J Sched (2014) 17:185–197
DOI 10.1007/s10951-013-0338-9

A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem

Vincent Boyer · Bernard Gendron · Louis-Martin Rousseau

Received: 13 April 2012 / Accepted: 27 May 2013 / Published online: 12 June 2013

Abstract The multi-activity multi-task shift scheduling problem requires the assignment of 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 a 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

V. Boyer · B. Gendron · L.-M. Rousseau
Centre interuniversitaire de recherche sur les réseaux d'Entreprise, la logistique et le transport, Université de Montréal, C.P. 6128, succursale Centre-ville, Montreal, QC H3C 3J7, Canada

B. Gendron
e-mail: bernard.gendron@cirrelt.ca

L.-M. Rousseau
e-mail: louis-martin.rousseau@cirrelt.ca

Present Address:
V. Boyer (✉)
Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León (UANL), 66451 San Nicolás de los Garza, Nuevo Leon, Mexico
e-mail: vincent.a.l.boyer@gmail.com

1 Introduction

Shift scheduling requires the assignment of 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 and 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 undercovering of this demand reflects 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. (2004a,b) give many references, but few of them consider simultaneous activity and task assignment.

Demassey et al. (2006) study the multi-activity case for a 24-h planning horizon with up to ten activities. They use a set covering formulation and apply 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. (2010a) present 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 propose rolling-horizon heuristics based on column generation for the largest instances. Lequy et al. (2010b,c) extend their approach to the multi-task case and consider 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. (2011a, 2011b, 2012). They propose a grammar-based model for the personalized multi-activity shift scheduling problem and a branch-and-price (B&P) approach with column generation to solve this problem. The results show that this approach can handle instances with up to 100 employees and 15 activities for a planning horizon of 7 days when the working periods are preassigned to employees. We extend this approach to include tasks with precedence constraints, and we present an extensive study of branching strategies. Furthermore, we present results for instances where the starting and ending times of a shift are not known a priori.

In the literature, column generation Desaulniers et al. (2005) and B&P Barnhart et al. (1998) approaches have been used with success to solve complex problems [see for instance Akker et al. (1999), Barnhart et al. (2000), Contreras et al. (2011), Tang et al. (2007), Tang et al. (2011)]. As a result, many different models have been proposed. As shown by Côté et al. (2011a), the flexibility provided by context-free grammars allows to encode all work regulation rules and simplifies the modelling of multi-activity shift scheduling problems. Furthermore, the good results obtained by these authors show the efficiency of this approach for solving complex scheduling problems. To the best of our knowledge, the present paper along with Lequy et al. (2010c) represent the first attempts at solving the multi-activity and multi-task case. However, Lequy et al. present a heuristic method for the case where the shifts already exist, while we propose an exact approach that does not require the preexistence of the shifts.

This paper is organized as follows. In Sect. 2, we present our mathematical model for the personalized multi-activity multi-task shift scheduling problem. In Sect. 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 B&P algorithm. Finally, in Sect. 5, we present computational results for a set of instances inspired by real cases.

2 The shift scheduling problem (SSP)

In the personalized multi-activity multi-task SSP, we must assign to each employee $e \in E$ a feasible shift $s \in \Omega^e$, where Ω^e is the set of shifts that can be assigned to the employee 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 use the term *job* to refer to either an activity or a task.

We consider that the planning horizon is divided into periods of equal length. In this context, a shift is a sequence of jobs corresponding to a continuous presence at work (that may include lunch and breaks). This definition of a shift is illustrated in Example 1.

Example 1 We consider a planning horizon divided into periods of 1 h. We denote by a and b two different activities and by l a lunch period. Then the shift s=*aaaalbbbb* represents 4 h of activity a, followed by 1 h of lunch l, followed by 4 h of activity b.

With each shift $s \in \Omega^e$ we associate a cost $c_s^e \geq 0$. 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_{ia}^u and c_{ia}^o be the costs for undercovering and overcovering, respectively, an activity $a \in A$ at period $i \in I$ and let c_t^u 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, noted (D), is an extension of that of Côté et al. (2011b), who used a classical set-covering formulation introduced by Dantzig (1954) for the shift scheduling problem:

$$f(D) = \min \sum_{e\in E} \sum_{s\in\Omega^e} c_s^e x_s^e + \sum_{i\in I} \sum_{a\in A} \left(c_{ia}^u u_{ia} + c_{ia}^o o_{ia}\right) + \sum_{t\in T} c_t^u u_t \tag{1}$$

subject to :

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

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

$$\sum_{s\in\Omega^e} x_s^e = 1 \qquad \forall e \in E \qquad (4)$$

$$x_s^e \in \{0,\ 1\} \qquad \forall e \in E,\ s \in \Omega^e \qquad (5)$$

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

$$u_t \geq 0 \qquad \forall t \in T, \qquad (7)$$

where:

- $\delta_{ias}^e = 1$ if activity $a \in A$ is **assigned** at period $i \in I$ in shift $s \in \Omega^e$ for employee $e \in E$, and $\delta_{ias}^e = 0$ otherwise;
- $\beta_{its}^e = 1$ if task $t \in T$ **starts** at period $i \in I$ in shift $s \in \Omega^e$ for employee $e \in E$, and $\beta_{its}^e = 0$ otherwise;
- $x_s^e = 1$ if employee $e \in E$ is assigned to shift $s \in \Omega^e$, and $x_s^e = 0$ 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) insures 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:

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

Lequy et al. (2010b) propose another formulation for the precedence constraints that reduces the total number of constraints:

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

where M is a sufficiently large integer.

In general, constraints (8) are more explicit, so they yield better bounds than 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. (2011a,b, 2012) use 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 (Σ, 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 ($A, B, C, \ldots$);
- 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 $\alpha \in (N \times N) \cup \Sigma$. Any grammar can be converted to this form (see Hopcroft et al. (2001)). However, for the sake of clarity, we will not present the grammars in Chomsky normal form.

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 2 (see Côté et al. (2011b)): 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 j and b respectively:

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

In the grammar, the symbol | specifies a choice of production. The non-terminal J and B represent, respectively, a sequence of jobs and a sequence of breaks.

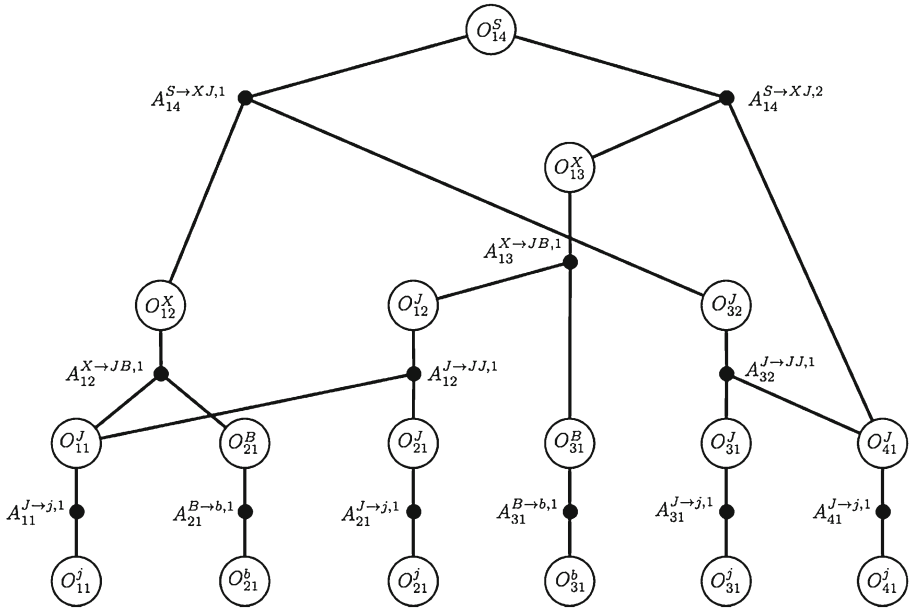

Fig. 1 DAG Γ for grammar from Example 2 on word of length 4

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:

- 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 (2007) that is inspired by an algorithm from Cooke, Younger, and Kasami (see Hopcroft et al. (2001)). Figure 1 shows the DAG for the grammar in Example 2 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 visited leaves define 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 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 3 The grammar G below defines the feasible shifts for one employee and a set of activities A. A shift has between 3 and 8 h of activities, including breaks b of 15 min, and, a lunch-break l if the employee works more than 6 h. The minimum duration of an activity is 1 h. The planning horizon is divided into periods of 15 min:

$$
\begin{array}{ll}
S \to RPR \mid RFR & J_a \to a \mid aJ_a,\ \forall\, a \in A \\
P_{[12,24]} \to JbJ & L \to llll \\
F_{[28,36]} \to PLP & R \to r|rR \\
J^{[tws_a,twe_a]}_{[4,\infty]} \to J_a,\ \forall\, a \in A &
\end{array}
$$

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

In this grammar, the non terminal P represents a sequence of jobs of 3–6 h including a 15-min break, and the nonterminal F represents a sequence of jobs of 6–8 h, plus a lunch L of 1 h. Furthermore, as the starting time of the shift is unknown, we introduce the artificial activity r, spanned from the nonterminal R, to represent the rest periods before and after a shift.

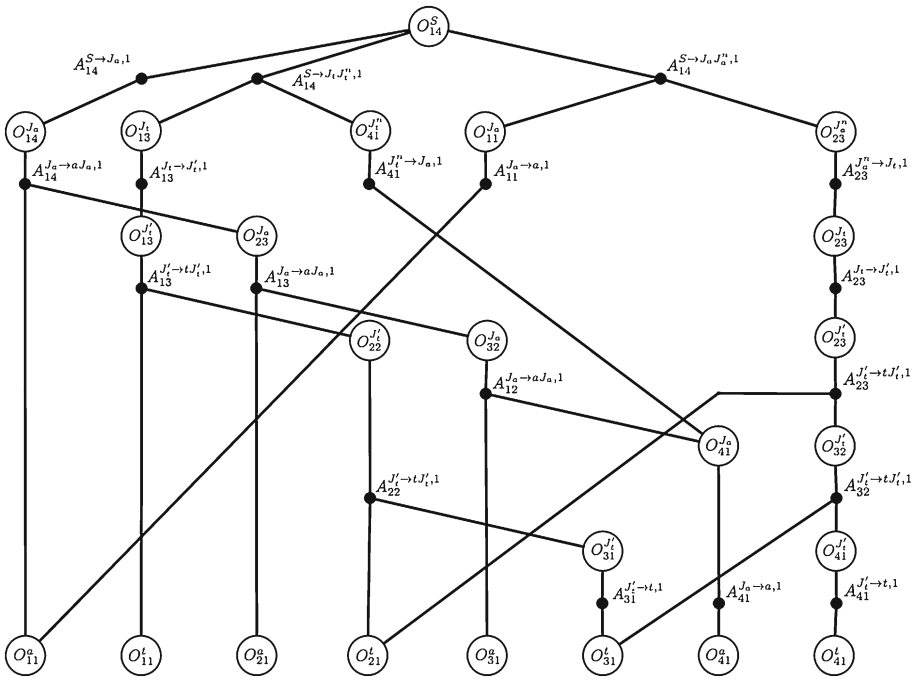

Fig. 2 DAG Γ for grammar from Example 4

3.4 Productions of tasks

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

$$J_t \, {}^{[tws_t, twe_t]}_{[l_t, l_t]} \rightarrow J'_t,$$
$$J'_t \rightarrow t \mid t J'_t.$$

As we have constraints on the length of the task t, i.e., exactly l_t, the nonterminal J'_t is used in order to introduce this rule in the production of the task.

Example 4 We consider a shift of 4 h divided into periods of 1 h 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 3 h. The associated grammar is defined by the following productions:

$$S_{[4,4]} \rightarrow J_a \mid J_a J_{\bar{a}} \mid J_t J_{\bar{t}} \qquad J_a \rightarrow a \mid a J_a,$$
$$J_{\bar{t}} \rightarrow J_a J_{\bar{a}} \mid J_a \qquad J'_t \rightarrow t \mid t J'_t$$
$$J_{\bar{a}} \rightarrow J_t J_{\bar{t}} \mid J_t$$
$$J_{t\ [3,3]} \rightarrow J'_t$$

In this grammar, for $j \in \{a, t\}$, the nonterminal J_j represents a span of the job j, and $J_{\bar{j}}$, a span of a sequence of jobs not starting with the job j. This representation allows to model the transition between the jobs. The corresponding DAG is given in Fig. 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:

$$J_t \, {}^{[tws_t, twe_t]}_{[l_t, l_t]} \rightarrow s_t J'_t \mid s_t,$$
$$J'_t \rightarrow t \mid t J'_t.$$

4 Grammar-based B&P algorithm

We now present the grammar-based 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 algorithm 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:

$$f(RMP) = \min \sum_{e \in E} \sum_{s \in \tilde{\Omega}^e} c^e_s x^e_s + \sum_{i \in I} \sum_{a \in A} \left(c^u_{ia} u_{ia} + c^o_{ia} o_{ia} \right) + \sum_{t \in T} c^u_t u_t \quad (10)$$

subject to :

$$\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e}\delta^e_{ias}x^e_s + u_{ia} - o_{ia} = b_{ia} \quad \forall i \in I,\ a \in A \tag{11}$$

$$\sum_{i\in I}\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e}\beta^e_{its}x^e_s + u_t = 1 \quad \forall t \in T \tag{12}$$

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

$$x^e_s \geq 0 \quad \forall e \in E,\ s \in \tilde{\Omega}^e \tag{14}$$

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

$$u_t \geq 0 \quad \forall t \in T. \tag{16}$$

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

$$\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e}\beta^e_{its}x^e_s - \sum_{i'+l_{t'}\leq i}\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e}\beta^e_{i't's}x^e_s \leq 0,$$
$$\forall i \in I,\ \forall t \in T,\ \forall t' \in P(t) \neq 0 \tag{17}$$

or

$$-Mu_t - \sum_{i\in I}\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e} i\beta^e_{its}x^e_s + Mu_{t'}$$
$$+\sum_{i\in I}\sum_{e\in E}\sum_{s\in\tilde{\Omega}^e}(i+l_{t'})\beta^e_{it's}x^e_s \leq 0,\ \forall t \in T, t' \in P(t). \tag{18}$$

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 work regulations and the employee's skills and availabilities. 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^e_s of the shift $s \in \tilde{\Omega}^e$ for employee $e \in E$ can be decomposed as:

$$c^e_s = \sum_{i\in I}\left(\sum_{a\in A}\delta^e_{ias}c^e_{ia} + \sum_{t\in T}\beta^e_{its}c^e_{it}\right), \tag{19}$$

where c^e_{ij}, $j \in A \cup T$, is the cost for employee e to perform job j at period i. The reduced cost of the shift s is then:

$$\tilde{c}^e_s = \sum_{i\in I}\left(\sum_{a\in A}(c^e_{ia} - \lambda_{ia})\delta^e_{ias} + \sum_{t\in T}(c^e_{it} - \theta_t - r_{it})\beta^e_{its}\right) - \sigma^e,\ \forall e \in E,\ s \in \Omega^e, \tag{20}$$

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 $P^*(t) = \{t' \in T \mid t \in P(t')\}$.

To solve the pricing subproblem for employee e, we associate a cost $k_{ia} = c^e_{ia} - \lambda_{ia}$, $a \in A$, with each leaf of the DAG corresponding to the activity a performed at period $i \in I$ and a cost $k_{it} = (c^e_{it} - \theta_t - r_{it})$, $t \in T$, with each leaf of the DAG corresponding to the task t starting at period $i \in I$ (labelled s_t in the grammar). 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 (2009) 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^e_s can be extended because a cost can be assigned to some productions of the grammar as defined in Sect. 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. (2011b), which is adapted from the strategy presented by Barnhart et al. (2000) for solving integer multi-commodity flow problems.

At each node of the B&P tree, 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 $s^{e'}(1)$ and $s^{e'}(2)$, 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 insures 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 $c_{i'j}^{e'}$ where j is a forbidden job in $J_{i'}^{e'}(l)$ for $l \in \{1,\ 2\}$. This insures that all shifts with the job j at position i' lead to a positive reduce cost, hence these shifts are never selected by the dynamic programming algorithm. Then, we do not need to modify the pricing subproblems at each step of the algorithm.

In Côté et al. (2011b), 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 insures 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, with this strategy, the B&P approach can have more difficulty to converge towards an optimal 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.

Note that, when no branching occurs with the different proposed branching strategies, that means that no fractional solution exists, and, hence, the solution is integer [see Barnhart et al. (2000), Côté et al. (2011b)].

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 times 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 where, for an employee, the working periods are fixed a priori or are flexible. 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. (2009). The algorithm stops when the optimality gap is less than 1 % or when the processing time for an instance reaches 2 h (which is, in practice, an acceptable processing time considering that the planning is done once for a whole week).

Four different strategies are tested in this section. First, we consider the two formulations of the precedence constraints described in Sect. 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 Sect. 4.3, i.e., S1, where branching is performed only on the activities, S2, where branching is performed on the tasks and the activities, and S3, the hybrid strategy.

For all the instances, to obtain an initial upper 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 30 min. In some cases, this initial solution has an optimality gap lower than 1 % and it is then not necessary to run the B&P algorithm.

6.1 The instances

We consider that an employee can perform a shift of 4 h or a shift of 8 h with a break of 1 h for lunch. The hours of work are from 6 a.m. to 7 p.m. When the working periods are fixed, 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 working periods are flexible, an employee can perform a shift of 4 or 8 h which can start at any period within a given time window. Within a shift of 8 h, after 4 h of consecutive work, we must place a lunch of 1 h.

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 30 min and its maximum length is 4 h. 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 $t \in T$.

The instances are available online at the following address: http://w1.cirrelt.umontreal.ca/~louism/Random_instances.zip

6.2 Instances with fixed working periods

For the instances with fixed working periods, the start and end times of the shift pieces, i.e, a continuous sequence of work for each employee, are fixed a priori. Hence, the position of the lunch periods are also known. We consider a planning horizon of 1 week, which is divided into periods of 15 min (672 periods in total) with 5 activities and 50 tasks. Shifts pieces of 4 h are assigned such that no employee works more than 40 h. Furthermore, no more than two shift pieces, separated by a break of 1 h, can be assigned to an employee per day. 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 piece. These grammars are represented by the following productions:

- $S \ {}_{[16,16]} \xrightarrow{c_{tr}} J_j J_{\bar{j}} \mid J_j, \ \forall j \in A_e \cup T_e$;
- $J_a \ {}_{[2,16]} \rightarrow J'_a, \ \forall a \in A_e$;
- $J'_a \rightarrow a \mid a J'_a, \ if \ a \in A_e$;
- $J_t \ {}^{[tws_t, twe_t]}_{[l_t, l_t]} \rightarrow s_t J'_t, \ \forall t \in T_e$;
- $J'_t \rightarrow t \mid t J'_t, \ \forall t \in T_e$;
- $J_{\bar{j}} \rightarrow J_{j'}, \ \forall j \in A_e \cup T_e, \ \forall j' \in A_e \cup T_e - \{j\}$;
- $J_{\bar{j}} \xrightarrow{c_{tr}} J_{j'} J_{\bar{j'}}, \ \forall j \in A_e \cup T_e, \ \forall j' \in A_e \cup T_e - \{j\}$.

Table 1 Solution at the root with 20 employees and fixed working periods

Instance	C1			C2		
	# columns	Gap (%)	Time (s)	# columns	Gap (%)	Time (s)
PF_20_0	41,213	3.39	95.54	39,974	–	1800.00
PF_20_1	38,170	2.53	79.17	38,700	2.53	87.05
PF_20_2	40,166	35.96	102.68	41,621	23.52	90.58
PF_20_3	39,195	0.00	187.86	40,056	22.90	174.37
PF_20_4	41,742	3.85	132.28	42,566	3.96	81.77
PF_20_5	44,384	7.42	121.26	44,678	25.08	98.08
PF_20_6	41,123	3.10	102.09	40,553	7.49	113.84
PF_20_7	37,314	0.00	93.80	40,923	0.00	107.03
PF_20_8	40,964	2.43	95.09	40,468	15.66	98.51
PF_20_9	37,804	8.73	86.23	39,593	13.11	74.94
Average	40207.50	6.74	109.61	40913.20	12.69[a]	102.91[a]

[a] Average values do not include instance PF_20_0

Table 2 Solution at the root with 50 employees and fixed working periods

Instance	C1			C2		
	# columns	Gap (%)	Time (s)	# columns	Gap (%)	Time (s)
PF_50_0	83,330	0.86	203.57	83,173	38.20	313.42
PF_50_1	84,118	13.73	244.54	81,939	17.84	345.62
PF_50_2	82,221	6.31	150.91	80,122	17.34	271.11
PF_50_3	80,909	0.08	172.90	80,479	27.20	238.51
PF_50_4	81,200	17.48	204.43	83,163	27.49	173.23
PF_50_5	79,080	6.00	157.10	83,451	0.85	193.42
PF_50_6	84,123	27.24	247.54	87,119	64.42	549.77
PF_50_7	84,811	3.10	173.84	81,421	13.13	214.22
PF_50_8	82,465	2.88	243.40	82,585	37.95	282.23
PF_50_9	83,727	9.63	321.62	82,771	35.94	295.28
Average	82598.40	8.73	211.98	82622.30	28.04	287.68

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.

6.2.2 B&P

The B&P algorithm 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 is already small at the root ($< 5\,\%$), it is only slightly improved, but when the average optimality gap at the root is large ($\geq 5\,\%$), the improvement is generally significant.

Table 3 Optimality gap (%) with 20 employees and fixed working periods

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	3.18
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	4.16
PF_20_3	0.00	0.00	0.00	1.60	1.60	0.05
PF_20_4	3.85	3.85	3.85	3.87	3.87	3.85
PF_20_5	2.79	2.79	2.92	3.10	2.79	2.79
PF_20_6	3.10	3.10	3.06	3.11	3.11	3.10
PF_20_7	0.04	0.04	0.04	0.00	0.00	0.00
PF_20_8	2.34	2.29	2.34	2.47	2.61	2.29
PF_20_9	3.90	3.94	3.90	3.94	3.90	3.90
Average	2.59	2.59	2.60	3.21	3.19	2.59

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

Table 4 Optimality gap (%) with 50 employees and fixed working periods

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	0.70
PF_50_1	10.01	10.01	9.93	10.24	10.24	9.97
PF_50_2	1.58	1.57	1.58	1.70	1.70	1.58
PF_50_3	0.12	0.12	0.12	0.94	0.94	0.12
PF_50_4	2.43	2.43	2.39	4.57	4.57	2.35
PF_50_5	0.98	0.98	0.98	0.85	0.85	1.02
PF_50_6	7.64	7.64	7.56	8.71	8.71	7.56
PF_50_7	3.02	3.02	3.02	4.38	4.38	2.97
PF_50_8	2.75	2.75	2.75	2.75	2.75	2.75
PF_50_9	5.11	5.11	5.11	5.84	6.08	5.11
Average	3.43	3.43	3.41	4.23	4.25	3.41

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

6.3 Instances with flexible working periods

For the instances with flexible working periods, 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 h, i.e., two shift pieces of 4 h separated with a lunch-break of 1 h. Hence, an 8-h shift last, in fact, 9 h (8 h of work plus 1 h of lunch-break). Since these instances are more difficult, we consider a planning horizon of 1 day divided into periods of 15 min (52 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.

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 use the artificial activity r. The grammar is described by the following productions:

- $S_{[52,52]} \rightarrow RPR \mid RFR \mid PR \mid RP \mid FR \mid RF$;
- $P_{[16,16]} \xrightarrow{c_{tr}} J_j J_{\bar{j}} \mid J_j,\ \forall j \in A_e \cup T_e$;
- $F_{[36,36]} \rightarrow PLP$;
- $J_{a\ [2,16]} \rightarrow J'_a,\ \forall a \in A_e$;
- $J'_a \rightarrow a \mid a J'_a,\ \forall a \in A_e$;
- $J_{t\ [l_t,l_t]}^{[tws_t,twe_t]} \rightarrow s_t J'_t,\ \forall t \in T_e$;
- $J'_t \rightarrow t \mid t J'_t,\ \forall t \in T_e$;
- $J_{\bar{j}} \rightarrow J_{j'},\ \forall j \in A_e \cup T_e,\ \forall j' \in A_e \cup T_e - \{j\}$;
- $J_{\bar{j}} \xrightarrow{c_{tr}} J_{j'} J_{\bar{j'}},\ \forall j \in A_e \cup T_e,\ \forall j' \in A_e \cup T_e - \{j\}$;
- $R \rightarrow r \mid r R$;
- $L_{[4,4]} \rightarrow r \mid r R$.

6.3.1 Solution at the root

Tables 5 and 6 give the optimality gaps obtained by solving the RMP at 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.

6.3.2 B&P

Tables 7 and 8 give the optimality gaps obtained with the B&P approach on the instances with flexible working periods. 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 min. With 50 employees, 6 of the 10 instances are solved exactly in less than 2 h. The best average gap for all the strategies is less than 4 %. On average, the best solu-

Table 5 Solution at the root with 20 employees and with flexible working periods

Instance	C1			C2		
	# columns	Gap (%)	Time (s)	# columns	Gap (%)	Time (s)
PV_20_0	18,847	73.32	41.82	18,098	52.71	40.38
PV_20_1	21,292	0.00	34.98	20,399	0.00	27.64
PV_20_2	18,282	23.92	76.25	18,312	43.94	47.50
PV_20_3	19,565	27.05	59.64	18,590	27.23	72.01
PV_20_4	15,328	52.09	45.77	14,685	35.11	29.03
PV_20_5	22,390	24.26	64.88	19,945	42.64	62.12
PV_20_6	18,918	21.50	40.36	19,084	28.75	31.47
PV_20_7	17,662	55.55	44.28	17,699	61.98	86.47
PV_20_8	17,185	54.92	38.08	17,569	48.12	48.16
PV_20_9	12,985	0.03	19.61	13,106	34.83	28.43
Average	18245.40	33.26	46.57	17748.70	37.53	47.32

Table 6 Solution at the root with 50 employees and with flexible working periods

Instance	C1			C2		
	# columns	Gap (%)	Time (s)	# columns	Gap (%)	Time (s)
PV_50_0	36,332	67.88	168.68	36,828	72.51	38.93
PV_50_1	31,523	64.79	143.84	30,161	66.20	139.56
PV_50_2	34,808	69.71	208.12	33,857	70.50	315.94
PV_50_3	31,524	68.82	127.62	32,172	77.23	178.71
PV_50_4	44,878	67.03	223.51	42,345	72.36	228.00
PV_50_5	38,473	54.92	173.53	41,203	69.57	233.83
PV_50_6	43,448	69.00	279.66	45,571	77.17	236.65
PV_50_7	33,798	68.87	149.30	31,163	79.64	142.08
PV_50_8	36,418	69.00	182.54	39,064	77.22	143.63
PV_50_9	31,669	64.34	149.36	33,415	66.05	256.86
Average	36282.11	66.44	180.62	36577.90	72.84	191.50

Table 7 Optimality gap (%) with 20 employees and with flexible working periods

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	0.08	0.08	0.08	15.55	0.08
PV_20_4	0.18	0.18	0.06	0.07	0.07	0.07
PV_20_5	0.06	0.06	0.06	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	16.36	16.43	24.96	44.14	18.96
PV_20_8	11.95	11.95	37.45	12.97	12.97	12.97
PV_20_9	0.03	0.03	0.03	0.03	0.03	0.03
Average	2.07	2.88	5.42	3.91	7.38	3.31

Table 8 Optimality gap (%) with 50 employees and with flexible working periods

Instance	C1 + S1	C1 + S2	C1 + S3	C2 + S1	C2 + S2	C2 + S3
PV_50_0	2.53	2.53	2.53	2.53	0.00	2.53
PV_50_1	6.32	6.32	6.32	6.32	6.32	6.32
PV_50_2	4.72	22.25	4.72	27.17	38.33	19.49
PV_50_3	0.00	0.00	0.00	0.00	0.00	0.00
PV_50_4	0.00	0.00	0.00	0.00	0.00	0.00
PV_50_5	0.00	0.00	0.00	34.53	34.70	34.44
PV_50_6	14.10	14.29	37.59	23.41	13.32	13.02
PV_50_7	0.00	0.00	0.00	0.00	0.00	0.00
PV_50_8	0.00	0.00	0.00	0.00	0.00	0.00
PV_50_9	10.09	0.00	0.10	5.07	5.07	5.07
Average	3.78	4.54	5.13	9.90	9.77	8.09

tions are obtained with C1 + S1. However, for the instances with 50 employees, C1 + S2 and C1 + S3 are able to obtain a gap lower than 1 % for PV_20_9 . 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 towards a good feasible solution.

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 with flexible working periods, 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 interesting 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 with flexible working periods. The S2 and S3 branching decisions are stronger because they try 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_7, PV_50_2, and PV_50_6) it degrades the convergence towards a good solution. Hence, S1 appears to be a more stable strategy for formulation C1.

With formulation C2, however, strategies S2 and S3 lead to better results than strategy S1. In this case, the precedence constraints are less explicit than in formulation C1, hence branching on the positioning of a task seems to be the better strategy.

For the strategies C1 + S1 and C2 + S3, which gave on average the best results for, respectively, formulation C1 and formulation C2, Table 9 displays the total number of nodes explored by the B&P (# Nodes), the time to reach the best solution (CPU BS), the average time spent in solving the grammar subproblems (CPU GR), the average time spent in

Table 9 B&P statistics

	Instances with fixed working periods		Instances with flexible working periods	
Number of employees	20	50	20	50
C1 + S1				
# Nodes	6361.80	3309.10	369.10	1297.10
CPU BS (s.)	169.49	115.80	190.95	1183.35
CPU GR (s.)	708.37	681.50	119.65	404.72
CPU CG (s.)	6223.64	5468.44	1677.83	4548.44
CPU Total (s.)	6248.02	5488.45	1679.49	4552.52
C2 + S3				
# Nodes	5095.60	2998.30	320.70	1050.00
CPU BS (s.)	121.26	192.76	316.92	2121.89
CPU GR (s.)	726.45	938.41	83.70	404.41
CPU CG (s.)	7009.57	6247.80	1663.16	5400.12
CPU Total (s.)	7027.76	6262.73	1664.06	5405.91

solving the linear relaxation of the model with the column generation (CPU CG), and the average total computation time (CPU Total). In all cases, we can see that less than 15 % of the processing time is used to generate new columns for the column generation. Almost all of the processing time is spent in the column generation for solving the linear relaxation of the model. Furthermore, we note that, in general, formulation C1 converges faster towards a good solution than formulation C2.

Our B&P algorithm finds integer solutions for the SSP and gives an approximate optimality gap. The convergence towards 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.

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 availabilities and skills. This model is solved via a B&P 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 three branching strategies. We considered instances with up to 50 employees where the working periods are fixed a priori, with a time horizon of one week, and where they are flexible, with a time horizon of one day. The results showed that the B&P algorithm can find an integer solution for all the instances within 2 h with an optimality gap lower than 5 % in the best case.

Other branching strategies could be implemented, for instance, one could consider the choice of the employee to branch on or the choice of the activities to forbid. However, in our tests, we did not identify any better criteria for the selection of the employee and the activities to forbid than the ones proposed in this paper.

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

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

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, *46*(3), 316–329.

Contreras, I., Díaz, J., & Fernández, E. (2011). Branch and price for large-scale capacitated hub location problems with single assignment. *INFORMS Journal of Computing*, *23*, 41–55.

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

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

Côté, M. C., Gendron, B., & Rousseau, L. M. (2012). Grammar-based column generation for personalized multi-activity shift scheduling. *INFORMS Journal of Computing*. doi:10.1287/ijoc.1120.0514.

Crainic, T. G., Frangioni, A., Gendron, B., & Guertin, F. (2009). OOBB: An object-oriented library for parallel branch-and-bound. In: *CORS/INFORMS international conference, Toronto, Canada.*

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

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

Desaulniers, G., Desrosiers, J., & Solomon, M. M. (2005). *Column generation*. New York: Springer.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., Owens, B., & Sier, D. (2004a). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, *127*(1), 21–144.

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

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). *Introduction to automata theory, languages and computability*. Boston: Addison-Welsey.

Lequy, Q., Bouchard, M., Desaulniers, G., Soumis, F., & Tachefine, B. (2010a). Assigning multiple activities to work shifts. *Journal of Scheduling*, *15*(2), 239–251.

Lequy, Q., Desaulniers, G., Solomon, M. M. (2010b) Assigning team tasks and multiple activities to fixed work shifts. Cahiers du GERAD (G-2010-71). Montreal: HEC Montreal.

Lequy, Q., Desaulniers, G., Solomon, M. M. (2010c) A two-stage heuristic for multi-activity and task assignment to work shifts. Cahiers du GERAD (G-2010-28). Montreal: HEC Montreal.

Quimper, C. G., & Rousseau, L. M. (2009). A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, *16*(3), 373–392.

Quimper, C. G., & Walsh, T. (2007). Decomposing global grammar constraints. In: *Principles and practice of constraint programming-CP 2007. Lecture notes in computer science* (Vol. 4741, pp. 590–604). New York: Springer.

Tang, L., Wang, G., & Liu, J. (2007). A branch-and-price algorithm to solve the molten iron allocation problem in iron and steel industry. *Computers & Operations Research*, *34*(10), 3001–3015.

Tang, L., Wang, G., Liu, J., & Liu, J. (2011). A combination of lagrangian relaxation and column generation for order batching in steelmaking and continuous-casting production. *Naval Research Logistics*, *58*(4), 370–388.

van den Akker, J. M., Hoogeveen, J. A., & van de Velde, S. L. (1999). Parallel machine scheduling by column generation. *Operations Research*, *47*(6), 862–872.