CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

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

María I. Restrepo
Bernard Gendron
Louis-Martin Rousseau

November 2015

CIRRELT-2015-57

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

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

María I. Restrepo[1,2,*], Bernard Gendron[1,3], Louis-Martin Rousseau[1,2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Mathematics and Industrial Engineering, Polytechnique Montréal, P.O. Box 6079, Station Centre-Ville, Montréal, Canada H3C 3A7

[3] Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

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.

Acknowledgements. The authors would like to thank the Fonds de recherche du Québec - Nature et technologies (FRQNT) which supported this work with a grant.

---

* Corresponding author: Maria-Isabel.Restrepo@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2015

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 discontinuous* 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 intervals* of equal length. For each time interval, 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 interval, 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 interval 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 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 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* [5], 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 [32]. 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 [11] by a state-of-the-art MIP solver on instances with up to 30 work activities.

The paper is organized as follows. In Section 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 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 4, we describe the decomposition method that combines BD and CG. Computational experiments are presented and discussed in Section 5. Concluding remarks follow in Section 6.

2 Background Material

We first review the literature on shift and tour scheduling problems (Section 2.1), focusing in particular on multi-activity versions of these problems (Section 2.2). Then, Section 2.3 presents some background material related with the use of context-free grammars for shift scheduling.

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. [17, 18], Alfares [1] and Van den Bergh et al. [35] 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 [14]. 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 interval are met. Later, in one of the first attempts to solve shift scheduling problems with an implicit model, Moondra [26] 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 [4], with an implicit formulation where shifts are grouped into shift types according to their start time, length and break window. Thompson [34] combines the work of Moondra [26] and Bechtold and Jacobs [4] to implicitly model meal breaks, but also to schedule rest breaks and to allow the use of overtime. Aykin [2] 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 intervals.

Jarrah et al. [21] 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 [20] present an implicit model that allows start time flexibility within continuous (i.e., shifts can span over multiple days) and discontinuous employee tours, but that does not consider meal breaks. Brusco and Jacobs [8] integrate the work of Bechtold and Jacobs [4] and Jacobs and Brusco [20] 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 [6] 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. [25], Ni and Abeledo [27], Brunner and Stolletz [7]). 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. [31] use BD in a continuous tour scheduling problem to prove that the forward and backward constraints introduced by Bechtold and Jacobs [4] 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.

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. [11] propose to solve the scalability issues identified in Côté et al. [10] 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. [15] present a CP-based column generation algorithm as a way to model complex regulation constraints to solve large MASSP instances. Quimper and Rousseau [29] 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. [12] 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 (i.e., one week). Restrepo et al. [32] 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 [13] propose a heuristic based on tabu search and B&B to solve the personalized MASSP over multiple days. In this method, days-off are previously assigned to the employees and some constraints related with the composition of feasible tours are not considered (i.e., minimum and maximum number of working hours per week). Detienne et al. [16] 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. [11].

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 [30].

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

In the previous example, productions $W \rightarrow w_1$, $W \rightarrow w_2$, $B \rightarrow b$ and $R \rightarrow r$ 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 $W \rightarrow WW$ 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. Production $F \rightarrow XW$ 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 $S \rightarrow RF$ and $S \rightarrow FR$. 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 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 $A^{S\rightarrow RF,1}_{15}$. Continuous-line or-nodes are part of the parse trees associated with and-node $A^{S\rightarrow FR,1}_{15}$.

Note that the children of the root node $ch(O^{S}_{15}) = \{A^{S\rightarrow RF,1}_{15}, A^{S\rightarrow FR,1}_{15}\}$ 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 $A^{\Pi,k}_{d1n}$ are characterized by their starting time $t^{\Pi,k}_{d1n}$, working length $w^{\Pi,k}_{d1n}$, length including breaks $l^{\Pi,k}_{d1n}$ and finish time $f^{\Pi,k}_{d1n} = t^{\Pi,k}_{d1n} + l^{\Pi,k}_{d1n} - 1$. In DAG Γ, and-node $A^{S\rightarrow RF,1}_{15}$ generates shifts $rwbww$ and $rwwbw$, while and-node $A^{S\rightarrow FR,1}_{15}$ generates shifts $wbwwr$ and $wwbwr$.

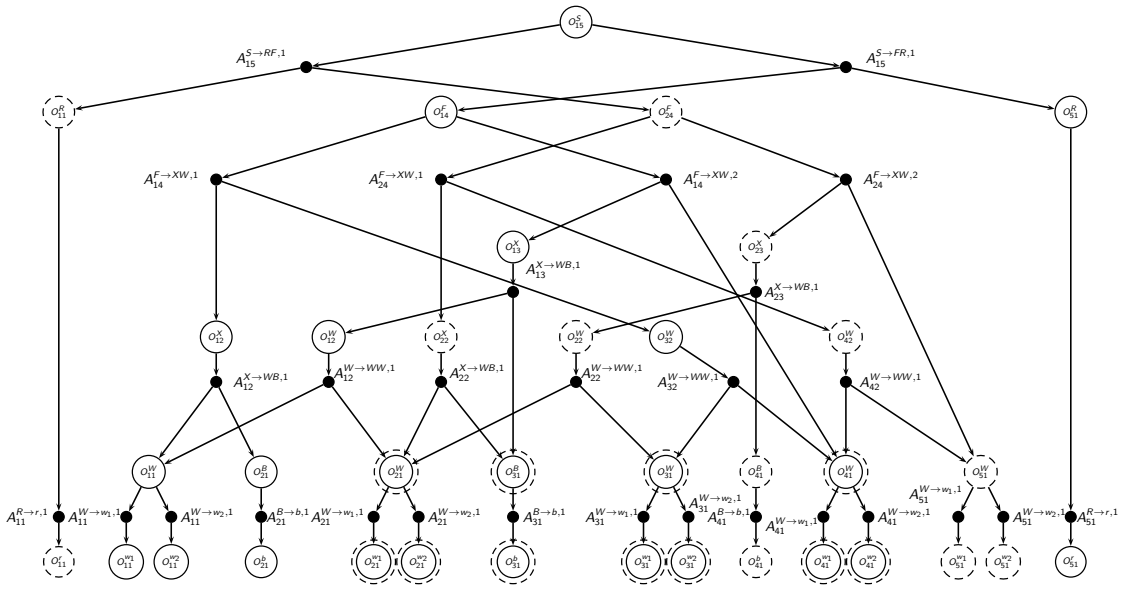

Figure 1 – DAG Γ on words of length five and two work activities

3 Grammar-Based Model

The work of Côté et al. [11] 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 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 intervals 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 interval i and activity j;

c_{dij}: non-negative cost associated to one employee working on activity j, at time interval i, at day d;

c^+_{dij}, c^-_{dij}: non-negative overcovering and undercovering costs of staff requirements for day d, time interval i and activity j, respectively.

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 interval i, at day d;

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

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

$$Z(G_{\mathcal{S}}) = \min \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij} y_{dij} + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} (c_{dij}^{+} s_{dij}^{+} + c_{dij}^{-} s_{dij}^{-}) \tag{1}$$

$$y_{dij} - s_{dij}^{+} + s_{dij}^{-} = b_{dij}, \forall\, d \in D,\ i \in I_d,\ j \in J, \tag{2}$$

$$\sum_{A_{dil}^{\Pi,k} \in ch(O_{dil}^{\pi})} v_{dil}^{\Pi,k} = \sum_{A_{dil}^{\Pi,k} \in par(O_{dil}^{\pi})} v_{dil}^{\Pi,k}, \forall\, d \in D,\ O_{dil}^{\pi} \in O_d \setminus \{O_{d1n}^{S} \cup L_d\}, \tag{3}$$

$$y_{dij} = \sum_{A_{di1}^{\Pi,1} \in par(O_{di1}^{j})} v_{di1}^{\Pi,1}, \forall\, d \in D,\ i \in I_d,\ j \in J, \tag{4}$$

$$\sum_{A_{d1n}^{\Pi,k} \in ch(O_{d1n}^{S})} v_{d1n}^{\Pi,k} \leq |E|, \forall\, d \in D, \tag{5}$$

$$v_{dil}^{\Pi,k} \geq 0, \forall\, d \in D,\ A_{dil}^{\Pi,k} \in A_d, \tag{6}$$

$$s_{dij}^{+},\ s_{dij}^{-} \geq 0, \forall\, d \in D,\ i \in I_d,\ j \in J, \tag{7}$$

$$y_{dij} \geq 0 \text{ and integer}, \forall\, d \in D,\ i \in I_d,\ j \in J. \tag{8}$$

The objective of $G_{\mathcal{S}}$, (1), is to minimize the total staffing cost plus the penalization for overcovering and undercovering of staff requirements. Constraints (2) ensure that staff requirements per day d, time interval i and work activity j are met. Constraints (3) guarantee, for every or-node in Γ_d, $d \in D$, excluding the root node O_{d1n}^{S} 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 (4) set the value of variables y_{dij} as the summation of the value of the parents of leaf nodes O_{di1}^{j}. Constraints (5) guarantee that at most $|E|$ employees are assigned to the daily shift shells (children of the root node) at each day d. Constraints (6)-(8) set the non-negativity of variables $v_{dil}^{\Pi,k}$, s_{dij}^{+}, s_{dij}^{-} and the non-negativity and integrality of variables y_{dij}.

The solution obtained from model (1)-(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_{151}^{2} is reached, it means that activity 2 should be inserted at position 5 in the schedule of day 1).

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 2 – Weekly tours composed of and-nodes (shift shells) from Γ

Observe that model (1)-(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 2 presents an example of three tours composed with the shift shells presented in Figure 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 intervals, 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 $A^{S\rightarrow RF,1}_{d15}$ generating shifts $\{rwbww, rwwbw\}$, S_2 corresponds to and-node $A^{S\rightarrow FR,1}_{d15}$ generating shifts $\{wbwwr, wwbwr\}$ and Do corresponds to allocating a day-off.

Model (1)-(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 introduce a set of decision variables x_t denoting the number of employees assigned to tour $t \in \mathcal{T}$. Constraints (9) set the link between these variables and shift shell variables $v^{\Pi,k}_{d1n}, A^{\Pi,k}_{d1n} \in ch(O^S_{d1n})$. Constraints (5) are replaced by constraint (10), which guarantees that exactly $|E|$ employees are assigned to the set of tours $\mathcal{T}$. Constraints (11) set the non-negativity and integrality of tour-based variables x_t.

$$v^{\Pi,k}_{d1n} = \sum_{t\in\mathcal{T}} \delta^{\Pi,k}_{dt} x_t,\ \forall\, d \in D,\ A^{\Pi,k}_{d1n} \in ch(O^S_{d1n}), \tag{9}$$

$$\sum_{t\in\mathcal{T}} x_t = |E|, \tag{10}$$

$$x_t \geq 0 \text{ and integer},\ \forall\, t \in \mathcal{T}. \tag{11}$$

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

$$Z(G_{\mathcal{T}}) = \min \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} c_{dij} y_{dij} + \sum_{d \in D} \sum_{i \in I_d} \sum_{j \in J} (c^{+}_{dij} s^{+}_{dij} + c^{-}_{dij} s^{-}_{dij})$$
$$\text{subject to} \quad (2)-(4) \text{ and } (6)-(11).$$

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 $G_{\mathcal{T}}$.

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 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 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 4.3.

4.1 Benders Decomposition

The structure of model $G_{\mathcal{T}}$ suggests to partition the set of $v^{\Pi,k}_{dil}$ variables into two sets. Indeed, due to the linking constraints (9), when variables x_t, $t \in \mathcal{T}$, are fixed and satisfy constraints (10)-(11), variables $v^{\Pi,k}_{d1n}$, $A^{\Pi,k}_{d1n} \in ch(O^{S}_{d1n})$, associated with the shift shells (children of root node O^{S}_{d1n}) 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 interval i, at day d of length $l < n$, denoted as $v^{\Pi,k}_{dil}$, $A^{\Pi,k}_{dil} \in A_d \setminus ch(O^{S}_{d1n})$. This partition of the variables of model $G_{\mathcal{T}}$ is the basic idea of the proposed BD approach.

4.1.1 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 subproblem*, denoted as $\mathcal{Q}(\overline{\mathbf{v}_d})$ for a given day d, is as follows:

$$Z(\mathcal{Q}(\overline{\mathbf{v}_d})) = \min \sum_{i \in I_d} \sum_{j \in J} c_{dij} y_{dij} + \sum_{i \in I_d} \sum_{j \in J} (c^+_{dij} s^+_{dij} + c^-_{dij} s^-_{dij}) \tag{12}$$

$$y_{dij} - s^+_{dij} + s^-_{dij} = b_{dij}, \; \forall \, i \in I_d, \; j \in J, \tag{13}$$

$$\sum_{A^{\Pi,k}_{dil} \in ch(O^\pi_{dil})} v^{\Pi,k}_{dil} = \sum_{A^{\Pi,k}_{dil} \in par(O^\pi_{dil})} \overline{v^{\Pi,k}_{d1n}}, \; \forall \, O^\pi_{dil} \in ch(A^{\Pi,k}_{d1n}) \setminus L_d, \tag{14}$$

$$\sum_{A^{\Pi,t}_{dil} \in ch(O^\pi_{dil})} v^{\Pi,k}_{dil} = \sum_{A^{\Pi,k}_{dil} \in par(O^\pi_{dil})} v^{\Pi,k}_{dil}, \; \forall \, O^\pi_{dil} \in O_d \setminus \{O^S_{d1n} \cup L_d \cup ch(A^{\Pi,k}_{d1n})\}, \tag{15}$$

$$y_{dij} = \sum_{A^{\Pi,k}_{di1} \in par(O^j_{di1})} v^{\Pi,1}_{di1}, \; \forall \, i \in I_d, \; j \in J, \tag{16}$$

$$v^{\Pi,k}_{dil} \geq 0, \; \forall \, A^{\Pi,k}_{dil} \in A_d \setminus ch(O^S_{d1n}), \tag{17}$$

$$s^+_{dij}, s^-_{dij} \geq 0, \; \forall \, i \in I_d, \; j \in J, \tag{18}$$

$$y_{dij} \geq 0 \text{ and integer}, \; \forall \, i \in I_d, \; j \in J. \tag{19}$$

For a given number of employees assigned to each shift shell (fixed variables $\overline{v^{\Pi,k}_{d1n}}$), the objective, (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 (13) guarantee that staff requirements are met. Constraints (14)-(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 (17)-(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 subproblems (12)-(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 (19) on variables y_{dij}. Second, we will generate integer Benders cuts to guarantee the convergence of the method to an optimal solution.

4.1.2 Classical Benders Cuts

Let $\overline{\mathcal{Q}}(\overline{\mathbf{v}_d})$ denote the LP relaxation of model (12)-(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 (13) and (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 non-negative 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:

$$\theta_d \geq \sum_{i \in I_d} \sum_{j \in J} \rho_{dij} b_{dij} + \sum_{O^{\pi}_{dil} \in ch(A^{\Pi,k}_{d1n})} \gamma^{\pi}_{dil} \sum_{A^{\Pi,k}_{d1n} \in par(O^{\pi}_{dil})} v^{\Pi,k}_{d1n}, \; \forall \, d \in D, \; (\rho_d, \gamma_d) \in E_{\mathbf{\Delta}_d}. \quad (20)$$

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 $B_{\mathcal{T}}$:

$$\begin{aligned} Z(B_{\mathcal{T}}) = \min & \sum_{d \in D} \theta_d \\ \text{subject to} \quad & (20), (9) - (11) \text{ and} \\ & v^{\Pi,k}_{d1n} \geq 0, \; \forall \, d \in D, \; A^{\Pi,k}_{d1n} \in ch(O^{S}_{d1n}). \end{aligned}$$

Optimality cuts (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 (12)-(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.

4.1.3 Algorithmic Strategy

Consider an iterative BD approach to generate optimality cuts (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 (12)-(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 $\overline{v^{\Pi,k}_{d1n}}(l)$ denote the values of the shift shell variables $v^{\Pi,k}_{d1n}$ 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 in Section 4.1.4) that tend to eliminate solution $\overline{v^{\Pi,k}_{d1n}}(l)$ 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 (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 (12)-(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 (12)-(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 4.3.

4.1.4 Integer Benders Cuts

Constraints (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) 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)-(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^e_{di}, 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:

$$\sum_{e \in E} z^e_{di} = z_{di}, \; \forall d \in D, \; i \in S_d, \tag{21}$$

$$z^{e+1}_{di} \leq z^e_{di}, \; \forall d \in D, \; i \in S_d, \; e \in E \setminus \{|E|\}, \tag{22}$$

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

Equations (21) guarantee, for every day $d \in D$, that the sum of the binary variables z^e_{di} is equal to the number of employees assigned to $i \in S_d$, while (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 $z^e_{di} = 1$ for $e \leq m$ and $z^e_{di} = 0$ 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^e_{di}.

Let $\overline{z^e_{di}}(l)$ be the value of variable z^e_{di} corresponding to the current solution $\overline{v^{\Pi,k}_{d1n}}(l)$ for day d and $\overline{\mathcal{B}_d}(l) = \{(i, e) \in S_d \times E \,|\, \overline{z^e_{di}}(l) = 0\}$. Since the objective is to eliminate the current solution $\overline{v^{\pi,k}_{d1nl}}$ 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:

$$\theta_d \geq s_d(l) - (s_d(l) - \theta_d(l)) \left(\sum_{(i,e) \in \overline{\mathcal{B}_d}(l)} z^e_{di} \right). \tag{23}$$

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 smaller than or equal to $\theta_d(l)$. This optimality cut exploits the fact that exactly $|E|$ binary variables take value 1 and is the adaptation to this case of the optimality cut used in the integer L-shaped method for stochastic programming [22]. Other types of cuts based on 0-1 variables have been used in variants of Benders decomposition, such as logic-based Benders decomposition [19], branch-and-cut-based Benders decomposition [33] and combinatorial Benders decomposition [9].

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.

4.2.1 Pricing Subproblem for Tour Generation

Let $\lambda_{d1n}^{\Pi,k}$ and δ be the dual variables associated with constraints (9) and (10), respectively. The reduced cost $\bar{c}_t$ of column (tour) t is given by:

$$\bar{c}_t = \left(\sum_{d \in D} \sum_{A_{d1n}^{\Pi,k} \in ch(O_{d1n}^S)} \lambda_{d1n}^{\Pi,k} \delta_{dt}^{\Pi,k} \right) - \sigma. \tag{24}$$

Expression (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 with the minimum and maximum number of working days in a tour (Λ_l and Λ_u, respectively); the minimum and maximum tour length in time intervals (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 with 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 with 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 $\bar{c}_t < 0$.

4.2.2 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 4.1.3 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. [3], Côté et al. [12]). 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 $\bar{c}_t \geq 0$ 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.

4.3 Overall Algorithm

The core of the BD/CG algorithm corresponds to the three-step algorithmic strategy presented in Section 4.1.3, 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 [24]. This is achieved by alternating between the CG method (without B&P) and the generation of classical Benders cuts (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 [28], which is an alternative to solving the extra auxiliary subproblem introduced in Magnanti and Wong [23]. 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 (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 (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 (23) and corresponding binary variables with their defining constraints (21)-(22). 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 non-negative 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 to 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

$l = 0, Z^L(l) = -\infty, \overline{Z^U}(l) = \infty, Z^U(l) = \infty, Int =$ false, $Cut =$ true
while $((Z^U(l) - Z^L(l))/Z^U(l) > \epsilon_1)$ and $(Cut{=}{=}\text{true})$ **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` (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 $(Cut{=}{=}\text{false})$ **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` (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`

Algorithm 1 – BD/CG algorithm for the MATSP

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.*

Proof. Because the maximum number of classical Benders cuts (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 4.1.4, 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. 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 (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 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 5.1, 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. [32] for the personalized variant of the problem. In Section 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. [11].

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.

5.1.1 Problem Definition and Grammar

Tour generation

1. The planning horizon is seven days, where each day is divided into 96 time intervals 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 interval 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 interval 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 $\Pi \in P$, $\Pi \rightarrow_{[\min,\ \max]}$ restrict the subsequences generated by a given production to a length between a minimum and maximum number of periods. The grammar and the productions that define the anonymous discontinuous MATSP are as follows:

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

5.1.2 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 interval in the planning horizon. Undercovering and overcovering of staff requirements are generated by randomly adding or removing demand.

Table 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.

Group	*Nb.Act*	*Nb.Emp*	*Nb.ChRoot*	*Nb.AndNodes*	*Nb.OrNodes*	*Nb.Leaves*	*Nd.* $G(\mathcal{N},\mathcal{A})$	*Arcs* $G(\mathcal{N},\mathcal{A})$
G1	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
	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
G2	1	18	139	6,535	5,320	246	973	275,745
	2	22	139	8,819	6,567	318	973	275,745
	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

Table 1 – Size of MATSP instances

5.1.3 Results

Tables 2 and 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.

Nb.Act	1	2	3	4	5	6	7	8	9	10
BD/CG										
T.time	24.59	94.02	97.30	722.25	343.04	3,269.97	2,099.40	1,362.20	1,933.60	6,285.78
T. MP	6.75	53.80	36.86	573.17	206.72	2,807.81	1,548.53	1,029.31	1,487.80	3,933.20
T. CG	9.89	15.97	17.34	68.21	26.19	230.78	341.89	35.73	50.21	615.17
T. LR BSP	7.53	23.51	41.83	78.51	105.95	220.41	191.52	230.69	371.24	623.38
T.MIP BSP	0.24	0.44	0.82	1.59	3.45	11.76	7.64	65.51	22.68	1,113.64
IBC/Nb. P	0/0	0/0	0/0	0/0	0/0	5/4	2/2	0/0	0/0	3/2
Gap	0.80%	0.75%	0.75%	0.89%	0.75%	0.86%	1.10%	0.58%	0.66%	2.37%
Opt.	10	10	10	10	10	10	9	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 2 – Results on MATSP instances with smooth demand shape

Nb.Act	1	2	3	4	5
BD/CG					
T.time	333.65	2,254.24	1,383.11	2,788.48	4,950.17
T. MP	240.10	2,029.86	1,162.28	1,908.43	2,151.28
T. CG	89.83	148.90	61.12	234.37	509.26
T. LR BSP	23.25	76.13	156.28	215.18	438.32
T.MIP BSP	0.29	30.45	2.70	429.15	1,853.03
IBC/Nb. P	0/0	2/2	0/0	5/2	3/1
Gap	0.89%	0.87%	0.98%	1.91%	3.03%
Opt.	10	10	9	7	4

Table 3 – Results on MATSP instances with erratic demand shape

From Tables 2 and 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 2.37%. For instances with erratic demand behaviour, computational times and solution quality are worse than those reported in Table 2. However, one can observe that even if the instances are not solved to optimality, the value of *Gap* does not exceed 3.03%. 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 4 activities and erratic demand behavior needed four 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.

5.2 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 with 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 (9)-(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 with 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. [11]. Before analyzing the results, we first present the instances used in our tests.

5.2.1 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 that follow 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 4. The notation used in this table is the same as the one used in Table 1. Ten different staff requirements were generated for each instance.

Group	*Nb.Act*	*Nb.Emp*	*Nb.ChRoot*	*Nb.AndNodes*	*Nb.OrNodes*	*Nb.Leaves*
G1	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
	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
G2	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
	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

Table 4 – Size of MASSP instances

5.2.2 Results

Table 5 presents the computational results on instances dealing with up to 30 work activities for the anonymous discontinuous 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 bond (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.*

				BD					***GB***	
Nb.Act	*Tot. time*	*T. BMP*	*T. LR BSP*	*T. MIP BSP*	*IBC/Nb.P*	*Gap*	*Opt.*	*Tot. time*	*Gap*	*Opt.*
1	**208.65**	0.63	89.17	118.85	6/2	0.10%	10	272.19	0.09%	10
2	**194.45**	0.58	145.67	48.20	15/2	0.32%	10	241.15	0.17%	10
3	**243.57**	1.36	238.24	3.97	4/3	0.45%	10	1,159.92	0.25%	10
4	289.47	0.88	285.95	2.64	1/1	0.49%	10	178.69	0.25%	10
5	396.14	1.61	390.52	4.01	3/3	0.28%	10	367.62	0.05%	10
6	490.45	2.05	480.80	7.60	5/5	0.36%	10	351.04	0.05%	10
7	773.08	4.63	749.08	19.37	9/5	0.64%	10	438.28	0.17%	10
8	633.34	3.08	600.86	29.40	13/7	0.67%	10	428.83	0.08%	10
9	**496.03**	0.91	487.03	8.09	3/3	0.52%	10	588.02	0.14%	10
10	**621.62**	1.73	603.17	16.72	5/4	0.64%	10	772.57	0.12%	10
11	**702.99**	0.48	658.80	43.72	1/1	0.61%	10	3,096.72	0.04%	10
12	**793.93**	0.57	664.22	129.14	4/4	0.61%	10	3,680.88	0.00%	10
13	**678.30**	0.38	649.76	28.17	0/0	0.62%	10	3,013.46	0.06%	10
14	**877.10**	0.51	776.61	99.99	3/2	0.56%	10	3,453.49	2.69%	9
15	**935.29**	0.76	907.70	26.83	1/1	0.66%	10	2,724.59	0.08%	10
16	**1,201.13**	1.66	1,151.41	48.06	2/2	0.46%	10	3,120.24	0.05%	10
17	**881.29**	0.55	849.94	30.80	2/2	0.64%	10	2,976.84	0.26%	10
18	**1,168.27**	1.94	1,107.95	58.37	3/3	0.59%	10	3,685.44	0.87%	9
19	**1,389.03**	3.06	1,353.20	32.77	1/1	0.52%	10	3,380.61	0.18%	10
20	**1,375.73**	2.70	1,309.36	63.67	4/4	0.67%	10	3,416.84	0.00%	10
30	**4,026.44**	0.84	3,245.40	780.20	4/3	0.78%	10	6,130.31	7.16%	4

Table 5 – Results on MASSP instances

From Table 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 12 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 with the LP solution of the Benders 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 58 out of 210 instances, the generation of integer Benders cuts was needed.

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 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. [11], our approach presented competitive and often better solution times.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[31] M. Rekik, J.-F. Cordeau, and F. Soumis. Using Benders decomposition to implicitly model tour scheduling. *Annals of Operations Research*, 128(1-4):113–133, 2004.

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

[33] S. Sen and H.D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006.

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

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