A Branch-and-Price Algorithm for the Multi-Activity Multi-Task Shift Scheduling Problem

Vincent Boyer
Bernard Gendron
Louis-Martin Rousseau

May 2012

CIRRELT-2012-17

A Branch-and-Price Algorithm for the Multi-Activity Multi-Task Shift Scheduling Problem

Vincent Boyer^{1,2*}, Bernard Gendron^{1,2}, Louis-Martin Rousseau^{1,3}

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

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

[3] Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, C.P. 6079, succursale Centre-ville, Montréal, Canada H3C 3A7

Abstract. The multi-activity multi-task shift scheduling problem consists in assigning interruptible activities and uninterruptible tasks to a set of employees in order to satisfy a demand function. In this paper, we consider the personalized variant of the problem where the employees have different qualifications, preferences, and availabilities. We present an exact branch-and-price algorithm to solve this problem. The pricing subproblems in column generation are formulated with context-free grammars that are able to model complex rules in the construction of feasible shifts for an employee. We present results for a large set of instances inspired by real cases and show that this approach is sufficiently flexible to handle different classes of problems.

Keywords. Multi-activity multi-task shift scheduling problem, precedence constraints, branch-and-price, context free grammar.

Acknowledgements. Partial funding for this project has been provided by Mitacs and by the Fonds de recherche du Québec – Nature et Technologies (FRQNT). This support is gratefully acknowledged.

* Corresponding author: Vincent.Boyer@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1 Introduction

Shift scheduling consists in assigning a sequence of activities and tasks to a set of employees for each time period in a planning horizon. A shift consists of a continuous sequence of activities/tasks, which may include breaks and a lunch-break. The content of a shift is generally constrained by rules arising from regulation agreements and ergonomic considerations. In this paper, we consider the shift scheduling problem with multiple activities and multiple tasks where all the employees are different, i.e., they can perform only a subset of the activities and tasks and have different periods of availability.
Activities represent daily operations in a company, such as assisting customers. There is a demand that should be met, and the overcovering and undercovering of this demand reflect the quality of service provided by the company. An activity can be interrupted and can be assigned to several employees at the same time.
Tasks represent small but essential operations, such as unloading cargo or preparing a stall, and there is a large penalty for uncovering them. Tasks have a fixed length and there are usually hard constraints on their completion time and on the sequence in which they should be executed. They must be executed without interruption by a fixed number of employees.
To the best of our knowledge, few papers in the literature have addressed this problem. However, the personnel scheduling problem is a well-known problem of operations research and has been widely studied. The surveys of Ernst et al. [9, 8] give many references, but few of them consider simultaneous activity and task assignment.
Demassey et al. [7] studied the multi-activity case for a 24-hour planning horizon with up to 10 activities. They used a set covering formulation and applied a column generation approach where the pricing subproblem is solved with constraint programming. The results show that this method can solve instances with only up to three activities.
Lequy et al. [11] presented two integer programming (IP) models and a column generation approach based on multi-commodity flow formulations for the multi-activity shift scheduling problem where shifts and breaks are assigned a priori to the employees. These models lead to very large IPs, and the authors proposed rolling-horizon heuristics based on column generation for the larger instances. More recently, Lequy et al. [12, 13] extended their approach to the multi-task case and considered the possibility that a task could be performed by more than one employee with synchronization. Two-stage heuristics are proposed: first the tasks and then the activities are assigned to the employees.
The present work is an extension of Côté et al. [2, 4, 3]. They proposed a grammar-based model for the personalized multi-activity shift scheduling problem and a branch-and-price approach with column generation to solve this problem exactly. The results showed that this approach can handle instances of up to 100 employees and 15 activities for a planning horizon of 7 days when shifts are preassigned to employees. We have extended this approach to include tasks with precedence constraints, and we present an extensive study of the branching strategy. Furthermore, we present results for instances where no shifts are preassigned.
This paper is organized as follows. In Section 2 we present our mathematical model

for the personalized multi-activity multi-task shift scheduling problem. In Section 3 we introduce grammar theory and show how it is used to construct feasible shifts for each employee. Section 4 deals with the general framework of the proposed grammar-based branch-and-price algorithm. Finally, in Section 5 we present computational results for a set of instances inspired by real cases.

2 The Shift Scheduling Problem

In the personalized multi-activity multi-task shift scheduling problem (SSP), we must assign to each employee $e \in E$ a feasible shift $s \in \Omega^e$ to cover at a minimum cost the demand for activities and tasks over a planning horizon I. We assume that each employee has different characteristics and thus can perform only a subset of the activities A and tasks T. Different employees have different availabilities over the planning horizon. The set of feasible shifts for each employee is determined by these characteristics and also by rules arising from work agreement regulations or ergonomic considerations. We will use the term *job* to refer to either an activity or a task.
With each shift $s \in \Omega^e$ we associate a cost $c^e_s \geq 0$. This can include different components, such as the cost for an employee to perform a job and the cost of transition from one job to another. Furthermore, we allow undercovering for activities and tasks and overcovering only for activities, since tasks cannot be repeated. Let c^u_{ia} and c^o_{ia} be the costs for undercovering and overcovering, respectively, an activity $a \in A$ at period $i \in I$ and let c^u_t be the cost for undercovering a task $t \in T$.
We assume that the demand b_{ia} for activity $a \in A$ is known at period $i \in I$.

2.1 Mathematical Formulation

Our model for the personalized multi-activity multi-task SSP, model (D), is an extension of that of Côté et al. [3], who used a classical set-covering formulation introduced by Dantzig [6] for the shift scheduling problem:

$$f(D) = \min \sum_{e \in E} \sum_{s \in \Omega^e} c^e_s x^e_s + \sum_{i \in I} \sum_{a \in A} (c^u_{ia} u_{ia} + c^o_{ia} o_{ia}) + \sum_{t \in T} c^u_t u_t \tag{1}$$

$$\sum_{e \in E} \sum_{s \in \Omega^e} \delta^e_{ias} x^e_s + u_{ia} - o_{ia} = b_{ia} \qquad \forall i \in I,\ a \in A \tag{2}$$

$$\sum_{i \in I} \sum_{e \in E} \sum_{s \in \Omega^e} \beta^e_{its} x^e_s + u_t = 1 \qquad \forall t \in T \tag{3}$$

$$\sum_{s \in \Omega^e} x^e_s = 1 \qquad \forall e \in E \tag{4}$$

$$x^e_s \in \{0,\ 1\} \qquad \forall e \in E,\ s \in \Omega^e \tag{5}$$

$$u_{ia} \geq 0,\ o_{ia} \geq 0 \qquad \forall i \in I,\ \forall a \in A \tag{6}$$

$$u_t \geq 0 \qquad \forall t \in T, \tag{7}$$

where

- $\delta^e_{ias} = 1$ if activity $a \in A$ is **assigned** at period $i \in I$ in shift $s \in \Omega^e$ for employee $e \in E$, and $\delta^e_{ias} = 0$ otherwise;
- $\beta^e_{its} = 1$ if task $t \in T$ **starts** at period $i \in I$ in shift $s \in \Omega^e$ for employee $e \in E$, and $\beta^e_{its} = 0$ otherwise;
- $x^e_s = 1$ if employee $e \in E$ is assigned to shift $s \in \Omega^e$, and $x^e_s = 0$ otherwise;
- u_{ia} and o_{ia} represent the undercovering and overcovering at period $i \in I$ of activity $a \in A$; and
- u_t is the variable associated with the undercovering of task $t \in T$.

The objective (1) is composed of the cost of assigning a shift to the employee, the cost for undercovering and overcovering activities, and the cost for undercovering tasks. Constraints (2) and (3) represent the satisfaction of the demand for activities and tasks in the planning horizon. Constraint (4) ensures that only one shift is assigned to each employee.

2.2 Precedence Constraints

If there are precedence constraints, i.e., constraints on the sequence of tasks, further constraints must be added to model (D). For a task $t \in T$, we denote by l_t its fixed length and by $P(t)$ the set of tasks that should be performed before t. In a shift we know the start and end times of a task, so one way to formulate these constraints is to assume that task $t \in T$ cannot be assigned at period $i \in I$ if the tasks in $P(t)$ have not been assigned and finished:

$$\sum_{e\in E}\sum_{s\in\Omega^e}\beta^e_{its}x^e_s - \sum_{i'+l_{t'}\leq i}\sum_{e\in E}\sum_{s\in\Omega^e}\beta^e_{i't's}x^e_s \leq 0, \qquad \forall i \in I,\ \forall t \in T,\ \forall t' \in P(t) \neq \emptyset. \qquad (8)$$

Lequy et al. [12] propose another formulation for the precedence constraints that reduces the total number of constraints:

$$-Mu_t - \sum_{i\in I}\sum_{e\in E}\sum_{s\in\Omega^e} i\beta^e_{its}x^e_s + Mu_{t'} + \sum_{i\in I}\sum_{e\in E}\sum_{s\in\Omega^e}(i+l_{t'})\beta^e_{it's}x^e_s \leq 0, \quad \forall t \in T,\ t' \in P(t), \qquad (9)$$

where M is a sufficiently large integer.
In general, constraints (8) are more explicit, so they yield to better bounds than do constraints (9). However, the latter lead to a model with fewer constraints that can be solved more efficiently. We consider both formulations in our computational study.

3 Modeling Shifts with Grammars

To solve the above model directly, we need to know all the feasible shifts for each employee. Côté et al. [2, 4, 3] used a formal language based on a context-free grammar to represent these shifts.

3.1 Context-Free Grammar

A context-free grammar G is defined by the tuple $(\Sigma,\ N,\ P,\ S)$ where

- Σ is an alphabet that contains letters $(a, b, c, ...)$, also called terminal symbols;
- N is a set of nonterminal symbols $(A, B, C, ...)$;
- P is a set of productions of the form $X \rightarrow \alpha$, where $X \in N$ and α is a sequence of terminal and/or nonterminal symbols;
- S is the starting nonterminal.

A sequence of letters from the alphabet Σ, called a word, is recognized by the grammar G if it can be generated by the successive application of productions from P, starting from the starting nonterminal S. The set of words recognized by a grammar is called a language. A context-free grammar is in Chomsky normal form when all productions are of the form $X \rightarrow \alpha$ where $X \in N$ and $\alpha \in (N \times N) \cup \Sigma$. However, for clarity, we will not present the grammars in Chomsky normal form. This is not restrictive since any grammar can be converted to this form (see [10]).
For instance, a word can represent a sequence of activities, tasks, and breaks, which corresponds to a shift. Hence, the words recognized by the grammar are feasible shifts for an employee. The length of the word corresponds to the planning horizon considered.

Example 1 [3]: The following grammar G defines all feasible shifts for one employee and one activity. A shift has a length equal to the planning horizon and contains one break that cannot be placed at the beginning or end of the shift. Work and break periods are represented by w and b respectively:

$$G = (\Sigma = (j,\ b),\ N = (S,\ X,\ J,\ B), P, S) \text{ where } P \text{ is}$$
$$S \rightarrow XJ,\ X \rightarrow JB,\ J \rightarrow JJ \mid j,\ B \rightarrow b$$

The shifts jbj, $jjjjjbj$, and $jbjj$, among others, are recognized by G.

3.2 Directed Acyclic Graph

All the derivations that a grammar associates with a word of a given length n can be represented by the directed acyclic graph (DAG) Γ. This DAG has an and-or structure with two types of nodes:

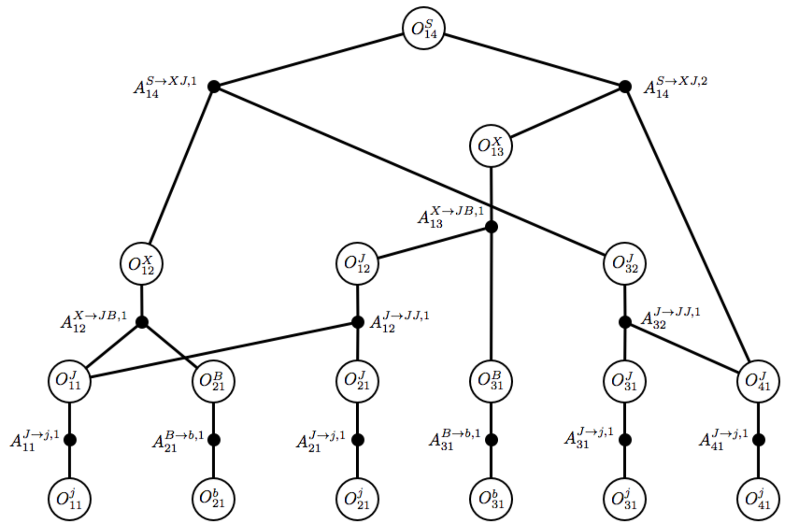

Figure 1: DAG Γ for grammar from Example 1 on word of length 4

- the or-nodes O represent the nonterminal symbols from N; and
- the and-nodes A represent the productions from P.

We denote by O^{π}_{il} the or-node associated with the nonterminal or letter π that generates a subsequence of length l from position i. Hence, if $\pi \in \Sigma$, the or-node is a leaf and $l = 1$, and the root is represented by O^S_{1n}. Likewise, $A^{\Pi,k}_{il}$ is the and-node associated with the production Π that generates a subsequence of length l from position i, with k representing the index of the subsequence. A DAG is built by a procedure suggested by Quimper and Walsh [15] that is inspired by an algorithm from Cooke, Younger, and Kasami (see [10]). Figure 1 shows the DAG for the grammar in Example 1 for a word of length 4.
To derive words from the DAG Γ, we start at the root O^S_{1n}. An or-node O^{π}_{il} is visited by selecting exactly one child, which is an and-node, and an and-node $A^{\Pi,t}_{il}$ is visited by selecting all its children. Γ is traversed in this way until the only unvisited nodes are leaves. The word formed by the remaining unvisited leaves defines a word recognized by the grammar. Likewise, starting from the leaves associated with a word w, we can traverse Γ backwards to check if this word belongs to the grammar.

3.3 Enriched Grammar

The productions of a grammar can be enriched in order to include more constraints in the derivation of a word and thus the construction of the associated DAG. These constraints appear in the SSP when some jobs must be done within a time window and/or there are restrictions on their minimum and maximum durations. The notation $A^{[tws,\ twe]}_{[lmin,\ lmax]} \xrightarrow{c} BC$ indicates that the subsequence generated from A should be spanned within the positions $[tws,\ twe]$, has a length between $lmin$ and $lmax$, and, if it is used, has a cost of c.

Example 2: The grammar G below defines the feasible shifts for one employee and a set of activities A. A shift has between 3 and 8 hours of activities, including breaks, and, if the employee works more than 6 hours, a lunch-break. The minimum duration of an activity is 1 hour. The planning horizon is divided into periods of 15 minutes:

$$\begin{array}{ll} S \to RPR \mid RFR & J_a \to a \mid aJ_a,\ for\ a \in A \\ P_{[13,24]} \to JbJ & L \to llll \\ F_{[30,38]} \to PLP & R \to r|rR \\ J_{[4,\infty]}^{[tws_a,twe_a]} \to J_a,\ for\ a \in A & \end{array}$$

where $[tws_a, twe_a]$ represents the time window associated with activity $a \in A$.

3.4 Productions of Tasks in Grammar

A task $t \in T$ is fully defined by its length l_t and its associated time window $[tws_t, twe_t]$. With the enriched grammar, we can define a task as the span of an activity with a fixed length and a time window. Therefore, the productions associated with the span of a task $t \in T$ can be defined as

$$\begin{array}{l} J_{[l_t,l_t]}^{[tws_t,twe_t]} \to J_t, \\ J_t \to t \mid tJ_t. \end{array}$$

Example 3: We consider a shift of four hours divided into periods of one hour with the start time of the shift fixed a priori. An employee can perform an activity a or a task t at each period. The length of task t is three hours. The associated grammar is defined by the following productions:

$$\begin{array}{ll} S_{[4,4]} \to J_a \mid J_aJ_a^n \mid J_tJ_t^n & J_a \to a \mid aJ_a, \\ J_t^n \to J_aJ_a^n \mid J_a & J_t' \to t \mid tJ_t' \\ J_n^a \to J_tJ_t^n \mid J_t & \\ J_{t\ [3,3]} \to J_t' & \end{array}$$

The corresponding DAG is given in Figure 2.
However, for the column generation approach, we need to identify in the grammar the beginning of a task so that the precedence constraints can be applied. With the previous productions involved in the span of a task, the corresponding terminals in the DAG represent one period of this task. To determine the starting period of a task $t \in T$, we use a different label in the DAG for the first period of the span, i.e., s_t. The productions are then rewritten as follows:

$$\begin{array}{l} J_{[l_t,l_t]}^{[tws_t,twe_t]} \to s_tJ_t \mid s_t, \\ J_t \to t \mid tJ_t. \end{array}$$

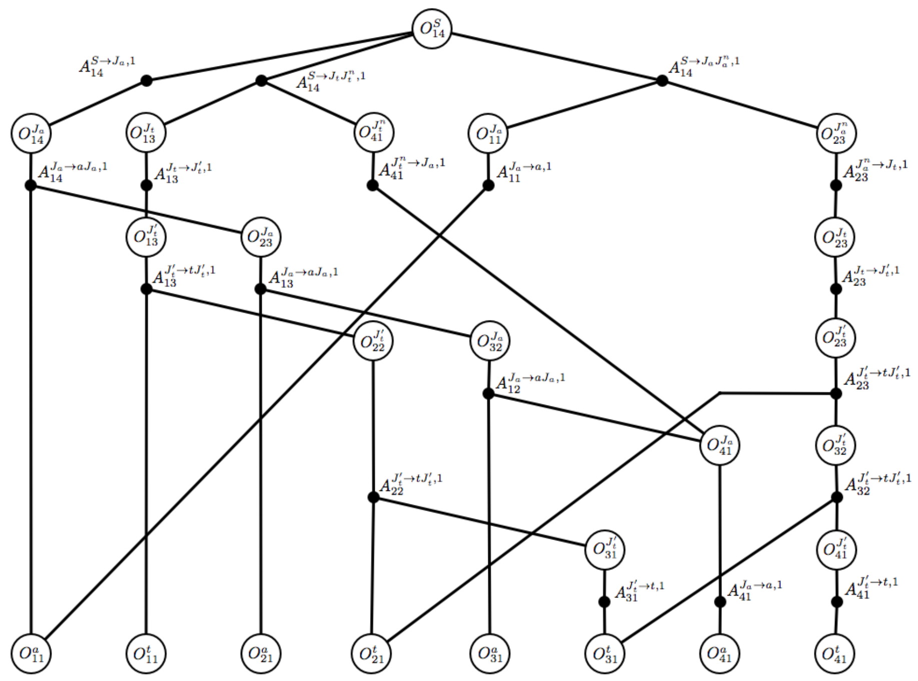

Figure 2: DAG Γ for grammar from Example 4

4 Grammar-Based Branch-and-Price Algorithm

We now present the grammar-based branch-and-price (B&P) algorithm to solve the multi-activity multi-task SSP. Enumerating all possible shifts for each employee and solving the model (D) directly leads to a problem with a large number of variables that is, in practice, too hard to solve. Therefore, at each iteration of the B&P we solve the linear relaxations of a sequence of restrictions of model (D), called the restricted master problems $(RMPs)$, via a column generation approach.

4.1 Restricted Master Problem

The RMP is defined by allowing only a subset of the feasible shifts $\tilde{\Omega}^e \subset \Omega^e$ for each employee $e \in E$, as follows:

$$f(RMP) = \min \sum_{e\in E}\sum_{s\in\tilde{\Omega}^e} c_s^e x_s^e + \sum_{i\in I}\sum_{a\in A}(c_{ia}^u u_{ia} + c_{ia}^o o_{ia}) + \sum_{t\in T} c_t^u u_t \tag{10}$$

$$\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e} \delta_{ias}^e x_s^e + u_{ia} - o_{ia} = b_{ia} \qquad \forall i\in I,\ a\in A \tag{11}$$

$$\sum_{i\in I}\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e} \beta_{its}^e x_s^e + u_t = 1 \qquad \forall t\in T \tag{12}$$

$$\sum_{s\in\tilde{\Omega}^e} x_s^e = 1 \qquad \forall e\in E \tag{13}$$

$$x_s^e \in \{0,1\} \qquad \forall e\in E,\ s\in\tilde{\Omega}^e \tag{14}$$

$$u_{ia}\ge 0,\ o_{ia}\ge 0 \qquad \forall i\in I,\ \forall a\in A \tag{15}$$

$$u_t \ge 0 \qquad \forall t\in T. \tag{16}$$

If there are precedence constraints, we also require constraints (8) or (9):

$$\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e} \beta_{its}^e x_s^e - \sum_{i'+l_{t'}\le i}\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e} \beta_{i't's}^e x_s^e \le 0, \qquad \forall i\in I,\ \forall t\in T,\ \forall t'\in P(t)\ne 0 \tag{17}$$

or

$$-Mu_t - \sum_{i\in I}\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e} i\beta_{its}^e x_s^e + Mu_{t'} + \sum_{i\in I}\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e}(i+l_{t'})\beta_{it's}^e x_s^e \le 0, \quad \forall t\in T,\ t'\in P(t). \tag{18}$$

At each iteration of the column generation method, we solve the current RMP and then for each employee we try to identify columns with negative reduced costs by solving a pricing subproblem. If no such columns are found, we have obtained the optimal solution of the linear relaxation of (D) by generating only a (typically small) subset of the feasible shifts.

4.2 Pricing Subproblem

To generate new columns for the RMP we use the DAG to represent the shifts that each employee can perform. For each employee $e\in E$, a grammar G^e is formulated according to the employee's skills and availability and the work regulations. The associated DAG^e is then created, and it is used to solve the pricing subproblem for employee e.
To compute the reduced cost of a column, we need the dual variables of the current RMP. Let

- λ_{ia}, $i\in I$, $a\in A$ be the dual variables associated with constraints (11);

- θ_t, $t \in T$ be the dual variables associated with constraints (12);
- σ^e, $e \in E$ be the dual variables associated with constraints (13).

We assume that the cost c_s^e of the shift $s \in \tilde{\Omega}^e$ for employee $e \in E$ can be decomposed as $c_s^e = \sum_{i \in I} \left(\sum_{a \in A} \delta_{ias}^e c_{ia}^e + \sum_{t \in T} \beta_{its}^e c_{it}^e \right)$ where c_{iw}^e, $w \in A \cup T$, is the cost for employee e to perform w at period i. The reduced cost of the shift s is then

$$\tilde{c}_s^e = \sum_{i \in I} \left(\sum_{a \in A} (c_{ia}^e - \lambda_{ia}) \delta_{ias}^e + \sum_{t \in T} (c_{it}^e - \lambda_t - r_{it}) \beta_{its}^e \right) - \sigma^e \quad \forall e \in E,\ s \in \Omega^e, \tag{19}$$

where, for $i \in I$, $t \in T$, r_{it} is the term generated by the precedence constraints.
If there are no precedence constraints, then r_{it}=0, for $i \in I$, $t \in T$. Otherwise,

- $r_{it} = \sum_{t' \in P(t)} \xi_{itt'} - \sum_{i' \geq i+l_t} \sum_{t' \in P^*(t)} \xi_{i't't}$ if the precedence constraints are represented by (17), where $\xi_{itt'}$, $i \in I$, $t \in T$, $t' \in P(t)$, are the associated dual variables;
- $r_{it} = -\sum_{t' \in P(t)} i\gamma_{tt'} + \sum_{t' \in P^*(t)} (i + l_t)\gamma_{t't}$ if the precedence constraints are represented by (18), where $\gamma_{tt'}$, $t \in T$, $t' \in P(t)$, are the associated dual variables;

with $P^*(t) = \{t' \in T \ / \ t \in P(t')\}$.
To solve the pricing subproblem for employee e, we associate a cost k_{iw} with each leaf of the DAG corresponding to an activity or task $w \in A \cup T$ performed at period $i \in I$. If $w = a \in A$, $k_{ia} = c_{ia}^e - \lambda_{ia}$, and if $w = t \in T$, $k_{it} = (c_{it}^e - \lambda_t - \Xi_{it})/l_t$. The other nodes of the graph have their costs initialized to zero. The subproblem is then solved by a dynamic programming algorithm proposed by Quimper and Rousseau [14] to find a minimum parse tree in a grammar-based DAG.
Starting from the leaves, we traverse the DAG and assign to the visited and-nodes the sum of the costs of their children and to the or-nodes the minimum of the costs of their children. Hence, every child of the root node of the DAG that has a negative cost represents a column with a negative reduced cost that can be added to the RMP. If no such child exists for any employee, the solution of the current RMP is the optimal solution of the linear relaxation, because no column with a negative reduced cost can be generated.
Note that the definition of the cost c_s^e can be extended because a cost can be assigned to some productions of the grammar as defined in Section 3.3. These costs could represent, for instance, the transition cost for an employee to switch from one activity or task to another. In the dynamic programming algorithm, the corresponding and-nodes would be initialized with these transition costs, which would be added to the total cost of the node.

4.3 Branching Strategy

Since solving the RMP at each node of the B&P gives a fractional solution, we must branch to derive an optimal integer solution. Our branching strategy is based on that proposed by Côté et al. [3], which was adapted from the strategy presented by Barnhart et al. [1] for solving integer multi-commodity flow problems.
At each node of the B&P, we choose from the fractional solution an employee $e' \in E$ with at least two assigned shifts that have associated variables with fractional values. If no such employee exists, then the solution is integer and the exploration of this node is complete. Otherwise, we select the two shifts $s^{e'}(1)$ and $s^{e'}(2)$ for which the corresponding variables have the largest fractional values. We compare these two shifts and find the first divergent position i', that is, the first period where they differ in terms of the job. Let $j(1) \in A \cup T$ and $j(2) \in A \cup T$ be the assigned jobs at period i' in shifts $s_1^{e'}$ and $s_2^{e'}$, respectively. Let $J_{i'}^{e'} \subset A \cup T$ be the subsets of activities and tasks that can be performed by employee e' at period i'. We create a partition of $J_{i'}^{e'}$ into two subsets $J_{i'}^{e'}(1)$ and $J_{i'}^{e'}(2)$ such that $j(l) \in J_{i'}^{e'}(l)$, for $l \in \{1, 2\}$. In practice the remaining activities and tasks in $J_{i'}^{e'} - \{j(1),\ j(2)\}$ are equally distributed between the two partitions. Finally, we generate two nodes where each ensures that the employee e' does not perform a job in $J_{i'}^{e'}(l)$ at period i', for $l \in \{1, 2\}$.
This rule can be easily handled when solving the RMP and the pricing subproblems. Indeed, it suffices to assign a large value to the cost $c_{i'w}^{e'}$ where w is a forbidden job in $J_{i'}^{e'}(l)$ for $l \in \{1,\ 2\}$. This ensures that the corresponding reduced cost is positive and the corresponding shifts are never selected by the dynamic programming algorithm.
In Côté et al. [3], branching is performed by selecting the first employee e' with a fractional value. We have improved this approach by choosing $s^{e'}(1)$ such that its corresponding variable has the largest fractional value (closest to 1). This tends to improve the convergence of the algorithm toward an integer solution. We call this strategy **S1**.
In the multi-task case, we can enhance this strategy by preventing some tasks from starting at certain periods. The goal is to first find a pattern for the position of the tasks and then place the activities. We define a two-step branching strategy where

- in the first step, we try to branch on the starting time of a task; and
- in the second step, if branching on the tasks is not possible, we apply the branching strategy to the activities as in S1.

In the fractional solution of the RMP, we identify two shifts where the same task $t \in T$ is assigned but it starts at different periods, $i(1)$ and $i(2)$. Without loss of generality, consider $i(1) < i(2)$. We then create two nodes where in the first we do not allow task t to start before $i(2)$, and in the second we do not allow task t to start after $i(2)$. This branching ensures that these two shifts do not appear in the same solution. Moreover, by reducing the time window where a task can be performed, we try to position the tasks before assigning the activities. We call this strategy **S2**.
This last branching strategy tries to reduce the time window associated with the tasks. Because the placement of the tasks can have a significant impact on the objective, this

strategy can reduce the quality of the solution. Hence, we propose a third branching strategy, **S3**, that is a hybrid of S1 and S2. It applies the first branching strategy, S1, but gives priority to shifts containing a task. Specifically, we try to find a shift $s^{e'}(1)$, $e' \in E$, such that its corresponding variable has the largest fractional value and it contains at least one task. If no such shift exists, then we apply S1. Otherwise, we select the shift $s^{e'}(2) \neq s^{e'}(1)$ with the second highest fractional value and we use the same separation as in S1.

5 Preprocessing of Precedence Constraints

Precedence constraints have a significant impact on the difficulty of the problem. In some cases, a quick analysis of these constraints allows us to reduce the time window associated with a task and to eliminate some of them. For a task $t \in T$, let s_t and e_t be the start and end time of the time window associated with t. A simple two-step preprocessing can be implemented:

- In step 1, we try to reduce the time window of the tasks, i.e., for $t \in T$ and $t' \in P(t)$, $s_t = max\{s_{t'} + l_{t'},\ s_t\}$. This is because t' must be performed before t and so t cannot start in $[s_{t'}, s_{t'} + l_{t'}]$.
- In step 2, we check whether some precedence constraints are always satisfied. That is, for $t \in T$ and $t' \in P(t)$, if $e_{t'} < s_t$ then $P(t) = P(t) - t'$, since the two intervals are disjoint.

We repeat the preprocessing until no time window is reduced. The reduction of the time windows associated with the tasks decreases the number of nodes involved in the DAG and therefore the time required to solve the pricing subproblem. The preprocessing can also reduce the number of constraints in the model.

6 Computational Experiments

We now present the results obtained with the grammar-based B&P algorithm for the multi-activity multi-task SSP. We consider the cases with and without pre-assignment of shifts to employees. Furthermore, we compare the different formulations of the precedence constraints and the different branching strategies.
The experiments were performed sequentially on a two-processor quad-core Intel Xeon 2.4 GHz. The restricted master problem was solved by CPLEX 12 with the dual method. The B&P algorithm was implemented in C++ using the OOBB framework from Crainic et al. [5]. The algorithm stops when the optimality gap is less than 1% or when the processing time for an instance reaches two hours.
Four different strategies are tested in this section. First, we consider the two formulations of the precedence constraints described in Section 2.2. **C1** is the case where constraints (8) are used and **C2** is the case where constraints (9) are used. Secondly, we test the three different branching strategies of Section 4.3, i.e., S1, where branching is performed only

on the activities, S2, where the branching is performed on the tasks and the activities, and S3, the hybrid strategy.
For all the instances, to obtain an initial bound for the B&P algorithm, we use a diving strategy to try to construct an initial integer solution from the restricted master problem of the root node. This strategy repeatedly fixes to one the variable with fractional value closest to one until a feasible solution is found. We consider only the columns added by the column generation at the root, and the maximum processing time is thirty minutes. In some cases, this initial solution has an optimality gap lower than 1% and it is then not necessary to run the B&P.

6.1 The Instances

We consider that an employee can perform a shift of four hours or a shift of eight hours with a break of one hour for lunch. The hours of work are from 6 a.m. to 7 p.m. When the shifts are preassigned, we know a priori the length and the start time of the shifts assigned to an employee, and we have to decide which activities and tasks he/she will perform. When the shifts are not preassigned, an employee can perform a shift of four or eight hours within the time window.
The instances are generated as follows. We start by generating a feasible schedule for each employee. From this schedule, we derive the associated demand profile, and we randomly add or remove demand in each time period to generate undercovering and overcovering. We also derive a set of precedence constraints satisfied by this schedule. The time windows for each task are then generated such that they are centered on the start time of the corresponding task in the generated schedule. We thus ensure that the precedence constraints are feasible. We consider that an employee has the necessary skills to perform half of the set of activities and tasks.
Furthermore, the minimum length of an activity is thirty minutes and its maximum length is four hours. We minimize the number of transitions in a shift via a penalty c_{tr}. We denote by A^e and T^e, respectively, the subset of activities and tasks that can be performed by employee $e \in E$ and by $[wmin_t, wmax_t]$ the time window associated with a task $t \in T$.

6.2 Instances with Preassigned Shifts

For the instances with preassigned shifts, we know *a priori* the start and end times of the shifts for each employee. We consider a planning horizon of 1 week, which is divided into periods of 15 minutes (672 periods in total) with 5 activities and 50 tasks. The shifts are assigned such that no employee works more than 40 hours. Two sets of instances have been generated, one with 20 employees and another with 50 employees.
For employee $e \in E$, we consider a grammar for each assigned shift. These grammars are represented by the following productions:

- $S_{\ [16,16]} \xrightarrow{c_{tr}} J_j J_j^n \mid J_j,\ \forall j \in A_e \cup T_e$;
- $J_{a\ [2,16]} \rightarrow J'_a,\ \forall\ a \in A_e$;

- $J'_a \rightarrow a \mid aJ'_a,\ if\ a \in A_e;$
- $J_t \, {}^{[tws_t, twe_t]}_{[l_t, l_t]} \rightarrow s_t J'_t,\ \forall\ t \in T_e;$
- $J'_t \rightarrow t \mid tJ'_t,\ \forall\ t \in T_e;$
- $J^n_j \rightarrow J_{j'},\ \forall j \in A_e \cup T_e,\ \forall j' \in A_e \cup T_e - \{j\};$
- $J^n_j \xrightarrow{c_{tr}} J_{j'} J^n_{j'},\ \forall j \in A_e \cup T_e,\ \forall j' \in A_e \cup T_e - \{j\}.$

6.2.1 Solution at the Root

Tables 1 and 2 give the results obtained by solving the RMP at the root node using a diving strategy. The results show that, for C1, the solution provided by the root node is generally close to optimality. However, for C2, in one case no feasible solution was found within the allowed processing time, and the final gaps are generally larger than for C1.

	C1		C2	
Instance	gap (%)	time (s)	gap (%)	time (s)
PF_20_0	3.39	95.54	-	1800.00
PF_20_1	2.53	79.17	2.53	87.05
PF_20_2	35.96	102.68	23.52	90.58
PF_20_3	0.00	187.86	22.90	174.37
PF_20_4	3.85	132.28	3.96	81.77
PF_20_5	7.42	121.26	25.08	98.08
PF_20_6	3.10	102.09	7.49	113.84
PF_20_7	0.00	93.80	0.00	107.03
PF_20_8	2.43	95.09	15.66	98.51
PF_20_9	8.73	86.23	13.11	74.94
Average	6.74	109.61	12.69*	102.91*

Table 1: Solution at root with 20 employees and preassigned shifts
* Average values do not include instance PF_20_0

6.2.2 Branch and Price

The B&P is initialized with the bound obtained at the root. Tables 3 and 4 give the optimality gaps obtained with the different strategies. When the average optimality gap was already small at the root (< 5%), it has been only slightly improved, but when the average optimality gap at the root was large ($\geq$ 5%), the improvement is generally significant.
The best results are obtained for formulation C1, and the three tested branching strategies give similar results. Using formulation C1, five of the 20 instances are solved within

	C1		C2	
Instance	gap (%)	time (s)	gap (%)	time (s)
PF_50_0	0.86	203.57	38.20	313.42
PF_50_1	13.73	244.54	17.84	345.62
PF_50_2	6.31	150.91	17.34	271.11
PF_50_3	0.08	172.90	27.20	238.51
PF_50_4	17.48	204.43	27.49	173.23
PF_50_5	6.00	157.10	0.85	193.42
PF_50_6	27.24	247.54	64.42	549.77
PF_50_7	3.10	173.84	13.13	214.22
PF_50_8	2.88	243.40	37.95	282.23
PF_50_9	9.63	321.62	35.94	295.28
Average	8.73	211.98	28.04	287.68

Table 2: Solution at root with 50 employees and preassigned shifts

Instance	C1+S1	C1+S2	C1+S3	C2+S1	C2+S2	C2+S3
PF_20_0	3.18	3.18	3.18	7.18	7.18	7.18
PF_20_1	2.45	2.45	2.45	2.45	2.45	2.45
PF_20_2	4.25	4.25	4.25	4.41	4.41	4.41
PF_20_3	0.00	0.00	0.00	1.60	1.60	1.60
PF_20_4	3.85	3.85	3.85	3.87	3.87	3.87
PF_20_5	2.97	2.79	2.84	3.10	2.84	2.93
PF_20_6	3.10	3.10	3.10	3.11	3.11	3.11
PF_20_7	0.00	0.00	0.00	0.00	0.00	0.00
PF_20_8	2.25	2.34	2.25	2.57	2.52	2.61
PF_20_9	3.94	4.17	3.94	3.94	3.90	3.94
Average	2.60	2.62	2.59	3.22	3.18	3.21

Table 3: Optimality gap (%) with 20 employees and preassigned shifts

Instance	C1+S1	C1+S2	C1+S3	C2+S1	C2+S2	C2+S3
PF_50_0	0.70	0.70	0.70	2.31	2.31	2.31
PF_50_1	10.01	10.01	9.97	10.24	10.24	10.24
PF_50_2	1.57	1.57	1.57	1.66	1.58	1.58
PF_50_3	0.12	0.12	0.12	0.94	0.94	0.94
PF_50_4	2.43	2.42	2.43	4.57	4.57	4.57
PF_50_5	0.98	0.98	0.98	0.85	0.85	0.85
PF_50_6	7.64	7.64	7.64	8.71	8.71	8.71
PF_50_7	3.01	3.01	3.01	4.38	4.38	4.38
PF_50_8	2.75	2.75	2.75	2.75	2.75	2.75
PF_50_9	5.11	5.11	5.11	5.96	6.08	5.68
Average	3.43	3.43	3.43	4.24	4.24	4.20

Table 4: Optimality gap (%) with 50 employees and preassigned shifts

the time limit (2 hours) with an optimality gap lower than 1%. The remaining instances, except three, are solved with an optimality gap lower than 5%.

6.3 Instances Without Preassigned Shifts

For the instances without preassigned shifts, we do not know *a priori* when the shifts of each employee start and end. An employee can work between 6 a.m. and 7 p.m. and perform a shift of 4 or 8 hours with a break of 30 minutes. Since these instances are more difficult, we consider a planning horizon of 1 day divided into periods of 15 minutes (96 periods in total) with 5 activities and 5 tasks. Two sets of instances have been generated, one with 20 employees and one with 50 employees. Tables 7 and 8 give the results. For each employee $e \in E$, we build a grammar corresponding to the possible shifts of the workday. To represent the periods when the employee is not working, we create an artificial activity r. The grammar is described by the following productions:

- $S_{\ [52,52]} \rightarrow RPR \mid RFR \mid PR \mid RP \mid FR \mid RF$;
- $P_{\ [16,16]} \xrightarrow{c_{tr}} J_j J_j^n \mid J_j, \ \forall j \in A_e \cup T_e$;
- $F_{\ [34,34]} \rightarrow PLP$;
- $J_{a\ [2,16]} \rightarrow J'_a, \ \forall\ a \in A_e$;
- $J'_a \rightarrow a \mid aJ'_a, \ \forall a \in A_e$;
- $J_t{}^{[tws_t, twe_t]}_{[l_t, l_t]} \rightarrow s_t J'_t, \ \forall\ t \in T_e$;
- $J'_t \rightarrow t \mid tJ'_t, \ \forall\ t \in T_e$;
- $J_j^n \rightarrow J_{j'}, \ \forall j \in A_e \cup T_e, \ \forall j' \in A_e \cup T_e - \{j\}$;

- $J_j^n \xrightarrow{C_{tr}} J_{j'} J_{j'}^n, \ \forall j \in A_e \cup T_e, \ \forall j' \in A_e \cup T_e - \{j\}$;
- $R \ \rightarrow r|rR$;
- $L_{\ [2,2]} \rightarrow r|rR$.

6.3.1 Solution at the Root

Tables 5 and 6 give the optimality gaps obtained by solving the RMP for the root node using a diving strategy. In contrast to the previous instances, we can see an important optimality gap for each formulation. This shows that these instances are harder: it is more difficult to derive a good feasible solution.

	C1		C2	
Instance	gap (%)	time (s)	gap (%)	time (s)
PV_20_0	73.32	41.82	52.71	40.38
PV_20_1	0.00	34.98	0.00	27.64
PV_20_2	23.92	76.25	43.94	47.50
PV_20_3	27.05	59.64	27.23	72.01
PV_20_4	52.09	45.77	35.11	29.03
PV_20_5	24.26	64.88	42.64	62.12
PV_20_6	21.50	40.36	28.75	31.47
PV_20_7	55.55	44.28	61.98	86.47
PV_20_8	54.92	38.08	48.12	48.16
PV_20_9	0.03	19.61	34.83	28.43
Average	33.26	46.57	37.53	47.32

Table 5: Solution at root with 20 employees and without preassigned shifts

	C1		C2	
Instance	gap (%)	time (s)	gap (%)	time (s)
PV_50_0	67.88	168.68	72.51	38.93
PV_50_1	64.79	143.84	66.20	139.56
PV_50_2	69.71	208.12	70.50	315.94
PV_50_3	68.82	127.62	77.23	178.71
PV_50_4	67.03	223.51	72.36	228.00
PV_50_5	54.92	173.53	69.57	233.83
PV_50_6	69.00	279.66	77.17	236.65
PV_50_7	68.87	149.30	79.64	142.08
PV_50_8	69.00	182.54	77.22	143.63
PV_50_9	64.34	149.36	66.05	256.86
Average	66.44	180.62	72.84	191.50

Table 6: Solution at root with 50 employees and without preassigned shifts

6.3.2 Branch and Price

Tables 7 and 8 give the optimality gaps obtained with the B&P approach on the instances without preassigned shifts. All the instances with 20 employees, except for PV_20_7 and PV_20_8, are solved with an optimality gap lower than 1% in less than 8 minutes. With 50 employees, 6 of the 10 instances are solved exactly in less than 2 hours. The best average gap for all the strategies is less than 4%. On average, the best solutions are obtained with C2+S3 for the set of 20 employees and with C1+S1 for the set of 50 employees. However, for the instances with 50 employees, C1+S3 provides better solutions than C1+S1 except in one case, PV_50_9, which significantly increased the average gap. We also note that formulation C1 shows a better behavior on the largest instances than formulation C2, which seems to have more difficulty in converging toward a good feasible solution.

Instance	C1+S1	C1+S2	C1+S3	C2+S1	C2+S2	C2+S3
PV_20_0	0.09	0.09	0.09	0.93	0.93	0.93
PV_20_1	0.00	0.00	0.00	0.00	0.00	0.00
PV_20_2	0.00	0.00	0.00	0.00	0.00	0.00
PV_20_3	0.08	16.03	0.08	0.08	0.08	0.08
PV_20_4	0.06	0.06	0.06	0.07	0.07	0.07
PV_20_5	0.06	0.00	0.26	0.06	0.06	0.06
PV_20_6	0.00	0.00	0.00	0.00	0.00	0.00
PV_20_7	8.31	8.47	8.39	39.73	10.44	10.68
PV_20_8	22.46	11.95	22.46	23.36	23.36	12.97
PV_20_9	0.03	0.03	0.03	0.03	0.03	0.03
Average	3.11	3.66	3.14	6.43	3.50	2.48

Table 7: Optimality gap (%) with 20 employees and without preassigned shifts

Instance	C1+S1	C1+S2	C1+S3	C2+S1	C2+S2	C2+S3
PV_50_0	2.64	3.17	2.53	2.53	13.47	2.53
PV_50_1	6.43	6.32	6.32	6.32	6.32	6.32
PV_50_2	4.72	59.70	4.72	10.72	53.14	53.14
PV_50_3	0.00	0.00	0.00	65.07	65.07	65.07
PV_50_4	0.00	0.00	0.00	0.00	0.00	0.00
PV_50_5	0.00	54.81	0.00	54.40	54.40	54.40
PV_50_6	14.39	74.66	13.90	23.09	72.43	72.43
PV_50_7	0.00	74.36	0.00	0.00	75.58	75.58
PV_50_8	0.00	0.00	0.00	14.59	0.00	0.00
PV_50_9	0.00	0.00	10.09	5.07	63.22	63.22
Average	2.82	27.30	3.76	35.03	40.36	39.27

Table 8: Optimality gap (%) with 50 employees and without preassigned shifts

6.4 Comparison of Different Strategies

From these results, we see that formulation C1 tends to provide the best solutions. In particular, for the instances with 50 employees and without preassigned shifts, C2 failed to find good integer solutions. C2 provides a model with fewer constraints for which the linear relaxation is generally easier to solve, but the relatively important optimality gap penalizes the solution process of the B&P algorithm.
The branching strategies S1, S2, and S3 gave similar results. If we focus on formulation C1 which provides better results in general, strategy S1 seems to lead to the best gap on average, in particular for the instances without preassigned shifts. Strategy S2 imposes a decision for all employees, in contrast to S1 and S3, which consider one employee at each branching step. The S2 branching decision is stronger because it tries to force the placement of a task. It appears from our results that the solution quality is quite sensitive to this placement: in some cases (see for instance PV_20_3 and PV_50_2) it degrades the convergence toward a good solution. Hence, S1 and S3 appear to be more stable strategies.
Our B&P algorithm finds integer solutions for the SSP and gives an approximate optimality gap. The convergence toward an integer solution is relatively fast, but the symmetries of the problem lead to numerous equivalent solutions and make it difficult to prove optimality. These symmetries arise because many employees, although they have different skills, can perform the same job in the same period. Thus, the same shift can be assigned to many employees.
The column generation allows us to solve large instances. The grammar formulation of the feasible shifts for each employee can include many complex rules generally involved in SSP, such as time windows for the tasks, employee skills, the placement of breaks, and the length of an uninterrupted working period.

7 Conclusion

We have presented a set covering model for the shift scheduling problem with multiple activities and multiple tasks, together with two formulations of the constraints on the sequence of execution of the tasks. We have considered the personalized case, i.e., each employee has different availability and skills. This model is solved via a branch and price algorithm using a grammar-based column generation for the solution of the restricted master problem. This restricted problem contains only a subset of the feasible shifts for each employee.
We have shown that the grammar can be used efficiently to model feasible shifts with complex constraints such as time windows for the tasks, employee skills, the placement of breaks, and the length of an uninterrupted working period. The resulting grammar can be used to find shifts with the most negative reduced costs and hence to select nodes to add at each iteration of the column generation. Grammars are convenient because they can be expressed via simple rules that are used to construct the associated decision tree used in our algorithm.

We have presented results for instances inspired by real cases with two formulations of the precedence constraints and three branching strategies. We considered instances where the shifts are preassigned, with a time horizon of one week, and where they are not preassigned, with a time horizon of one day. In the former case, we know exactly when the employee will work and we must assign the set of activities and tasks that he/she will perform. In the latter case, we know only the availability of the employee and we must decide the start and finish times. We considered that an employee can be assigned to a shift of four hours or to a shift of eight hours with one hour for lunch.
The results showed that the B&P algorithm can find an integer solution for all the instances within two hours with an optimality gap lower than 5% in the best case. These problems contained up to 50 employees and the column generation allows us to solve large instances without overflow. We also compared three different branching strategies and two formulations for the precedence constraints.
Many different branching strategies could be implemented. Our tests showed that the strategies presented in this paper gave good results. However, one could consider the choice of the employee to branch on or the choice of the activities to forbid. Furthermore, our B&P approach could be improved with the use of a heuristic to compute upper bounds at each node. This heuristic should handle the precedence constraints and the demand for tasks, because the objective value is sensitive to their placement. One approach could be a local search on the columns added so far. Such a heuristic could improve the convergence of our method and hence reduce the overall number of nodes that must be explored.

References

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

[2] Marie-Claude Côté, Bernard Gendron, Claude-Guy Quimper, and Louis-Martin Rousseau. Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16(1):54–76, 2011.

[3] Marie-Claude Côté, Bernard Gendron, and Louis-Martin Rousseau. Grammar-based integer programming models for multiactivity shift scheduling. *Management Sciences*, 57(1):151–163, 2011.

[4] Marie-Claude Côté, Bernard Gendron, and Louis-Martin Rousseau. Grammar-based column generation for personalized multi- activity shift scheduling. *INFORMS Journal of Computing*, forthcoming.

[5] Teodor Gabriel Crainic, Antonio Frangioni, Bernard Gendron, and François Guertin. OOBB: An object-oriented library for parallel branch-and-bound. In *CORS/INFORMS International Conference, Toronto, Canada*, 2009.

[6] George 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.

[7] Sophie Demassey, Gilles Pesant, and Louis-Martin Rousseau. A cost-regular based hybrid column generation approach. *Constraints*, 11(4):315–333, 2006.

[8] Andreas T. Ernst, Houyuan Jiang, Mohan Krishnamoorthy, Bowie Owens, and David Sier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127:21–144, 2004.

[9] Andreas T. Ernst, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[10] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computability.* Addison-Welsey, 2001.

[11] Quentin Lequy, Mathieu Bouchard, Guy Desaulniers, François Soumis, and Beyime Tachefine. Assigning multiple activities to work shifts. *Journal of Scheduling*, 2010.

[12] Quentin Lequy, Guy Desaulniers, and Marius M. Solomon. Assigning team tasks and multiple activities to fixed work shifts. *Cahiers du GERAD*, (G-2010-71), 2010.

[13] Quentin Lequy, Guy Desaulniers, and Marius M. Solomon. A two-stage heuristic for multi-activity and task assignment to work shifts. *Cahiers du GERAD*, (G-2010-28), 2010.

[14] Claude-Guy Quimper and Louis-Martin Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–392, 2009.

[15] Claude-Guy Quimper and Toby Walsh. Decomposing global grammar constraints. In *Principles and Practice of Constraint Programming – CP 2007*, volume 4741 of *Lecture Notes in Computer Science*, pages 590–604. 2007.