem, where columns correspond to daily shifts and tours are assembled in the master problem by means of extra constraints. Second, the *Tour-based* formulation, where columns correspond to tours that are built in a two-phase procedure. The first phase involves the composition of daily shifts, while the second phase assembles those shifts to generate tours.

The outline of the paper is the following. Section 4.2 presents the literature review related to personnel scheduling problems, as well as some background material. The definition of the problem, the two formulations and some properties of them are presented in Section 4.3. The B&P algorithms for the Daily-based formulation and the Tour-based formulation are presented in Sections 4.4 and 4.5, respectively. Computational experiments are discussed in Section 4.6. Finally, Section 4.7 presents the concluding remarks and future work.

4.2 Background and Related Research

In this section, we present a literature review on the models and methods proposed to solve shift scheduling, multi-activity and tour scheduling problems. Then, we present an introduction on the use of grammars in the context of multi-activity shift scheduling.

4.2.1 Shift Scheduling

Two different modeling approaches can be distinguished in the literature on shift scheduling problems: *explicit* and *implicit* models. Explicit models allow to consider flexibility in terms of break placement, start times and shift length by representing each feasible shift with a different variable. On the contrary, implicit models define one variable for every shift and break type, seeking to reduce the number of decision variables by compromising model flexibility.

Dantzig [34] was the first author to introduce an explicit model for the SSP. The model is based on a *set covering* formulation in which the objective is to minimize the total labor cost, ensuring that labor requirements at every time period are met. Trying to reduce the number of variables, Moondra [60] proposes a method that implicitly represents shifts and considers flexibility regarding multiple shift lengths and start times. Break flexibility is considered in Bechtold and Jacobs [12] with an implicit formulation where shifts are grouped into shift types according to their start time, length and break window.

Aykin [5] and Rekik et al. [74] present two extensions of Bechtold and Jacobs' formulation. In the first extension, the author tackles multiple rests, meal breaks and break windows by

introducing integer variables for the number of employees assigned to a shift and starting their breaks at different time periods. The second extension deals with two implicit models that include a reformulation of forward and backward constraints and a transportation structure to match shifts with admissible breaks.

Column generation (CG) and constraint programming (CP) have been recently proposed as alternatives to solve complex shift scheduling problems. CG is used when the incorporation of flexibility in the composition of shifts causes a considerable increase in the number of variables. In this method, two strategies can be adopted to reach integrality in the problem: a heuristic approach, where the master problem is solved by forcing the integrality constraints on the decision variables, and an exact method, where the CG procedure is embedded into a B&P algorithm. In that vein, Mehrotra et al. [59] use B&P and exploit the advantages of a set covering formulation to solve a shift scheduling problem with multiple rest breaks, one meal break and break windows. On the other hand, CP is used to model shift scheduling problems where work rules involve non trivial relationships between variables. As an illustration, Côté et al. [29] take advantage of the expressiveness of a *Deterministic finite automaton* (DFA) that, used in a 0-1 mixed integer programming model, helps to implicitly express a large set of rules and represent all possible patterns for an employee timetabling problem.

4.2.2 Multi-Activity Shift Scheduling

In one of the first attempts to solve multi-activity shift scheduling problems, Ritzman et al. [78] develop a heuristic approach by integrating a construction method and a simulation component to schedule employees in a post office over a planning horizon of one week. Although the authors tackled the multi-activity context, they do not consider breaks nor rules related to switching between activities.

A few decades later, the idea of using context-free grammars in the context of CP to solve combinatorial problems was introduced in Pesant [66], Sellmann [80] and Quimper and Walsh [70]. The first author [66] presents the *Regular* constraint that forces a sequence of characters to belong to a regular language. The *Grammar* constraint is introduced in Sellmann [80] and Quimper and Walsh [70]. This constraint forces a sequence of characters to belong to a context-free language. Later, Kadioglu and Sellmann [51] and Kadioglu and Sellmann [52] present and modify a memory- and time-efficient filtering algorithm for context-free grammars under the absence and presence of a linear objective function.

The previous ideas have been applied in the context of multi-activity shift scheduling in Demassey et al. [37], Quimper and Rousseau [69] and Côté et al. [28, 30]. In the first approach [37], a CP-based column generation algorithm is presented as a way to model complex regula-

tion constraints to solve large instances of multi-activity shift scheduling problems. Quimper and Rousseau [69] solve multi-activity shift scheduling problems with up to ten work activities by using specialized graph structures that are derived from formal languages and solved via Large neighborhood search. Côté et al. [28] propose two models: one that uses an automaton to derive a network flow model, and another one that benefits from context-free grammars to obtain a MIP model in which an and/or graph structure is used. Despite their ability to easily handle complex rules, the models present some scalability issues when the number of employees and the number of activities increase. Côté et al. [30] seek to solve the scalability issues of their previous models by introducing an implicit formulation that encapsulates model symmetry by using integer variables. The authors show, based on the work of Pesant et al. [67], that the proposed integer programming model has a LP relaxation bound equivalent to that of the classical set covering model. Computational results suggest that, in the mono-activity case, the solution times of the model are comparable and sometimes better than the results presented in the literature and that, in the multi-activity case, the model is able to solve to optimality instances with up to ten work activities.

To solve the personalized version of the problem introduced in Côté et al. [30], Côté et al. [31] present a grammar-based column generation method where the pricing subproblems are formulated using grammars and solved with a dynamic programming algorithm. Although the expressiveness of grammars enables to encode a large set of work rules over shifts, some limitations are present regarding shift total length over longer planning horizons (e.g., one week). Boyer et al. [16] extends the previous work, where besides considering multiple activities, it also includes multiple tasks. An extensive study of branching strategies is made, showing that the method is able to find, in a reasonable amount of time, the solution for all test instances with an integrality gap lower than 5%.

Instead of working with explicit and implicit models or with CP, some authors have proposed different methods to tackle multi-activity shift scheduling problems. Some of them include Tabu search [33], heuristic column generation [77], decomposition techniques [38], or a simplification of the multi-activity shift scheduling problem [42, 55] by fixing sequences of work, rest days, shift types and breaks.

4.2.3 Tour Scheduling

Since the introduction of Dantzig's model for shift scheduling [34], a lot of research has been conducted to consider more realistic and complex versions of the problem. One example of such extensions is the work of Morris and Showalter [61], which introduces the first integer programming formulation to solve tour scheduling problems based on Dantzig's set covering

model. The authors combine LP and heuristic methods to solve a tour scheduling problem with a two-phase solution method. In the first phase, daily schedules are generated and sent to the second phase where weekly tours are constructed. Trying to introduce flexibility without increasing dramatically the size of the problem, Bailey [7] presents a formulation to implicitly model the start times in a tour scheduling problem. A construction heuristic is used to assign shifts under a limited staff size constraint, while a rounding heuristic is used to achieve integrality of the variables.

Jarrah et al. [50] propose an implicit model with aggregated variables that decomposes the weekly tour scheduling problem into seven daily-shift scheduling subproblems. A transportation model and a post-processor are used to assign breaks to shifts and shifts to tours, respectively. The authors test their method on a real-world application, running several scenarios including flexibility in start times and break allocation. Jacobs and Brusco [49] present a compact implicit model to demonstrate the importance of start time bandwidth flexibility in tour scheduling. Computational results suggest that allowing start-time flexibility reduces significantly the required workforce size when compared with fixed start times.

Attempting to develop more realistic models, Brusco and Jacobs [20] consider both start time and meal break flexibility in an implicit integer programming model to solve the continuous tour scheduling problem. The authors evaluate their approach on a real-world application, showing that the effect of the scheduling policies on the optimal workforce size can vary significantly depending on the level of other policies.

In order to solve a discontinuous tour scheduling problem (shifts are allowed to overlap from one day to the next) in a call center, Çezik et al. [26] propose a model with two components: a daily-shift generator and a days-off generator. In the former, seven daily shift schedules are generated through implicit modeling, while in the latter, weekly requirements are met via network flows. The proposed model is able to solve half of the real instances to optimality by using a heuristic B&P algorithm.

In a recent work, Brusco and Johns [24] introduce an integrated approach to overcome the lack of integration between start time selection and tour construction by using Tabu search and a cutting plane method. The former is useful for the start time selection, while the latter handles tour construction. Computational experiments are conducted with and without consistency in demand patterns. In both cases, the method has a good performance in terms of computational time.

Decomposition techniques and column generation are also often used in the context of tour scheduling problems. As an illustration, Rekik et al. [72] use Benders decomposition to solve a continuous tour scheduling problem where the subproblems are modeled with a transportation

structure, and the forward and backward constraints introduced by Bechtold and Jacobs [12] are used as a set of initial feasibility cuts. After conducting an extensive analysis, the authors conclude that the proposed model considerably decreases the number of variables at the cost of a small increase in the number of constraints. Ni and Abeledo [62] present a B&P algorithm to solve the continuous version of the problem where weekly tours are decomposed into daily shifts and start-time patterns. Computational experiments show that for large-scale instances where implicit methods often fail to find a feasible solution, the proposed method is able to find near-optimal solutions.

Because achieving integrality for large-scale instances is a difficult task, Al-Yakoob and Sherali [2] propose a heuristic column generation to schedule, over one week, different categories of employees. Employee preferences for work centers, shift types and days-off are considered. Given previously defined shift types, the method is able to generate, in a reasonable amount of time, employee schedules for up to 90 stations and 336 employees.

Combining implicit and explicit shift definitions and developing an exact B&P algorithm, Brunner and Bard [17] solve a discontinuous tour scheduling problem over a one-week planning horizon for postal service employees. The authors analyze the flexibility impact on workforce size, costs and utilization rate by considering several scenarios in the computational experiments.

Brunner and Stolletz [18] present a stabilized B&P algorithm to solve a discontinuous tour scheduling problem that includes flexibility regarding labor regulations and assignment of lunch breaks. The master problem is based on a set covering formulation. Computational experiments show that the model convergence is faster when the stabilization technique is used.

The reader is referred to Van den Bergh et al. [82] for a comprehensive review of recent papers on tour scheduling problems.

The literature review on tour scheduling problems reveals that no method has been proposed to integrate tour scheduling and multi-activity shift scheduling problems. In particular, no rules for the allocation and transition between activities were ever taken into account in the existing research. The literature review on multi-activity shift scheduling problem (Section 4.2.2) shows that some authors have addressed these problems over planning horizons longer than one day, but only in situations where either weekly patterns were previously defined or rules concerning total tour length and days-off were not considered. Additionally, the previous works on multi-activity shift scheduling show that although the expressiveness of grammars enables to encode a large set of work rules over shifts, some limitations are present regarding total shift length over longer planning horizons (e.g., one week).

The present paper addresses these gaps in the literature by proposing models and methods that integrate the tour scheduling and multi-activity shift scheduling problems. We include ideas from previous works on multi-activity shift scheduling, specifically we make use of context-free grammars to efficiently handle the multi-activity context, but we propose a new approach to tackle longer planning horizons and include work rules and flexibility for the composition of employee tours. The use of grammars in the context of shift scheduling is reviewed next.

4.2.4 Grammars and Shift Scheduling Problems

We define a *context-free grammar* as a tuple $G = \langle \Sigma, N, P, S \rangle$ where Σ is an alphabet of characters called the *terminal symbols*, N is a set of *non-terminal symbols*, $S \in N$ is the starting symbol and P is a set of *productions* represented as $A \to w$, where $A \in N$ is a non-terminal symbol and w is a sequence of terminal and non-terminal symbols. The productions of a grammar can be used to generate new symbol sequences until only terminal symbols are part of the sequence. A *Context-free language* is the set of sequences accepted by a context-free grammar.

A *parse tree* is a tree where each leaf is labeled with a terminal and each inner-node is labeled with a non-terminal. A grammar recognizes a sequence if and only if there exists a parse tree where the leaves, when listed from left to right, reproduce the sequence. An *and/or graph* is a graph where each leaf corresponds to an assignment that can either be true or false. An and-node is true if all of its children are true. An or-node is true if one of its children is true. The root is true if the grammar accepts the sequence encoded by the leaves. The and/or graph embeds every possible parse tree of a grammar.

Finally, a *DAG* Γ is a directed acyclic graph that embeds all parse trees associated with words of a given length n recognized by a grammar. The DAG Γ has an and/or structure where the and-nodes represent productions from P and or-nodes represent non-terminals from N and letters from Σ. The DAG Γ is built by a procedure proposed in Quimper and Walsh [71].

In the shift scheduling context, the use of grammars allows both to include work rules regarding the definition of shifts and to handle the multi-activity context in an easy way. Thus, feasible shifts can be represented as words in a context-free language. For example, words $rw_1w_1bw_2$ and $w_1bw_2w_1r$ are recognized as valid sequences in a two-activity shift scheduling problem where letters w_1, w_2, b and r represent working on activity 1, working on activity 2, break and rest periods, respectively. The time horizon consists of five time periods, shifts have a length of four time periods and must contain exactly one break of one time period that can be placed anywhere during the shift except at the first or the last

period. The grammar that defines the shifts on this example follows:

$G = (\Sigma = (w_1, w_2, b, r), N = (S, F, X, W, B, R), P, S)$,
where productions P are: $S \to RF|FR$, $F \to XW$, $X \to WB$, $W \to WW|w_1|w_2$, $B \to b$, $R \to r$, and symbol $|$ specifies the choice of production.

In the previous example, productions $W \to w_1$, $W \to w_2$, $B \to b$ and $R \to r$ generate the terminal symbols associated with working on activity 1, working on activity 2, having a break or having a rest time period inside of the shift, respectively. Production $W \to WW$ generates two non-terminal symbols, W, meaning that the shift will include a working subsequence. Production $X \to WB$ means that the shift will include working time and then it will be followed by a break. Production $F \to XW$ generates a subsequence of length four (the daily shift), which includes working time followed by a break to finish with more working time. Finally, the last two productions are $S \to RF$ and $S \to FR$. The former generates a sequence starting with a time period of rest followed by the daily shift. The latter generates a sequence starting with the daily shift followed by a time period of rest.

Let O^{π}_{dil} be the or-nodes associated with $\pi \in N \cup \Sigma$, i.e., with non-terminals from N or letters from Σ, that generate a subsequence at day d, position i of length l. Note that if $\pi \in \Sigma$, the node is a leaf and l is equal to one. On the contrary, if $\pi \in N$, the node represents a non-terminal symbol and $l > 1$. $A^{\Pi,k}_{dil}$ is the kth and-node representing productions $\Pi \in P$ generating a subsequence at day d, from position i of length l. There are as many $A^{\Pi,k}_{dil}$ nodes as there are ways of using P to generate a sequence of length l from position i during day d. The sets of or-nodes and and-nodes from day d are denoted by O_d and A_d, respectively. The root node is described by O^{S}_{d1n} and its children by $A^{\Pi,k}_{d1n}$. Figure 4.1 represents the DAG Γ associated with the grammar of the example, where dashed-line or-nodes are part of the parse trees associated with and-node $A^{S \to RF,1}_{15}$, while continuous-line or-nodes are part of the parse trees associated with and-node $A^{S \to FR,1}_{15}$. For clarity, we did not include the subscript of the day in the notation of the nodes.

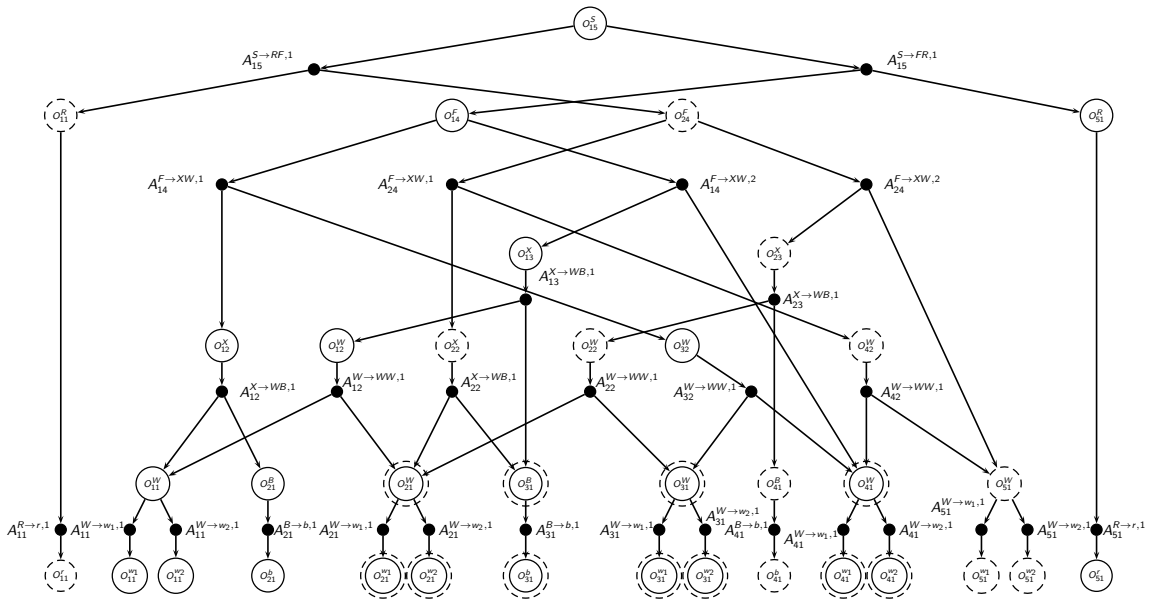
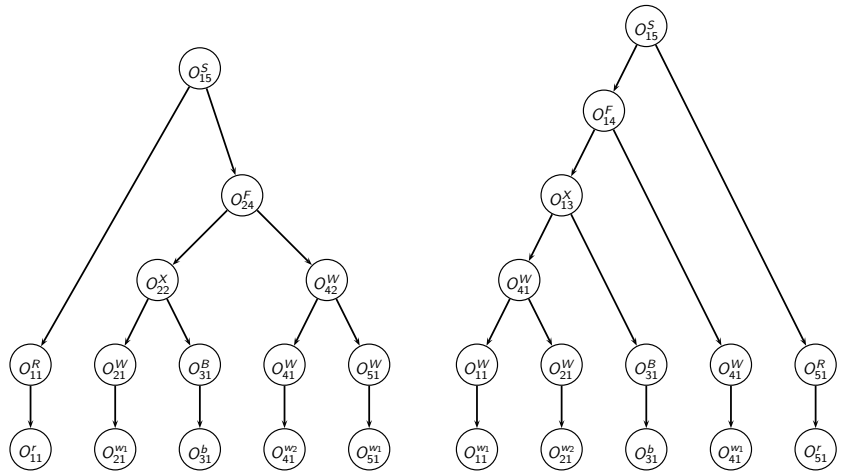


Figure 4.1 DAG Γ on words of length five and two work activities.

Figure 4.2 shows two of the 32 parse trees that are embedded into the DAG Γ presented in Figure 4.1. Note that the leaves correspond to letters from Σ that form a word, in this case of length five, when listed from left to right.



(a) Word: $rw_1bw_2w_1$ (b) Word: $w_1w_2bw_1r$

Figure 4.2 Parse trees derived from DAG Γ on words of length five and two work activities.

4.3 Problem Definition and Formulations

In this section, we first introduce the problem studied and its notation to later define the mathematical models for the Daily-based and the Tour-based approaches. We then show that the Tour-based formulation yields a better linear programming (LP) relaxation bound than the Daily-based formulation.

4.3.1 Problem Definition and Notation

The problem addressed in this paper is a Tour scheduling problem in a multi-activity context where the set of activities is denoted by J. The planning horizon is at least one week in which each day $d \in D = \{1, ..., |D|\}$ is divided into $I_d = \{1, ..., |I_d|\}$ time periods of equal length. Each employee $e \in E$ has different skills, meaning that the personalized version of the problem is considered. The set of feasible tours for each employee e is denoted by $\mathcal{T}^e$, while the set of feasible shifts for each employee e at each day d is denoted by $\mathcal{S}^e_d$. The notation used in both formulations is as follows:

General parameters

b_{dij}: staff requirements for day d, time period i and activity j;

c^e_{dij}: cost of employee e working on day d, time period i and activity j;

c_{tr}: transition cost between two work activities;

c^+_{dij}, c^-_{dij}: overcovering and undercovering costs of employee requirements for day d, time period i and activity j, respectively;

J^e_{di}: set of work activities that employee e can perform at time period i of day d.

Daily shift parameters

ϵ: minimum rest time between two consecutive daily shifts;

$\mathcal{S}^e_{di}(1)$: set of shifts that finish at time period i during day d for employee e;

$\mathcal{S}^e_{di}(2)$: set of shifts whose start time period goes from 1 up to time period $i + \epsilon - |I_d|$ at day d for employee e ($i > |I_d| - \epsilon$);

δ^e_{dijs}: parameter that takes value 1 if shift s covers time period i and work activity j during day d for employee e, and assumes value 0 otherwise;

c^e_{ds}: cost associated with shift s during day d for employee e (includes de working cost c^e_{dij} and the transition cost c_{tr})

Every daily shift s from day d has a set of attributes: its start time period t_{ds}, its length l_{ds} (considering breaks), its working time w_{ds}, and its end time period $f_{ds} = t_{ds} + l_{ds} - 1$.

Tour parameters

Δ_l, Δ_u: minimum and maximum number of working days in a tour, respectively;

Θ_l, Θ_u: minimum and maximum tour working length in time periods, respectively;

$\Phi = |D| - \Delta_l$: maximum number of days-off in a tour;

ϕ^e_{dst}: parameter that takes value 1 if tour t includes shift s at day d for employee e, and assumes value 0 otherwise;

ρ^e_{dijt}: parameter that takes value 1 if tour t covers time period i and work activity j at day d for employee e, and assumes value 0 otherwise ($\rho^e_{dijt} = \sum_{s \in \mathcal{S}^e_d} \phi^e_{dst} \delta^e_{dijs}$);

c^e_t: cost of tour t for employee e ($c^e_t = \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J^e_{di}} \delta^e_{dijt} c^e_{dij} = \sum_{d \in D} \sum_{s \in \mathcal{S}^e_d} \phi^e_{dst} c^e_{ds}$).

4.3.2 The Daily-Based Formulation

In order to solve the MATSP, we propose, as a first formulation, a natural extension of the model presented in Côté et al. [31]. In this extension, daily shifts are linked into tours by means of extra constraints that assure the minimum and maximum tour length, the minimum and maximum number of working days per tour and the minimum rest time between consecutive shifts. We define the following variables:

x^e_{ds}: binary variable that takes value 1 if shift s on day d is assigned to employee e, and assumes value 0 otherwise;
y^+_{dij}, y^-_{dij}: slack variables representing overcovering and undercovering of employee requirements for day d, time period i and activity j, respectively.

The Daily-based formulation, denoted $F_{\mathcal{S}}$, is as follows:

$$f(F_{\mathcal{S}}) = \min \sum_{e \in E} \sum_{d \in D} \sum_{s \in \mathcal{S}_d^e} c_{ds}^e x_{ds}^e + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij}^+ y_{dij}^+ + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij}^- y_{dij}^- \tag{4.1}$$

$$\sum_{e \in E} \sum_{s \in \mathcal{S}_d^e} \delta_{dijs}^e x_{ds}^e - y_{dij}^+ + y_{dij}^- = b_{dij}, \; \forall\, d \in D, \; i \in I_d, \; j \in J, \tag{4.2}$$

$$\sum_{s \in \mathcal{S}_d^e} x_{ds}^e \leq 1, \; \forall\, e \in E, \; \forall\, d \in D, \tag{4.3}$$

$$\Delta_l \leq \sum_{d \in D} \sum_{s \in \mathcal{S}_d^e} x_{ds}^e \leq \Delta_u, \; \forall\, e \in E, \tag{4.4}$$

$$\Theta_l \leq \sum_{d \in D} \sum_{s \in \mathcal{S}_d^e} w_{ds} x_{ds}^e \leq \Theta_u, \; \forall\, e \in E, \tag{4.5}$$

$$\sum_{s \in \mathcal{S}_{di}^e(1)} x_{ds}^e + \sum_{s \in \mathcal{S}_{d+1i}^e(2)} x_{d+1s}^e \leq 1, \; \forall\, e \in E, \; d \in D \backslash \{|D|\}, i = |I_d| - \epsilon + 1, ..., |I_d|, \tag{4.6}$$

$$x_{ds}^e \in \{0, 1\}, \; \forall\, e \in E, \; d \in D, \; s \in \mathcal{S}_d^e, \tag{4.7}$$

$$y_{dij}^+, \; y_{dij}^- \geq 0, \; \forall\, d \in D, \; i \in I_d, \; j \in J. \tag{4.8}$$

The objective of $F_{\mathcal{S}}$, (4.1), is to minimize the total staffing cost plus the penalization for overcovering and undercovering of demand. Constraints (4.2) ensure that the total number of employees working on day $d \in D$, time period $i \in I_d$ and work activity $j \in J$ is equal to the demand subject to some adjustments related to undercovering and undercovering. Constraints (4.3) guarantee that every employee is assigned to at most one shift per day. Constraints (4.4) and (4.5) enforce a minimum and maximum number of working days (Δ_l and Δ_u) and tour length (Θ_l and Θ_u), respectively. Constraints (4.6) ensure a minimum rest time between consecutive shifts. Finally, constraints (4.7)-(4.8) set the binary nature of variables x_{ds}^e and the non-negativity of variables y_{dij}^+, y_{dij}^-.

4.3.3 The Tour-Based Formulation

The Tour-based formulation makes use of slack variables as in the Daily-based formulation, but also includes the following decision variables related to tours:

x_t^e: binary variable that takes value 1 if tour t is assigned to employee e, and assumes value 0 otherwise.

Hence, the constraints related to the link between daily shifts and tours are considered in the definition of feasible tours and not in the mathematical model. The Tour-based formulation, denoted $F_{\mathcal{T}}$, is as follows:

$$f(F_{\mathcal{T}}) = \min \sum_{e \in E} \sum_{t \in \mathcal{T}^e} c_t^e x_t^e + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij}^+ y_{dij}^+ + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij}^- y_{dij}^- \tag{4.9}$$

$$\sum_{e \in E} \sum_{t \in \mathcal{T}^e} \rho_{dijt}^e x_t^e - y_{dij}^+ + y_{dij}^- = b_{dij}, \; \forall\, d \in D, \; i \in I_d, \; j \in J, \tag{4.10}$$

$$\sum_{t \in \mathcal{T}^e} x_t^e = 1, \; \forall\, e \in E, \tag{4.11}$$

$$x_t^e \in \{0, 1\}, \; \forall\, e \in E, \; t \in \mathcal{T}^e, \tag{4.12}$$

$$y_{dij}^+, \; y_{dij}^- \geq 0, \; \forall\, d \in D, \; i \in I_d, \; j \in J. \tag{4.13}$$

The objective of $F_{\mathcal{T}}$, (4.9), is to minimize the total staffing cost plus the penalization for overcovering and undercovering of demand. Constraints (4.10) ensure that the total number of employees working on day $d \in D$, time period $i \in I_d$ and work activity $j \in J$ is equal to the demand subject to some adjustments related to undercovering and undercovering. Constraints (4.11) guarantee that every employee is assigned to exactly one tour. Finally, constraints (4.12)-(4.13) set the binary nature of variables x_t^e and the non-negativity of variables y_{dij}^+, y_{dij}^-.

4.3.4 Comparison Between the Two Formulations

Let $f(\overline{F_{\mathcal{S}}})$ and $f(\overline{F_{\mathcal{T}}})$ be the LP relaxation bounds of the Daily-based formulation and the Tour-based formulation, respectively, with $\overline{F_{\mathcal{S}}}$ and $\overline{F_{\mathcal{T}}}$ the corresponding LP relaxations.

Proposition 1: $f(\overline{F_{\mathcal{S}}}) \leq f(\overline{F_{\mathcal{T}}})$.

Proof: Consider the Daily-based formulation (4.1)-(4.8) to which we add the following redundant constraints:

$$y_{dij}^+, \; y_{dij}^- \leq b_{dij}, \; \forall\, d \in D, \; i \in I_d, \; j \in J. \tag{4.14}$$

Note that the LP relaxation of the resulting model is equivalent to $\overline{F_{\mathcal{S}}}$. Using the resulting model, we dualize constraints (4.2) to obtain the following Lagrangian subproblem, denoted by $F_{\mathcal{S}}(\beta)$, where β_{dij} are the Lagrange multipliers associated with constraints (4.2):

$$f(F_{\mathcal{S}}(\beta)) = \sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J}\beta_{dij}b_{dij} +$$

$$\min \sum_{e\in E}\sum_{d\in D}\sum_{s\in\mathcal{S}_d^e} c_{ds}^e x_{ds}^e - \sum_{e\in E}\sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J}\sum_{s\in\mathcal{S}_d^e}\beta_{dij}\delta_{dijs}^e x_{ds}^e$$

$$\sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J}(c_{dij}^- - \beta_{dij})y_{dij}^- + \sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J}(c_{dij}^+ + \beta_{dij})y_{dij}^+$$

subject to constraints (4.3)-(4.8) and (4.14).

Note that $F_{\mathcal{S}}(\beta)$ decomposes into two subproblems: one expressed in terms of the x variables only, with feasible set X, and the other involving only the y variables, with feasible set Y. The subproblem that depends only on the x variables is defined by:

$$\min \sum_{e\in E}\sum_{d\in D}\sum_{s\in\mathcal{S}_d^e}\left(c_{ds}^e - \sum_{i\in I_d}\sum_{j\in J}\beta_{dij}\delta_{dijs}^e\right)x_{ds}^e$$

subject to constraints (4.3)-(4.7).

This subproblem can itself be decomposed by employee and its feasible set is defined as the finite set of points $X = \Pi_{e\in E}X^e$, where X^e corresponds to (4.3)-(4.7) for each employee e. The subproblem involving only the y variables is defined by:

$$\min \sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J}(c_{dij}^- - \beta_{dij})y_{dij}^- + \sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J}(c_{dij}^+ + \beta_{dij})y_{dij}^+$$

subject to the non-negativity constraints (4.8) and the bound constraints (4.14).

Therefore, the feasible set $X \times Y$ of the Lagrangian subproblem $F_{\mathcal{S}}(\beta)$ can be written as $X \times Y = (\Pi_{e\in E}X^e)\times Y$.The corresponding Lagrangian dual can be written as $Z = \max_\beta f(F_{\mathcal{S}}(\beta))$. By Lagrangian duality theory [47], this Lagrangian dual is equivalent to the following LP model:

$$
\begin{aligned}
Z = \ \min \ & \sum_{e \in E} \sum_{d \in D} \sum_{s \in \mathcal{S}_d^e} c_{ds}^e x_{ds}^e + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij}^+ y_{dij}^+ + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij}^- y_{dij}^- \\
& \sum_{e \in E} \sum_{s \in \mathcal{S}_d^e} \delta_{dijs}^e x_{ds}^e - y_{dij}^+ + y_{dij}^- = b_{dij}, \ \forall \, d \in D, \ i \in I_d, \ j \in J, \\
& (x, y) \in conv(X \times Y).
\end{aligned}
$$

where $conv(A)$ is the convex hull of any set A. Clearly, $Z \geq f(\overline{F_{\mathcal{S}}})$, since $conv(X \times Y)$ is contained in the feasible set of $\overline{F_{\mathcal{S}}}$.

Since $X \times Y = (\Pi_{e \in E} X^e) \times Y$ and Y is a bounded polyhedron, we have $conv(X \times Y) = (\Pi_{e \in E} conv(X^e)) \times Y$. Now, for any employee $e \in E$, every point x^e in $conv(X^e)$ can be written as a convex combination of the extreme points of $conv(X^e)$. Any such extreme point $\xi^e(t)$ corresponds to a tour $\mathcal{T}^e$ for employee e. Thus, we can write $x^e = \sum_{t \in \mathcal{T}^e} \mu^e(t) \xi^e(t)$, where $\mu^e(t)$ is the convex combination weight associated to tour $t \in \mathcal{T}^e$, i.e., $\sum\limits_{t \in \mathcal{T}^e} \mu^e(t) = 1, \forall \, e \in E$ and $\mu^e(t) \geq 0, \ \forall \, t \in \mathcal{T}^e, \ e \in E$. Thus, we have:

$$
\begin{aligned}
Z = \min & \sum_{e \in E} \sum_{t \in \mathcal{T}^e} \left(\sum_{d \in D} \sum_{s \in \mathcal{S}_d^e} c_{ds}^e \xi_{ds}^e(t) \right) \mu^e(t) + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij}^+ y_{dij}^+ + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij}^- y_{dij}^- \\
& \sum_{e \in E} \sum_{t \in \mathcal{T}^e} \left(\sum_{s \in \mathcal{S}_d^e} \delta_{dijs}^e \xi_{ds}^e(t) \right) \mu^e(t) - y_{dij}^+ + y_{dij}^- = b_{dij}, \ \forall \, d \in D, \ i \in I_d, \ j \in J, \\
& \sum_{t \in \mathcal{T}^e} \mu^e(t) = 1, \ \forall \, e \in E, \\
& \mu^e(t) \geq 0, \ \forall \, t \in \mathcal{T}^e, e \in E, \\
& 0 \leq y_{dij}^+, \ y_{dij}^- \leq b_{dij}, \ \forall \, d \in D, \ i \in I_d, \ j \in J.
\end{aligned}
$$

We remark that $\xi_{ds}^e(t) = \phi_{dst}^e$, for any $e \in E$, $d \in D$,$s \in \mathcal{S}_d^e$ and $t \in \mathcal{T}^e$. If we let $\mu^e(t) = x_t^e$, for each $t \in \mathcal{T}^e$ and $e \in E$, we obtain the LP relaxation model of the Tour-based formulation (4.9)-(4.13) with the redundant constraints (4.14). Thus, $f(\overline{F_{\mathcal{T}}}) = Z \geq f(\overline{F_{\mathcal{S}}})$. $\square$.

Appendix A presents an example where a feasible solution of $\overline{F_{\mathcal{S}}}$ is not feasible for $\overline{F_{\mathcal{T}}}$, which implies that, in this case, the optimal LP relaxation value of the Daily-based formulation is lower than the optimal LP relaxation value of the Tour-based formulation.

4.4 B&P for the Daily-Based Formulation

In this section, we present a B&P algorithm for the Daily-based formulation. First, we introduce the restricted master problem. Second, we describe the pricing subproblems. Finally, we present the branching rule used to find near optimal integer solutions.

4.4.1 Restricted Master Problem

In practical scenarios, the complete enumeration of the set of feasible shifts $\mathcal{S}_d^e$ for every employee e and every day d is intractable due to the incorporation of flexibility regarding shift length, break placement, shift start times, among others. Therefore, we define a restricted master problem $F_{\widetilde{\mathcal{S}}}$ as the LP relaxation of problem (4.1)-(4.8) over restricted sets of shifts $\widetilde{\mathcal{S}}_d^e \subseteq \mathcal{S}_d^e$.

Let β_{dij}, λ_d^e, δ^e, θ^e and α_{di}^e be the dual variables associated with constraints (4.2) to (4.6), respectively. The reduced cost $\bar{c}_{ds}^e$ of column (shift) s during day d for employee e is given by:

$$\bar{c}_{ds}^e = \sum_{i \in I_d} \sum_{j \in J_{di}^e} (c_{dij}^e - \beta_{dij})\delta_{dijs}^e - \lambda_d^e - \delta^e - w_{ds}\theta^e - \alpha_{df_{ds}}^e - \sum_{i=|I_{d-1}|-\epsilon+t_{ds}}^{|I_{d-1}|} \alpha_{d-1i}^e. \tag{4.15}$$

The objective in the column generation procedure is to find, at every iteration, columns with negative reduced cost, i.e., $\bar{c}_{ds}^e < 0$. Hence, (4.15) is used in the objective function of the pricing subproblems for every employee and every day of the planning horizon. Under a B&P algorithm, if no negative reduced cost columns can be generated, the solution to the restricted master problem is optimal for the node under evaluation.

4.4.2 Pricing Subproblems for Daily Shift Generation

New columns are generated based on the work presented in Côté et al. [31] and the concepts of Section 4.2.4. For each employee e and day d, a grammar G_d^e is generated. Such grammar represents the possible shifts that an employee can perform during the day and its generation is made by considering the employee skills, his preferences, availability, work rules and regulations such as minimum and maximum shift length, start times, minimum and maximum number of activities and breaks per shift, position of breaks and rules for the transition between activities.

From each grammar G_d^e, we generate a DAG Γ_d^e that is used to find negative reduced cost shifts for every employee e and day d. In Γ_d^e each child of the root node $A_{d1n}^{\pi,k} \in ch(O_{d1n}^S)$ represents a daily shift and its corresponding reduced cost is computed by using the dynamic programming algorithm presented in Quimper and Rousseau [69]. The algorithm traverses the DAG from the leaves to the root node by summing up the costs of the children of the and-nodes and choosing the minimum cost child of the or-nodes. It should be noted that, at every iteration of the column generation, the cost of every node in Γ_d^e has to be updated. If the node is a leaf that corresponds to working on activity j at time period i during day d, its cost is updated with $c_{dij}^e - \beta_{dij}$; if the node is an and-node $A_{d1n}^{\pi,k} \in ch(O_{d1n}^S)$ its cost is updated with $(-\lambda_d^e - \delta^e - w_{ds}\gamma^e - \alpha_{df_{ds}}^e - \sum_{i=|I_{d-1}|-\epsilon+t_{ds}}^{|I_{d-1}|} \alpha_{d-1i}^e)$; otherwise, the cost is zero.

4.4.3 Branching Rule

We use the branching rule suggested in Côté et al. [31], where at each node of the B&P algorithm, we select the employee e' such that one of its shift variables, $x_{ds}^{e'}$, holds the largest fractional value in the optimal LP solution of the node under evaluation. For that employee, we select two daily shifts on the same day d, $s_d^{e'}(1)$ and $s_d^{e'}(2)$, corresponding to the associated variables with the largest fractional values. Then, we identify the first divergent time perido i' between $s_d^{e'}(1)$ and $s_d^{e'}(2)$, meaning that both shifts differ in their activities (work-activity, rest or break). Let $j(1) \in J$ and $j(2) \in J$ be the assigned activities at time period i' for shifts $s_d^{e'}(1)$ and $s_d^{e'}(2)$, respectively. As mentioned before, $J_{di'}^{e'}$ is the set of activities (in this case including also rests and breaks) that employee e' can perform at time period i' during day d. A partition of $J_{di'}^{e'}$ into subsets $J_{di'}^{e'}(1)$ and $J_{di'}^{e'}(2)$ is created, such that $j(l) \in J_{di'}^{e'}(l)$, for $l = 1, 2$. The rest of the activities in $J_{di'}^{e'}$ are equally distributed between the two partitions. After generating the partitions, two nodes are created. At each node $l = 1, 2$ it is ensured that employee e' does not perform the activities in $J_{di'}^{e'}(l)$ at time period i'. The rule is easily handled in the subproblem, since if an activity j is forbidden at time period i during day d, the associated leaf O_{di1}^j receives a large cost in the DAG Γ_d^e ensuring that tours containing shifts with activity j at time period i for day d will not be generated. Therefore, the suggested branching rule preserves the structure of the pricing subproblem.

4.5 B&P for the Tour-Based Formulation

In this section, we present a B&P algorithm for the Tour-based formulation. First, we introduce the restricted master problem. Second, we describe a two-phase procedure to solve the pricing subproblems. Finally, we present the branching rule used to identify integer

solutions.

4.5.1 Restricted Master Problem

The complete enumeration of the set of feasible tours $\mathcal{T}^e$ for every employee is intractable due to the incorporation of shift and tour flexibility. Hence, we define a restricted master problem $F_{\widetilde{\mathcal{T}}}$ as the LP relaxation of problem (4.9)-(4.13) over restricted sets of tours $\widetilde{\mathcal{T}}^e \subseteq \mathcal{T}^e$.

Let β_{dij} and σ^e be the dual variables associated with constraints (4.10) and (4.11), respectively. The reduced cost of column (tour) t for employee e is given by:

$$\bar{c}_t^e = \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J_{di}^e} (c_{dij}^e - \beta_{dij}) \rho_{dijt}^e - \sigma^e. \tag{4.16}$$

Expression (4.16) is used in the objective function of the pricing subproblems for every employee. Under a B&P algorithm, if a new column with negative reduced cost is found, the column is added to the restricted master problem $F_{\widetilde{\mathcal{T}}}$. If no negative reduced columns can be generated, the solution of $F_{\widetilde{\mathcal{T}}}$ is optimal for the node under evaluation.

4.5.2 Pricing Subproblems for Tour Generation

Tours are composed of a combination of daily shifts and days-off over a time horizon of at least one week. Considering the above, the approach presented in this section seeks to build tours with a two-phase process. In the first phase, daily shifts are generated with the same procedure introduced in Section 4.4.2, except for a change in the cost of every node in Γ_d^e. In this case, if the node is a leaf that corresponds to working on activity j at time period i from day d, its cost is $c_{dij}^e - \beta_{dij}$, otherwise the cost corresponds to zero. In the second phase, the proposed method assembles shifts into tours by considering the constraints related to the tour length, the number of working days and the minimum rest time between consecutive shifts. Thus, in this second phase of the subproblem, the multi-activity property does not affect the assembling of feasible tours and the only attributes that must be considered are the shift starting time and shift length. The two-phase procedure is exact because, as it will be described further, at every iteration of the column generation, we include the shift with the lowest reduced cost for every start time and length at each day in the planning horizon.

The objective in the second phase of the subproblem is to generate tours with negative reduced cost for every employee. Taking this into consideration, let $\mathcal{K}^e = \bigcup_{d \in D} A_{d1n}^{\pi,k} \in ch(O_{d1n}^S)$

be the union over the set of days of all the children of the root node O^S_{d1n} of Γ^e_d. Therefore, $\mathcal{K}^e$ can be seen as the set of shift "shells" containing all the possible combinations of shift start times and lengths but without considering breaks and work activity allocation. Let $G^e(\mathcal{N}, \mathcal{A})$ be a directed acyclic graph with a set of nodes $\mathcal{N} = \{v_k \mid k \in \mathcal{K}^e \cup \{v_s, v_f\}\}$, where v_k corresponds to shift k for employee e, while v_s and v_f are the source and sink nodes, respectively. The set of arcs $\mathcal{A}$ is divided into three types: arcs going from the source node to a "shift" node $\mathcal{A}_1 = \{(v_s, v_k) \mid v_k \in \mathcal{N}, d_k \leq \Phi + 1\}$; arcs connecting two "shift" nodes $\mathcal{A}_2 = \{(v_k, v_{k'}) \mid v_k, v_{k'} \in \mathcal{N}, k \neq k', t_{k'} - f_k \geq \epsilon, d_{k'} - d_k \leq \Phi + 1\}$; and arcs connecting a "shift" node to the sink, $\mathcal{A}_3 = \{(v_k, v_f) \mid v_k \in \mathcal{N}, d_k \geq \Delta_l\}$; d_k is the corresponding day of shift k.

Each node in graph $G^e(\mathcal{N}, \mathcal{A})$ has, besides a list of immediate successors $\mathcal{N}(v_k) = \{v_i \in \mathcal{N} | (k, i) \in \mathcal{A}\}$, a set of attributes (start time period t_k, length l_k, considering breaks, working time w_k, end time period $f_k = t_k + l_k - 1$, and reduced cost $\bar{c}_k$) inherited from its corresponding daily shift. The source node has a cost equal to zero, the sink node has a cost equal to the negative of the dual variable σ^e and the remaining nodes have a cost equal to $\bar{c}_k$. The list of successors of each node is generated according to the work rules for tour composition, employee preferences and availability, as expressed in the arc types definition. Thus, successors of source node v_s are nodes that, depending on their start day, allow enough time to meet the constraints related to the minimum number of working days required in a tour. In the same way, sink node v_f is a successor of node v_k if the finish day of v_k is greater or equal to the minimum number of working days required in a tour. Finally, a node $v_{k'}$ is a successor of node v_k if first, its start time allows to guarantee that there is a minimum rest time between both shifts and secondly, if its start day allows to meet the constraints related to the minimum and maximum number of days-off and their consecutiveness (i.e. two days off in a row).

Tours are resource constrained paths along graph $G^e(\mathcal{N}, \mathcal{A})$. In this case, if the cost of a tour is negative, that means that a column with negative reduced cost was found and it deserves to be sent to the restricted master problem. It should be noted that every feasible path should meet the constraints related to the resources considered in the problem: the total tour length and the number of working days. Furthermore, the method is optimal, since graph $G^e(\mathcal{N}, \mathcal{A})$ contains the shifts with the lowest reduced cost for all the possible shift structures.

Figure 4.3 presents an example of graph $G^e(\mathcal{N}, \mathcal{A})$. In the graph, the number of days in the planning horizon is 7, the minimum and maximum number of working days are 5 and 6, respectively, the minimum and maximum tour length are 15 and 18, respectively, and shifts are built with the DAG Γ^e_d presented in Figure 4.1. Nodes with an odd number represent

shifts obtained from and-node $A_{d15}^{S\rightarrow RF,1}$ that generates the structures $rwbww$, $rwwbw$, whereas nodes with an even number represent shifts obtained from and-node $A_{d15}^{F\rightarrow FR,1}$ with structures $wbwwr$, $wwbwr$. In this case, for every and-node there are sixteen possible shifts, where $rw_1bw_1w_1$, $rw_1bw_1w_2$,..., $rw_1w_2bw_2$,...,$rw_2w_2bw_2$ are some possible shifts for the first structure and $w_1bw_1w_1r$, $w_1bw_1w_2r$,..., $w_1w_2bw_2r$,...,$w_2w_2bw_2r$ are some possible shifts for the second. The shift considered at every node is the one with the minimum reduced cost.

For clarity, Figure 4.3 does not include all the possible arcs, but three examples of paths that consider every arc type are presented: a path with a total length of 18 working time periods and one day-off (dashed path): $s-1-3-5-8-10-12-e$, a path with a total length of 15 time periods and two days-off in a row (dotted path): $s-2-4-6-8-13-e$ and path with a total length of 15 time periods and two nonconsecutive days-off (bold path): $s-3-7-9-11-14-e$.

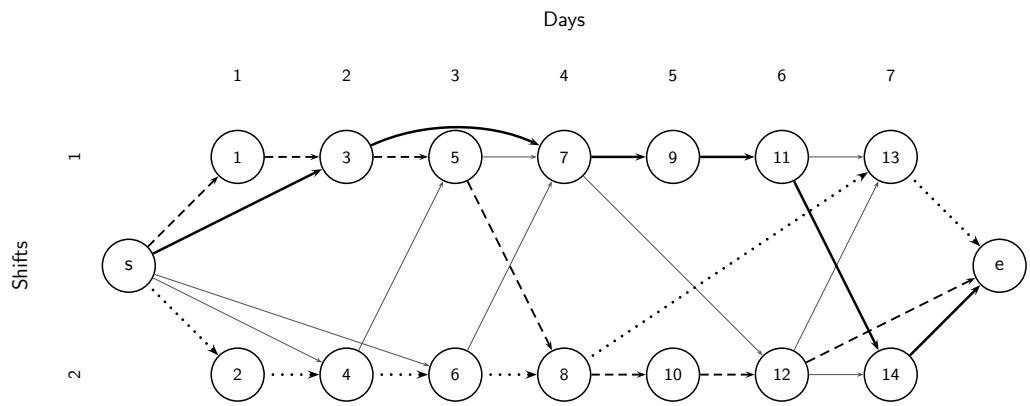


Figure 4.3 $G^e(\mathcal{N},\mathcal{A})$ over a planning horizon of seven days.

Two processes are considered in the solution method for the tour composition: a graph preprocessing and the solution of a shortest path problem with resource constraints (SPPRC). The aim of the graph preprocessing is to reduce the size of the tour graph $G^e(\mathcal{N},\mathcal{A})$ by deleting nodes that are similar to each other. Therefore, a node $v_k \in \mathcal{N}$ is deleted if it is dominated by a node $v_{k'} \in \mathcal{N}$, $k' \neq k$. Considering the above, $v_{k'}$ dominates v_k if both nodes represent shifts starting the same day, if both cover the same number of time periods and if the cost of $v_{k'}$ is lower than the cost of v_k. Note that the preprocessing step allows to solve the pricing subproblems in a fast but heuristic way, since some nodes that could be in the optimal solution of the SPPRC are deleted. Therefore, to guarantee the optimality of the column generation approach, when no more columns with negative reduced cost can be found, we call the pricing subproblems once again, this time without performing the preprocessing step, and solve the SPPRC with the complete graph.

To solve the shortest path problem with resource constraints, we implemented a label setting algorithm where the total length of the tour and the number of working days are global resources that are consumed by the labels while they are extended. In order to avoid exploring paths that in some point will be infeasible or dominated, we included pruning strategies related to bounds, the consumption of global resources and the dominance property between labels. The pseudocode of the algorithm and its description are presented in Appendix B.

4.5.3 Branching Rule

Let e_1 be the employee such that one of its variables $x_t^{e_1}$ holds the largest fractional value in the optimal LP solution at the node under evaluation. Let e_2 be the employee such that one of its variables $x_{t_1}^{e_2}$ holds the the second largest fractional value, therefore $x_t^{e_1} \geq x_{t_1}^{e_2}$. Let $x_{t_2}^{e_2}$ be the variable holding the second largest fractional value for employee e_2, hence $x_{t_1}^{e_2} \geq x_{t_2}^{e_2}$. The branching rule used for the Tour-based formulation is an aggressive variable fixing strategy combined with the rule presented for the Daily-based formulation. At each node of the B&P tree, we choose employees e_1 and e_2, fix to one variable $x_t^{e_1}$ for e_1 and select two tours, t_1 and t_2, for e_2 corresponding to variables $x_{t_1}^{e_2}$ and $x_{t_2}^{e_2}$ to apply the branching rule presented in Section 4.4.3.

4.6 Computational Experiments

In this section, we present the computational experiments to test the performance of the B&P algorithms. First, we generate random instances to test and compare the algorithms. Second, since no method has been proposed in the literature to solve personalized multi-activity tour scheduling problems, we test the Tour-based B&P algorithm for the mono-activity tour scheduling problem introduced by Brunner and Bard [17].

4.6.1 Results on Random Instances

We perform the computational experiments on several random instances that are generated as follows. We start creating a set of feasible schedules for each employee to randomly choose one. From these schedules, we derive the associated demand profile along the planning horizon. Undercovering and overcovering of employee requirements are generated by randomly adding or removing demand. All instances are defined over a one-week planning horizon where days are divided into 96 time periods of 15 minutes and shifts are not allowed to span from one day to another (discontinuous version of the problem). The number of working days in the tour should fall between 5 and 6 and the tour length should fall between 35 and

40 working hours per week. Activities have a minimum and a maximum length that range between 2 and 24 time periods, depending on the instance. Each tour has an associated cost corresponding to the sum of the costs of performing activity j at time period i plus the sum of transition costs c_{tr} between work activities. Instances are divided into three groups and are labeled with the format `E_J_D_V_G` where `E`, `J`, `D`, `V` and `G` represent the number of employees, number of activities, length of the planning horizon in days, version of the instance and group, respectively. The instances and the productions used in the grammars to create the shifts are described as follows, where: s_d and e_d are the first and last time periods in day d with demand greater than zero, respectively; a_l and a_u are the minimum and maximum length of activity a, respectively; A_e is the set of skills of employee e, and artificial activity r represents either a break or a rest time period.

G1: Instances with flexible start times and three types of shifts

In this group of instances, shifts may start at any time period of the day allowing enough time to complete its length and three types of shifts are considered: 9-hour shifts with a 1-hour lunch break in the middle, 4-hour and 6-hour shifts without breaks.

$S_{[s_d,e_d]} \rightarrow RPR \mid RFR \mid RQR \mid PR \mid RP \mid FR \mid RF \mid QR \mid RQ$;
$P_{[16,16]} \overset{c_{tr}}{\rightarrow} J_j J_{\bar{j}} | J_j$, $\forall j \in A_e$;
$Q_{[24,24]} \overset{c_{tr}}{\rightarrow} J_j J_{\bar{j}} | J_j$, $\forall j \in A_e$;
$F \rightarrow PLP$;
$J_{a[a_l,a_u]} \rightarrow J_a^{'}$, $\forall a \in A_e$;
$J_a^{'} \rightarrow a | a J_a^{'}$, $\forall a \in A_e$;
$J_{\bar{j}} \rightarrow J_{j'}$, $\forall j \in A_e$, $\forall j^{'} \in A_e \backslash \{j\}$;
$J_{\bar{j}} \overset{c_{tr}}{\rightarrow} J_{j'} J_{\bar{j}'}$, $\forall j \in A_e$, $\forall j^{'} \in A_e \backslash \{j\}$;
$R \rightarrow r | rR$;
$L_{[4,4]} \rightarrow r | rR$.

G2: Instances with flexible start times and two types of shifts

In this group, we allow flexibility in terms of shift start time, meaning that shifts may start at any time period of the day and two types of shifts are considered: 9-hour shifts with a 1-hour lunch break in the middle and 4-hour shifts without breaks.

$S_{[s_d,e_d]} \rightarrow RPR \mid RFR \mid PR \mid RP \mid FR \mid RF$;
$P_{[16,16]} \overset{c_{tr}}{\rightarrow} J_j J_{\bar{j}} | J_j$, $\forall j \in A_e$;
$F \rightarrow PLP$;
$J_{a[a_l,a_u]} \rightarrow J_a^{'}$, $\forall a \in A_e$;

$J_a^{'} \rightarrow a|aJ_a^{'}$, $\forall a \in A_e$;
$J_{\bar{j}} \rightarrow J_{j'}$, $\forall j \in A_e$, $\forall j^{'} \in A_e \backslash \{j\}$;
$J_{\bar{j}} \overset{c_{tr}}{\rightarrow} J_{j'} J_{\bar{j}'}$, $\forall j \in A_e$, $\forall j^{'} \in A_e \backslash \{j\}$;
$R \rightarrow r|rR$;
$L_{[4,4]} \rightarrow r|rR$.

G3: Instances with restricted start times and three types of shifts

In this third group of instances, shifts cannot start at any time period, therefore a set of start times is considered. Three types of shifts are generated: 9-hour shifts with a 1-hour lunch break in the middle, 4-hour and 6-hour shifts without breaks. The start times are 1, 17, 33, 49 and 61.

$S_{[36,36]} \rightarrow PR \mid QR \mid F$;
$P_{[16,16]} \overset{c_{tr}}{\rightarrow} J_j J_{\bar{j}} | J_j$, $\forall j \in A_e$;
$Q_{[24,24]} \overset{c_{tr}}{\rightarrow} J_j J_{\bar{j}} | J_j$, $\forall j \in A_e$;
$F \rightarrow PLP$;
$J_{a[a_l,a_u]} \rightarrow J_a^{'}$, $\forall a \in A_e$;
$J_a^{'} \rightarrow a|aJ_a^{'}$, $\forall a \in A_e$;
$J_{\bar{j}} \rightarrow J_{j'}$, $\forall j \in A_e$, $\forall j^{'} \in A_e \backslash \{j\}$;
$J_{\bar{j}} \overset{c_{tr}}{\rightarrow} J_{j'} J_{\bar{j}'}$, $\forall j \in A_e$, $\forall j^{'} \in A_e \backslash \{j\}$;
$R \rightarrow r|rR$;
$L_{[4,4]} \rightarrow r|rR$.

Since restricted start times are being considered, in this grammar $s_d = e_d = 36$ represents the maximum allowed shift length (9-hour shifts) and not the first and last time period with demand greater than zero. Observe that the same set of productions are used to generate the shifts for each start time.

Table 4.1 presents the average size of the graphs for the instances with flexible and restricted start times. The name of the instance is presented in Column 1. Column 2 shows the average number of nodes in the DAG Γ for each employee at every day in the planning horizon. Columns 3 and 4 present the average number of nodes and arcs in graph $G^e(\mathcal{N}, \mathcal{A})$.

Table 4.1 Average number of nodes and arcs in the graphs for instances with flexible and restricted start times.

Instance	Nodes DAG Γ	Nodes $G(\mathcal{N},\mathcal{A})$	Arcs $G(\mathcal{N},\mathcal{A})$
20_1_7_1_G1	6,980	998	291,606
20_1_7_2_G1	11,899	1,496	627,300
25_1_7_1_G1	7,006	1,001	292,689
25_1_7_2_G1	11,998	1,505	633,753
40_1_7_1_G1	7,006	1001	292,689
40_1_7_2_G1	11,998	1,505	633,753
20_3_7_1_G2	24,802	652	124,072
20_3_7_2_G2	19,420	983	270,476
20_3_7_3_G2	19,147	972	265,352
20_1_7_1_G3	300	63	1,215
20_1_7_2_G3	300	96	2,700
25_1_7_1_G3	300	63	1,215
25_1_7_2_G3	300	105	3,141
40_1_7_1_G3	300	63	1,215
40_1_7_2_G3	300	105	3,141
20_3_7_1_G3	3,918	63	1,215
20_3_7_2_G3	1,851	101	2,906
20_3_7_3_G3	1,851	99	2,802
20_5_7_1_G3	1,131	53	1,127
20_5_7_2_G3	1,154	51	1,007
20_5_7_3_G3	1,379	84	2,285

The computational experiments were performed on a 64-bit GNU/Linux operating system, 96 GB of RAM and 1 processor Intel® Xeon® X5675 running at 3.07GHz. The B&P algorithms were implemented in C++ using the object-oriented branch-and-bound library (OOBB) developed by Crainic et al. [32]. The RMP was solved by using the barrier method of CPLEX version 12.5.0.1.

Solution at the Root Node.

Tables 4.2 - 4.3 show the computational effort, at the root node, for the B&P algorithms for the proposed formulations: the Daily-based formulation ($F_{\widetilde{\mathcal{S}}}$) and the Tour-based formulation ($F_{\widetilde{\mathcal{T}}}$) for instances with flexible and restricted start times, respectively. We report the value (LP val.) and the required time in seconds (CPU time), to solve the LP relaxation of the RMP. The difference between the lower bound of $F_{\widetilde{\mathcal{T}}}$ against the lower bound of $F_{\widetilde{\mathcal{S}}}$ is

presented in Column 6.

According to the results, $F_{\widetilde{\mathcal{T}}}$ achieved better lower bounds for all the 21 instances when compared with $F_{\widetilde{\mathcal{S}}}$. It is worth noting that the difference between lower bounds tends to increase as the problem flexibility increases. This can be seen in the results where the average LB difference for instances in G1-G2 (flexible start time) is 3.79%, while the average LB difference for instances in G3 (restricted start time) is 1.77%. Regarding the computational time to solve the LP relaxation at the root node, the Daily-based formulation shows a better performance in 15 out of 21 instances and, as expected, results also show that the average computational time of both formulations increases as the number of activities or employees grow.

The reason for the differences in the time performance for both formulations is mainly due to the structure of the pricing subproblems. Subproblems of $F_{\widetilde{\mathcal{S}}}$ only deal with the composition of daily shifts, while subproblems in $F_{\widetilde{\mathcal{T}}}$ handle the composition of daily shifts and the assembling of tours. However, as it will be shown in the computational results for the B&P, it is worth to invest more time in the solution of the root node if the lower bounds obtained are better, especially when the model exhibits symmetry that makes difficult to prove optimality due to many equivalent solutions.

Table 4.2 Computational effort to solve the LP relaxation at the root node, for instances with flexible start times.

	Tour-based		Daily-based		
Instance	LP val.	CPU time	LP val.	CPU time	LB Dif.
20_1_7_1_G1	52,080	**21.02**	50,080	24.72	3.84%
20_1_7_2_G1	49,440	30.84	47,440	**18.20**	4.05%
25_1_7_1_G1	60,560	**16.10**	58,060	33.47	4.13%
25_1_7_2_G1	72,660	**6.03**	70,160	32.51	3.44%
40_1_7_1_G1	100,410	**10.52**	96,410	52.64	3.98%
40_1_7_2_G1	98,390	89.17	94,390	**52.95**	4.07%
20_3_7_1_G2	55,270	185.11	53,195	**70.20**	3.75%
20_3_7_2_G2	60,120	133.69	58,045	**54.36**	3.45%
20_3_7_3_G2	60,450	97.80	58,375	**51.51**	3.43%
Average	-	65.60	-	49.40	3.79%

Table 4.3 Computational effort to solve the LP relaxation at the root node, for instances with restricted start times.

	Tour-based		Daily-based		
Instance	LP val.	CPU time	LP val.	CPU time	LB Dif.
20_1_7_1_G3	82,825	3.59	82,145	**0.68**	0.82%
20_1_7_2_G3	64,675	4.4	63,775	**1.79**	1.39%
25_1_7_1_G3	99,995	2.83	98,732.5	**0.91**	1.26%
25_1_7_2_G3	72,660	**1.31**	70,160	3.03	3.44%
40_1_7_1_G3	202,050	**1.35**	198,050	1.99	1.98%
40_1_7_2_G3	117,677	9.49	116,280	**6.84**	1.19%
20_3_7_1_G3	73,440	61.05	71,440	**7.69**	2.72%
20_3_7_2_G3	61,655.5	130.6	59,635.6	**17.32**	3.28%
20_3_7_3_G3	63,575	86.61	61,560	**15.89**	3.17%
20_5_7_1_G3	95,078.3	509.53	94,865.6	**8.44**	0.22%
20_5_7_2_G3	93,962.2	156.92	93,280.1	**10.25**	0.73%
20_5_7_3_G3	131,824	126.63	130,432	**12.22**	1.06%
Average	-	91.2	-	7.25	1.77%

We tested a heuristic approach in which the MIP version of the problem, including only the set of generated columns at the root node, was solved by a state-of-the-art B&B code. Tables 4.4 - 4.5 present the best upper bound found (IP RMP) and the computational time required (CPU time) to find it, for the Tour-based formulation and the Daily-based formulation on instances with flexible start time and restricted start time, respectively. The value of the integrality gap for both formulations is computed as: $\frac{\text{(IP RMP - LP val.)}}{\text{IP RMP}}$ and reported in Columns 4 and 7. We set a time limit of 600 sec. or a relative MIP gap tolerance of less than 1% to solve the problem when the integrality constraints are considered.

Table 4.4 Computational effort to find an upper bound at the root node, for instances with flexible start times.

	Tour-based			Daily-based		
Instance	IP RMP	CPU time	Gap	IP RMP	CPU time	Gap
20_1_7_1_G1	**52,080**	**0.29**	0.00%	52,280	10.51	4.21%
20_1_7_2_G1	49,660	**0.16**	0.44%	**49,640**	100.77	4.43%
25_1_7_1_G1	**60,560**	**0.59**	0.00%	60,760	38.09	4.44%
25_1_7_2_G1	**72,660**	**0.29**	0.00%	73,100	27.36	4.02%
40_1_7_1_G1	100,850	**0.25**	0.44%	**100,810**	49.89	4.36%
40_1_7_2_G1	**98,940**	**1.35**	0.56%	98,990	551.02	4.65%
20_3_7_1_G2	57,625	600	4.09%	**55,380**	89.97	3.94%
20_3_7_2_G2	62,350	600	3.58%	**60,450**	264.32	3.98%
20_3_7_3_G2	62,050	600	2.58%	**60,700**	116.32	3.83%
Average	-	200.33	1.30%	-	138.69	4.21%

Table 4.5 Computational effort to find an upper bound at the root node, for instances with restricted start times.

	Tour-based			Daily-based		
Instance	IP RMP	CPU time	Gap	IP RMP	CPU time	Gap
20_1_7_1_G3	**82,860**	**0.07**	0.04%	83,280	0.65	1.36%
20_1_7_2_G3	65,280	**1.51**	0.93%	**65,040**	9.18	1.94%
25_1_7_1_G3	**100,380**	**0.64**	0.38%	100,650	3.46	1.90%
25_1_7_2_G3	73,100	**0.3**	0.60%	**72,860**	2.02	3.71%
40_1_7_1_G3	**202,250**	**0.03**	0.10%	203,450	1.45	2.65%
40_1_7_2_G3	118,170	**0.97**	0.42%	118,170	13.50	1.60%
20_3_7_1_G3	76,030	600	3.41%	**74,070**	7.16	3.55%
20_3_7_2_G3	65,445	600	5.79%	**62,245**	45.20	4.19%
20_3_7_3_G3	67,330	600	5.58%	**64,285**	59.45	4.24%
20_5_7_1_G3	102,170	600	6.94%	**96,035**	9.30	1.22%
20_5_7_2_G3	102,860	600	8.65%	**94,690**	11.40	1.49%
20_5_7_3_G3	137,135	600	3.87%	**133,430**	55.74	2.25%
Average	-	300.29	3.06%	-	18.21	2.51%

According to the results, the Tour-based formulation achieved better solution times for all the 12 mono-activity instances when compared to the Daily-based formulation. On the other hand, the Daily-based formulation had a better performance as the number of activities grow.

Observe that if feasible integer solutions are required for multi-activity problems, avoiding the B&P process and using only the variables generated at the root node with the Daily-based formulation could be a good option. On the contrary, as shown by the results of the next section, if the objective is to solve the problems to optimality, a B&P algorithm should be used taking advantage of the formulation with the best lower bound, in this case the Tour-based formulation.

Branch-and-Price.

Tables 4.6 - 4.7 show the results of the B&P algorithm for the instances with flexible start time and restricted start time, respectively, with both formulations, the Tour-based formulation and the Daily-based formulation. Columns 2 and 6 present the value of the best integer solution found. Columns 3 and 7 report the total computational time required to find the integer solutions, while Columns 4 and 8 contain the number of nodes explored in the B&P trees. Finally, Columns 5 and 9 show the value of the integrality gap (defined as the percentage difference between the best upper bound minus the best lower bound). The algorithm stops when the gap is less than 1% or when the total time reaches one hour. The search strategy used was a depth-first search where the upper bound of the algorithm was initialized with the heuristic integer solution found at the root node.

For the Tour-based formulation, our method was able to find high-quality integer solutions for all the instances in the three groups. All the mono-activity instances were closed with the heuristic at the root node in less than two minutes. On the contrary, for the multi-activity instances, it was necessary to explore several nodes in the B&P tree to find integer solutions with an integrality gap less than 1%. Note that, as can be seen in the results for instances with five activities, the computational time and the number of nodes explored in the B&P tree increase with the number of activities. The average integrality gap for instances in G3 is the highest among the three groups, with a value of 0.56%.

On the other hand, the B&P algorithm implemented for the Daily-based formulation did not have a good performance when tested with the three groups of instances. The algorithm was not able to improve, within one hour time limit, the integer solution found at the root node, and when compared with the Tour-based formulation, it only achieved better integer solutions in 2 out of 21 instances. As a result, if the objective is to find near optimal solutions, the Tour-based formulation could be considered stronger because it exhibits a better LP relaxation bound that makes easier to close the integrality gap.

Table 4.6 Computational effort in the B&P algorithm for instances with flexible start times.

	Tour-based				Daily-based			
Instance	IP val.	CPU time	Nb. Nodes	Gap	IP val.	CPU time	Nb. Nodes	Gap
20_1_7_1_G1	**52,080**	21.31	1	0.00%	52,280	3,600	777	4.21%
20_1_7_2_G1	**49,440**	31	1	0.44%	49,640	3,600	229	4.43%
25_1_7_1_G1	**60,560**	16.69	1	0.00%	60,760	3,600	369	4.44%
25_1_7_2_G1	**72,660**	6.32	1	0.00%	73,100	3,600	545	4.02%
40_1_7_1_G1	**100,520**	10.77	1	0.44%	100,810	3,600	217	4.36%
40_1_7_2_G1	**98,390**	90.52	1	0.56%	98,990	3,600	57	4.65%
20_3_7_1_G2	55,380	2366.81	39	0.20%	55,380	3,600	51	3.95%
20_3_7_2_G2	**60,120**	1735.07	35	0.00%	60,450	3,600	159	3.98%
20_3_7_3_G2	60,780	1507.83	39	0.54%	**60,700**	3,600	33	3.83%
Average	-	643	13	0.24%	-	3,600	271	4.20%

Table 4.7 Computational effort in the B&P algorithm for instances with restricted start times.

	Tour-based				Daily-based			
Instance	IP val.	CPU time	Nb. Nodes	Gap	IP val.	CPU time	Nb. Nodes	Gap
20_1_7_1_G3	**82,860**	3.66	1	0.04%	83,280	3,600	615	1.36%
20_1_7_2_G3	**64,730**	5.91	1	0.93%	65,040	3,600	677	1.94%
25_1_7_1_G3	**100,250**	3.47	1	0.38%	100,650	3,600	703	1.9%
25_1_7_2_G3	**72,660**	1.61	1	0.6%	72,860	3,600	1055	3.71%
40_1_7_1_G3	**202,050**	1.38	1	0.1%	203,450	3,600	971	2.65%
40_1_7_2_G3	**117,730**	10.46	1	0.42%	118,170	3,600	1393	1.6%
20_3_7_1_G3	**73,625**	1,070.66	39	0.25%	74,070	3,600	1113	3.55%
20_3_7_2_G3	**62,125**	1,408.06	37	0.76%	62,245	3,600	405	4.19%
20_3_7_3_G3	**63,950**	1,370.42	37	0.59%	64,285	3,600	953	4.24%
20_5_7_1_G3	**95,890**	2,033.89	41	0.85%	96,035	3,600	465	1.22%
20_5_7_2_G3	94,800	2,256.76	127	0.88%	**94,690**	3,600	431	1.49%
20_5_7_3_G3	**133,035**	1,929.55	91	0.91%	133,430	3,600	410	2.25%
Average	-	841.32	32	0.56%	-	3,600	766	2.51%

4.6.2 Results on Instances From Brunner and Bard [17]

This mono-activity problem consists of a discontinuous tour scheduling problem over one week with two types of employees: full time workers (Reg) and part time workers (Flex). In the problem, each day is divided into 48 time periods of 30 minutes each. Each Reg employee must be given an 8.5-hour shift on each working day and the starting time of shifts can vary by day (nine start times). In contrast, Flex employees can be assigned to one of five shift types ranging from 4 to 8.5 hours, which can have different starting times (12 start times). If an employee works 6 hours or more a day, its assigned shift must have a 0.5-hour lunch break.

In Brunner and Bard [17], the master problem is based on a set covering model that seeks to minimize the total cost of the tours plus the undercovering cost of employee requirements. The model guarantees that the total number of employees that are on duty cover the requirements for each time period during the time horizon, and that the ratio between the number of Reg employees and the number of Flex employees is at least a given value.

Table 4.8 presents the description of the eleven scenarios analyzed. Column 1 gives the name of the instance. Column 2 presents the value of the ratio between Reg and Flex employees. Columns 3 and 4 specify the types of employees having shifts with flexible length and flexible start times, respectively. Column 5 shows if shifts have a break. Additionally, tours are not forced to have two days off in a row. The full description of the parameters to build daily shifts and weekly tours, as well as the employee requirements are presented in the appendix of the work in Brunner and Bard [17].

Table 4.8 Description of Brunner and Bard [17] problem's scenarios.

Instance	Ratio	Flex_length	Flex_start	Breaks
B1	4	Flex	No	Yes
B2	4	No	Flex	Yes
B3	4	Flex	Flex	Yes
B4	4	Flex	Reg,Flex	Yes
B5	4	No	Flex	No
B6	4	Flex	Flex	No
B7	4	Flex	Reg,Flex	No
B8	1	Flex	Reg,Flex	No
B9	2	Flex	Reg,Flex	No
B10	3	Flex	Reg,Flex	No
B11	5	Flex	Reg,Flex	No

Table 4.9 shows the output statistics for the Tour-based formulation and the model presented in Brunner and Bard [17], now called Brunner's model. Since the number of employees is not

known, the Daily-based formulation cannot be used in the context of this problem because tours are composed for each employee through constraints (4.3)-(4.6). In the same way, the proposed B&P algorithm cannot be used in the context of this problem, because the branching rule is based on employee branching. That is why we decided to solve the Tour-based formulation with the root node heuristic presented in Section 4.6.1, where we observed a good performance on mono-activity instances (average integrality gap of 0.33% in less than 2 sec. of computational time). Columns 2 and 4 present the IP value at the root node for the eleven instances evaluated with the Tour-based formulation and Brunner's model, respectively. As it was done in Brunner and Bard [17] we set a time limit of 300 sec. to solve the IP at the root node. Columns 3 and 5 show the total CPU time in sec. required to solve the LP relaxation and to obtain the IP value for both formulations. Finally, the value of the best IP solution found with the B&P algorithm presented in Brunner and Bard [17] and the total computational time in seconds to obtain that value are reported in Columns 6 and 7, respectively.

Table 4.9 Computational effort at the root node to solve the integer problem.

	Tour-based		Brunner's model			
Instance	IP val.	CPU time	IP val.	CPU time	Best IP val.	Total time
B1	95,944	**303.95**	95,864	488.06	95,640	2,882.39
B2	95,120⋆	**304.34**	95,640	496.62	95,120	5,743.36
B3	95,056⋆	**331.24**	95,832	492.20	95,120	14,428.84
B4	94,960⋆	**341.52**	95,920	406.75	95,456	6,293.98
B5	93,442.5	365.60	93,402.5	348.35	93,282.5	2,514.85
B6	93,290.5⋆	342.93	93,426.5	322.10	93,322.5	1,909.29
B7	93,266.5	**304.50**	93,298.5	322.93	93,178.5	1,871.9
B8	83,501⋆	**323.80**	84,421	331.44	84,061	1,853.11
B9	88,295⋆	**340.10**	88,727	342.46	88,439	1,919.47
B10	90,967.5⋆	344.34	91,135.5	335.27	90,999.5	1,891.51
B11	94,812.5	321.32	94,964.5	316.10	94,796.5	1,860.59
Average	-	329.42	-	382.02	-	3,924.48

From Table 4.9, we can conclude that the Tour-based formulation shows competitive solution times when compared with Brunner's model. Regarding the quality of the solution (IP value obtained) the star (⋆) in Column 2 means that our model was able to achieve, in less than 6 minutes, the same or a better integer solution than the one reported as the best in Brunner and Bard [17], where the computational time to find this value was more than 30 minutes for all the instances (values reported in Column 7). Regarding the problem difficulty, Brunner and Bard [17] found that the instances with a lunch break were the most difficult to solve (instances B1 to B4) and that the time tended to decrease when more flexibility was

introduced in the model. On the contrary, we did not observe any significant change in the execution time when considering either more flexibility or breaks.

4.7 Concluding Remarks

In this paper we introduced two B&P algorithms to solve the personalized multi-activity tour scheduling problem. Two formulations were presented in which the master problem is modeled as a generalized set partitioning problem. With respect to the pricing subproblems, in the Daily-based formulation, columns (daily shifts) are modeled using context-free grammars. In the Tour-based formulation, columns (tours) are built with an exact two-phase procedure. In the first phase, daily shifts are modeled by using context-free grammars, where in the second phase, the daily shifts are assembled into tours by using a shortest path algorithm with resource constraints.

Although our computational experiments suggest that the Daily-based formulation finds solutions for the LP relaxation at the root node in a shorter execution time when compared with the Tour-based formulation, we show that the second formulation is stronger in terms of its LP relaxation lower bound.

Two methods were tested to find integer solutions. A heuristic approach in which we impose the integrality constraints at the root node and an exact approach corresponding to a B&P. The Daily-based formulation exhibited better solution times than the Tour-based formulation for the heuristic approach. On the other hand, the Tour-based formulation had a better performance in the exact approach being able to find, within 1 hour, integer solutions for all the instances with an integrality gap lower than 1%. We also tested the Tour-based formulation on a mono-activity problem presented in Brunner and Bard [17]. The experiments suggested that the solution times and quality of our formulation are comparable with the solution times and quality reported by Brunner and Bard [17].

Despite the ability of our models to handle complex work rules, convergence and scalability issues arise when the number of employees and activities increase. One solution to this problem could be to implement an implicit model in order to avoid the dimension associated with the number of employees. Another option might be to implement new branching strategies related to the shift length in order to reduce the number of nodes explored in the B&P tree and reach integrality in a shorter time.

CHAPTER 5 ARTICLE 2: COMBINING BENDERS DECOMPOSITION AND COLUMN GENERATION FOR MULTI-ACTIVITY TOUR SCHEDULING

Explicit approaches for the MATSP begin to suffer from scalability and symmetry issues when the number of employees and work activities increase. In this chapter of the thesis, we present an approach that combines BD and CG approaches to solve the anonymous MATSP in a discontinuous environment. The model, decomposable by days, consists of a Benders master problem and a set of Benders daily subproblems. The Benders master problem includes the variables associated with employee tours and with daily shifts shells (shifts without activity and break allocation). The Benders subproblems include the variables related to the allocation of work activities and breaks to daily shifts and to the undercovering and overcovering of employee requirements. Since the number of feasible tours might be too large to be completely enumerated, we propose to solve the Benders master problem by CG. New columns for the Benders master problem are generated with a label setting algorithm for the resource-constrained shortest-path problem over a directed acyclic graph, where the set of nodes correspond to daily shift shells and the set of arcs correspond to feasible connections between shifts, made according to the rules for the allocation of days-off and to the rest time between shifts. Since one of the goals of the method is to eliminate the problems related to the employee dimension, the Benders subproblems were modeled with the implicit grammar-based IP model presented in Côté et al. [30]. In the model, a grammar is built at each day by including the work rules for the composition of shifts and the allocation of work activities and breaks to the shifts. From each grammar, a directed acyclic graph (DAG), containing all the possible set of shifts, is derived. Finally, the logical clauses associated with the DAG are translated into linear constraints on integer variables.

Since Benders primal subproblems do not possess the integrality property, we propose a new algorithmic strategy in which, in addition to the generation of classical Benders optimality cuts, we generate a set of valid integer Benders cuts to guarantee the convergence of the method. We test and compare the proposed approach on two set of instances: real-world instances and randomly generated instances for the MATSP (one-week planning horizon) and for the MASSP (one-day planning horizon). Results on weekly instances show that our method exhibits faster solution times and provides better upper bounds when compared with a B&P method. Results on daily instances show that the Benders decomposition approach is able to solve to optimality instances with up to thirty work activities and when the method is compared with the grammar-based IP approach presented in Côté et al. [30], it shows

competitive and often better solution times.

The next article was submitted to *Management Science* in October 2015.

Combining Benders Decomposition and Column Generation for Multi-Activity Tour Scheduling

María I. Restrepo
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, Canada

Bernard Gendron
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département d'informatique et de recherche opérationelle, Université de Montréal, Montréal, Canada

Louis-Martin Rousseau
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, Canada

Abstract. This paper presents a method that combines Benders decomposition and column generation to solve the multi-activity tour scheduling problem. The Benders decomposition approach iterates between a master problem that links daily shifts with tour patterns and a set of daily subproblems that assign work activities and breaks to the shifts. Due to its structure, the master problem is solved by column generation. We exploit the expressiveness of context-free grammars to model and solve the Benders subproblems. Computational results show that our method outperforms a branch-and-price approach and is able to find high-quality solutions for weekly instances dealing with up to ten work activities. The adaptation of the method to the shift scheduling problem (the special case defined on a single day) is also shown to outperform the solution of a grammar-based model by a state-of-the-art mixed-integer programming solver on instances with up to 30 work activities.

Keywords. Multi-activity tour scheduling problem, Benders decomposition, Column generation, Context-free grammars.

5.1 Introduction

In this paper, we consider the *multi-activity tour scheduling problem* (MATSP), when employees have the same skills and shifts do not span over several days (the *anonymous dis-*

continuous version of the problem). In the MATSP, tours (or working schedules) are defined over a *planning horizon* of at least one week, where each day is divided into *time periods* of equal length. For each time period, it must be specified if a *work activity*, a *break* or a *rest period* is performed. Multi-activity daily shifts are characterized by their start time, working length, break allocation and activity placement at any time period, while tours are defined by their working length, number of working days and consecutiveness in the days-off. The composition of daily shifts and tours is usually constrained by *work rules* and *employee agreements*. The objective of the MATSP is to determine a minimum cost set of tours and to assign them to each employee, so that staff requirements are guaranteed for each work activity at each time period in the planning horizon.

Tour scheduling problems (including *shift scheduling* problems, their special cases defined on a single day) have been typically modeled using two different approaches: *explicit* and *implicit* models. In explicit models, each feasible working schedule is represented by an integer variable. These models allow to consider a high degree of flexibility with the drawback that the resulting problem is difficult to solve, due to the large number of variables involved in the formulation. On the contrary, implicit models compromise model flexibility seeking to reduce the size of the problem by defining variables that implicitly represent feasible working schedules. In shift scheduling problems, implicit variables represent shift and break types. In tour scheduling problems, implicit variables represent work and non-work days across the planning horizon, daily start times, daily end times and breaks.

Both explicit and implicit models for tour scheduling problems become computationally elusive as the problem size increases. Decomposition methods, in particular column generation (CG) and Benders decomposition (BD), arise as interesting approaches to efficiently solve tour scheduling problems. As we will see in Section 5.2, the literature shows examples of such decomposition methods, but only for simplified versions of scheduling problems with multiple activities, where flexibility regarding shift and tour composition is usually limited. Such simplifications might lead to unrealistic versions of the problem. For instance, when days-off are fixed, the problem simplifies to a multi-day problem, which is easier to solve, since some of the constraints characterizing the feasibility of schedules over multiple days do not have to be guaranteed. By considering flexibility regarding shift and tour composition, realistic, but complex, problems arise. In particular, when there is a large number of employees or activities, these problems do not scale well.

In this paper, we propose a BD method for the anonymous discontinuous MATSP, which is particularly well-suited for solving large-scale instances arising from practical problems. We take advantage of the block-angular structure of the problem, decomposable by days.

The variables corresponding to tour patterns are represented in the Benders master problem and linked with variables related to daily shifts. Due to its structure that involves a large number of tour-based variables, the Benders master problem is solved by a CG method, where flexibility regarding shift start time, shift length, tour length and days-off is included. Regarding the Benders subproblems, we exploit the expressiveness of context-free grammars and use an implicit mixed-integer programming (MIP) model that captures all the rules for the composition of shifts in a compact way. The allocation of work activities and breaks to daily shifts is handled at the subproblem level.

Our contributions are threefold:

- We propose a new model for the anonymous discontinuous MATSP that combines an explicit tour scheduling modeling approach with an implicit grammar-based shift scheduling formulation.
- We develop an innovative decomposition method that combines BD and CG. In particular, the Benders subproblems are MIP models that do not possess the integrality property. Thus, in addition to *classical Benders cuts* [14], the method generates *integer Benders cuts* to guarantee the convergence to an optimal solution under mild conditions.
- By performing computational experiments on a large set of weekly instances with up to ten work activities, we show that our method is able to find high-quality solutions and outperforms a recently proposed branch-and-price (B&P) algorithm for the *personalized* (i.e., employees have different skills) discontinuous MATSP [76]. In addition, the adaptation of the method to the multi-activity shift scheduling problem is shown to outperform the solution of a grammar-based model [30] by a state-of-the-art MIP solver on instances with up to 30 work activities.

The paper is organized as follows. In Section 5.2, we review the relevant literature on shift and tour scheduling problems, and we give a short introduction on the use of grammars for multi-activity shift scheduling problems. In Section 5.3, we present our model for the anonymous discontinuous MATSP, which is derived from a grammar-based model for multi-activity multi-day shift scheduling problems. In Section 5.4, we describe the decomposition method that combines BD and CG. Computational experiments are presented and discussed in Section 5.5. Concluding remarks follow in Section 5.6.

5.2 Background Material

We first review the literature on shift and tour scheduling problems (Section 5.2.1), focusing in particular on multi-activity versions of these problems (Section 5.2.2). Then, Section 5.2.3 presents some background material related to the use of context-free grammars for shift scheduling.

5.2.1 Shift and Tour Scheduling

Shift and tour scheduling problems have been extensively studied during the last few decades. Several modeling techniques and solution methods have been proposed to tackle the different characteristics of the problems. Ernst et al. [43, 44], Alfares [4] and Van den Bergh et al. [82] present comprehensive surveys in which more than a thousand papers are classified according to the type of problem, the application area and the solution method.

The first author to introduce an explicit model for shift scheduling problems is Dantzig [34]. The model is based on a set covering formulation in which the objective is to minimize the total labor cost, ensuring that staff requirements at every time period are met. Later, in one of the first attempts to solve shift scheduling problems with an implicit model, Moondra [60] proposes a method for banking operations that includes shift flexibility regarding multiple shift lengths and start times. Meal-break placement flexibility is considered in Bechtold and Jacobs [12], with an implicit formulation where shifts are grouped into shift types according to their start time, length and break window. Thompson [81] combines the work of Moondra [60] and Bechtold and Jacobs [12] to implicitly model meal breaks, but also to schedule rest breaks and to allow the use of overtime. Aykin [5] presents an extension of Bechtold and Jacobs' formulation that considers multiple rest breaks, meal breaks and break windows by introducing integer variables for the number of employees assigned to a shift and starting their breaks at different time periods.

Jarrah et al. [50] propose an implicit model to solve a discontinuous weekly tour scheduling problem, which is decomposed into seven daily shift scheduling subproblems. A transportation component and a post-processor are used to assign breaks to shifts and shifts to tours, respectively. Jacobs and Brusco [49] present an implicit model that allows start time flexibility within continuous (e.g., shifts can span over multiple days) and discontinuous employee tours, but that does not consider meal breaks. Brusco and Jacobs [20] integrate the work of Bechtold and Jacobs [12] and Jacobs and Brusco [49] in an implicit integer programming model that considers both start time and meal break flexibility to solve continuous tour scheduling problems. More recently, Brunner and Bard [17] take advantage of implicit and

explicit shift definitions to solve, with a B&P algorithm, a discontinuous tour scheduling problem over one-week planning horizons.

CG is presented as an interesting method when the introduction of flexibility in the composition of shifts and tours is handled by explicit models that cause a considerable increase in the number of variables (see, e.g., Mehrotra et al. [59], Ni and Abeledo [62], Brunner and Stolletz [18]). Although BD appears to be an appropriate method to solve large problems that feature a special block structure, few papers addressing shift and tour scheduling problems with this technique have appeared. Rekik et al. [72] use BD in a continuous tour scheduling problem to prove that the forward and backward constraints introduced by Bechtold and Jacobs [12] are valid, but do not suffice to model break-window or start time extraordinary overlap. After conducting an extensive analysis, the authors conclude that the model derived from BD considerably decreases the number of variables, at the cost of a small increase in the number of constraints.

5.2.2 Multi-Activity Shift and Tour Scheduling

Implicit modeling has also been used in the context of multi-activity shift scheduling. In particular, for the anonymous version of the problem, Côté et al. [30] propose to solve the scalability issues identified in Côté et al. [28] by taking advantage of context-free grammars to model the rules for the composition of daily shifts and to derive an implicit model that addresses symmetry by using general integer variables. Computational results show that, in the monoactivity case, solving the model with a state-of-the-art MIP solver is comparable and sometimes superior to the results presented in the literature and that, in the multi-activity case, this approach is able to solve to optimality instances with up to ten work activities.

Methods involving CG, BD, constraint programming (CP), formal languages, branch-and-bound (B&B), and heuristics have also been proposed in order to solve both the multi-activity shift scheduling problem (MASSP) and the MATSP. Demassey et al. [37] present a CP-based column generation algorithm as a way to model complex regulation constraints to solve large MASSP instances. Quimper and Rousseau [69] introduce a model that uses formal languages to derive specialized graph structures that are handled via large neighborhood search for solving the MASSP. Côté et al. [31] attempt to solve the personalized version of the MASSP with a B&P method that uses grammars to formulate the pricing subproblems. Although the expressiveness of grammars enables to encode a large set of work rules over shifts, some limitations are present regarding shift total length over long planning horizons (e.g., one week). Restrepo et al. [76] attempt to overcome these limitations by proposing two B&P approaches that address the personalized MATSP. In the first approach, columns correspond

to daily shifts, while in the second approach, columns correspond to tours. Although the authors show that the second formulation is stronger in terms of its LP relaxation bound, both formulations suffer from scalability issues when the number of employees, the number of work activities and the flexibility increase. Dahmen and Rekik [33] propose a heuristic based on tabu search and B&B to solve the personalized MASSP over multiple days. Although some constraints related to the composition of feasible tours are included (e.g., minimum and maximum number of working hours per week), days-off are previously assigned to the employees. Detienne et al. [38] solve an employee timetabling problem, where besides using Lagrangian relaxation and a heuristic based on a cut generation process, a BD method is also proposed. In their work, the multi-activity case is considered, but tour patterns over the time horizon are previously defined. Computational results suggest that the BD method is computationally more expensive when compared with the cut generation based heuristic, because of the large amount of time invested in solving the master problem.

The following section presents some basic concepts on the use of context-free grammars for shift scheduling. For a more extensive review on the subject, the reader is referred to Côté et al. [30].

5.2.3 Grammars

In a multi-day planning horizon, where D represents the set of days and d the subscript for a given day, a *context-free grammar* is a tuple $G_d = \langle \Sigma_d, N_d, S_d, P_d \rangle$ where Σ_d is an alphabet of characters called the *terminal symbols*, N_d is a set of *non-terminal symbols*, $S_d \in N_d$ is the starting symbol, and P_d is a set of *productions* represented as $A \rightarrow \alpha$, where $A \in N_d$ is a non-terminal symbol and α is a sequence of terminal and non-terminal symbols. The productions of a grammar can be used to generate new symbol sequences until only terminal symbols are part of the sequence. A *context-free language* is the set of sequences accepted by a context-free grammar.

A *parse tree* is a tree where each inner-node is labeled with a non-terminal symbol and each leaf is labeled with a terminal symbol. A grammar recognizes a sequence if and only if there exists a parse tree where the leaves, when listed from left to right, reproduce the sequence. An *and/or graph* is a graph where each leaf corresponds to an assignment that can either be true or false. An and-node is true if all of its children are true. An or-node is true if one of its children is true. The root node is true if the grammar accepts the sequence encoded by the leaves. The and/or graph embeds every possible parse tree of a grammar.

A *DAG* Γ_d is a *directed acyclic graph* that embeds all parse trees associated with words of a given length n recognized by a grammar. The DAG Γ_d has an and/or structure where the

and-nodes represent productions from P_d and or-nodes represent non-terminals from N_d and letters from Σ_d. The DAG Γ_d is built by a procedure proposed in Quimper and Walsh [71].

In the MASSP, the use of grammars allows to include work rules regarding the definition of shifts and to handle the allocation of multiple work activities to the shifts in an easy way. Thus, feasible shifts can be represented as words in a context-free language. For example, words $rw_1w_1bw_2$ and $w_1bw_2w_1r$ are recognized as valid shifts in a two-activity shift scheduling problem where letters w_1, w_2, b and r represent working on activity 1, working on activity 2, break and rest periods, respectively. The time horizon consists of five time periods, shifts have a length of four periods and must contain exactly one break of one period that can be placed anywhere during the shift except at the first or the last period. We remove the subscript d, since there is only one day in the planning horizon. The grammar that defines the multi-activity shifts on this example follows:

$G = (\Sigma = (w_1, w_2, b, r), N = (S, F, X, W, B, R), S, P)$,
where productions P are: $S \to RF|FR$, $F \to XW$, $X \to WB$, $W \to WW|w_1|w_2$, $B \to b$, $R \to r$ and symbol $|$ specifies the choice of production.

In the previous example, productions $W \to w_1$, $W \to w_2$, $B \to b$ and $R \to r$ generate the terminal symbols associated with working on activity 1, working on activity 2, having a break or having a rest period inside of the shift, respectively. Production $W \to WW$ generates two non-terminal symbols, W, meaning that the shift will include a working subsequence. Production $X \to WB$ means that the shift will include working time followed by a break. Production $F \to XW$ generates a subsequence of length four (the daily shift), which includes working time followed by a break to finish with more working time. Finally, the last two productions are $S \to RF$ and $S \to FR$. The former generates a sequence starting with a period of rest followed by the daily shift. The latter generates a sequence starting with the daily shift followed by a period of rest.

Let O^{π}_{dil} be the or-nodes associated with $\pi \in N_d \cup \Sigma_d$, i.e., with non-terminals from N_d or letters from Σ_d, that generate a subsequence at day d, from position i of length l. Note that if $\pi \in \Sigma_d$, the node is a leaf and l is equal to one. On the contrary, if $\pi \in N_d$, the node represents a non-terminal symbol and $l > 1$. $A^{\Pi,k}_{dil}$ is the kth and-node representing production $\Pi \in P_d$ generating a subsequence at day d, from position i of length l. There are as many $A^{\Pi,k}_{dil}$ nodes as there are ways of using P_d to generate a sequence of length l from position i during day d. The sets of or-nodes, and-nodes and leaves of day d are denoted by O_d, A_d and L_d, respectively. The root node is described by O^{S}_{d1n} and its children by $A^{\Pi,k}_{d1n}$. The children of or-node O^{π}_{dil} are represented by $ch(O^{\pi}_{dil})$ and its parents by $par(O^{\pi}_{dil})$. Similarly, the children of and-node $A^{\Pi,k}_{dil}$ are represented by $ch(A^{\Pi,k}_{dil})$ and its parents by $par(A^{\Pi,k}_{dil})$.

Figure 5.1 represents the DAG Γ associated with the grammar of the example (we do not include the subscript of the day in the notation of the nodes). Dashed-line or-nodes are part of the parse trees associated with and-node $A_{15}^{S\to RF,1}$. Continuous-line or-nodes are part of the parse trees associated with and-node $A_{15}^{S\to FR,1}$.

Note that the children of the root node $ch(O_{15}^{S}) = \{A_{15}^{S\to RF,1}, A_{15}^{S\to FR,1}\}$ can be seen as shift "shells" because they do not consider the allocation of specific work activities and breaks to the shifts, only the shift starting time and its length. Hence, and-nodes $A_{d1n}^{\Pi,k}$ are characterized by their starting time $t_{d1n}^{\Pi,k}$, working length $w_{d1n}^{\Pi,k}$, length including breaks $l_{d1n}^{\Pi,k}$ and finish time $f_{d1n}^{\Pi,k} = t_{d1n}^{\Pi,k} + l_{d1n}^{\Pi,k} - 1$. In DAG Γ, and-node $A_{15}^{S\to RF,1}$ generates shifts $rwbww$ and $rwwbw$, while and-node $A_{15}^{S\to FR,1}$ generates shifts $wbwwr$ and $wwbwr$.

5.3 Grammar-Based Model

The work of Côté et al. [30] on the anonymous MASSP is one example of the use of context-free grammars to represent the work rules involved in the composition of shifts. The authors present an implicit grammar-based integer programming model where the word length n corresponds to the number of periods in the planning horizon, the set of work activities corresponds to letters in the alphabet Σ_d and each employee $e \in E$ is allowed to work at any activity. In the model, the logical clauses associated with Γ_d are translated into linear constraints on integer variables. Each and-node A_d and each leaf L_d in Γ_d are represented by an integer variable denoting the number of employees assigned to a specific subsequence of work.

In this section, we first present a straightforward extension of this grammar-based integer programming model that addresses a discontinuous multi-day MASSP. This formulation is then used as a basis for a new grammar-based model for the anonymous discontinuous MATSP. The set of work activities is denoted by J. The planning horizon is at least one week, where each day $d \in D$ is divided into I_d time periods of equal length. The notation used for the formulation of the discontinuous multi-day MASSP is as follows:

Parameters:

b_{dij}: staff requirements for day d, time period i and activity j;

c_{dij}: nonnegative cost associated with one employee working on activity j, at time period i, at day d;

c_{dij}^{+}, c_{dij}^{-}: nonnegative overcovering and undercovering costs of staff requirements for day d, time period i and work activity j, respectively.

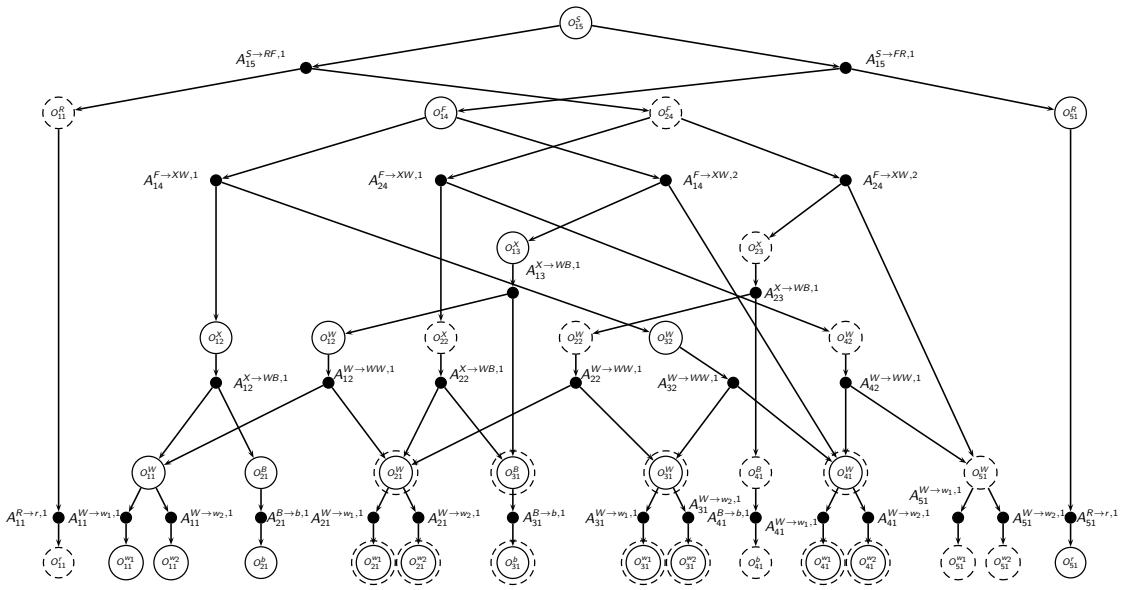


Figure 5.1 DAG Γ on words of length five and two work activities

Decision variables:

$v_{dil}^{\Pi,k}$: variable that denotes the number of employees assigned to the kth and-node, representing production Π from Γ_d producing a sequence from i of length l at day d;

y_{dij}: variable that denotes the number of employees assigned to leaf O_{di1}^{j}, that represents working on activity j, at time period i, at day d;

s_{dij}^{+} and s_{dij}^{-}: slack variables that denote overcovering and undercovering of staff requirements of work activity j, at time period i, for day d, respectively.

The grammar-based formulation of the discontinuous multi-day MASSP, denoted as $G_{\mathcal{S}}$, is as follows:

$$Z(G_{\mathcal{S}}) = \min \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij} y_{dij} + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} (c^+_{dij} s^+_{dij} + c^-_{dij} s^-_{dij}) \tag{5.1}$$

$$y_{dij} - s^+_{dij} + s^-_{dij} = b_{dij}, \; \forall\, d \in D, \; i \in I_d, \; j \in J, \tag{5.2}$$

$$\sum_{A^{\Pi,k}_{dil} \in ch(O^{\pi}_{dil})} v^{\Pi,k}_{dil} = \sum_{A^{\Pi,k}_{dil} \in par(O^{\pi}_{dil})} v^{\Pi,k}_{dil}, \; \forall\, d \in D, \; O^{\pi}_{dil} \in O_d \setminus \{O^S_{d1n} \cup L_d\}, \tag{5.3}$$

$$y_{dij} = \sum_{A^{\Pi,1}_{di1} \in par(O^j_{di1})} v^{\Pi,1}_{di1}, \; \forall\, d \in D, \; i \in I_d, \; j \in J, \tag{5.4}$$

$$\sum_{A^{\Pi,k}_{d1n} \in ch(O^S_{d1n})} v^{\Pi,k}_{d1n} \leq |E|, \; \forall\, d \in D, \tag{5.5}$$

$$v^{\Pi,k}_{dil} \geq 0, \; \forall\, d \in D, \; A^{\Pi,k}_{dil} \in A_d, \tag{5.6}$$

$$s^+_{dij}, \; s^-_{dij} \geq 0, \; \forall\, d \in D, \; i \in I_d, \; j \in J, \tag{5.7}$$

$$y_{dij} \geq 0 \text{ and integer}, \; \forall\, d \in D, \; i \in I_d, \; j \in J. \tag{5.8}$$

The objective of $G_{\mathcal{S}}$, (5.1), is to minimize the total staffing cost plus the penalization for overcovering and undercovering of staff requirements. Constraints (5.2) ensure that the total number of employees working on day $d \in D$, time period $i \in I_d$ and work activity $j \in J$ is equal to the demand subject to some adjustments related to undercovering and undercovering. Constraints (5.3) guarantee, for every or-node in Γ_d, $d \in D$, excluding the root node O^S_{d1n} and the leaves L_d, that the summation of the value of its children is the same as the summation of the value of its parents. Constraints (5.4) set the value of variables y_{dij} as the summation of the value of the parents of leaf nodes O^j_{di1}. Constraints (5.5) guarantee that at most $|E|$ employees are assigned to the daily shift shells (children of the root node) at each day d. Constraints (5.6)-(5.8) set the non-negativity of variables $v^{\Pi,k}_{dil}$, s^+_{dij}, s^-_{dij} and the non-negativity and integrality of variables y_{dij}.

The solution obtained from model (5.1)-(5.8) is implicit. As a result, a post-processing algorithm should be used to build the individual schedules. This algorithm traverses $\Gamma_d, d \in D$, from the root node to the leaves, visiting the nodes with value greater than zero. Once a node is visited, its value is decreased by one, and, when a leaf is reached, its value is inserted to the current schedule at the right position (for instance, if leaf O^2_{151} is reached, it means that activity 2 should be inserted at position 5 in the schedule of day 1).

Observe that model (5.1)-(5.8) does not account for the constraints characterizing the feasibility of tour patterns, namely: minimum and maximum tour length, minimum and maximum

working days, minimum rest time between consecutive shifts and consecutiveness in the days-off. To circumvent this issue, we define a set $\mathcal{T}$ containing all the feasible tour patterns that can be built given the work rules for tour composition. In this context, we define a tour as a combination of days-off and daily shift shells (children of root nodes O^S_{d1n}) over the set of days in the planning horizon. Figure 5.2 presents an example of three tours composed with the shift shells presented in Figure 5.1. In this example, we assume that the DAG Γ_d for each day $d \in D$ is the same and corresponds to Γ. The planning horizon consists of seven days, the working length should fall between 15 and 18 time periods, the number of working days must fall between 5 and 6 and there are no rules for the allocation of days-off and for the rest time between consecutive shifts. Additionally, S_1 corresponds to and-node $A^{S\to RF,1}_{d15}$ generating shifts $\{rwbww, rwwbw\}$, S_2 corresponds to and-node $A^{S\to FR,1}_{d15}$ generating shifts $\{wbwwr, wwbwr\}$ and Do corresponds to allocating a day-off.

Model (5.1)-(5.8) should be modified to solve the discontinuous MATSP. To this end, we define $\delta^{\Pi,k}_{dt}$ as a parameter that takes value 1, if tour t includes the kth children of the root node O^S_{d1n} built with production Π for day d, and assumes value 0 otherwise. We generate a set of decision variables x_t denoting the number of employees assigned to tour $t \in \mathcal{T}$. Constraints (5.9) set the link between these variables and shift shell variables $v^{\Pi,k}_{dil}, A^{\Pi,k}_{dil} \in ch(O^S_{d1n})$. Constraints (5.5) are replaced by constraint (5.10), which guarantees that exactly $|E|$ employees are assigned to the set of tours $\mathcal{T}$. Constraints (5.11) set the non-negativity and integrality of tour-based variables x_t.

$$v^{\Pi,k}_{d1n} = \sum_{t\in\mathcal{T}} \delta^{\Pi,k}_{dt} x_t, \; \forall\, d \in D, \; A^{\Pi,k}_{d1n} \in ch(O^S_{d1n}), \tag{5.9}$$

$$\sum_{t\in\mathcal{T}} x_t = |E|, \tag{5.10}$$

$$x_t \geq 0 \text{ and integer}, \; \forall\, t \in \mathcal{T}. \tag{5.11}$$

The grammar-based model for the anonymous discontinuous MATSP, denoted as $G_{\mathcal{T}}$, is as follows:

$$Z(G_{\mathcal{T}}) = \min \sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J} c_{dij} y_{dij} + \sum_{d\in D}\sum_{i\in I_d}\sum_{j\in J} (c^+_{dij} s^+_{dij} + c^-_{dij} s^-_{dij})$$

$$\text{subject to} \quad (5.2) - (5.4) \text{ and } (5.6) - (5.11).$$

Tours \ Days	1	2	3	4	5	6	7
1	Do	Do	S_1	S_1	S_2	S_1	S_2
2	S_2	S_2	S_2	S_2	S_2	Do	S_1
3	Do	S_1	S_2	S_2	Do	S_2	S_1

Figure 5.2 Weekly tours composed of and-nodes (shift shells) from Γ

When the number of work activities and days grow and when the problem accounts for a large number of work rules for the composition of shifts and tours, solving $G_{\mathcal{T}}$ becomes a difficult task because of the large number of constraints and variables involved in the formulation. Next, we present the decomposition method we propose to solve large instances of model $G_{\mathcal{T}}$.

5.4 Benders Decomposition/Column Generation Algorithm

Two ideas can be exploited in order to efficiently solve model $G_{\mathcal{T}}$. First, note that if tour-based variables are fixed to particular values $\overline{x}_t$, $t \in \mathcal{T}$, model $G_{\mathcal{T}}$ can be decomposed by days due to its particular block structure. The BD approach that exploits this idea is presented in Section 5.4.1. Second, observe that tour-based variables x_t do not need to be exhaustively enumerated, since only a small subset of them will be present in an optimal solution. Section 5.4.2 describes the CG method that results from this idea. By combining these two ideas, we obtain an exact algorithm, which is presented and analyzed in Section 5.4.3.

5.4.1 Benders Decomposition

The structure of model $G_{\mathcal{T}}$ suggests to partition the set of $v_{dil}^{\Pi,k}$ variables into two sets. Indeed, due to the linking constraints (5.9), when variables x_t, $t \in \mathcal{T}$, are fixed and satisfy constraints (5.10)-(5.11), variables $v_{d1n}^{\Pi,k} A_{d1n}^{\Pi,k} \in ch(O_{d1n}^{S})$, associated with the shift shells (children of root node O_{d1n}^{S}) are also fixed. Thus, the first set contains these variables, while the second set contains the other variables, corresponding to the and-nodes that generate a subsequence of work from time period i, at day d of length $l < n$, denoted as $v_{dil}^{\Pi,k}$, $A_{dil}^{\Pi,k} \in A_d \setminus ch(O_{d1n}^{S})$. This partition of the variables of model $G_{\mathcal{T}}$ is the basic idea of the proposed BD approach.

Benders Daily Subproblems

After fixing tour-based variables to particular values $\overline{x}_t$, $t \in \mathcal{T}$, the resulting model decomposes into $|D|$ independent Benders subproblems, one for each day in the planning horizon. Each subproblem includes the variables associated with the and-nodes in Γ_d and with the allocation of work activities and breaks to the shift shells. The formulation of the *Benders primal subproblem*, denoted as $\mathcal{Q}(\overline{\mathbf{v}_d})$ for a given day d, is as follows:

$$Z(\mathcal{Q}(\overline{\mathbf{v}_d})) = \min \sum_{i \in I_d} \sum_{j \in J} c_{dij} y_{dij} + \sum_{i \in I_d} \sum_{j \in J} (c^+_{dij} s^+_{dij} + c^-_{dij} s^-_{dij}) \tag{5.12}$$

$$y_{dij} - s^+_{dij} + s^-_{dij} = b_{dij}, \; \forall\, i \in I_d, \; j \in J, \tag{5.13}$$

$$\sum_{A^{\Pi,k}_{dil} \in ch(O^{\pi}_{dil})} v^{\Pi,k}_{dil} = \sum_{A^{\Pi,k}_{dil} \in par(O^{\pi}_{dil})} \overline{v^{\Pi,k}_{d1n}}, \; \forall\, O^{\pi}_{dil} \in ch(A^{\Pi,k}_{d1n}) \setminus L_d, \tag{5.14}$$

$$\sum_{A^{\Pi,t}_{dil} \in ch(O^{\pi}_{dil})} v^{\Pi,k}_{dil} = \sum_{A^{\Pi,k}_{dil} \in par(O^{\pi}_{dil})} v^{\Pi,k}_{dil}, \; \forall\, O^{\pi}_{dil} \in O_d \setminus \{O^S_{d1n} \cup L_d \cup ch(A^{\Pi,k}_{d1n})\}, \tag{5.15}$$

$$y_{dij} = \sum_{A^{\Pi,k}_{di1} \in par(O^{j}_{di1})} v^{\Pi,1}_{di1}, \; \forall\, i \in I_d, \; j \in J, \tag{5.16}$$

$$v^{\Pi,k}_{dil} \geq 0, \; \forall\, A^{\Pi,k}_{dil} \in A_d \setminus ch(O^S_{d1n}), \tag{5.17}$$

$$s^+_{dij}, s^-_{dij} \geq 0, \; \forall\, i \in I_d, \; j \in J, \tag{5.18}$$

$$y_{dij} \geq 0 \text{ and integer}, \; \forall\, i \in I_d, \; j \in J. \tag{5.19}$$

For a given number of employees assigned to each shift shell (fixed variables $\overline{v^{\Pi,k}_{d1n}}$), the objective, (5.12), of $\mathcal{Q}(\overline{\mathbf{v}_d})$ is to assign work activities to these shifts in order to minimize the staffing cost plus the undercovering and overcovering of staff requirements. Constraints (5.13) guarantee that staff requirements are met. Constraints (5.14)-(5.16) ensure that certain work rules are guaranteed for the composition of shifts and the allocation of work activities and breaks to the shifts. Constraints (5.17)-(5.19) set the non-negativity of variables $v^{\Pi,k}_{dil}$, s^+_{dij}, s^-_{dij} and the non-negativity and integrality of variables y_{dij}.

Since variables y_{dij} are required to be integer and Benders primal subproblems (5.12)-(5.19) do not possess the integrality property, the classical BD approach needs to be modified to ensure convergence to an optimal solution. First, we will generate classical Benders cuts by relaxing the integrality constraints (5.19) on variables y_{dij}. Second, we will generate integer Benders cuts to guarantee the convergence of the method to an optimal solution.

Classical Benders Cuts

Let $\overline{\mathcal{Q}}(\overline{\mathbf{v}_d})$ denote the LP relaxation of model (5.12)-(5.19). Observe that, due to the allowance of undercovering and overcovering of staff requirements, $\overline{\mathcal{Q}}(\overline{\mathbf{v}}_d)$ is always feasible. The polyhedra that define the Benders dual subproblems are thus bounded and contain no ray. Therefore, when $\overline{\mathcal{Q}}(\overline{\mathbf{v}_d})$ is solved to obtain classical Benders cuts, no feasibility cuts will be generated.

To define optimality cuts, we introduce the following notation for each day $d \in D$. Let ρ_{dij}, γ^{π}_{dil} be the dual variables associated with constraints (5.13) and (5.14) from $\overline{\mathcal{Q}}(\overline{\mathbf{v}_d})$, respectively. Let $\mathbf{\Delta}_d$ be the polyhedron obtained from the projection, over the space of variables ρ_{dij} and γ^{π}_{dil}, of the set of feasible solutions to the dual of model $\overline{\mathcal{Q}}(\overline{\mathbf{v}}_d)$. Let $E_{\mathbf{\Delta}_d}$ be the set of extreme points of $\mathbf{\Delta}_d$. Let θ_d be a nonnegative variable that represents the value of the objective of $\overline{\mathcal{Q}}(\overline{\mathbf{v}_d})$. The Benders optimality cuts are then defined as follows:

$$\theta_d \geq \sum_{i \in I_d} \sum_{j \in J} \rho_{dij} b_{dij} + \sum_{O^{\pi}_{dil} \in ch(A^{\Pi,k}_{d1n})} \gamma^{\pi}_{dil} \sum_{A^{\Pi,k}_{d1n} \in par(O^{\pi}_{dil})} v^{\Pi,k}_{d1n}, \; \forall\, d \in D, \; (\rho_d, \gamma_d) \in E_{\mathbf{\Delta}_d} \tag{5.20}$$

Optimality cuts ensure that the value of each variable θ_d is larger than or equal to the LP relaxation value of its corresponding Benders daily subproblem. To derive these cuts, we have assumed that Benders subproblems are linear programs, i.e., the integrality of variables y_{dij} is relaxed. The relaxation of model $G_{\mathcal{T}}$ obtained by relaxing the integrality of variables y_{dij} can thus be reformulated as the following master problem, denoted as $B_{\mathcal{T}}$:

$$\begin{aligned} Z(B_{\mathcal{T}}) = \min \sum_{d \in D} \theta_d & \\ \text{subject to} \quad & (5.20), (5.9) - (5.11) \text{ and} \\ & v^{\Pi,k}_{d1n} \geq 0, \; \forall\, d \in D, \; A^{\Pi,k}_{d1n} \in ch(O^{S}_{d1n}). \end{aligned}$$

Optimality cuts (5.20) do not need to be exhaustively generated since only a subset of them will be active in an optimal solution. An iterative cutting plane algorithm can thus be used to generate only the subset of cuts that will yield an optimal solution of $B_{\mathcal{T}}$. Because Benders subproblems (5.12)-(5.19) are MIP models that do not possess the integrality property, this cutting plane algorithm will not, in general, identify a feasible solution of $G_{\mathcal{T}}$. Therefore, a more complex algorithmic strategy must be adopted, which is presented next.

Algorithmic Strategy

Consider an iterative BD approach to generate optimality cuts (5.20) where $l \geq 1$ is the index of each iteration. Let $\theta_d(l)$ denote the optimal values of variables θ_d at iteration l. Note that $Z^L(l) = \sum_{d \in D} \theta_d(l)$ is a lower bound on $Z(G_{\mathcal{T}})$ at iteration l. Let $\overline{s_d}(l)$ be the optimal value of Benders subproblem (5.12)-(5.19) for day d at iteration l when integrality constraints on variables y_{dij} are relaxed. Note that $\overline{Z^U}(l) = \sum_{d \in D} \overline{s_d}(l)$ is an upper bound on $Z(B_{\mathcal{T}}) \leq Z(G_{\mathcal{T}})$ at iteration l, which we call the *approximate upper bound*. Finally, let $\overline{v_{d1n}^{\Pi,k}}(l)$ denote the values of the shift shell variables $v_{d1n}^{\Pi,k}$ used to solve the Benders daily subproblems corresponding to the approximate upper bound at iteration l.

The proposed algorithmic strategy iterates between three steps. In the first step, we solve relaxation $B_{\mathcal{T}}$ of $G_{\mathcal{T}}$ through a classical BD method obtained by relaxing the integrality constraints on variables y_{dij}. When a solution of $B_{\mathcal{T}}$ is found, a feasibility check (second step) is performed in order to verify if the approximate upper bound $\overline{Z^U}(l)$ is a valid upper bound on $Z(G_{\mathcal{T}})$. If it is the case, we stop the computations. Otherwise, the third step will generate cuts (the integer Benders cuts to be described below) that tend to eliminate solution $\overline{v_{d1n}^{\Pi,k}}(l)$ from the master problem, unless it is part of an optimal solution of $G_{\mathcal{T}}$. The three steps are described as follows.

- *First step:* In this step, we solve model $B_{\mathcal{T}}$ (when integrality constraints on variables y_{dij} are relaxed) through a classical BD approach. In particular, a classical Benders optimality cut (5.20) is generated for each day d at each iteration l until the difference between the approximate upper bound $\overline{Z^U}(l)$ and the lower bound $Z^L(l)$ is small enough. A solution of $B_{\mathcal{T}}$, $(\theta_d(l), \overline{s_d}(l), \overline{v_{d1n}^{\Pi,k}}(l))$, $d \in D$, is recovered at the end of this step.

- *Feasibility check:* The objective of this step is to verify if $\overline{Z^U}(l)$ represents a valid upper bound for the original problem. In particular, since integrality constraints on variables y_{dij} were relaxed in the first step, it might happen that the approximate upper bound $\overline{Z^U}(l)$ obtained at the end of the classical BD approach underestimates the optimal value of the original problem, even if the first step is solved to optimality, i.e., $\overline{Z^U}(l) = Z^L(l)$. Clearly, if all Benders daily subproblems corresponding to the approximate upper bound $\overline{Z^U}(l)$ have an optimal solution for which all variables y_{dij} take integer values, then $\overline{Z^U}(l)$ is an upper bound on $Z(G_{\mathcal{T}})$ and the computations are stopped. If this case does not happen, we solve the MIP of each Benders daily subproblem (5.12)-(5.19) by using the values $\overline{v_{d1n}^{\Pi,k}}(l)$ obtained at the end of the first step. Then, the optimal value $s_d(l)$ of each Benders daily subproblem (5.12)-(5.19) is computed and compared with $\overline{s_d}(l)$ for each day $d \in D$. If $\overline{s_d}(l) < s_d(l)$ for at least one day $d \in D$, the value of $\overline{Z^U}(l)$ does not represent

a valid upper bound for the original problem and the solution $\overline{v_{d1n}^{\Pi,k}}(l)$ must be eliminated from the Benders master problem $B_{\mathcal{T}}$, unless it can be shown that it is part of an optimal solution of $G_{\mathcal{T}}$. Otherwise, if $s_d(l) = \overline{s_d}(l)$ for each $d \in D$, the approximate upper bound is valid and the computations are stopped.

- *Third step:* This step adds an integer Benders cut to the master problem $B_{\mathcal{T}}$ for each day $d \in D$ such that $\overline{s_d}(l) < s_d(l)$. Integer Benders cuts tend to eliminate the solution $\overline{v_{d1n}^{\Pi,k}}(l)$ from model $B_{\mathcal{T}}$ by changing at least one employee from its assigned shift shell to another. The three steps are then repeated until $\overline{Z^U}(l)$ is a valid upper bound on $Z(G_{\mathcal{T}})$. The detailed algorithm, along with its convergence analysis, are presented in Section 5.4.3.

Integer Benders Cuts

Constraints (5.5) state that the total number of employees assigned at each day $d \in D$ to the shift shells is lower than or equal to the total number of employees $|E|$. The slack variables in constraints (5.5) represent the number of employees having a day-off on day d. If we denote these variables as v_d^R, we have

$$\sum_{A_{d1n}^{\Pi,k} \in ch(O_{d1n}^S)} v_{d1n}^{\Pi,k} + v_d^R = |E|, \; \forall \, d \in D.$$

To simplify the presentation, we define a set $S_d, d \in D$, composed by the children of the root node O_{d1n}^S plus an element corresponding to a day-off. Variables $v_{d1n}^{\Pi,k}$, $A_{d1n}^{\Pi,k} \in ch(O_{d1n}^S)$, and v_d^R are then rewritten as z_{di}, denoting the number of employees allocated to $i \in S_d$ during day d. Therefore, we have, corresponding to constraints (5.5)-(5.6), a set of feasible solutions described by the following relations:

$$\sum_{i \in S_d} z_{di} = |E|, \; \forall \, d \in D,$$

$$z_{di} \geq 0 \text{ and integer}, \; \forall \, d \in D, \; i \in S_d.$$

In order to derive an integer Benders cut for the problem, we express each variable z_{di} with $|E|$ binary variables z_{di}^e, which take value 1 if employee $e \in E$ is assigned to $i \in S_d$ at day $d \in D$, and assume value 0 otherwise. We assume that employees are ordered arbitrarily such that $E = \{1, ..., |E|\}$. The binary variables are then defined as follows:

$$\sum_{e \in E} z_{di}^{e} = z_{di},\ \forall d \in D,\ i \in S_d, \tag{5.21}$$

$$z_{di}^{e+1} \leq z_{di}^{e},\ \forall d \in D,\ i \in S_d,\ e \in E \setminus \{|E|\}, \tag{5.22}$$

$$z_{di}^{e} \in \{0,1\},\ \forall\, d \in D,\ i \in S_d,\ e \in E.$$

Equations (5.21) guarantee, for every day $d \in D$, that the sum of the binary variables z_{di}^{e} is equal to the number of employees assigned to $i \in S_d$, while (5.22) are symmetry breaking constraints. Due to these constraints, each integer variable z_{di} has a unique representation in terms of the binary variables: if $z_{di} = m$, then $z_{di}^{e} = 1$ for $e \leq m$ and $z_{di}^{e} = 0$ for $e > m$. As a consequence, any solution representing an assignment of shifts (work or rest) on day d to $|E|$ employees is represented uniquely with the binary variables z_{di}^{e}.

Let $\overline{z_{di}^{e}}(l)$ be the value of variable z_{di}^{e} corresponding to the current solution $\overline{v_{d1n}^{\Pi,k}}(l)$ for day d and $\overline{\mathcal{B}_d}(l) = \{(i,e) \in S_d \times E \,|\, \overline{z_{di}^{e}}(l) = 1\}$. Since the objective is to eliminate the current solution $\overline{v_{d1nl}^{\pi,k}}$ by swapping at least one employee from its assigned shift (work or rest) to a different one, the integer Benders cut for a given day d, is as follows:

$$\theta_d \geq s_d(l) \left(1 + \sum_{(i,e) \in \overline{\mathcal{B}_d}(l)} z_{di}^{e} - |E| \right). \tag{5.23}$$

Because the value of θ_d is to be minimized, this constraint tends to eliminate the current shift shell assignment on day d, since the value of the right-hand side is then maximized and is equal to $s_d(l)$, the value of the Benders daily subproblem $\mathcal{Q}(\overline{\mathbf{v}_d})$. For any other shift shell assignment on day d, this constraint is trivially valid, since the right-hand side is then nonpositive. This optimality cut exploits the fact that exactly $|E|$ binary variables take value 1 and is similar to other types of cuts based on 0-1 variables used in variants of Benders decomposition, such as integer L-shaped [54], logic-based Benders decomposition [48] and combinatorial Benders decomposition [27].

5.4.2 Column Generation

In model $B_{\mathcal{T}}$, it is assumed that the complete set of tours $\mathcal{T}$ is known. However, with the incorporation of shift and tour flexibility, the complete enumeration of the set of feasible tours might be intractable. Therefore, we propose a CG method in which the master problem is

defined as the LP relaxation of model $B_{\mathcal{T}}$ over a restricted set of tours $\widetilde{\mathcal{T}} \subseteq \mathcal{T}$. The CG method alternates between this master problem and a *pricing subproblem*. The variables are generated iteratively by the pricing subproblem according to their reduced cost.

Pricing Subproblem for Tour Generation

Let $\lambda_{d1n}^{\Pi,k}$ and σ be the dual variables associated with constraints (5.9) and (5.10), respectively. The reduced cost $\overline{c}_t$ of column (tour) t is given by:

$$\overline{c}_t = \left(\sum_{d \in D} \sum_{A_{d1n}^{\Pi,k} \in ch(O_{d1n}^S)} \lambda_{d1n}^{\Pi,k} \delta_{dt}^{\Pi,k} \right) - \sigma. \tag{5.24}$$

Expression (5.24) corresponds to the objective function to be minimized in the pricing subproblem, whose goal is to build tours that meet the work rules related to the minimum and maximum number of working days in a tour (Λ_l and Λ_u, respectively); the minimum and maximum working tour length in time periods (Θ_l and Θ_u, respectively); the maximum number of days-off in a tour ($\Phi = |D| - \Lambda_l$); and the minimum rest time between two consecutive daily shifts (β). To find these tours, we define the set $\mathcal{S} = \bigcup_{d \in D} ch(O_{d1n}^S)$ as the union, over the set of days in the planning horizon, of all the children of root node O_{d1n}^S, $d \in D$. Shift shell $s \in \mathcal{S}$ inherits a set of attributes from its corresponding and-node: start period (t_s), working time (w_s), length considering breaks (l_s), end period ($f_s = t_s + l_s - 1$) and day (d_s). In addition, we define a directed acyclic graph $G(\mathcal{N}, \mathcal{A})$, composed by a set of nodes $\mathcal{N} = \{v_s \mid s \in \mathcal{S} \cup \{v_b, v_e\}\}$, where v_s corresponds to shift shell s and v_b, v_e are the source and sink nodes, respectively. The set of arcs $\mathcal{A}$, is divided into three types: arcs going from the source node to a shift shell node $\mathcal{A}_1 = \{(v_b, v_s) \mid v_s \in \mathcal{N}, d_s \leq \Phi + 1\}$; arcs connecting two shift shell nodes $\mathcal{A}_2 = \{(v_s, v_{s'}) \mid v_s, v_{s'} \in \mathcal{N}, s \neq s', t_{s'} - f_s \geq \beta, d_{s'} - d_s \leq \Phi + 1\}$; and arcs connecting a shift shell node to the sink, $\mathcal{A}_3 = \{(v_s, v_e) \mid v_s \in \mathcal{N}, d_s \geq \Lambda_l\}$.

Each node in graph $G(\mathcal{N}, \mathcal{A})$ has, besides a list of immediate successors $\mathcal{N}(v_s) = \{v_i \in \mathcal{N} | (s, i) \in \mathcal{A}\}$, a cost that represents its contribution to the reduced cost of the column. The source node has a cost equal to zero, the sink node has a cost equal to the negative of dual variable σ and the remaining nodes have a cost given by the corresponding value of their dual variables $\lambda_{d1n}^{\Pi,k}$. The list of successors of each node is generated according to the work rules for tour composition, as expressed in the arc types definition. Thus, successors of source node v_b are nodes that, depending on their start day, allow enough time to meet the constraints related to the minimum number of working days required in a tour. In the same way, sink

node v_e is a successor of node v_s if the day associated with v_s is greater than or equal to the minimum number of working days required in a tour. Finally, a node $v_{s'}$ is a successor of node v_s if its start time guarantees that there is a minimum rest time between both shifts, and if its start day meets the constraints related to the minimum and maximum number of days-off and their consecutiveness.

New variables (tours) for the master problem are generated by using a label setting algorithm for the resource-constrained shortest-path problem over the directed acyclic graph $G(\mathcal{N}, \mathcal{A})$. In the algorithm, the total length of the tour and the number of working days, represent global resources that are consumed by the labels while they are extended. If a column with negative reduced cost is found, the column is sent to the master problem, which is re-optimized to start a new iteration. The CG method stops when it is not possible to find any column t with $\overline{c}_t < 0$.

Branch-and-Price Algorithm

The CG method solves the LP relaxation of $B_{\mathcal{T}}$, with some classical and integer Benders cuts added. However, step 1 of the algorithmic strategy of Section 5.4.1 requires solving $B_{\mathcal{T}}$ with all the integer tour-based variables. That is why we embedded the CG method within a B&P algorithm, where integrality is obtained by branching.

At any node of the B&P tree, we denote by x_t^* the optimal LP relaxation value of x_t. Two cases are considered. In the first case, we search for a tour variable with a fractional value x_t^* greater than one. If such a variable exists, we create two nodes. In the left node, we impose the constraint $x_t \leq \lfloor x_t^* \rfloor$, while in the right node, we impose the constraint $x_t \geq \lceil x_t^* \rceil$. The second case occurs when all the fractional variables have a value lower than one. In this situation, we cannot impose the constraint $x_t = 0$ because it would result in the same tour being generated again by the subproblem, unless a specialized algorithm for the resource-constrained shortest-path problem is used. Henceforth, a different branching rule should be applied to the problem, which is an adaptation of existing rules (see, e.g., Barnhart et al. [11], Côté et al. [31]). We select two tours, $x_t(1)$ and $x_t(2)$, corresponding to the associated variables with the largest fractional values. Then, we identify the first divergent day d' between $x_t(1)$ and $x_t(2)$, meaning that both tours differ in their shift shells. Let $s(1) \in \mathcal{S}_d{}'$ and $s(2) \in \mathcal{S}_d{}'$ be the assigned shift shells at day d' for tours $x_t(1)$ and $x_t(2)$, respectively. A partition of $\mathcal{S}_d{}'$ into subsets $\mathcal{S}_d{}'(1)$ and $\mathcal{S}_d{}'(2)$ is created, such that $s(l) \in \mathcal{S}_d{}'(l)$, for $l = 1, 2$. The rest of the shift shells in $\mathcal{S}_d{}'$ are equally distributed between the two partitions. After generating the partitions, two nodes are created. At each node $l = 1, 2$ it is ensured that the tour generated will not include the shift shells in $\mathcal{S}_d{}'(l)$ at day d'. The rule is easily

handled in the subproblem, since if a shift shell s is forbidden at day d, the associated node v_s receives a large cost in graph $G(\mathcal{N}, \mathcal{A})$. Therefore, the suggested branching rule preserves the structure of the pricing subproblem.

At each node of the B&P algorithm, we perform the CG method until $\overline{c}_t \geq 0$ for each $t \in \mathcal{T}$. At the root node, a lower bound ζ^L on $Z(G_{\mathcal{T}})$ is thus obtained and the MIP of the master problem including only the current set of generated columns is solved by a state-of-the-art B&B code until the gap between the lower and upper bounds is small enough. In particular, the upper bound ζ^U found by the B&B code corresponds to a feasible solution of $B_{\mathcal{T}}$. This solution might not be optimal for $B_{\mathcal{T}}$, since only a reduced set of tours $\widetilde{\mathcal{T}} \subseteq \mathcal{T}$ has been considered. However, the quality of this solution can be measured against the lower bound ζ^L. The bounds computed at the root are then improved by branching and as soon as the gap between them is small enough, the B&P algorithm is stopped (this might happen at the root node, even before any branching is performed). Note that the B&P algorithm always produces at least one integer solution (in variables x_t), which corresponds to the upper bound ζ^U.

5.4.3 Overall Algorithm

The core of the BD/CG algorithm corresponds to the three-step algorithmic strategy presented in Section 5.4.1, but some modifications and refinements are included to enhance its performance. The pseudocode of the algorithm is presented in Algorithm 1, where l is the iteration counter, $Z^L(l)$ is the best lower bound, $\overline{Z^U}(l)$ is the best approximate upper bound and $Z^U(l)$ is the best upper bound on $Z(G_{\mathcal{T}})$.

The algorithm consists in two phases. In the first phase, the algorithm solves the LP relaxation of model $B_{\mathcal{T}}$, following McDaniel and Devine [58]. This is achieved by alternating between the CG method (without B&P) and the generation of classical Benders cuts (5.20) until no more cuts can be found or the gap between the approximate upper bound $\overline{Z^U}(l)$ and the lower bound $Z^L(l)$ is small enough. The generation of classical Benders cuts is improved by adopting the method presented in Papadakos [64], which is an alternative to solving the extra auxiliary subproblem introduced in Magnanti and Wong [57]. The algorithm then enters the second phase, where it seeks integer solutions by performing B&P. In this phase, the algorithm alternates between the B&P algorithm and the generation of classical Benders cuts (5.20), but when no more of these cuts can be generated, the algorithm solves the MIPs of the Benders subproblems. Then, a feasible solution is computed and integer Benders cuts (5.23) are generated, if needed, in which case the algorithm restarts with the cycle B&P/classical BD.

In the algorithm, we denote by $B^+_{\mathcal{T}}$, with optimal value $Z(B^+_{\mathcal{T}})$, the master problem corresponding to model $B_{\mathcal{T}}$, to which we add integer Benders cuts (5.23). Note that $B^+_{\mathcal{T}}$ plays the role of the BD master problem, where both classical and integer Benders cuts are gradually added, but also acts as the CG master problem, where tour-based variables are gradually generated.

The algorithm uses the following Boolean variables: Int indicates if the algorithm is in the first (Int=false) or in the second (Int=true) phase and Cut indicates if some cuts, classical or integer, have been generated (Cut=true) or not (Cut=false). In addition, $\overline{\mathbf{v}_d}(l)$ is the primal solution (in variables $v^{\Pi,k}_{d1n}$) obtained by performing the CG method (when Int=false) or the B&P algorithm (when Int=true). Since the B&P method can be stopped before optimality is proven, we need to distinguish the values of variables θ_d that correspond to the lower (ζ^L) and upper (ζ^U) bounds computed by the B&P algorithm: $\theta_d(l)$ are the values of θ_d for the relaxed solution ($\zeta^L = \sum_{d\in D} \theta_d(l)$) and $\overline{\theta_d}(l)$ are the values of θ_d for the best feasible solution ($\zeta^U = \sum_{d\in D} \overline{\theta_d}(l)$). To simplify the algorithm description, we use the same notation when the CG method is used in the first phase, even though in that case, we have $\theta_d(l) = \overline{\theta_d}(l)$ for each $d \in D$, since the CG algorithm is performed until all columns have nonnegative reduced costs.

The algorithm uses five parameters $\epsilon_i \in [0, 1]$, $i = 1, ..., 5$, which represent thresholds on different relative gaps: ϵ_1 is used to stop the algorithm when the relative gap between $Z^U(l)$ and $Z^L(l)$ is small enough; ϵ_2 is used to stop the B&P algorithm when the relative gap between the upper bound $\zeta^U$and the lower bound ζ^L computed by the B&P method is small enough; ϵ_3 controls the generation of classical Benders cuts in case the relative gap between $\overline{\theta_d}(l)$ and the upper bound associated with the LP relaxation of the Benders subproblem $\mathcal{Q}(\overline{\mathbf{v}_d}(l))$ is large enough; ϵ_4 controls when the first phase (solving the LP relaxation of $B_{\mathcal{T}}$) is stopped using the relative gap between $\overline{Z^U}(l)$ and $Z^L(l)$, which has to be small enough; ϵ_5 controls the generation of integer Benders cuts in case the relative gap between the MIP and the LP relaxation bounds of the Benders subproblem $\mathcal{Q}(\overline{\mathbf{v}_d}(l))$ is small enough.

The next two propositions state that the algorithm, independently of the values of the tolerance parameters ϵ_i, delivers at least one feasible solution when it terminates and that it computes a lower bound on $Z(G_{\mathcal{T}})$ at every iteration. Then, we show that the algorithm converges to optimal solutions of $G_{\mathcal{T}}$ and its LP relaxation, when the appropriate tolerance parameters are set to 0.

Proposition 1. *The algorithm terminates with a feasible solution of $G_{\mathcal{T}}$ in a finite number of iterations.*

Algorithm 1 BD/CG algorithm for the MATSP

$l = 0, Z^L(l) = -\infty, \overline{Z^U}(l) = \infty, Z^U(l) = \infty, Int =$ false, $Cut =$ true
while $((Z^U(l) - Z^L(l))/Z^U(l) > \epsilon_1)$ and (Cut==true) **do**
  $l = l + 1$
  if Int==false **then**
    `Perform CG until` $\overline{c_t} \geq 0, \forall t \in \mathcal{T}$ `(solve LP relaxation of` $B^+_{\mathcal{T}}$`), return` $\theta_d(l), \overline{\theta_d}(l), \overline{\mathbf{v}_d}(l), d \in D$
  else
    `Perform B&P until` $(\zeta^U - \zeta^L)/\zeta^U \leq \epsilon_2$ `(solve` $B^+_{\mathcal{T}}$`), return` $\theta_d(l), \overline{\theta_d}(l), \overline{\mathbf{v}_d}(l), d \in D$
  $Z^L(l) = \sum_{d \in D} \theta_d(l)$
  $Cut =$ false
  for $d \in D$ **do**
    `Solve the LP relaxation of the Benders subproblem` $\mathcal{Q}(\overline{\mathbf{v}_d}(l))$, `return` $\overline{s_d}(l)$
    if $(\overline{s_d}(l) - \overline{\theta_d}(l))/\overline{s_d}(l) > \epsilon_3$ **then**
      `Add classical Benders cut` (5.20) `to` $B^+_{\mathcal{T}}$, $Cut =$ true
  if Int==false **then**
    $\overline{Z^U}(l) = \min\{\overline{Z^U}(l), \sum_{d \in D} \overline{s_d}(l)\}$
    if $(\overline{Z^U}(l) - Z^L(l))/\overline{Z^U}(l) \leq \epsilon_4$ or (Cut==false) **then**
      Int=true, Cut=true
  if Cut==false **then**
    for $d \in D$ **do**
      `Solve the MIP of the Benders subproblem` $\mathcal{Q}(\overline{\mathbf{v}_d}(l))$, `return` $s_d(l)$
      if $(s_d(l) - \overline{s_d}(l))/s_d(l) > \epsilon_5$ **then**
        `Add integer Benders cut` (5.23) `to` $B^+_{\mathcal{T}}$, Cut=true
    $Z^U(l) = \min\{Z^U(l), \sum_{d \in D} s_d(l)\}$
    if $Z^U(l) = \sum_{d \in D} s_d(l)$ **then**
      $\overline{\mathbf{v}_d} = \overline{\mathbf{v}_d}(l)$
`Use` $\overline{\mathbf{v}_d}$ `to find the working schedule for each employee`

Proof. Because the maximum number of classical Benders cuts (5.20) is bounded by the number of extreme points of $|D|$ polyhedra, the first phase (when Int=false) ends in a finite number of iterations. During the second phase (when Int=true), the B&P algorithm always generates an integer solution (in variables x_t). Since the number of classical Benders cuts is finite, the algorithm solves the MIP of each Benders subproblem $\mathcal{Q}(\overline{\mathbf{v}_d})$ at least one time, identifying then a feasible solution of $G_{\mathcal{T}}$. Finally, because the number of classical and integer Benders cuts is finite, the algorithm terminates in a finite number of iterations. ■

Proposition 2. *At every iteration l, the algorithm computes a lower bound $Z^L(l)$ on $Z(G_{\mathcal{T}})$.*

Proof. As long as Int=false, the algorithm computes, at every iteration l, a lower bound on the LP relaxation of $B_{\mathcal{T}}$, which is itself a relaxation of $G_{\mathcal{T}}$. After the algorithm has entered the second phase (Int=true), every iteration l performed until Cut=false computes a lower bound on the MIP relaxation of $B_{\mathcal{T}}$ where variables x_t take integer values, but variables y_{dij} might assume fractional values (again, a relaxation of $G_{\mathcal{T}}$). When Int=true and Cut=false for the first time, say at iteration l', either the algorithm terminates immediately or integer

Benders cuts are generated for each $d \in D$ such that $(s_d(l') - \overline{s_d}(l'))/s_d(l') > \epsilon_5$. Let us assume this last case happens. As shown in Section 5.4.1, each integer Benders cut for day d is valid, i.e., no feasible solution of $G_{\mathcal{T}}$ is removed by the addition of such cut to $B^+_{\mathcal{T}}$. Moreover, after adding an integer Benders cut for day d, variable θ_d always represents a lower bound on the optimal value of the corresponding daily Benders subproblem, since in case that cut becomes active at an optimal solution of $G_{\mathcal{T}}$, θ_d is then exactly equal to the optimal value of the Benders subproblem for day d (otherwise, θ_d is not influenced by that cut). This implies that, at the next iteration $l' + 1$, we have $Z^L(l' + 1) \leq Z(G_{\mathcal{T}})$. At any subsequent iteration l, the same arguments as above show that the lower bound $Z^L(l)$ on $Z(B^+_{\mathcal{T}})$ is also a lower bound on $Z(G_{\mathcal{T}})$. ■

Proposition 3. *If $\epsilon_3 = \epsilon_4 = 0$, the algorithm converges to an optimal solution of the LP relaxation of $G_{\mathcal{T}}$ in a finite number of iterations of the first phase (when Int=false).*

Proof. Since the number of classical Benders cuts (5.20) is finite, the first phase (when Int=false) terminates in a finite number of iterations. The CG algorithm is stopped only when the LP relaxation of $B_{\mathcal{T}}$ has been solved, which implies $\theta_d(l) = \overline{\theta_d}(l)$ for each $d \in D$. Since $\epsilon_4 = 0$, the first phase cannot stop prematurely and necessarily ends with $\overline{Z^U}(l) \leq Z^L(l)$ or Cut=false. In fact, these two conditions are equivalent when $\epsilon_3 = 0$, since Cut=false if $\overline{s_d}(l) \leq \overline{\theta_d}(l)$ for each $d \in D$, which implies that $\overline{Z^U}(l) \leq \sum_{d\in D} \overline{s_d}(l) \leq \sum_{d\in D} \overline{\theta_d}(l) = \sum_{d\in D} \theta_d(l) = Z^L(l)$. Because $\overline{Z^U}(l) \geq Z^L(l)$, the first phase ends with $\overline{Z^U}(l) = Z^L(l)$, meaning that the LP relaxation of $B_{\mathcal{T}}$ (hence, of $G_{\mathcal{T}}$) is solved. ■

Proposition 4. *If $\epsilon_1 = \epsilon_2 = \epsilon_3 = \epsilon_5 = 0$, the algorithm converges to an optimal solution of $G_{\mathcal{T}}$ in a finite number of iterations.*

Proof. Since the number of classical and integer Benders cuts is finite, the algorithm terminates in a finite number of iterations. Because $\epsilon_1 = 0$, the algorithm cannot stop prematurely and necessarily ends when $Z^U(l) \leq Z^L(l)$ or Cut = false (we show below that these two conditions are equivalent under the assumption that $\epsilon_2 = \epsilon_3 = \epsilon_5 = 0$). Since $\epsilon_2 = 0$, the B&P algorithm is exact and always produces an optimal solution (in variables x_t) to the current master problem, which implies $\theta_d(l) = \overline{\theta_d}(l)$ for each $d \in D$. Because $\epsilon_3 = 0$, we have, when Cut=false, $\overline{s_d}(l) \leq \overline{\theta_d}(l)$ for each $d \in D$, which implies that $\sum_{d\in D} \overline{s_d}(l) \leq \sum_{d\in D} \overline{\theta_d}(l) = \sum_{d\in D} \theta_d(l) = Z^L(l)$. At the last iteration l, no integer Benders cut is generated (Cut=false) and, since ϵ_5=0, we have $s_d(l) \leq \overline{s_d}(l)$ for each $d \in D$, which implies that $\sum_{d\in D} s_d(l) \leq \sum_{d\in D} \overline{s_d}(l) \leq Z^L(l)$. By definition of $Z^U(l)$ and using Proposition 2, we have $\sum_{d\in D} s_d(l) \geq Z^U(l) \geq Z(G_{\mathcal{T}}) \geq Z^L(l) \geq \sum_{d\in D} s_d(l)$, from which we conclude that $Z^L(l) = Z^U(l) = Z(G_{\mathcal{T}})$. ■

5.5 Computational Experiments

In this section, we present the computational experiments we have performed on our implementation of the BD/CG method. The computing environment used for the tests consists of a 1-processor Intel Xeon X5675 with 96 GB of RAM running at 3.07GHz and operating on a 64-bit GNU/Linux operating system. The BD/CG algorithm was implemented in C++. Both the LP relaxation of the master problem and the LP relaxation of the Benders subproblems were solved by using the barrier method of CPLEX version 12.5.0.1. A relative gap tolerance of 0.01 was set as a stopping criterion for solving the MIPs with CPLEX B&B. We used the following values for the tolerance parameters: $\epsilon_1 = \epsilon_3 = \epsilon_5 = 0.00001$ and $\epsilon_2 = \epsilon_4 = 0.01$.

In Section 5.5.1, we show the results obtained on MATSP instances defined over a one-week planning horizon, which are compared with the ones obtained when using the B&P approach presented in Restrepo et al. [76] for the personalized variant of the problem. In Section 5.5.2, we show the results obtained on MASSP instances, the special case of MATSP defined over a single day, which are compared with the ones obtained when using the grammar-based integer programming approach presented in Côté et al. [30].

5.5.1 Results on MATSP Instances

In this section, we present results on MATSP instances. First, we introduce the definition of the problem and the grammar used to create the daily shifts. Then, we present the set of instances used in the experiments. Finally, we present and analyze the computational results.

Problem Definition and Grammar

Tour generation

1. The planning horizon is seven days, where each day is divided into 96 time periods of 15 minutes.
2. Shifts are not allowed to span from one day to another (discontinuous problem).
3. The tour working length should fall between 35 and 40 hours per week.
4. The number of working days in the tour should fall between five and six.
5. There must be a minimum rest time of twelve hours between consecutive shifts.

Daily shift generation

1. Shifts can start at any time period during any day d, allowing enough time to complete their duration in day d.

2. Three types of shifts are considered: 8-hour shifts with 1-hour lunch break in the middle and two 15-minute breaks. 6-hour shifts with one 15-minute break and no lunch, and 4-hour shifts with one 15-minute break and no lunch.
3. If performed, the duration of a work activity is at least one hour and at most five hours.
4. A break (or lunch) is necessary between two different work activities.
5. Work activities must be inserted between breaks, lunch and rest stretches.
6. A fixed number of employees $|E|$ is given, therefore undercovering and overcovering of staff requirements is allowed.

Let a_j be a terminal symbol that defines a time period of work activity $j \in J$. Let b, l and r be the terminal symbols that represent break, lunch and rest periods, respectively. In productions $\Pi \in P$, $\Pi \rightarrow_{[\text{min, max}]}$ restricts the subsequences generated by a given production to a length between a minimum and maximum number of time periods. The grammar and the productions that define the anonymous discontinuous MATSP are as follows:

$$
\begin{aligned}
G =& (\Sigma = (a_j \ \forall j \in J, b, l, r), \\
& N = (S, F, Q, N, W, A_j \ \forall j \in J, B, L, R), P, S), \\
& S \rightarrow RFR|FR|RF|RQR|QR|RQ|RNR|NR|RN, \ B \rightarrow b, \ L \rightarrow llll, \\
& F \rightarrow_{[38,38]} NLN, \ Q \rightarrow_{[25,25]} WBW, \\
& N \rightarrow_{[17,17]} WBW, \ R \rightarrow Rr|r, \\
& W \rightarrow_{[4,20]} A_j \ \forall j \in J, \ A_j \rightarrow A_j a_j | a_j \ \forall j \in J.
\end{aligned}
$$

Instances

Instances are divided into two groups according to the shape of the demand profile: smooth demand behaviour and erratic demand behaviour. Instances with smooth demand behaviour correspond to real data from a small retail store, where the staff requirements for up to ten work activities vary slightly from one day to the next. Instances with erratic demand behaviour show significant variations in the staff requirements for up to five work activities. These instances were randomly generated in the following way. Given a fixed number of employees $|E|$, we start creating a set of feasible schedules (multi-activity tours), then randomly choose one schedule per employee $e \in E$. From these schedules, we derive the associated demand profile along the planning horizon. The demand profile represents the required number of employees for each work activity at each time period in the planning horizon. Undercovering and overcovering of staff requirements are generated by randomly

adding or removing demand.

Table 5.1 shows the size of the instances, divided into two groups: G1 includes instances with a smooth demand profile, while G2 includes randomly generated instances with an erratic demand profile. Ten different staff requirements were generated for each instance. For each set of instances, we present the number of activities (*Nb.Act*), the average number of employees (*Nb.Emp*) and several grammar-related statistics: the average number of children of the root node (*Nb.ChRoot*), the average number of and-nodes (*Nb.AndNodes*) without including the children of the root node, the average number of or-nodes (*Nb.OrNodes*) without including the leaves, and the average number of leaves (*Nb.Leaves*) of DAG Γ_d. We also present the average number of nodes in the directed acyclic graph from the pricing subproblem and the average number of arcs denoted by *Nd.* $G(\mathcal{N}, \mathcal{A})$ and *Arcs* $G(\mathcal{N}, \mathcal{A})$, respectively. Observe that the number of variables in the master problem is equal to *Nd.* $G(\mathcal{N}, \mathcal{A})$ + number of columns generated.

Table 5.1 Size of MATSP instances

Group	*Nb.Act*	*Nb.Emp*	*Nb.ChRoot*	*Nb.AndNodes*	*Nb.OrNodes*	*Nb.Leaves*	*Nd.* $G(\mathcal{N}, \mathcal{A})$	*Arcs* $G(\mathcal{N}, \mathcal{A})$
	1	8	106	4,997	4,044	202	744	166,184
	2	9	104	6,773	4,994	260	731	160,567
	3	11	107	8,808	6,138	325	749	166,933
	4	19	107	10,724	7,190	388	751	168,430
G1	5	24	109	12,843	8,379	455	768	175,379
	6	28	114	15,206	9,722	530	797	188,923
	7	32	112	17,057	10,721	590	789	185,010
	8	40	113	19,062	11,822	655	792	186,374
	9	37	111	20,830	12,772	713	782	182,525
	10	36	114	23,180	14,088	787	800	189,741
	1	18	139	6,535	5,320	246	973	275,745
	2	22	139	8,819	6,567	318	973	275,745
G2	3	29	139	11,103	7,814	390	973	275,745
	4	37	139	13,387	9,061	462	973	275,745
	5	43	139	15,671	10,308	534	973	275,745

Results

Tables 5.2 and 5.3 present the computational results on the smooth demand and the erratic demand instances, respectively, for the BD/CG algorithm (*BD/CG*) and for the B&P approach (*B&P*). We set a 2-hour time limit to solve the instances with up to five work activities and a 3-hour time limit to solve the instances with more than five work activities. For the BD/CG algorithm, we present the average of the total CPU time in seconds to solve the problem (*T. time*). The total CPU time is decomposed into four parts: the time spent

solving the MIP of the master problem (*T. MIP*), the time spent in the CG approach (*T. CG*) (time to solve the pricing subproblems + time to solve the LP relaxation of the problem when the method adds new columns), the time used to solve the LP relaxation of the Benders subproblems (*T. LR BSP*) and the time spent to solve the MIP of the Benders subproblems (*T. MIP BSP*). We also present the total number of integer Benders cuts generated over the total number of problems that required these cuts ($IBC/Nb.P$), the average gap (in %) between the upper bound (Z^U) and the lower bound (Z^L) of the problem computed as: $Gap = 100 \times (Z^U - Z^L)/Z^U$, and the number of instances solved to optimality (*Opt.*). The solution of an instance is considered to be optimal if no more Benders cuts (classical and integer) need to be added when the algorithm stops. Results for the B&P approach are presented in the rows labeled *B&P*. The average CPU time in seconds to solve the LP relaxation of the problem at the root node is presented in *T. root*. *T. time* shows the average total time to solve the problem (LP relaxation at the root node + branching). *Gap* presents the integrality gap between the best upper bound (Z^U) and best lower bound (Z^L). *Gap* is defined in a similar fashion as for the BD/CG algorithm. *Opt.* shows the number of instances solved to optimality. In this case, the solution of an instance is considered to be optimal if the integrality gap is less than or equal to 1%. Results for instances with an erratic demand profile, as well as results for the smooth demand profile with more than four work activities are not reported for the B&P approach, because the method exhibited convergence issues for these instances.

Table 5.2 Results on MATSP instances with smooth demand shape

Nb.Act	1	2	3	4	5	6	7	8	9	10
BD/CG										
T.time	198.95	127.83	254.67	1,363.31	1,464.55	1,745.39	2,418.80	1,324.42	1,526.01	7,767.48
T. MIP	10.28	92.97	186.93	1,207.38	1,295.90	1,516.43	1,638.96	1,016.45	1,128.59	5,411.78
T. CG	9.62	13.86	18.01	71.23	35.29	47.08	547.57	32.97	36.72	594.98
T. LR BSP	7.99	20.22	48.37	81.69	127.85	170.93	221.45	200.20	309.60	635.41
T.MIP BSP	0.22	0.44	0.81	1.98	4.29	9.48	8.30	73.78	49.52	1,124.75
IBC/Nb. P	0/0	0/0	1/1	3/3	2/1	17/3	1/1	3/3	3/2	3/2
Gap	0.51%	0.62%	0.41%	0.56%	0.35%	0.67%	1.24%	0.53%	0.68%	2.25%
Opt.	10	10	10	10	10	9	6	9	9	6
B&P										
T. root	20.13	243.08	598.49	2,008.12	-	-	-	-	-	-
T. time	245.94	6,619.27	7,884.65	7,251.86	-	-	-	-	-	-
Gap	0.44%	8.15%	37.32%	56.69%	-	-	-	-	-	-
Opt.	10	2	0	0	-	-	-	-	-	-

Table 5.3 Results on MATSP instances with erratic demand shape

Nb.Act	1	2	3	4	5
BD/CG					
T.time	333.65	2,571.05	1,382.79	3,354.43	4,574.66
T. MIP	223.47	2,342.27	1,158.34	1,848.66	2,124.39
T. CG	85.83	115.06	61.11	357.87	473.70
T. LR BSP	23.25	81.96	159.84	258.55	428.85
T.MIP BSP	0.29	30.50	2.74	887.91	1,549.62
IBC/Nb. P	0/0	2/2	0/0	6/3	4/2
Gap	0.84%	1.02%	0.94%	2.50%	2.74%
Opt.	10	9	9	6	4

From Tables 5.2 and 5.3, one can observe that the time to solve the master problem is the highest among the four components. Solving the LP relaxation of the master problem does not require too much time, but when the algorithm switches to the integer version of the master problem and more optimality cuts are added, the time to solve the master problem increases with each iteration. The time spent to find new columns, as well as the time to solve the LP relaxation of the Benders subproblems, represent small portions of the total time.

Note that the BD/CG algorithm is able to find high-quality integer solutions for almost all instances with smooth demand behaviour and, when optimality is not reached within the time limit, the value of *Gap* is most often within 1% and does not exceed 2.25%. For instances with erratic demand behaviour, computational times and solution quality are worse than those reported in Table 5.2. However, one can observe that even if the instances are not solved to optimality, the value of *Gap* does not exceed 2.75%. We found that the instances that have more quantity of overcovering than undercovering are easier to solve than the instances that have a similar quantity of undercovering and overcovering. Observe that few instances required the generation of integer Benders cuts and, among these instances, the majority required just one or two cuts (only one instance in the group of 6 activities needed 10 integer Benders cuts).

The comparison between the proposed method and the B&P approach developed for the personalized variant of the problem suggests that the BD/CG algorithm is a better alternative when employees have the same skills. Notably, in almost all the instances, the average total CPU time to solve the instances when using the BD/CG algorithm is smaller than the average time to solve the LP relaxation at the root node when the B&P approach is used. This can be mostly attributed to the symmetry issues exhibited by the B&P method when all employees have the same skills.

5.5.2 Results on MASSP Instances

In this section, we present computational results on MASSP instances. In this case, some modifications have been done to the proposed approach in order to solve daily problems. First, it is not necessary to generate columns, since the master problem only includes variables related to shift shells. Thus, the B&P method used to solve the master problem is replaced by a call to a state-of-the-art B&B code (we use CPLEX). Second, constraints (5.9)-(5.10) in the master problem are replaced by $\sum_{A^{\Pi,k}_{d1n} \in ch(O^S_{d1n})} v^{\Pi,k}_{d1n} = |E|$. Third, the integer Benders cuts do not include the variables related to employee days-off. Observe that the Benders subproblem has the same structure, but that only one Benders subproblem is solved per iteration. The detailed definition of the problem and the grammar used to compose shifts are presented in Section 5.2 of Côté et al. [30]. Before analyzing the results, we first present the instances used in our tests.

Instances

Instances are divided into two groups. The first group, G1, contains smooth demand profile instances from a small retail store, allowing up to ten work activities. The second group, G2, consists of instances with a demand profile that follows a normal distribution, allowing from eleven up to thirty work activities. The shape of the demand profile was generated based on the number of employees $|E|$, the number of activities $|J|$, a random standard deviation and a total demand to distribute along the planning horizon. The characteristics and size of the instances are summarized in Table 5.4. The notation used in this table is the same as the one used in Table 5.1. Ten different staff requirements were generated for each instance.

Problems dealing with up to 30 work activities (and often more) are typically found in call centers, retail stores and industries where employees can move from one work station to another (i.e.: parking companies).

Table 5.4 Size of MASSP instances

Group	*Nb.Act*	*Nb.Emp*	*Nb.ChRoot*	*Nb.AndNodes*	*Nb.OrNodes*	*Nb.Leaves*
	1	5	119	33,879	8,229	229
	2	7	119	35,365	9,227	288
	3	9	119	37,649	10,415	351
	4	18	120	39,965	11,621	415
	5	22	121	42,308	12,842	480
G1	6	28	124	46,064	14,473	556
	7	36	123	47,562	15,477	616
	8	38	123	49,521	16,611	679
	9	38	123	51,479	17,745	742
	10	37	124	53,957	19,044	811
	11	42	143	68,409	23,295	1,006
	12	41	143	70,730	24,562	1,079
	13	41	143	73,051	25,829	1,152
	14	42	143	75,372	27,096	1,225
	15	44	143	77,693	28,363	1,298
G2	16	45	143	80,014	29,630	1,371
	17	49	143	82,335	30,897	1,444
	18	48	143	84,656	32,164	1,517
	19	53	143	86,977	33,431	1,590
	20	55	143	89,298	34,698	1,663
	30	137	143	112,508	47,368	2,393

Results

Table 5.5 presents the computational results on instances dealing with up to 30 work activities for the anonymous MASSP. We set a 2-hour time limit to solve these instances. For the Benders decomposition approach (*BD*), we present the average total CPU time in seconds to solve the problem (*Tot. time*). This time is divided into three parts: the time required to solve both the LP relaxation and the MIP of the master problem (*T. BMP*), the time spent to solve the LP relaxation of the Benders subproblem (*T. LR BSP*) and the time required to solve the MIP of the Benders subproblem (*T. MIP BSP*). We also present the total number of integer Benders cuts generated over the total number of problems that required those cuts ($IBC/Nb.P$), the average gap between the upper bound (Z^U) and the lower bound (Z^L) of the problem ($Gap = 100 \times (Z^U - Z^L)/Z^U$), and the number of instances solved to optimality (*Opt.*), which corresponds to the number of instances for which no additional Benders cuts could be generated. Results for the grammar-based integer programming approach are presented in the columns labeled *GB*, where *T. time* presents the average CPU time to find an integer solution with a relative MIP gap tolerance lower than 1%, *Gap* shows the average relative MIP gap between the best upper bound (Z^U) and the best lower bound (Z^L), where $Gap = 100 \times (Z^U - Z^L)/Z^L$. The number of instances solved to optimality is presented in

the column labeled *Opt.*

Table 5.5 Results on MASSP instances

	BD							***GB***		
Nb.Act	*Tot. time*	*T. BMP*	*T. LR BSP*	*T. MIP BSP*	*IBC/Nb.P*	*Gap*	*Opt.*	*Tot. time*	*Gap*	*Opt.*
1	**199.53**	0.47	85.87	0.61	6/2	0.05%	10	272.19	0.09%	10
2	**192.41**	0.54	141.67	1.33	15/2	0.29%	10	241.15	0.17%	10
3	**227.19**	1.34	222.79	1.14	4/3	0.41%	10	1,159.92	0.25%	10
4	279.79	0.86	275.92	2.13	2/1	0.43%	10	**178.69**	0.25%	10
5	413.51	2.07	402.80	1.85	8/3	0.31%	10	**367.62**	0.05%	10
6	472.95	2.01	463.70	4.34	5/5	0.43%	10	**351.04**	0.05%	10
7	666.59	3.74	653.06	4.57	5/5	0.65%	10	**438.28**	0.17%	10
8	617.21	3.72	580.09	5.52	9/6	0.68%	10	**428.83**	0.08%	10
9	**485.77**	0.91	476.83	7.46	3/3	0.52%	10	588.02	0.14%	10
10	**618.89**	1.78	589.91	10.03	6/4	0.71%	10	772.57	0.12%	10
11	**700.65**	0.48	657.85	41.82	1/1	0.64%	10	3,096.72	0.04%	10
12	**924.81**	0.69	685.80	103.09	7/4	0.69%	10	3,680.88	0.00%	10
13	**647.74**	0.36	617.45	29.35	0/0	0.66%	10	3,013.46	0.06%	10
14	**855.55**	0.53	763.39	61.14	2/2	0.57%	10	3,453.49	2.69%	9
15	**915.54**	0.72	885.93	28.02	1/1	0.66%	10	2,724.59	0.08%	10
16	**1,121.04**	0.97	1,073.41	37.51	2/2	0.54%	10	3,120.24	0.05%	10
17	**915.31**	0.59	868.85	43.08	3/2	0.61%	10	2,976.84	0.26%	10
18	**1,201.08**	1.99	1,131.84	30.78	3/3	0.59%	10	3,685.44	0.87%	9
19	**1,389.34**	3.19	1,355.90	28.53	1/1	0.52%	10	3,380.61	0.18%	10
20	**1,274.71**	2.50	1,215.68	32.80	4/4	0.59%	10	3,416.84	0.00%	10
30	**3,636.53**	0.81	3,179.98	454.41	3/3	0.80%	10	6,130.31	7.16%	4

From Table 5.5, we can conclude that the BD approach succeeds to find, within the computational time limit, high-quality solutions, within 1% of optimality, for all the instances tested. When the proposed approach is compared with the grammar-based integer programming approach, results show that BD presents a better average total CPU time for 16 out of 21 instances, and that, in the best case, the method is five times faster (instances with 3 and 13 work activities). The difference in performance of the two methods can be attributed to the fact that solving the problem with a B&B method requires more effort than solving the problem by adding Benders cuts. The proposed BD approach takes advantage of the structure of the problem by fixing the shift shell variables to efficiently solve the Benders subproblem (which is the part that requires more time).

Regarding CPU times, note that, contrary to what we observed for the MATSP instances, the most time-consuming component is related to the LP solution of the Benders primal subproblem, for which CPU times increase with the number of activities. Solving the master problem (both LP relaxation and MIP) was the part that required the least effort. This can

be attributed to the fact that allocating the work activities and the breaks to the shifts in order to minimize undercovering and overcovering is more difficult than assigning daily shifts to employees. Finally, observe that in only 57 out of 210 instances, the generation of integer Benders cuts was needed.

5.6 Concluding Remarks

In this paper, we presented a combined Benders decomposition and column generation approach to solve the MATSP. Due to its structure, the master problem is solved by column generation. Benders subproblems were modelled with context-free grammars to implicitly tackle all the work rules for the composition of shifts and to allocate work activities and breaks to the shifts. Although the Benders primal subproblems do not possess the integrality property, we showed that the generation of integer Benders cuts, in addition to classical Benders cuts, guarantees the convergence of the method under mild assumptions.

The proposed approach was tested on real-world instances and randomly generated instances of the MATSP (one-week planning horizon) and the MASSP (one-day planning horizon). Results on MATSP instances showed that our method was able to find high-quality integer solutions for instances dealing with up to ten work activities. When compared with a B&P approach, our method exhibited faster solution times and provided better upper bounds for the most difficult instances. Regarding the MASSP, the Benders decomposition approach was able to solve, within 1% of optimality, instances with up to 30 work activities. When the method was compared with the grammar-based integer approach presented in Côté et al. [30], our approach presented competitive and often better solution times.

CHAPTER 6 ARTICLE 3: A TWO-STAGE STOCHASTIC PROGRAMMING APPROACH FOR MULTI-ACTIVITY TOUR SCHEDULING

Companies that operate outside the standard 8-hours shift, 5-days per week schedule and that face wide fluctuations in demand for services continuously struggle to optimize the allocation and composition of their workforce. Recent research has shown that alternative personnel scheduling approaches that take demand uncertainty into account can lead to significant reductions in labor costs [10].

In this chapter of the thesis we address the stochastic extension of the discontinuous anonymous MATSP presented in Chapter 5. We formulate the problem as a two-stage stochastic programming model decomposable by days and by scenarios, where the random vector representing stochastic perturbations of demand is assumed to be non-negative and to have finite support. The two-stage stochastic programming model allows to make a decision on the employee schedule (tours and daily shift shells) before a realization of the daily demand is known. Then, after demand becomes known, a recourse action is implemented to compensate deficiencies in the previously made schedules (e.g., allocation of work activities and breaks to daily shifts, undercovering and overcovering of demand).

As a solution approach, we implemented a heuristic multi-cut L-shaped method. The first-stage problem is solved by a heuristic CG approach. The second-stage problems are modeled with context-free grammars, as explained in Chapter 5. Computational experiments on real and randomly generated instances allow to conclude that that the use of the stochastic model prevents to incur additional staffing costs, when compared with the expected value problem.

The next article was submitted to *European Journal of Operations Research* in October 2015.

A Two-Stage Stochastic Programming Approach for Multi-Activity Tour Scheduling

María I. Restrepo
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, Canada

Bernard Gendron
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département d'informatique et de recherche opérationelle, Université de Montréal, Montréal, Canada

Louis-Martin Rousseau
Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, CIRRELT
Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, Canada

Abstract.

This paper addresses a discontinuous multi-activity tour scheduling problem under demand uncertainty and when employees have identical skills. The problem is formulated as a two-stage stochastic programming model, where first-stage decisions correspond to the assignment of employees to weekly tours, while second-stage decisions are related to the allocation of work activities and breaks to daily shifts. A multi-cut L-shaped method is presented as a solution approach. Computational results on real and randomly generated instances show that the use of the stochastic model helps to prevent additional costs, when compared with the expected-value problem solutions.

Keywords. Stochastic multi-activity tour scheduling problem, Two-stage stochastic programming model, L-shaped method, Column generation, Context-free grammars.

6.1 Introduction

The *multi-activity tour scheduling problem* (MATSP) is the integration of two problems, *the multi-activity shift scheduling problem* (MASSP) and the *days-off scheduling problem.* In the MASSP, the *planning horizon* is usually one day divided into *time periods* of equal length. Since employees can perform different *work activities* during the same shift, the MASSP

is concerned with choosing the work activities and the rest periods to assign to shifts to respond to a demand for service, that is translated into a *demand* for the number of employees required for each work activity and time period. The days-off scheduling problem deals with the selection of employee *working days* and *days-off* over a planning horizon of at least one week. In the MATSP, the constraints characterizing the feasibility of daily shifts and weekly tours, as well as the *work rules* for the allocation of work activities and rest breaks to the shifts, are usually defined by *employee regulations* and *workplace agreements*. The MATSP can be categorized into different variants depending on the characteristics considered. For instance, the *personalized* version of the MATSP appears when employees have individual *preferences* and *skills*. The *anonymous* version of the MATSP corresponds to the case when employees have identical skills. When shifts are allowed to span from one day to another, the *continuous* version of the MATSP arises; otherwise, we have the *discontinuous* version of the problem. In this paper, we consider the discontinuous anonymous version of the MATSP.

Realistic applications of the MATSP for companies that operate outside the standard 8-hours shift, 5-days per week schedule and that face wide fluctuations in demand become challenging due to several factors. First, complex large-scale models result as a consequence of considering multiple work activities and flexibility in the composition of daily shifts and weekly tours. Second, since demand is typically unknown when scheduling decisions needs to be taken, specialized solution techniques that allow to include this variability should be developed. Specifically, such techniques allow to make a decision on the employee schedule before a realization of the demand is known. Then, after demand becomes known, a recourse action should be implemented to compensate deficiencies in the previously made schedules (e.g., undercovering and overcovering of demand).

In this paper, we address the discontinuous *stochastic multi-activity tour scheduling problem* (SMATSP) for employees with identical skills. In this problem, a long-term staffing decision needs to be made while hedging for the short-term demand uncertainty. The problem is formulated as a two-stage stochastic programming model, decomposable by days and by scenarios, in which first-stage decisions correspond to the assignment of employees to weekly tours, while second-stage decisions (recourse actions) are related to the allocation of work activities and breaks to daily shifts. The contribution of this paper lies in the proposal of an approach to efficiently solve practical instances of the problem. A heuristic multi-cut L-shaped method is implemented as a solution approach. Because the complete enumeration of weekly tours makes the problem intractable, the first-stage problem is solved via column generation. Additionally, the second-stage problems benefit from the use of *context-free grammars* to include work rules regarding the definition of shifts and to efficiently handle the multi-activity context.

The paper is organized as follows. In Section 6.2, we review the relevant literature on shift scheduling and tour scheduling problems with multiple work activities and stochastic demand. Then, we present some background material related to the use of grammars for multi-activity shift scheduling problems. In Section 6.3, we describe the two-stage model for the SMATSP. In Section 6.4, we introduce the solution approach to solve the problem. Computational experiments are presented and discussed in Section 6.5. The concluding remarks are presented in Section 6.6.

6.2 Background Material

In this section, we review some literature on the models and methods to solve the MASSP and the MATSP. We also present some references on workforce problems under stochastic demand. Then, we finish with an introduction on the use of grammars for the MASSP.

6.2.1 Literature Review on Multi-Activity Shift and Tour Scheduling

Although mono-activity shift scheduling and tour scheduling problems have been extensively studied in the literature during the last few decades [4, 43, 44, 82], only recently some attention has been given to the problem that deals with multiple work activities. Ritzman et al. [78] propose one of the first approaches to solve the MATSP. The method is based on a heuristic solution approach that integrates a construction method with a simulation component. Although employees are assigned to specific operations, breaks and rules related to switching between work activities are not considered. Heuristic approaches that use column generation (CG) [77] and tabu search [33] are also proposed to solve multi-activity shift scheduling problems over multiple days. Even though both approaches tackle long time horizons, the constraints characterizing the feasibility of weekly tours (e.g., total tour length) are not included in the formulation of the problem. In a similar way, Detienne et al. [38] and Lequy et al. [55] solve a multi-activity assignment problem by using decomposition techniques and heuristics based on CG and branch-and-bound (B&B) as solution methods.

Fixing the sequences of work, rest days, shift types and breaks, might reduce the complexity of personnel scheduling problems, but it can also lead to sub-optimal solutions. Constraint programming (CP) techniques aim to solve that difficulty by offering modeling languages to handle complex optimization problems. Demassey et al. [37] present a CP-based CG algorithm to model complex regulation constraints in a real-world MASSP. Quimper and Rousseau [69] use formal languages to model the work rules related to the composition of shifts in a multi-activity context. Côté et al. [28] propose two approaches for the MASSP: the first

one uses an automaton to derive a network flow model, while the second one takes advantage of context-free grammars to obtain a MIP model in which an and/or graph structure is used. Côté et al. [30] present an implicit grammar-based model for the MASSP that addresses symmetry issues by using general integer variables. Computational results show that, in the mono-activity case, the solution times of the model are comparable and sometimes superior to the results presented in the literature and that, in the multi-activity case, the model is able to solve to optimality instances with up to ten work activities. Côté et al. [31] and Boyer et al. [16] present grammar-based CG methods to solve the personalized MASSP and the personalized multi-activity multi-task shift scheduling problem, respectively. Both approaches use formal languages and dynamic programming to efficiently formulate and solve the pricing subproblems, but some limitations regarding long time horizons (e.g., one week) are present. To overcome these issues, Restrepo et al. [76] and Restrepo et al. [75] present approaches based on branch-and-price (B&P) and Benders decomposition (BD), respectively. In the former approach [76], two B&P algorithms are presented for the personalized MATSP. In the latter approach [75], a combined BD and CG method is introduced for the anonymous MATSP. In both approaches, the work rules for the composition of multi-activity shifts are expressed with context-free grammars, while some constraints that guarantee the feasibility of weekly tours are embedded into a directed acyclic graph.

6.2.2 Literature Review on Stochastic Shift and Tour Scheduling

Different models and solution approaches have been proposed in the literature to deal with stochastic demand in personnel scheduling problems. As an illustration, Easton and Rossin [40] and Easton and Mansour [39] develop heuristic methods that aim at tackling problems where demand is uncertain. Easton and Rossin [40] propose a tabu search method to solve a stochastic goal programming model that integrates and optimizes labor demand and employee scheduling. Easton and Mansour [39] present a genetic algorithm to solve shift scheduling problems in which the recourse decisions are related to the undercovering and overcovering of demand. Although both approaches aim to solve problems over a one-week planning horizon, employee patterns are previously defined and only a small set of stochastic scenarios is considered. Bard et al. [10] propose a heuristic two-stage model that addresses tour scheduling problems over a one-week planning horizon. First-stage variables are related to the number of full-time and part-time employees hired, while second-stage decisions correspond to the allocation of employees to specific shifts during the week. Computational experiments on real instances that consider three stochastic scenarios (high, medium and low demand) show that significant savings are likely when the recourse problem is used.

Some studies that use decomposition approaches have been recently proposed as alternatives to solve workforce planning problems when demand is uncertain. Pacqueau and Soumis [63] propose a heuristic two-stage model to solve a shift scheduling problem. The proposed model is based on a decomposition of Aykin's [5] implicit model, where first-stage variables are associated with the allocation of full-time shifts to the employees and recourse decisions correspond to hiring part-time employees, using overtime for full-time shifts, the allocation of breaks and the allowance of understaffing. Punnakitikashem et al. [68] introduce a stochastic nurse scheduling problem that aims to minimize staffing costs and excess workload. The authors present a BD approach, a Lagrangian relaxation with a BD approach and a nested BD approach as solution methods. Computational results suggest that simultaneously considering nurse staffing and assignment is more desirable than doing them sequentially. Similarly, Kim and Mehrotra [53] present an integrated staffing and scheduling approach applied to nurse management when demand is uncertain. The problem is formulated as a two-stage stochastic integer program, where daily shifts and weekly patterns are previously enumerated. First-stage decisions correspond to the number of employees assigned to daily shifts and to weekly patterns, while second-stage decisions correspond to: 1) the possibility of adding or canceling daily shifts for every working pattern; 2) allowing undercovering or overcovering of demand. A set of valid mixed-integer rounding inequalities that describe the convex hull of feasible solutions in the second-stage problem are included. Consequently, the integrality of the second-stage decision variables can be relaxed. Computational experiments show that the use of the stochastic model prevents the hospital from being overstaffed. An L-shaped method is presented in Robbins and Harrison [79] to solve a combined server-sizing and staff scheduling problem for call centers in which a service level agreement must be satisfied. First-stage decisions correspond to the employee staffing, while second-stage decisions correspond to the computation of a telephone service shortfall. Computational results show that ignoring variability is a costly decision, since the value of the stochastic solution for the model is substantially high.

Very limited literature is available on stochastic workforce planning for employees who have various skills to work on different activities, tasks or unit departments. Zhu and Sherali [85] address a workforce planning problem for employees with multiple skills between service centers. A two-stage model under demand fluctuations is presented, where first-stage decisions correspond to personnel recruiting and allocation of employees to multiple locations, while second-stage decisions consists in reassigning the workforce among the locations. The scheduling of cross-trained workers in a multi-department service environment with random demand is addressed in Campbell [25]. The author presents a two-stage model decomposable by days and by scenarios, where first-stage decisions are related to the scheduling of days-off

and second-stage decisions correspond to the allocation of available employees at the beginning of each day. In the approach, days-off are previously defined and only a small number of scenarios is considered (10 in total). Parisio and Jones [65] present a two-stage stochastic model for a multi-skill tour scheduling problem in retail outlets, where first-stage variables are associated with the assignment of employees to weekly schedules, while recourse decisions correspond to the allocation of overtime and to the undercovering and overcovering of demand. Although multiple work activities are included in the problem, the authors assume employees are allowed to work in only one activity per shift.

Even though some authors have tried to tackle personnel scheduling problems under stochastic demand, none of the previous studies consider the integration of days-off scheduling with shift scheduling in a multi-activity context. The method proposed in this paper addresses the discontinuous MATSP when demand is uncertain and employee skills are identical. Unlike the previous approaches, employee patterns and daily shifts are not previously fixed and a high degree of flexibility is included in their composition. Additionally, the multi-activity context is efficiently handled with context-free grammars, which are reviewed next.

6.2.3 Grammars for Multi-activity Shift Scheduling

In shift scheduling, a *context-free grammar* (CFG) can be defined as a finite set of work rules that are used to generate valid sequences of work (shifts) for a given day $d \in D$, where $|D|$ denotes the number of days in the planning horizon. A CFG consists of a tuple $G_d = \langle \Sigma_d, N_d, S_d, P_d \rangle$, where:

- Σ_d represents an alphabet of characters called the *terminal symbols* for day d, which consists of work activities, breaks, lunch breaks, and rest stretches.
- N_d is a finite set of *non-terminal symbols* for day d.
- $S_d \in N_d$ is the *starting symbol* for day d.
- P_d is a set of *productions* for day d, represented as $A \rightarrow \alpha$, where $A \in N_d$ is a non-terminal symbol and α is a sequence of terminal and non-terminal symbols. The work rules used to generate shifts are represented by the set of productions. The productions of a grammar can be used to generate new symbol sequences until only terminal symbols are part of the sequence.

A *parse tree* is a tree where each inner-node is labeled with a non-terminal symbol N_d and each leaf is labeled with a terminal symbol Σ_d. A grammar recognizes a sequence if and only

if there exists a parse tree where the leaves, when listed from left to right, reproduce the sequence.

A *DAG* Γ_d is a *directed acyclic graph* that embeds all parse trees associated with words (shifts) for day d of a given length n recognized by a grammar. The DAG Γ_d has an and/or structure where the and-nodes represent productions (work rules) from P_d and the or-nodes represent non-terminals from N_d and letters from Σ_d. An and-node is true if all of its children are true. An or-node is true if one of its children is true. The root node is true if the grammar accepts the sequence encoded by the leaves. In Γ_d, O^{π}_{dil} denotes the or-node associated with $\pi \in N_d \cup \Sigma_d$, i.e., with non-terminals from N_d or letters from Σ_d, that generates a subsequence at position i of length l for day d. Note that if $\pi \in \Sigma_d$, the node is a leaf and l is equal to one. On the contrary, if $\pi \in N_d$, the node represents a non-terminal symbol and $l > 1$. $A^{\Pi,k}_{dil}$ is the kth and-node representing production $\Pi \in P_d$ that generates a subsequence from position i of length l at day d. There are as many $A^{\Pi,k}_{dil}$ nodes as there are ways of using P_d to generate a sequence of length l from position i. In Γ_d, the root node is described by O^{S}_{d1n} and its children by $A^{\Pi,k}_{d1n} \in ch(O^{S}_{d1n})$. The children of or-node O^{π}_{dil} are represented by $ch(O^{\pi}_{dil})$ and its parents by $par(O^{\pi}_{dil})$. Similarly, the children of and-node $A^{\Pi,k}_{dil}$ are represented by $ch(A^{\Pi,k}_{dil})$ and its parents by $par(A^{\Pi,k}_{dil})$. The sets of or-nodes, and-nodes and leaves in Γ_d are denoted by O_d, A_d and L_d, respectively. The DAG Γ_d is built by a procedure proposed in Quimper and Walsh [71].

Grammar G_1 presents an example on the use of context-free grammars for multi-activity shift scheduling. Two activities, w_1 and w_2, must be scheduled, shifts have a length of $n = 4$ time periods and should contain exactly one break, b, of one time period that can be placed anywhere during the shift except at the first or the last time period. For clarity, we do not include the subscript of the day in the notation of grammar G_1 and nodes from Γ_1. The grammar that defines the set of feasible shifts on this example follows:

$G_1 = (\Sigma = (w_1, w_2, b), N = (S, X, W, B), S, P)$,
where productions P are: $S \rightarrow XW$, $X \rightarrow WB$, $W \rightarrow WW|w_1|w_2$, $B \rightarrow b$,
and symbol $|$ specifies the choice of production.

In the previous example, productions $W \rightarrow w_1$, $W \rightarrow w_2$ and $B \rightarrow b$ generate the terminal symbols associated with working on activity 1, working on activity 2, or having a break, respectively. Production $W \rightarrow WW$ generates two non-terminal symbols, W, meaning that the shift will include a working subsequence. Production $X \rightarrow WB$ means that the shift will include working time followed by a break. Finally, production $S \rightarrow XW$ generates a

sequence of length four (the daily shift), which includes working time followed by a break to finish with more working time.

Figure 6.1 represents the DAG Γ_1 associated with G_1. Observe that there are 16 parse trees (different shifts) embedded in Γ_1. As an illustration, we present a dotted-parse tree that generates shift $w_1bw_1w_2$, and a dashed-line parse tree that generates shift $w_2w_2bw_1$.

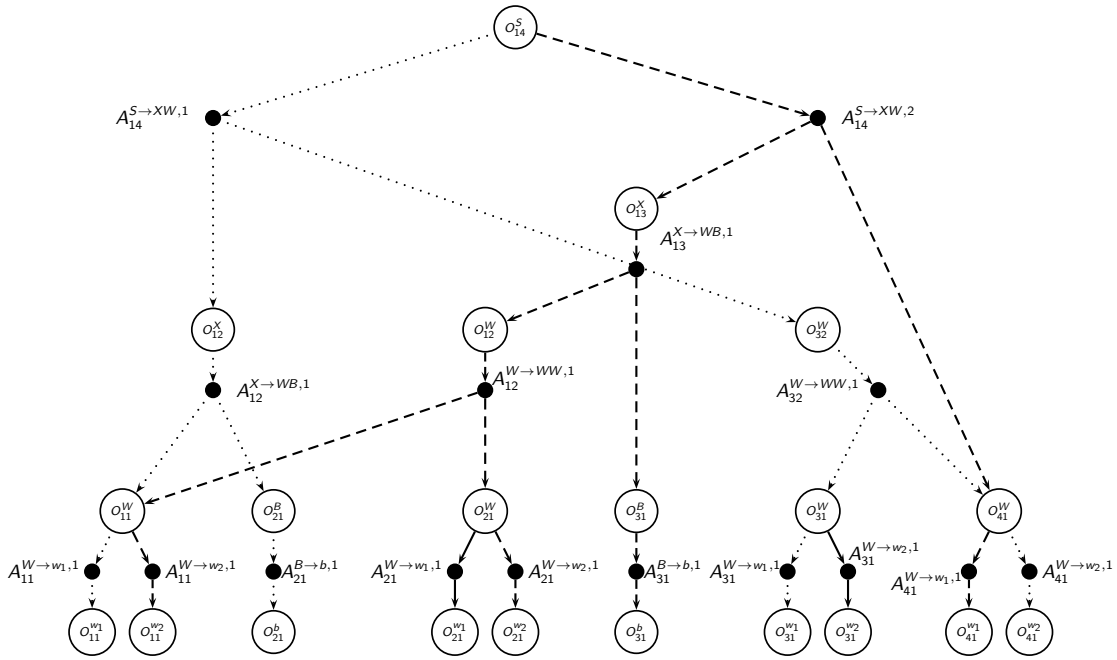


Figure 6.1 DAG Γ_1 on words of length four and two work activities.

Note that the children of the root node ($\{A_{14}^{S\to XW,1}, A_{14}^{S\to XW,2}\} \in ch(O_{14}^{S})$) can be seen as shift "shells" because they do not consider the allocation of specific work activities to the shifts, only the shift starting time and its length. Hence, and-nodes $A_{d1n}^{\Pi,k}$ are characterized by their starting time $t_{d1n}^{\Pi,k}$, working length $w_{d1n}^{\Pi,k}$, and length including breaks $l_{d1n}^{\Pi,k}$. In Γ_1, and-node $A_{14}^{S\to XW,1}$ generates shift $wbww$, while and-node $A_{14}^{S\to XW,2}$ generates shift $wwbw$. Both shifts have a working length of three time periods, a total length of four time periods and both start at time period one ($i = 1$).

Although the expressiveness of grammars allow to encode a large number of work rules for the composition of daily shifts, some limitations regarding shift total length are present when long planning horizons are included in the problem (e.g., one week). To circumvent this problem, Restrepo et al. [75] present an approach that combines BD and CG to solve the deterministic discontinuous MATSP for employees with identical skills. The model combines

an explicit definition of weekly tours with the implicit definition of daily shifts from Côté et al. [30]. Since the model presents a nice block structure decomposable by days, it is used in the formulation of the two-stage stochastic problem, presented next.

6.3 Two-Stage Stochastic Problem

Stochastic shift and tour scheduling formulations extend and adapt deterministic models to allow schedule modifications at a time closer to the actual demand realization. Two-stage stochastic programming models give an example of such extensions. In these models, some decisions must be made in the first-stage before values of random variables are observed. Then, in the second-stage, a recourse action can be adopted after observing the actual values of the random variables to adjust any bad decision previously taken. In the model proposed, first-stage decisions correspond to the number of employees assigned to each tour and to each daily shift shell, while second-stage decisions (recourse actions) correspond to the allocation of breaks and work activities to daily shifts and to the undercovering or overcovering of demand.

The second-stage problem is formulated with the implicit model proposed in Côté et al. [30]. In this approach, the authors translate the logical clauses associated with $\Gamma_d, d \in D$, into linear constraints on integer variables, where the number of employees assigned to each and-node (A_d), each or-node (O_d) and each leaf (L_d) in Γ_d are represented by an integer variable.

In the first-stage problem, we define a feasible tour as the integration of daily shift shells (children of root nodes $O^S_{d1n}, d \in D$) and days-off, over the set of days in the planning horizon. Tours must meet the work rules related to the total working length, to the number of working days, to the rest time between consecutive shifts and to the allocation of days-off. Figure 6.2 presents an example of three tours composed with the shifts presented in Γ_1. In this example, we assume that the DAG Γ_d for each day $d \in D$ is the same. Additionally, the planning horizon corresponds to seven days, the working length should fall between 15 and 18 time periods, the number of working days must fall between 5 and 6, and there are no rules for the allocation of days-off and for the rest time between shifts. Finally, S_1 corresponds to $A^{S \to XW,1}_{d14} \to wbww$, S_2 corresponds to $A^{S \to XW,2}_{d14} \to wwbw$ and DO corresponds to a day-off.

Tours \ Days	1	2	3	4	5	6	7
1	S_1	DO	DO	S_1	S_2	S_1	S_2
2	S_1	S_1	S_2	S_2	S_2	DO	S_2
3	DO	S_1	S_2	S_2	DO	S_2	S_1

Figure 6.2 Weekly tours composed of shifts from Γ_1.

In defining a model for the discontinuous SMATSP, we assume that, at the moment we can act on second-stage variables, the scenario for day $d \in D$ is fully known. Hence, staffing decisions (allocation of tours and daily shifts to employees) that are feasible to schedule without knowing in advance the demand, will be generated well ahead in time, while adjustments (allocation of work activities, position of breaks, overcovering and undercovering of demand) are made once improved (daily) demand information is available. We also assume that the random vector ξ representing the stochastic perturbations of demands has a finite support. Henceforth, we define Ω as the set of its possible realizations and $p^{(w)} > 0$ as the probability of occurrence of scenario $w \in \Omega$ with $\sum_{w \in \Omega} p^{(w)} = 1$. The notation for the stochastic model follows.

Sets

J: set of work activities;

D: set of days in the planning horizon;

I_d: set of time periods at day $d \in D$;

E: set of employees;

$\mathcal{T}$: set of feasible tours.

First-stage problem

Parameter

$\delta_{dt}^{\Pi,k}$: parameter that takes value 1, if tour t includes the kth shift shell built with production Π for day d (variable $v_{d1n}^{\Pi,k}$), and assumes value 0 otherwise.

Decision variables

x_t: integer variable that represents the number of employees assigned to tour t;

$v_{d1n}^{\Pi,k}$: variable that represents the number of employees assigned to the kth and-node built with production Π (children of the root node O_{d1n}^{S} from Γ_d).

Second-stage problem

Parameters

b_{dij}: deterministic demand for day d, time period i and activity j;

$\xi_{dij}^{(w)}$: stochastic perturbation of demand for day d, time period i and activity j for scenario w;

$b_{dij}^{(w)}$: stochastic demand for day d, time period i and activity j for scenario w, $b_{dij}^{(w)} = b_{dij} + \xi_{dij}^{(w)}$;

c_{dij}: nonnegative cost associated with one employee working on activity j, at time period i, at day d;

c_{dij}^{+}, c_{dij}^{-}: demand overcovering and undercovering nonnegative costs for day d, time period i and activity j, respectively.

Decision variables

$y_{dij}^{(w)}$: variable that denotes the number of employees assigned to activity j, at time period i, for day d under scenario w;

$v_{dil}^{\Pi,k,(w)}$: variable that denotes the number of employees assigned to the kth and-node, representing production Π from Γ_d and that generates a sequence from i of length $l < n$, under scenario w (this set of variables excludes the children of the root node O_{d1n}^{S});

$s_{dij}^{+(w)}$, $s_{dij}^{-(w)}$: slack variables representing overcovering and undercovering of demand of activity j, at time period i, for day d under scenario w, respectively.

Additionally, let $\mathcal{V}$ denote the set of variables corresponding to the union, over the set of days $d \in D$, of variables $v_{d1n}^{\Pi,k}$. Given that notation, the formulation for the first-stage model, denoted $G_{\mathcal{T}}$, is as follows.

$$f(G_{\mathcal{T}}) = \min \quad \mathcal{Q}(\mathcal{V}) \tag{6.1}$$

$$v_{d1n}^{\Pi,k} = \sum_{t \in \mathcal{T}} \delta_{dt}^{\Pi,k} x_t, \quad \forall\, d \in D,\ A_{d1n}^{\Pi,k} \in ch(O_{d1n}^{S}), \tag{6.2}$$

$$\sum_{t \in \mathcal{T}} x_t = |E|, \tag{6.3}$$

$$x_t \geq 0 \text{ and integer}, \quad \forall\, t \in \mathcal{T}, \tag{6.4}$$

$$v_{d1n}^{\Pi,k} \geq 0, \quad \forall\, d \in D,\ A_{d1n}^{\Pi,k} \in ch(O_{d1n}^{S}). \tag{6.5}$$

The objective of $G_{\mathcal{T}}$, (6.1), is to minimize the expected recourse cost $\mathcal{Q}(\mathcal{V})$. Constraints (6.2) represent the link between daily shifts (children of root nodes O_{d1n}^{S} in $\Gamma_d, d \in D$) and tours. Since a fixed number of employees is given and all the employees have the same skills, constraint (6.3) guarantees that exactly $|E|$ employees are assigned to the tours. Finally, constraints (6.4)-(6.5) set the non-negativity and integrality of variables x_t and the non-negativity of variables $v_{d1n}^{\Pi,k}$.

The *expected recourse function* is denoted by $\mathcal{Q}(\mathcal{V}) \equiv \mathbb{E}_{\xi}[\mathcal{Q}(\mathcal{V}, \xi)]$. The *recourse function* $\mathcal{Q}(\mathcal{V}, \xi(w))$, for a given realization w of ξ, is represented by:

$$\mathcal{Q}(\mathcal{V}, \xi(w)) = \min \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij} y_{dij}^{(w)} + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} (c_{dij}^{+} s_{dij}^{+(w)} + c_{dij}^{-} s_{dij}^{-(w)}) \tag{6.6}$$

$$y_{dij}^{(w)} - s_{dij}^{+(w)} + s_{dij}^{-(w)} = b_{dij}^{(w)}, \quad \forall d \in D, i \in I_d,\ j \in J, \tag{6.7}$$

$$\sum_{A_{dil}^{\Pi,k} \in ch(O_{dil}^{\pi})} v_{dil}^{\Pi,k,(w)} = \sum_{A_{dil}^{\Pi,k} \in par(O_{dil}^{\pi})} v_{d1n}^{\Pi,k},$$

$$\forall d \in D, O_{dil}^{\pi} \in ch(A_{d1n}^{\pi,k}) \setminus L_d, \tag{6.8}$$

$$\sum_{A_{dil}^{\Pi,t} \in ch(O_{dil}^{\pi})} v_{dil}^{\Pi,k,(w)} = \sum_{A_{dil}^{\Pi,k} \in par(O_{dil}^{\pi})} v_{dil}^{\Pi,k,(w)},$$

$$\forall d \in D, O_{dil}^{\pi} \in O_d \setminus \{O_{d1n}^{S} \cup L_d \cup ch(A_{d1n}^{\pi,k})\}, \tag{6.9}$$

$$y_{dij}^{(w)} = \sum_{A_{di1}^{\Pi,k} \in par(O_{di1}^{j})} v_{di1}^{\Pi,1,(w)}, \quad \forall d \in D, i \in I_d,\ j \in J, \tag{6.10}$$

$$v_{dil}^{\Pi,k,(w)} \geq 0, \quad \forall\, d \in D, A_{dil}^{\Pi,k} \in A_d \setminus ch(O_{d1n}^{S}), \tag{6.11}$$

$$s_{dij}^{+(w)}, s_{dij}^{-(w)} \geq 0, \quad \forall d \in D, i \in I_d,\ j \in J, \tag{6.12}$$

$$y_{dij}^{(w)} \geq 0 \text{ and integer}, \quad \forall d \in D, i \in I_d,\ j \in J. \tag{6.13}$$

Constraints (5.2)

Problem (6.6)-(6.13) is based on the implicit model presented in Côté et al. [30]. The objective, (6.6), is to assign work activities to daily shifts in order to minimize the staffing cost plus the undercovering and overcovering of demand. Constraints (6.7) ensure that the total number of employees working on day $d \in D$, time period $i \in I_d$ and work activity $j \in J$ is equal to the demand realization w subject to some adjustments related to undercovering and undercovering. Due to the structure of Γ_d, $d \in D$, constraints (6.8)-(6.9) guarantee for every or-node O^{π}_{dil}, excluding the root node O^{S}_{d1n} and the leaves L_d, that the summation of the value of its children, $ch(O^{\pi}_{dil})$, is the same as the summation of the value of its parents, $par(O^{\pi}_{dil})$. Constraints (6.10) set the value of variables $y^{(w)}_{dij}$ equal to the summation of the value of the parents of leaf nodes O^{j}_{di1}. Constraints (6.11)-(6.13) set the non-negativity of variables $v^{\Pi,k,(w)}_{dil}$, $s^{+(w)}_{dij}$, $s^{-(w)}_{dij}$ and the non-negativity and integrality of variables $y^{(w)}_{dij}$.

Observe that problem (6.1)-(6.5) (*first-stage problem*) has *complete recourse* because for any realization of the random vector ξ and value of variables $v^{\Pi,k}_{d1n}$, problem (6.6)-(6.13) (*second-stage problem*) is always feasible due to the allowance of undercovering and overcovering of demand. Additionally, note that since we assumed that at the moment we can act on second-stage variables the scenario for day $d \in D$ is fully known and since problem $\mathcal{Q}(\mathcal{V}, \xi(w))$ is decomposable by days due to its particular block structure, $\mathcal{V}$, Ω and $p^{(w)} > 0$ can also be decomposed by days: $\mathcal{V}_d$, Ω_d, $p^{(w)}_d > 0$, $d \in D$ and the expected recourse function $\mathcal{Q}(\mathcal{V})$ can be represented as:

$$\mathcal{Q}(\mathcal{V}) \equiv \mathbb{E}_{\xi}[\sum_{d \in D} \mathcal{Q}(\mathcal{V}_d, \xi_d)] \equiv \sum_{d \in D} \mathbb{E}_{\xi}[\mathcal{Q}(\mathcal{V}_d, \xi_d)] \tag{6.14}$$

In the following, we present the solution method proposed to solve the SMATSP.

6.4 Heuristic Multi-cut L-shaped Method

The basic idea behind the L-shaped method is to approximate the nonlinear term, $\mathcal{Q}(\mathcal{V})$, in the objective function of the two-stage stochastic problem (6.1)-(6.5). In particular, since the expected recourse function involves solving all second-stage recourse problems, the main principle of the L-shaped method is to avoid numerous function evaluations by using an outer linearization of $\mathcal{Q}(\mathcal{V})$, as in BD. Since ξ_d follows a non-negative distribution with a finite support, with Ω_d as the set of its possible realizations for each day $d \in D$, and $p^{(w)}_d > 0$ as the probability of occurrence of scenario $w \in \Omega_d$ ($\sum_{d \in D} \sum_{w \in \Omega_d} p^{(w)}_d = 1$), $\mathcal{Q}(\mathcal{V})$ can be

expressed as $\mathcal{Q}(\mathcal{V}) = \sum_{d\in D}\sum_{w\in\Omega_d} p_d^{(w)}\mathcal{Q}(\mathcal{V}_d, \xi_d(w))$. By defining $\theta_d^{(w)}$ as an additional set of free variables, the two-stage stochastic problem $G_{\mathcal{T}}$ can be reformulated as the following model, denoted as $B_{\mathcal{T}}$.

$$f(B_{\mathcal{T}}) = \min \sum_{d\,\in\,D}\sum_{w\,\in\,\Omega_d} \theta_d^{(w)} \tag{6.15}$$

$$v_{d1n}^{\Pi,k} = \sum_{t\,\in\,\mathcal{T}} \delta_{dt}^{\Pi,k} x_t, \quad \forall\, d\,\in\,D,\; A_{d1n}^{\Pi,k}\,\in\,ch(O_{d1n}^{S}), \tag{6.16}$$

$$\theta_d^{(w)} \geq p_d^{(w)}\mathcal{Q}(\mathcal{V}_d, \xi_d(w)), \quad \forall\, d\,\in\,D,\; w\,\in\,\Omega_d, \tag{6.17}$$

$$\sum_{t\,\in\,\mathcal{T}} x_t = |E|, \tag{6.18}$$

$$x_t \geq 0 \text{ and integer}, \quad \forall\, t\,\in\,\mathcal{T}, \tag{6.19}$$

$$v_{d1n}^{\Pi,k} \geq 0, \quad \forall\, d\,\in\,D,\; A_{d1n}^{\Pi,k}\,\in\,ch(O_{d1n}^{S}). \tag{6.20}$$

Optimality cuts (6.17) ensure that the value of each variable $\theta_d^{(w)}$ is larger than or equal to the optimal value of its corresponding second-stage problem for each day $d \in D$ and each scenario $w \in \Omega_d$. Observe that the structure of problem (6.15)-(6.20) allows the L-shaped method to be extended to include multiple cuts at each iteration, i.e., one per day and per scenario, instead of adding one aggregated cut. Birge and Louveaux [15] showed that in an iterative algorithm, adding multiple cuts at the same iteration may speed up convergence and reduce the number of iterations.

Since the second-stage problems (6.6)-(6.13) are MILP models that do not possess the integrality property, we relax integrality constraints (6.13) on variables $y_{dij}^{(w)}$ because 1) the dual to the LP relaxation of each second-stage problem will produce a cut that forces $\theta_d^{(w)}$ to be at least as great as the objective value of the relaxation, which is a valid lower bound for the actual recourse function value; 2) Restrepo et al. [75] showed that, in practice, problems (6.6)-(6.13) do not exhibit a large integrality gap and that optimal or near-optimal solutions can be found by solving the LP relaxation of the second-stage problems.

Let $\overline{\mathcal{Q}}(\mathcal{V}_d, \xi_d(w))$ denote the LP relaxation of problem $\mathcal{Q}(\mathcal{V}_d, \xi_d(w))$. Let $\rho_{dij}^{(w)}$, $\gamma_{dil}^{\pi,(w)}$ be the dual variables associated with constraints (6.7) and (6.8) from $\overline{\mathcal{Q}}(\mathcal{V}_d, \xi_d(w))$, respectively. Let $\mathbf{\Delta}_d^{(w)}$ be the projection over the space of variables $\rho_{dij}^{(w)}$, $\gamma_{dil}^{\pi,(w)}$ of the polyhedron defined by the constraints associated with the dual of model $\overline{\mathcal{Q}}(\mathcal{V}_d, \xi_d(w))$. Note that $\mathbf{\Delta}_d^{(w)}$ is itself a polyhedron [84]. Let $E_{\mathbf{\Delta}_d^{(w)}}$ be the set of extreme points of $\mathbf{\Delta}_d^{(w)}$. Inequalities (6.17) in model $B_{\mathcal{T}}$ are replaced by the following ones, defining formulation $B'_{\mathcal{T}}$:

$$\theta_d^{(w)} \geq p_d^{(w)} \left(\sum_{i \in I_d} \sum_{j \in J} b_{dij}^{(w)} \rho_{dij}^{(w)} + \sum_{O_{dil}^{\pi} \in ch(A_{d1n}^{\pi,k}) \setminus L_d} \gamma_{dil}^{\pi,(w)} \sum_{A_{d1n}^{\Pi,k} \in par(O_{dil}^{\pi})} v_{d1n}^{\Pi,k} \right),$$
$$\forall\, d \in D,\ w \in \Omega_d,\ (\rho_d, \gamma_d) \in E_{\mathbf{\Delta}_d^{(w)}}. \tag{6.21}$$

Since these new optimality cuts are using linear approximations of $\mathcal{Q}(\mathcal{V}_d, \xi_d(w))$, model $B_{\mathcal{T}}^{'}$ is a MILP relaxation of $B_{\mathcal{T}}$ i.e., $f(B_{\mathcal{T}}) \geq f(B_{\mathcal{T}}^{'})$. Optimality cuts (6.21) do not need to be exhaustively generated, since only a subset of them are active in the optimal solution of the problem. Hence, an iterative algorithm can be used to generate only the subset of cuts that will represent the optimal solution.

The algorithm consists in a multi-cut version of the L-shaped method where, at each iteration $l \geq 1$, a relaxation of the first-stage problem is solved. Such relaxation is obtained by replacing the set of extreme points at each day $d \in D$ and each scenario $w \in \Omega_d$, by subsets $E^l_{\mathbf{\Delta}_d^{(w)}} \subseteq E_{\mathbf{\Delta}_d^{(w)}}$. Note that in the first-stage model, $B_{\mathcal{T}}^{'}$, it is assumed that the complete set of tours $\mathcal{T}$ is known. However, with the incorporation of shift and tour flexibility, the complete enumeration of the set of feasible tours might be intractable. To address this issue, we propose a heuristic CG approach in which a master problem $B^{LP}_{\widetilde{\mathcal{T}}}$, is defined as the LP relaxation of $B_{\mathcal{T}}^{'}$ over a restricted set of tours $\widetilde{\mathcal{T}} \subseteq \mathcal{T}$. We also define the MILP associated with $B^{LP}_{\widetilde{\mathcal{T}}}$ as $B^{MILP}_{\widetilde{\mathcal{T}}}$, where for a given subset of columns $\widetilde{\mathcal{T}} \subseteq \mathcal{T}$, the integrality constraints on x_t variables are imposed to obtain a heuristic integer solution (i.e., $B^{MILP}_{\widetilde{\mathcal{T}}}$ is solved by a state-of-the-art B&B method, using only the columns corresponding to $\widetilde{\mathcal{T}}$). The algorithm for the L-shaped method is then divided into two parts: multi-cut generation and CG.

Two algorithm enhancements were implemented to speed-up the convergence of the multi-cut L-shaped method. First, we adopted the strategy proposed in McDaniel and Devine [58], which consists in initially solving the LP relaxation of the first-stage problem to generate, in a fast way, a number of valid cuts. Then, when some criterion is met (i.e., the relative gap between the upper bound and the lower bound of the problem is smaller than a certain value), the method then solves the MILP of the first-stage problem. Second, we implemented the method presented in Papadakos [64] for the generation of strong optimality cuts. The author proposes an alternative to eliminate the necessity of solving the extra auxiliary subproblem introduced in Magnanti and Wong [57]. Additionally, since finding a *core point* for the problem is a difficult task, the author suggests to use an approximation of the core point that consists of a convex linear combination of the previously generated core point and the current solution for the first-stage problem.

Let ϵ_1 be the tolerance that defines if an optimality cut is added or not to the first-stage problem and let ϵ_2 be the tolerance we used to stop solving $B^{LP}_{\widetilde{\mathcal{T}}}$, i.e., stop the McDaniel and Devine [58] strategy. Let Int be a boolean variable that indicates whether the MILP ($B^{MILP}_{\widetilde{\mathcal{T}}}$) of the first-stage problem is solved (Int=true) or not (Int=false). Let $\theta^{(w)^*}_{dl}$ denote the optimal value of variables $\theta^{(w)}_d$ from $B^{LP}_{\widetilde{\mathcal{T}}}$, at iteration l. Note that $\underline{\theta}_l = \sum_{d\in D}\sum_{w\in\Omega_d}\theta^{(w)^*}_{dl}$ denotes a lower bound on $f(G_{\mathcal{T}})$ at iteration l. Since we are using a heuristic approach to find integer solutions for the first-stage model, we also denote $\overline{\theta}^{(w)^*}_{dl}$ as the optimal values of variables $\theta^{(w)}_d$ when $B^{MILP}_{\widetilde{\mathcal{T}}}$ is solved at iteration l. To simplify the algorithm description, we use the same notation when solving $B^{LP}_{\widetilde{\mathcal{T}}}$, even though in that case, we have $\theta^{(w)^*}_{dl} = \overline{\theta}^{(w)^*}_{dl}$ for each $d \in D, w \in \Omega_d$, since the CG algorithm is performed until all columns have non-negative reduced costs. Let $s^{(w)^*}_{dl}, \overline{s_{dl}}^{(w)^*}$ be the optimal value of second-stage problem (6.6)-(6.13) for day d, under scenario w at iteration l, when integrality constraints on variables $y^{(w)}_{dij}$ are imposed and relaxed, respectively. Note that $\bar{\theta}_l = \sum_{d\in D}\sum_{w\in\Omega_d} p^{(w)}_d s^{(w)^*}_{dl}$ is an upper bound on $f(G_{\mathcal{T}})$ at iteration l. Similarly, $\tilde{\theta}_l = \sum_{d\in D}\sum_{w\in\Omega_d} p^{(w)}_d \overline{s_{dl}}^{(w)^*}$ is an upper bound on $f(B'_{\mathcal{T}}) \leq f(G_{\mathcal{T}})$ at iteration l, which we call the approximated upper bound. Let ${v^{\pi,k}_{d1nl}}^*$, ${v^{\pi,k}_{d1nl}}^0$ denote an optimal solution and the core point approximation, respectively of first-stage variables $v^{\pi,k}_{d1n}$ from model $B'_{\mathcal{T}}$ at iteration l. The flow diagram of the algorithm is presented in Figure 6.3. The description of the multi-cut L-shaped method follows.

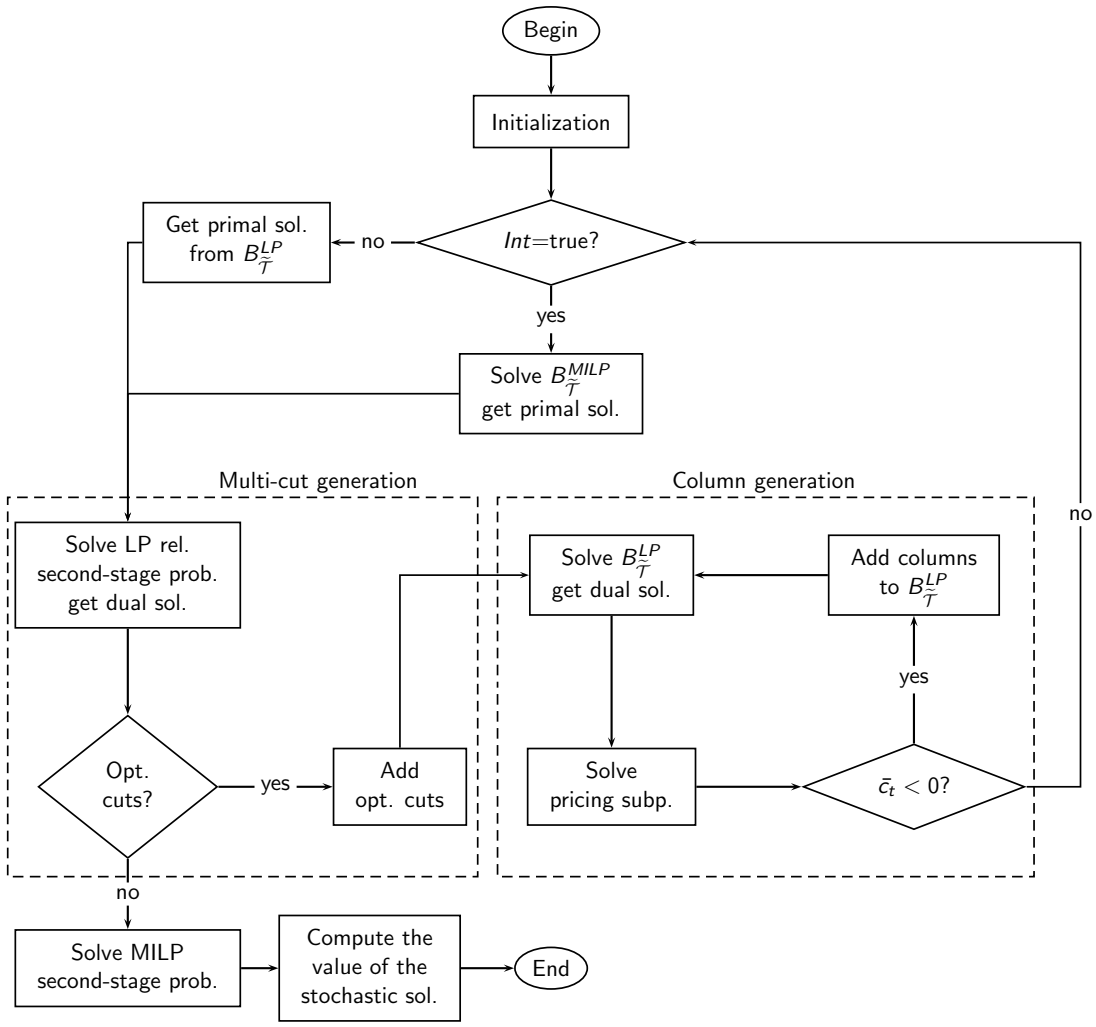


Figure 6.3 Flow chart for the multi-cut L-shaped method.

- *Initialization*: The multi-cut L-shaped algorithm starts with an empty set of optimality cuts, $E^l_{\mathbf{\Delta}^{(w)}_d} = \varnothing$, $d \in D, w \in \Omega_d$. In this step, the number of iterations l is set to zero, the lower and upper bounds of the problem are initialized as $\bar{\theta}_l = \infty, \tilde{\theta}_l = \infty, \underline{\theta}_l = -\infty$, the Boolean variable Int is set to false, and an initial set of columns, generated with the procedure shown in the *Column generation* step, is added to $B^{LP}_{\widetilde{\mathcal{T}}}$.

- *Int=true?*: In this step of the algorithm, we verify if the Boolean variable Int is true or false. If $Int = false$, we continue with the step *Get primal sol. from* $B^{LP}_{\widetilde{\mathcal{T}}}$. If $Int = true$ we continue with the step *Solve* $B^{MILP}_{\widetilde{\mathcal{T}}}$ *get primal sol.* The value of Int is changed from false to true when $(\tilde{\theta}_l - \underline{\theta}_l)/\tilde{\theta}_l < \epsilon_2$ and $Int = false$.

- *Get primal sol. from* $B^{LP}_{\widetilde{\mathcal{T}}}$: In this step of the algorithm, we get the primal solution

$v_{d1nl}^{\pi,k\ *}$, $\theta_{dl}^{(w)^*}$ from $B_{\widetilde{\mathcal{T}}}^{LP}$. Then, we calculate the approximation of the core point as: $v_{d1nl}^{\pi,k\ 0} = \frac{1}{2}v_{d1n\,l-1}^{\pi,k\ \ 0} + \frac{1}{2}v_{d1nl}^{\pi,k\ *}$, $\forall\, d \in D$, $A_{d1n}^{\Pi,k} \in ch(O_{d1n}^S)$ and we update the lower bound of the problem as: $\underline{\theta}_l = \sum_{d\in D}\sum_{w\in\Omega_d}\theta_{dl}^{(w)^*}$. The values of $v_{d1nl}^{\pi,k\ *}$, $v_{d1nl}^{\pi,k\ 0}$ are sent to the second-stage problems.

- *Solve $B_{\widetilde{\mathcal{T}}}^{MILP}$ get primal sol.*: In this step of the algorithm, we get the primal solution $\theta_{dl}^{(w)^*}$ from $B_{\widetilde{\mathcal{T}}}^{LP}$ to update the lower bound of the problem as: $\underline{\theta}_l = \sum_{d\in D}\sum_{w\in\Omega_d}\theta_{dl}^{(w)^*}$. Then we solve $B_{\widetilde{\mathcal{T}}}^{MILP}$ to get the primal solution $\overline{\theta_{dl}}^{(w)^*}, v_{d1nl}^{\pi,k\ *}$ and to calculate the approximation of the core point as: $v_{d1nl}^{\pi,k\ 0} = \frac{1}{2}v_{d1n\,l-1}^{\pi,k\ \ 0} + \frac{1}{2}v_{d1nl}^{\pi,k\ *}$, $\forall\, d \in D$, $A_{d1n}^{\Pi,k} \in ch(O_{d1n}^S)$. The values of $v_{d1nl}^{\pi,k\ *}$, $v_{d1nl}^{\pi,k\ 0}$ are sent to the second-stage problems.

- *Multi-cut generation*: The objective of the multi-cut step is to generate optimality cuts (6.21) in order to approximate the recourse function $\mathcal{Q}(\mathcal{V},\xi)$. The procedure to generate and to add new optimality cuts to the first-stage problem follows.

  - *Solve LP rel. second-stage prob.*: In this step of the algorithm, we solve the LP relaxation of the second-stage problems twice. First, we fix variables $v_{d1n}^{\pi,k}$ with the value of core point $v_{d1nl}^{\pi,k\ 0}$ to get a dual solution (ρ_d, γ_d). Second, we fix variables $v_{d1n}^{\pi,k}$ with the value of point $v_{d1nl}^{\pi,k\ *}$ to recover the real objective value of the second-stage problems and to update the approximated upper bound of the problem as: $\tilde{\theta}_l = \min\{\tilde{\theta}_l, \sum_{d\in D}\sum_{w\in\Omega_d} p_d^{(w)}\overline{s_{dl}}^{(w)^*}\}$.
  - *Opt. cuts?* and *add opt. cuts*: In order to add optimality cuts to the first-stage problem, we verify if $(\overline{s_{dl}}^{(w)^*} - \overline{\theta_{dl}}^{(w)^*})/\overline{s_{dl}}^{(w)^*} > \epsilon_1$, in which case a new optimality cut is added for scenario w and day d. After adding the optimality cuts, we increase by one the number of iterations l.

- *Column generation*: The CG method consists of a master problem $B_{\widetilde{\mathcal{T}}}^{LP}$ and a pricing subproblem. The former problem, as mentioned before, is the LP relaxation of model $B'_{\mathcal{T}}$ over a reduced set of tours $\widetilde{\mathcal{T}} \subseteq \mathcal{T}$. The latter problem is responsible for finding tours with negative reduced cost that will be added to $B_{\widetilde{\mathcal{T}}}^{LP}$ in an iterative way.
  Let $\lambda_{d1n}^{\Pi,k}$ and δ be the dual variables associated with the constraints (6.16) and (6.18) from $B_{\widetilde{\mathcal{T}}}^{LP}$, respectively. Let $\mathcal{S} = \bigcup_{d\in D} ch(O_{d1n}^S)$ be a set of shift shells, defined as the union, over the set of days in the planning horizon, of all the children of root nodes $O_{d1n}^S, d \in D$. Let $G(\mathcal{N},\mathcal{A})$ be a directed acyclic graph, composed of a set of nodes $\mathcal{N} = \{v_s \mid s \in \mathcal{S} \cup \{v_b, v_e\}\}$, where v_s corresponds to shift s and v_b, v_e are the source and sink nodes, respectively. Each shift $s \in \mathcal{S}$ holds, besides a set of attributes inherited from its corresponding and-node (start time period, working time, and length including

breaks), a "reduced cost contribution" corresponding to value of the dual variable $\lambda_{d1n}^{\Pi,k}$. The set of arcs $\mathcal{A}$ represents the connection between nodes depending on the work rules for the allocation of days-off and rest time between consecutive shifts.
New columns for $B_{\widetilde{\mathcal{T}}}^{LP}$ correspond to resource-constrained shortest paths over $G(\mathcal{N}, \mathcal{A})$. More specifically, each feasible tour $t \in \widetilde{\mathcal{T}}$ must meet the work rules related to the minimum and maximum number of working days in a tour, to the minimum and maximum tour length in time periods, to the maximum number of days-off, and to the minimum rest time between two consecutive daily shifts. Additionally, the reduced cost $\bar{c}_t$ of tour t is given by

$$\bar{c}_t = \left(\sum_{d \in D} \sum_{A_{d1n}^{\Pi,k} \in ch(O_{d1n}^{S})} \lambda_{d1n}^{\Pi,k} \delta_{dt}^{\Pi,k} \right) - \sigma. \tag{6.22}$$

The procedure to generate and add new columns for $B_{\widetilde{\mathcal{T}}}^{LP}$ is as follows:

- *Solve $B_{\widetilde{\mathcal{T}}}^{LP}$, get dual sol.*: In this step of the algorithm, we solve problem $B_{\widetilde{\mathcal{T}}}^{LP}$ to get the dual solution (λ, σ) that will be sent to the pricing subproblem.
- *Solve pricing subp.*: New variables (tours) for the $B_{\widetilde{\mathcal{T}}}^{LP}$ are generated by using a label setting algorithm for the resource-constrained shortest-path problem over graph $G(\mathcal{N}, \mathcal{A})$. In the algorithm, the total length of the tour and the number of working days represent global resources that are consumed by the labels while they are extended.
- $\bar{c}_t < 0$? and *add columns to $B_{\widetilde{\mathcal{T}}}^{LP}$*: In this step of the algorithm we evaluate if negative reduced cost columns were found by the pricing subproblem. If yes, such columns are sent to $B_{\widetilde{\mathcal{T}}}^{LP}$ which is re-optimized to start a new iteration.

- *Solve MILP second-stage prob.*: In this step of the algorithm, we solve the MILP of the second-stage problems when $v_{d1n}^{\pi,k}$ variables are fixed with the value of ${v_{d1nl}^{\pi,k}}^{*}$. In this step, we also compute the value of the upper bound as $\bar{\theta}_l = \sum_{w \in \Omega} \sum_{d \in D} p_d^{(w)} s_{dl}^{(w)*}$ and the gap with respect to the approximated upper bound: $100 \times (\bar{\theta}_l - \tilde{\theta}_l)/\bar{\theta}_l$. This gap helps to measure the quality of the solution obtained when the integrality constraints on second-stage variables $y_{dij}^{(w)}$ are relaxed.

- *Compute the value of the stochastic sol.*: The multi-cut L-shaped algorithm ends with the computation of the *value of the stochastic solution* (VSS) which is defined as $VSS = EEV - HN$. HN corresponds to the value of the two-stage stochastic programming

problem and EEV corresponds to the expected value of the *expected-value problem* (EV). Recall that, since second-stage problems are MILP models that do not possess the integrality property, the final (heuristic) solution of the two-stage stochastic problem might not be optimal and $HN = \bar{\theta}_l \geq \tilde{\theta}_l$.

6.5 Computational Experiments

In this section, we test the proposed multi-cut L-shaped method on real and randomly generated instances of the SMATSP. First, we describe the generation and the characteristics of the set of instances used. Second, we present the problem definition and the grammar built for the composition of daily shifts. Third, we report and analyze the computational results.

The computational experiments were performed on a 64-bit GNU/Linux operating system, 96 GB of RAM and 1 processor Intel Xeon X5675 running at 3.07GHz. The multi-cut L-shaped algorithm was implemented in C++. The LP relaxation of both the first-stage problem and the second-stage problems was solved by using the barrier method of CPLEX version 12.5.0.1. We set a time limit of 3 hours to solve each instance. Additionally, a relative gap tolerance of 0.01 was set as a stopping criterion for solving the MILPs with CPLEX. The value of tolerances ϵ_1, ϵ_2 were set to 0.0001 and 0.01, respectively.

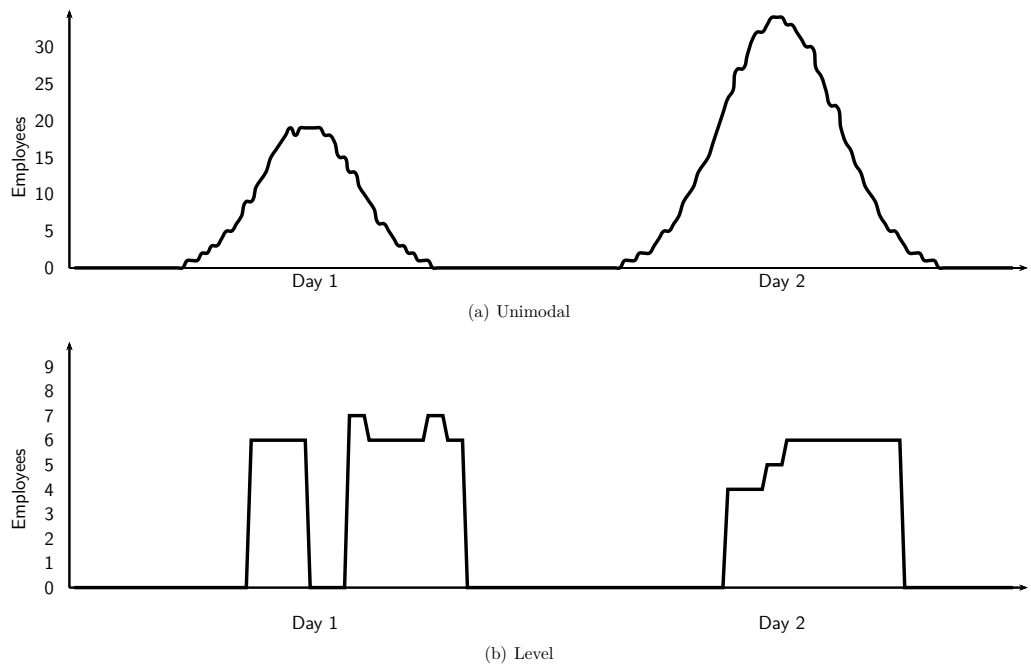
6.5.1 Instances Generation

The set of instances used to test our method is divided into two groups: randomly generated instances and real instances from a small retail shop. The deterministic demand profiles for the set of random instances were generated such that they follow a unimodal behavior. The deterministic demand profiles for the real instances are presented in Côté et al. [30]. These demand profiles present a constant (uniform) demand across hours (level behavior). Figure 6.4 shows an illustration, over two days, of the demand profiles used for the computational experiments.

Stochastic instances were created by adding to the deterministic demand profile, a random perturbation that follows a discrete uniform distribution. We created instances with 11, 49, 81 and 125 scenarios. Their description follows.

- Instances with 11 scenarios: In this group of instances, one large perturbation is generated for the complete week. Such perturbation follows a discrete uniform distribution between -5 and 5.

- Instances with 49 scenarios: In this group of instances, two perturbations that follow



(a) Unimodal

(b) Level

Figure 6.4 Deterministic demand profiles.

a discrete uniform distribution between -3 and 3 are generated. The first perturbation will affect the first 12 hours of each day during the week, while the second perturbation will affect the last 12 hours of each day during the week.

- Instances with 81 scenarios: In this group of instances, we generate perturbations for each day of the week every two hours between 10am and 6pm. Such perturbations follow a discrete uniform distribution between -1 and 1.
- Instances with 125 scenarios: In this group of instances, we generate perturbations for each day of the week every two hours between 11am and 5pm. The perturbations follow a discrete uniform distribution between -2 and 2.

It is important to highlight that the demand is not perturbed when $b_{dij}^{(w)} = b_{dij} + \xi_{dij}^{(w)} \leq 0$. However, this case only applies when $b_{dij} = 0$ because when the demand takes a positive value, this value is always higher than the value of the lower realization of the stochastic perturbation $\xi_{dij}^{(w)}$.

6.5.2 Problem Definition and Grammar

The work rules for shift and tour generation, as well as the grammar used in the problem are as follows.

Tour generation

1. The planning horizon is seven days, where each day is divided into 96 time periods of 15 minutes.
2. Shifts are not allowed to span from one day to another (discontinuous problem).
3. The tour working length should fall between 35 and 40 hours per week.
4. The number of working days in the tour should fall between five and six.
5. There must be a minimum rest time of twelve hours between consecutive shifts.

Daily shift generation

1. Shifts can start at any time period during any day d, allowing enough time to complete their duration in day d.
2. Three types of shifts are considered: 8-hour shifts with 1-hour lunch break in the middle and two 15-minute breaks. 6-hour shifts with one 15-minute break and no lunch, and 4-hour shifts with one 15-minute break and no lunch.
3. If performed, the duration of a work activity is at least one hour and at most five hours.
4. A break (or lunch) is necessary between two different work activities.
5. Work activities must be inserted between breaks, lunch and rest stretches.
6. A fixed number of employees $|E|$ is given, therefore undercovering and overcovering of demand is allowed.

Let a_j be a terminal symbol that defines a time period of work activity $j \in J$. Let b, l and r be the terminal symbols that represent break, lunch and rest periods, respectively. In productions $\Pi \in P$, $\Pi \rightarrow_{[\text{min, max}]}$ restricts the subsequences generated by a given production to a length between a minimum and maximum number of time periods. The grammar and the productions that define the multi-activity shifts are as follows:

$$
\begin{aligned}
G =&(\Sigma = (a_j \ \forall j \in J, b, l, r),\\
&N = (S, F, Q, N, W, A_j \ \forall j \in J, B, L, R), P, S),\\
&S \rightarrow RFR|FR|RF|RQR|QR|RQ|RNR|NR|RN,\ B \rightarrow b,\ L \rightarrow llll,\\
&F \rightarrow_{[38,38]} NLN,\ Q \rightarrow_{[25,25]} WBW,\\
&N \rightarrow_{[17,17]} WBW,\ R \rightarrow Rr|r,\\
&W \rightarrow_{[4,20]} A_j \ \forall j \in J,\ A_j \rightarrow A_j a_j | a_j \ \forall j \in J.
\end{aligned}
$$

6.5.3 Computational Results

Tables 6.1 - 6.2 present the computational results on stochastic weekly instances dealing with up to five work activities. Ten different demands were tested for each activity (*Nb.Act*) and for each version on the number of scenarios (*Scen.*). We present the average CPU time in seconds to solve the problem (*T. time*), the average CPU time spent in the CG approach (*Time CG*), which includes the time to solve the pricing subproblems and the time to solve the LP relaxation when new columns are added, the average CPU time to solve the first-stage problem (*Time F-S*), and the average CPU time to solve the second-stage problems (*Time S-S*). The average gap between the best upper bound and best lower bound is presented in *Gap1*. This gap is computed as: *Gap1*$= 100 \times (\bar{\theta} - \underline{\theta})/\bar{\theta}$. Since the second-stage problems are MILPs that do not possess the integrality property and we are relaxing integrality constraints on variables $y_{dij}^{(w)}$, we also calculate the average gap between the upper bound $\bar{\theta}$ and the approximated upper bound $\tilde{\theta}$: *Gap2*$= 100 \times (\bar{\theta} - \tilde{\theta})/\bar{\theta}$. *Conv.* presents the number of instances that converged to a near-optimal solution, i.e., the algorithm stopped when no more optimality cuts are added to the first-stage model. The average value of the stochastic solution (*VSS*), in percentage, is presented in the last column. This value is computed as: $VSS = 100 \times (EEV - HN)/EEV$ and it is only calculated for instances that converged (*Conv.*=1).

Table 6.1 Results on stochastic weekly instances with unimodal demand shape.

Scen.	*Nb. Act*	*T. time*	*Time CG*	*Time F-S*	*Time S-S*	*Gap1*	*Gap2*	*Conv.*	*VSS*
	1	242.75	23.04	11.14	185.99	1.01%	0.00%	10	5.40%
	2	1,587.45	97.11	188.8	1,123.27	0.91%	0.00%	10	9.05%
11	3	2,114.18	94.6	200.1	1,655.17	0.85%	0.00%	10	8.63%
	4	2,602.97	101.37	133.23	2,227.37	0.87%	0.00%	10	8.97%
	5	3,502.28	117.21	134.1	3,092.78	1.09%	0.02%	10	7.69%
	1	990.11	47.96	41.04	872.56	0.66%	0.00%	10	2.70%
	2	4,344.01	126.26	144.8	4,001.32	0.88%	0.00%	10	5.48%
49	3	8,069.22	568.29	544.45	6,390.64	0.95%	0.00%	7	8.01%
	4	8,463.06	385.56	386.02	7,226.97	1.14%	0.01%	6	8.08%
	5	10,664.8	647.37	488.32	8,632.84	11.85%	0.02%	0	-
	1	2,171.04	221.63	163.12	1,671.75	0.76%	0.00%	10	0.29%
81	2	8,577.45	669.39	661.43	7,023.38	0.86%	0.00%	8	0.92%
	3	9,603.11	292.43	285.7	8,909.98	0.82%	0.01%	7	1.54%
	4	10,537.88	267.65	250.83	9,927.05	1.34%	0.01%	2	1.61%
	1	2,189.07	212.81	237.51	1,630.95	0.89%	0.00%	10	1.51%
125	2	9,415.76	635.78	726.83	7,826.62	0.84%	0.00%	10	2.07%
	3	10,321.03	371.64	419.09	9,409.37	0.74%	0.01%	6	2.78%
	4	10,619.85	192.11	197.47	10,144.59	3.23%	0.01%	2	3.17%

Table 6.2 Results on stochastic weekly instances with level demand shape.

Scen.	*Nb. Act*	*T. time*	*Time CG*	*Time F-S*	*Time S-S*	*Gap1*	*Gap2*	*Conv.*	*VSS*
	1	152	18.02	5.34	81.44	0.56%	0.00%	10	1.96%
	2	279.16	14.31	7.78	203.67	0.46%	0.02%	10	0.87%
11	3	616.06	27.13	30.33	452.17	0.72%	0.02%	10	0.81%
	4	607.65	20.78	11.56	504.91	0.50%	0.02%	10	1.55%
	5	962.64	31.6	22.07	763.5	0.54%	0.03%	10	1.27%
	1	575.32	59.7	25.18	387.53	0.60%	0.00%	10	0.00%
	2	1,451.61	75.4	42.33	10,59.27	0.56%	0.01%	10	0.43%
49	3	2,838.99	87.35	153.01	2,123.57	0.83%	0.02%	10	0.42%
	4	3,166.09	93.83	78.68	2,625.82	0.54%	0.01%	10	1.11%
	5	4,557.47	94.44	127.76	3,813.78	0.59%	0.02%	10	0.95%
	1	1301.52	224.75	81.07	694.34	0.66%	0.00%	10	-0.75%
	2	2,812.83	269.58	117.27	1,804.07	0.65%	0.01%	10	-0.77%
81	3	6,177.93	298.59	611.18	4,174.32	0.80%	0.03%	9	0.01%
	4	6,261.87	486.49	169.65	4,719.92	0.53%	0.01%	10	-0.03%
	5	9,766.28	263.26	535.85	7,452.06	0.72%	0.01%	9	0.14%
	1	1,842.3	368.16	155.31	877.35	0.69%	0.00%	10	-0.47%
	2	4,322.38	597.87	283.89	2,376.46	0.58%	0.02%	10	-0.06%
125	3	7,510.87	513.18	436.21	4,891.14	0.64%	0.03%	10	0.37%
	4	8,129.87	607.49	324.75	5,912.99	0.51%	0.01%	10	1.53%
	5	10,818.7	371.01	388.89	7,572.3	22.19%	0.01%	0	-

From Table 6.1, we can conclude that the proposed approach was able to find high quality solutions for most of the instances with up to 125 demand scenarios and three work activities.

Our method was not able to find solutions that converged for any of the instances with five activities when evaluated on 49 stochastic scenarios, providing solutions with 11.85% average optimality gap. However, for most of the other instances, the final gap is, on average, less than 1.15%. From Table 6.2, we can conclude that our method was able to find near-optimal solutions for almost all the instances with up to five work activities when evaluated on 11, 49 and 81 demand scenarios and with up to four work activities when evaluated on 125 demand scenarios. Regarding the CPU time, we observe that the most time-consuming component is related to the LP solution of the second-stage problems, which increases significantly with the numbers of activities and scenarios.

Although integrality constraints on second-stage variables were relaxed, we can conclude that in most cases, the approximated upper bound $\tilde{\theta}$ was the same as or very close to the real upper bound $\bar{\theta}$. The above can be observed from the average values of $Gap2$, which are at most 0.03%.

We observe some negative values for the VSS since the algorithm is heuristic. These cases are rare, however: they are observed only on the instances with level demand shape and when the stochastic perturbations of demand do not exhibit a lot of variability (81 and 125 scenarios). The values of the stochastic solution (VSS) depend on the demand profile used. When a unimodal demand profile is tested, the stochastic model prevents the occurrence of additional staffing costs associated with the undercovering and overcovering of demand when compared with the expected value problem. On the contrary, when a level demand profile is used, a deterministic approach based on the expected demand appears to be sufficient, especially when 81 and 125 scenarios are used and not a lot of variability is included in the stochastic perturbation of demand.

6.6 Concluding Remarks

In this paper, we presented a two-stage stochastic programming approach to solve the discontinuous multi-activity tour scheduling problem when demand is uncertain and employees have identical skills. In the model, first-stage decisions correspond to the allocation of employees to weekly tours and to daily shifts, while second-stage decisions correspond to the allocation of work activities and breaks to shifts and to the undercovering and overcovering of demand. Since the number of tours becomes large with an increase in shift and tour flexibility, the first-stage problem was solved via CG. Second-stage problems were modeled with context-free grammars in order to efficiently handle the work rules for the composition of the shifts and the allocation of work activities to the shifts.

A heuristic multi-cut L-shaped method was implemented as a solution approach. Two algorithmic refinements were used to enhance the performance of the method. First, we adopted the strategy of McDaniel and Devine [58] in order to generate an initial set of valid cuts in a fast way. Second, we implemented the idea of Papadakos [64] to generate strong optimality cuts. Additionally, we showed that, although second-stage problems are MILP models that do not possess the integrality property, high-quality solutions can be achieved by relaxing the integrality constraints to generate optimality cuts.

Computational results suggest that the performance of the method depends on the distribution of the demand profile, as well as on the number of scenarios and work activities included. Specifically, the multi-cut L-shaped method exhibited a better performance, in terms of CPU time, when evaluated on instances with a level demand behavior than when evaluated on instances with a unimodal behavior. However, we observed that the use of the stochastic model has a larger impact on instances with unimodal demand behavior, since it prevents the occurrence of additional staffing costs when compared with the expected value problem. On the contrary, since the value of the stochastic solution is close to zero, a deterministic approach based on the expected demand often appears to be sufficient for instances with a level demand behavior.

CHAPTER 7 GENERAL DISCUSSION

In the articles presented in Chapters 4, 5, and 6 we introduced different modeling approaches that rely on the combination of decomposition methods with context-free grammars to solve the integration of the discontinuous tour scheduling problem with the multi-activity shift scheduling problem. In Chapters 4 and 5 we presented two exact methods for the personalized and the anonymous version of the problem, respectively. Then, an extension of the method presented in Chapter 5 was used to solve the multi-activity tour scheduling problem under stochastic demand. With these contributions we intended to address three different variants of the problem that, to the best of our knowledge, have not yet been studied in the literature.

In Chapters 4 and 5 we showed that the integration of previous work on multi-activity shift scheduling (the use of context-free grammars to efficiently handle the multi-activity context) with the development of new ideas for the composition of the tours, helps to overcome the limitations found in the literature regarding the composition of shifts over long time horizons (e.g., one week) and the introduction of flexibility and specific work rules for the tour and shift composition.

The theoretical comparison and experimental evaluation of the models and solution approaches presented in the thesis, allowed us to determine the characteristics and limitations of each one of them. First, the comparison of the two formulations presented in Chapter 4 allowed us to conclude that a Tour-based formulation in which columns correspond to tours is more suitable when exact solutions are required, but when fast-heuristic integer solutions are needed, a Daily-based formulation in which columns correspond to daily shifts works better. Second we found that, for the anonymous problem, the model presented in Chapter 5 is a better alternative than the Tour-based branch-and-price approach from Chapter 4, since the combination of an explicit tour scheduling modeling approach with an implicit grammar-based shift scheduling formulation allows to solve the scalability and symmetry issues arising with an increase in the numbers of employees and work activities.

The models and solution methods presented in the thesis were also adapted in order to compare them against some methods presented in the literature. First, since no method has been proposed in the literature to solve the personalized multi-activity tour scheduling problem, the Tour-based formulation from Chapter 4 was tested on a mono-activity problem presented in Brunner and Bard [17]. Second, the method presented in Chapter 5 was adapted to the multi-activity shift scheduling problem presented in Côté et al. [30]. The experimental results allowed us to conclude that our models are flexible enough to be easily adapted

depending on the characteristics of the problem and that they exhibited, in most of the cases, competitive and often better solution times.

CHAPTER 8 CONCLUSION

In the present thesis we have proposed different hybrid approaches that combine decomposition methods and formal languages to address the integration of two complex large-scale problems: the discontinuous tour scheduling and the multi-activity shift scheduling. In the following, we present a summary of the contributions achieved by the thesis in the area of personnel scheduling problems, then we discuss the limits of the approaches proposed, to finish with a presentation of future improvements and possible future work.

8.1 Advancement of Knowledge

To address the integration of the discontinuous tour scheduling problem with the MASSP, we proposed different modeling approaches based on decomposition methods and formal languages. While the use of formal languages (context-free grammars) allows to define the work rules for the composition of shifts and the allocation of work activities and breaks to shifts in an efficient way, the use of decomposition methods (CG and BD) allows to decompose the problem into smaller ones, easier to solve.

In Chapter 4 we introduced two B&P algorithms to solve the personalized MATSP in a discontinuous environment. Two formulations were presented: a Daily-based formulation and a Tour-based formulation in which columns correspond to shifts and tours, respectively. In the pricing subproblems for the Daily-based formulation, columns (shifts) are modeled using context-free grammars. In the Tour-based formulation, columns (tours) are composed with an exact two-phase procedure. In the first phase, daily shifts are modeled with context-free grammars. In the second phase, the daily shifts are assembled into tours by using a label setting algorithm for the resource-constrained shortest-path problem. The master problem of both formulations is based on a generalized set partitioning problem. We formally showed that the Tour-based formulation was stronger in terms of its LP relaxation lower bound when compared with the Daily-based formulation.

We tested and compared the two approaches on randomly generated instances for the mono-activity and multi-activity tour scheduling problem over a one week planning horizon. Two methods were tested to find integer solutions. A heuristic approach in which the MIP version of the problem including only the set of generated columns at the root node was solved by a state-of-the-art B&B code and an exact approach corresponding to a B&P where the branching rule was designed to preserve the structure of the pricing subproblems. The

Daily-based formulation exhibited better solution times than the Tour-based formulation for the heuristic approach. On the other hand, the Tour-based formulation had a better performance in the exact approach, being able to find integer solutions for all the instances with an optimality gap lower than 1%.

The Tour-based formulation was also tested on a mono-activity problem presented in Brunner and Bard [17]. The experiments suggested that the solution times and quality of our formulation are comparable with the solution times and quality reported by Brunner and Bard [17].

Since scalability issues arise as the number of employees becomes large, in Chapter 5, we presented an approach that combines BD and CG to solve the anonymous MATSP in a discontinuous environment. We took advantage of the special block structure of the problem to decompose it into smaller problems, easier to solve. Therefore, the model consists of a master problem and a set of Benders daily subproblems. The master problem links the tours with daily shift shells, while the Benders subproblems assign work activities and breaks to daily shifts. Due to its structure, the master problem is solved by CG. Additionally, the Benders subproblems are modeled with context-free grammars to implicitly tackle all the work rules for the composition of shifts and to efficiently allocate work activities and breaks to them. Although Benders primal subproblems do not possess the integrality property, we showed that adopting an alternative algorithmic strategy that combines the generation of integer Benders cuts with classical Benders cuts, guarantees the convergence of the method, to an optimal solution to the original problem.

The combined BD and CG approach was tested on real-world instances and randomly generated instances for the discontinuous anonymous MATSP (one-week planning horizon) and for the anonymous MASSP (one-day planning horizon). Results on weekly instances suggest that our method is able to find high quality integer solutions for instances dealing with up to ten work activities. When compared with a B&P approach, our method exhibited faster solution times and provided better upper bounds for the most difficult instances. Regarding the MASSP, the BD approach was able to solve to optimality instances with up to thirty work activities. When the method was compared with the grammar-based integer programming approach presented in Côté et al. [30], our method presented competitive and often better solution times.

Because alternative personnel scheduling approaches that include demand uncertainty can lead to significant reductions in labor costs, in Chapter 6 we presented a two-stage stochastic programming approach to solve the anonymous MATSP when demand is uncertain. The

method is an extension of the combined BD and CG approach presented in Chapter 5. Therefore, the first-stage problem is in charge of the decisions corresponding to the allocation of employees to weekly tours and to daily shifts, while the second-stage problems are in charge of the decisions related to the allocation of work activities and breaks to shifts and to the undercovering and overcovering of employee requirements (recourse actions). Since the number of tours becomes large with an increase in shift and tour flexibility, the first-stage problem was solved with a heuristic CG method. On the other hand, the second-stage problems are modeled with the grammar-based integer programming model presented in Côté et al. [30].

A multi-cut L-shaped method was implemented as a solution approach. The method was evaluated on instances with different demand profiles. The results suggest that the performance of the method depends on the distribution of the demand, as well as on the numbers of scenarios and work activities included. Specifically, the multi-cut L-shaped method exhibited a better performance, in terms of CPU time, when evaluated on instances with a level demand behavior than when evaluated on instances with a unimodal behavior. However, we observed that the use of the stochastic model has a larger impact on instances with unimodal demand behavior since it prevents the occurrence of additional staffing costs when compared with the expected value problem. On the contrary, since the value of the stochastic solution is close to zero, a deterministic approach based on the expected demand often appears to be sufficient for instances with a level demand behavior.

8.2 Limits and Constraints

The models and solution approaches proposed in this thesis present two types of limitations: practical and size-related limitations. First, the practical applications of our methods might be reduced specially for companies that operate 24-hours per day, 7-days per week, because the main assumption adopted in all the approaches was to suppose that the problem is discontinuous (shifts are not allowed to span from one day to another). Second, some size-related limitations of our approaches might arise with an increase in the numbers of work activities, days in the planning horizon, stochastic scenarios and flexibility. Specifically, the DAGs used for shift and tour generation in the CG approaches will probably become large and the dynamic programming algorithms might become slow to solve. Similarly, the grammar-based integer programming models used as subproblems in the methods from Chapters 5-6, might become difficult to solve with a state-of-the-art mathematical programming solver, as the number of variables and constraints might also become large.

8.3 Recommendations

In order to solve practical instances for the MATSP arising from continuous environments, it would be interesting to extend the proposed approaches to the case in which the problem is not decomposable by days. Therefore, in the B&P methods it would be a good idea to divide the planning horizon by pairs of days e.g., Monday and Tuesday, Tuesday and Wednesday,..., Sunday and Monday. In that way, the problem would still be decomposable and the B&P approaches proposed in this thesis would still apply to solve the problem. More in detail, if the day is divided into 24 time periods and shifts are allowed to span a maximum number of 8 time periods, the pair of days Monday and Tuesday will include the 24 time periods of Monday and the first 7 time periods of Tuesday. Additionally, at each pair of days, shifts can only start during the first day (i.e. in the pair of days Monday and Tuesday, shifts can only start on Monday). Regarding the approach that combines BD and CG, continuity can be included by assuming that the problem cannot be decomposed by days. Therefore, it would be necessary to define one large DAG (in the Benders subproblem) representing all possible shifts for the complete planning horizon. Note that in both approaches it would be necessary to modify the label setting algorithm for the resource-constrained shortest-path problem to guarantee the cyclical nature of the schedules. Additionally, the size of the DAG derived from the grammars, as well as the graph used to build the tours, would be larger than in the discontinuous case, since it would be necessary to (almost) duplicate the number of time periods and the possible number of shifts considered in the problem.

The B&P approaches presented in Chapter 5 can be integrated with the problem presented in Boyer et al. [16], where in addition to the allocation of work activities to the shifts it would also be necessary to include the allocation of uninterrupted tasks. In this case, the grammars and the DAGs should be modified to include the productions associated with the allocation of tasks to shifts, as well as the master problems to account for the precedence constraints between the tasks.

Since the computational complexity for solving the two-stage stochastic programming model presented in Chapter 6 gets worse with an increase in the number of demand scenarios, it would be interesting to use an algorithm for scenario reduction that will compute the (nearly) best approximation of the discrete probability distribution for employee requirements by a measure with fewer scenarios (smaller support).

REFERENCES

[1] Addou, I. and Soumis, F. (2007). Bechtold-Jacobs generalized model for shift scheduling with extraordinary overlap. *Annals of Operations Research*, 155:177–205.

[2] Al-Yakoob, S. M. and Sherali, H. D. (2008). A column generation approach for an employee scheduling problem with multiple shifts and work locations. *Journal of the Operational Research Society*, 59(1):34–43.

[3] Alfares, H. K. (2003). Four-day workweek scheduling with two or three consecutive days off. *Journal of Mathematical Modelling and Algorithms*, 2:67–80.

[4] Alfares, H. K. (2004). Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127(1-4):145–175.

[5] Aykin, T. (1996). Optimal shift scheduling with multiple break windows. *Management Science*, 42(4):591–602.

[6] Aykin, T. (2000). A comparative evaluation of modeling approaches to the labor shift scheduling problem. *European Journal of Operational Research*, 125(2):381–397.

[7] Bailey, J. (1985). Integrated days off and shift personnel scheduling. *Computers & Industrial Engineering*, 9(4):395–404.

[8] Baker, K. R. (1976). Workforce allocation in cyclical scheduling problems: A survey. *Operational Research Quarterly*, 27:155–167.

[9] Bard, J. F., Binici, C., and deSilva, A. H. (2003). Staff scheduling at the United States postal service. *Computers & Operations Research*, 30(5):745–771.

[10] Bard, J. F., Morton, D. P., and Wang, Y. M. (2007). Workforce planning at USPS mail processing and distribution centers using stochastic optimization. *Annals of Operations Research*, 155(1):51–78.

[11] Barnhart, C., Hane, C. A., and Vance, P. H. (2000). Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326.

[12] Bechtold, S. E. and Jacobs, L. W. (1990). Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science*, 36(11):1339–1351.

[13] Bechtold, S. E. and Jacobs, L. W. (1996). The equivalence of general set-covering and implicit integer programming formulations for shift scheduling. *Naval Research Logistics*, 43:233–249.

[14] Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

[15] Birge, J. R. and Louveaux, F. V. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392.

[16] Boyer, V., Gendron, B., and Rousseau, L.-M. (2014). A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem. *Journal of Scheduling*, 17(2):185–197.

[17] Brunner, J. O. and Bard, J. F. (2013). Flexible weekly tour scheduling for postal service workers using a branch and price. *Journal of Scheduling*, 16(1):129–149.

[18] Brunner, J. O. and Stolletz, R. (2014). Stabilized branch and price with dynamic parameter updating for discontinuous tour scheduling. *Computers & Operations Research*, 44:137–145.

[19] Brusco, M. J. and Jacobs, L. W. (1998). Personnel tour scheduling when starting-time restrictions are present. *Management Science*, 44(4):534–547.

[20] Brusco, M. J. and Jacobs, L. W. (2000). Optimal models for meal-break and start-time flexibility in continuous tour scheduling. *Management Science*, 46(12):1630–1641.

[21] Brusco, M. J. and Jacobs, L. W. (2001). Starting-time decisions in labor tour scheduling: An experimental analysis and case study. *European Journal of Operational Research*, 131(3):459–475.

[22] Brusco, M. J., Jacobs, L. W., Bongiorno, R. J., Lyons, D. V., and Tang, B. (1995). Improving personnel scheduling at airline stations. *Operations Research*, 43(5):741–751.

[23] Brusco, M. J. and Johns, T. R. (1996). A sequential integer programming method for discontinuous labor tour scheduling. *European Journal of Operational Research*, 95(3):537–548.

[24] Brusco, M. J. and Johns, T. R. (2011). An integrated approach to shift-starting time selection and tour-schedule construction. *Journal of the Operational Research Society*, 62(7):1357–1364.

[25] Campbell, G. M. (2011). A two-stage stochastic program for scheduling and allocating cross-trained workers. *Journal of the Operational Research Society*, 62(6):1038–1047.

[26] Çezik, T., Günlük, O., and Luss, H. (2001). An integer programming model for the weekly tour scheduling problem. *Naval Research Logistics*, 48(7):607–624.

[27] Codato, G. and Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

[28] Côté, M.-C., Gendron, B., Quimper, C.-G., and Rousseau, L.-M. (2011a). Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16(1):54–76.

[29] Côté, M.-C., Gendron, B., and Rousseau, L.-M. (2007). Modeling the regular constraint with integer programming. In Van Hentenryck, P. and Wolsey, L., editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 4510 of *Lecture Notes in Computer Science*, pages 29–43. Springer Berlin Heidelberg.

[30] Côté, M.-C., Gendron, B., and Rousseau, L.-M. (2011b). Grammar-based integer programming models for multiactivity shift scheduling. *Management Science*, 57(1):151–163.

[31] Côté, M.-C., Gendron, B., and Rousseau, L.-M. (2013). Grammar-based column generation for personalized multi-activity shift scheduling. *INFORMS Journal on Computing*, 25(3):461–474.

[32] Crainic, T. G., Frangioni, A., and Gendron, B. (2009). OOBB: An object-oriented library for parallel branch-and-bound. In *CORS/INFORMS International Conference, Toronto, Canada*.

[33] Dahmen, S. and Rekik, M. (2015). Solving multi-activity multi-day shift scheduling problems with a hybrid heuristic. *Journal of Scheduling*, 18(2):207–223.

[34] Dantzig, G. B. (1954). A comment on Edie's "traffic delays at toll booths". *Journal of the Operations Research Society of America*, 2(3):339–341.

[35] Defraeye, M. and Van Nieuwenhuyse, I. (2016). Staffing and scheduling under nonstationary demand for service: A literature review. *Omega*, 58:4–25.

[36] Demassey, S., Pesant, G., and Rousseau, L.-M. (2005). Constraint programming based column generation for employee timetabling. *Lecture Notes in Computer Science*, pages 140–154.

[37] Demassey, S., Pesant, G., and Rousseau, L.-M. (2006). A cost-regular based hybrid column generation approach. *Constraints*, 11(4):315–333.

[38] Detienne, B., Péridy, L., Pinson, É., and Rivreau, D. (2009). Cut generation for an employee timetabling problem. *European Journal of Operational Research*, 197(3):1178–1184.

[39] Easton, F. F. and Mansour, N. (1999). A distributed genetic algorithm for deterministic and stochastic labor scheduling problems. *European Journal of Operational Research*, 118(3):505–523.

[40] Easton, F. F. and Rossin, D. F. (1996). A stochastic goal program for employee scheduling. *Decision Sciences*, 27(3):541–568.

[41] Edie, L. C. (1954). Traffic delays at toll booths. *Journal of the Operations Research Society of America*, 2(2):107–138.

[42] Elahipanah, M., Desaulniers, G., and Lacasse-Guay, È. (2013). A two-phase mathematical-programming heuristic for flexible assignment of activities and tasks to work shifts. *Journal of Scheduling*, 16(5):443–460.

[43] Ernst, A., Jiang, H., Krishnamoorthy, M., Owens, B., and Sier, D. (2004a). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127:21–144.

[44] Ernst, A., Jiang, H., Krishnamoorthy, M., and Sier, D. (2004b). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27.

[45] Fischetti, M. and Liberti, L. (2012). Orbital shrinking. In Mahjoub, A., Markakis, V., Milis, I., and Paschos, V., editors, *Combinatorial Optimization*, volume 7422 of *Lecture Notes in Computer Science*, pages 48–58. Springer Berlin Heidelberg.

[46] Geoffrion, A. M. (1972). Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260.

[47] Geoffrion, A. M. (1974). *Lagrangean relaxation for integer programming.* Springer.

[48] Hooker, J. N. and Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming*, 96(1):33–60.

[49] Jacobs, L. W. and Brusco, M. J. (1996). Overlapping start-time bands in implicit tour scheduling. *Management Science*, 42(9):1247–1259.

[50] Jarrah, A. I. Z., Bard, J. F., and deSilva, A. H. (1994). Solving large-scale tour scheduling problems. *Management Science*, 40(9):1124–1144.

[51] Kadioglu, S. and Sellmann, M. (2008). Efficient context-free grammar constraints. In *AAAI*, pages 310–316.

[52] Kadioglu, S. and Sellmann, M. (2010). Grammar constraints. *Constraints*, 15(1):117–144.

[53] Kim, K. and Mehrotra, S. (forthcoming). A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. *Operations Research*.

[54] Laporte, G. and Louveaux, F. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142.

[55] Lequy, Q., Bouchard, M., Desaulniers, G., Soumis, F., and Tachefine, B. (2012). Assigning multiple activities to work shifts. *Journal of Scheduling*, 15(2):239–251.

[56] Loucks, J. S. and Jacobs, F. R. (1991). Tour scheduling and task assignment of a heterogeneous work force: A heuristic approach. *Decision Sciences*, 22:719–738.

[57] Magnanti, T. L. and Wong, R. T. (1981). Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484.

[58] McDaniel, D. and Devine, M. (1977). A modified Benders' partitioning algorithm for mixed integer programming. *Management Science*, 24(3):312–319.

[59] Mehrotra, A., Murphy, K. E., and Trick, M. A. (2000). Optimal shift scheduling: A branch-and-price approach. *Naval Research Logistics*, 47:185–200.

[60] Moondra, S. (1976). A linear programming model for work force scheduling for banks. *Journal of Bank Research*, 6:299–301.

[61] Morris, J. G. and Showalter, M. J. (1983). Simple approaches to shift, days-off and tour scheduling problems. *Management Science*, 29(8):942–950.

[62] Ni, H. and Abeledo, H. (2007). A branch-and-price approach for large-scale employee tour scheduling problems. *Annals of Operations Research*, 155:167–176.

[63] Pacqueau, R. and Soumis, F. (2014). Shift scheduling under stochastic demand. Technical Report 46, GERAD.

[64] Papadakos, N. (2008). Practical enhancements to the Magnanti–Wong method. *Operations Research Letters*, 36(4):444–449.

[65] Parisio, A. and Jones, C. N. (2015). A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand. *Omega*, 53:97–103.

[66] Pesant, G. (2004). A regular language membership constraint for finite sequences of variables. In Wallace, M., editor, *Principles and Practice of Constraint Programming - CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 482–495. Springer Berlin Heidelberg.

[67] Pesant, G., Quimper, C.-G., Rousseau, L.-M., and Sellmann, M. (2009). The polytope of context-free grammar constraints. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 223–232. Springer.

[68] Punnakitikashem, P., Rosenberber, J. M., and Buckley-Behan, D. F. (2013). A stochastic programming approach for integrated nurse staffing and assignment. *IIE Transactions*, 45(10):1059–1076.

[69] Quimper, C.-G. and Rousseau, L.-M. (2010). A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–392.

[70] Quimper, C.-G. and Walsh, T. (2006). Global grammar constraints. In *Principles and Practice of Constraint Programming-CP 2006*, pages 751–755. Springer.

[71] Quimper, C.-G. and Walsh, T. (2007). Decomposing global grammar constraints. In *Principles and Practice of Constraint Programming–CP 2007*, pages 590–604. Springer.

[72] Rekik, M., Cordeau, J.-F., and Soumis, F. (2004). Using Benders decomposition to implicitly model tour scheduling. *Annals of Operations Research*, 128(1-4):111–133.

[73] Rekik, M., Cordeau, J.-F., and Soumis, F. (2008). Solution approaches to large shift scheduling problems. *RAIRO-Operations Research*, 42:229–258.

[74] Rekik, M., Cordeau, J.-F., and Soumis, F. (2010). Implicit shift scheduling with multiple breaks and work stretch duration restrictions. *Journal of Scheduling*, 13(1):49–75.

[75] Restrepo, M. I., Gendron, B., and Rousseau, L.-M. (2015). Combining Benders decomposition and column generation for multiactivity tour scheduling. Technical Report 57, CIRRELT.

[76] Restrepo, M. I., Gendron, B., and Rousseau, L.-M. (forthcoming). Branch-and-price for multi-activity tour scheduling. *INFORMS Journal on Computing*.

[77] Restrepo, M. I., Lozano, L., and Medaglia, A. L. (2012). Constrained network-based column generation for the multi-activity shift scheduling problem. *International Journal of Production Economics*, 140(1):466–472.

[78] Ritzman, L. P., Krajewsky, L. J., and Showalter, M. J. (1976). The disaggregation of aggregate manpower plans. *Management Science*, 22(11):1204–1214.

[79] Robbins, T. R. and Harrison, T. P. (2010). A stochastic programming model for scheduling call centers with global service level agreements. *European Journal of Operational Research*, 207(3):1608–1619.

[80] Sellmann, M. (2006). The theory of grammar constraints. In *Principles and Practice of Constraint Programming-CP 2006*, pages 530–544. Springer.

[81] Thompson, G. M. (1995). Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science*, 41(4):595–607.

[82] Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L. (2012). Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385.

[83] Wang, Y. M. (2006). *A column generation approach for stochastic optimization problems*. PhD thesis, University of Texas at Austin.

[84] Wolsey, L. A. and Nemhauser, G. L. (2014). *Integer and combinatorial optimization*. John Wiley & Sons.

[85] Zhu, X. and Sherali, H. D. (2009). Two-stage workforce planning under demand fluctuations and uncertainty. *Journal of the Operational Research Society*, 60(1):94–103.

APPENDIX A EXAMPLE FOR THE COMPARISON BETWEEN THE TWO FORMULATIONS

The problem is a mono-activity tour scheduling with a three-day time horizon; each day is divided into four time periods; the total tour length ranges between 4 and 6 time periods; the minimum and maximum number of days are 1 and 2, respectively; there are no constraints for the minimum rest time; and the total number of employees is 1. The grammar used to compose the daily shifts is as follows:

$G = (\Sigma = (w, b), N = (S, X, W, B), P, S)$,
where productions P are: $S \rightarrow XW$, $X \rightarrow WB$, $W \rightarrow WW|w$, $B^{[2,2]} \rightarrow b$

Daily shifts have a working length of 3 time periods and must have one break allocated in their second time period (production $B^{[2,2]} \rightarrow b$). Table A.1 presents the employee requirements and the structure of the feasible shifts and tours. The cost of the activity per time period at each day is 1 and the costs of overcovering and undercovering of employee requirements are 1 and 2, respectively.

Table A.1 Employee requirements, shifts and tours structures.

Day (d)	1				2				3			
Time period (i)	1	2	3	4	1	2	3	4	1	2	3	4
Empl. req. (b_{dij})	1	1	0	0	1	1	0	0	1	1	0	0
Shift1 (x_{11})	1	0	1	1	-	-	-	-	-	-	-	-
Shift2 (x_{21})	-	-	-	-	1	0	1	1	-	-	-	-
Shift3 (x_{31})	-	-	-	-	-	-	-	-	1	0	1	1
Tour1 (x_1)	1	0	1	1	1	0	1	1	0	0	0	0
Tour2 (x_2)	1	0	1	1	0	0	0	0	1	0	1	1
Tour3 (x_3)	0	0	0	0	1	0	1	1	1	0	1	1

For the Daily-based formulation, since all the shifts have the same working length (3 time periods) it is easy to show that the constraint for the minimum and maximum number of working days (4.4) is redundant and that the summation of the value of the three decision variables (x_{11}, x_{21}, x_{31}) have to fall inside the interval [4/3, 6/3] because of constraints (4.5). Now, since all the shifts have the same structure and the employee requirements are the same

at each day, we can conclude that the value of $f(\overline{F_{\mathcal{S}}})$ is the same when we evaluate the point (1,1,0) or the point (2/3, 2/3, 2/3). Therefore, we can evaluate the value of $f(\overline{F_{\mathcal{S}}})$ when all the variables fall, with the same value, inside the interval [4/9, 6/9]. Figure A.1 presents the value of the objective function for the evaluated points, as well as the optimal solution $f(\overline{F^{\star}_{\mathcal{S}}}) = 16$ with $x^{\star}_{11} = x^{\star}_{21} = x^{\star}_{31} = 4/9$.

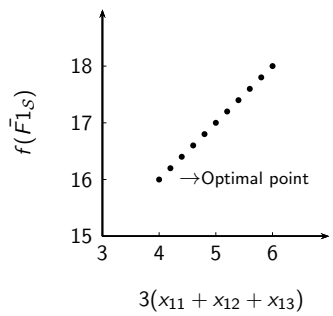


Figure A.1 Optimal solution for the LP relaxation of the example using the Daily-based formulation.

The solution of the problem with the Daily-based formulation is not feasible for the Tour-based formulation, since all possible tours must have a length of 6 time periods and must include 2 working days. As mentioned before, since the employee requirements and the shifts are the same for every day, the value of $f(\overline{F_{\mathcal{T}}})$ is the same when we evaluate all the points where $x_1 + x_2 + x_3 = 1$. In this case the optimal value of $f(\overline{F^{\star}_{\mathcal{T}}})$ is 18. Hence $f(\overline{F^{\star}_{\mathcal{S}}}) < f(\overline{F^{\star}_{\mathcal{T}}})$.

APPENDIX B LABEL SETTING ALGORITHM TO SOLVE THE SPPRC

Let $\mathcal{Q}$ be the set of labels. Each label $l \in \mathcal{Q}$ has an associated path $\mathcal{P}(l)$ and a set of attributes: its resident node $v(l)$, its predecessor node $p(l)$, its cost $c(l)$, its distance $t(l)$ and its number of working days $d(l)$ accumulated along $\mathcal{P}(l)$.

Algorithm 2 presents the pseudocode of the labeling algorithm. The inputs are the tour graph $G^e(\mathcal{N}, \mathcal{A})$ and the maximum number of tours α to generate per iteration. The output corresponds to a vector of paths $\vec{\mathcal{P}^e}$ with negative reduced cost. Line 1 returns an initial set of labels (partial paths from v_s to all its successors). Line 2 selects the first label to be processed according to its cost. Line 4 searches to prune by bound the current label before extending it. This pruning is done by calculating an optimistic prediction of the total cost of the path that might be generated by the current label. If such cost is negative, the label is not pruned, otherwise the label is removed from list $\mathcal{Q}$ without being processed. Line 5 seeks to extend label l_1 to all of its successors. A new label l_2 is created and stored in $\mathcal{Q}$ (Line 9) if it is feasible (Line 6), non-dominated (Line 7) and its resident node is different than the sink node v_f. If the resident node is v_f and the cost of label l_2 is negative, a new path is stored in $\vec{\mathcal{P}^e}$ (Line 11). The feasibility function checks if the label to be created can reach the minimum number of working days and tour length and, at the same time, if it does not exceed the maximum number of working days and tour length. The dominance function compares certain attributes of the label to be created with the rest of the labels in $\mathcal{Q}$. Hence, if the resident node, accumulated time and number of working days are the same for both labels and if the cost of label $l^{'}$ is lower or equal than the cost of label l, $l^{'}$ dominates l. The algorithm stops when either set $\mathcal{Q}$ is empty or the number of tours t generated is greater than or equal to α.

Algorithm 2 Label setting algorithm to solve the SPPRC

Input $G^e(\mathcal{N}, \mathcal{A}), \alpha$
Output $\vec{\mathcal{P}^e}$

```
initialization
selectLabel
while Q ≠ ∅ ∧ t < α do
    if pruneByBound (l_1)= false then
        for v_i ∈ N(v_k) do
            if pruneByFeasibility (l_1, v_i)= false then
                if dominance (l_2)= false then
                    if v_i ≠ v_f then
                        Q ← l_2
                    else
                        P⃗ ← P, t = t + 1
    remove l_1 from Q
    selectLabel
return P⃗^e
```