Grammar-Based Integer Programming Models for Multi-Activity Shift Scheduling

Abstract

This paper presents a new implicit formulation for shift scheduling problems, using context-free grammars **to model the rules for the composition of shifts**. From the grammar, we generate an integer programming (IP) model having a linear programming (LP) relaxation equivalent to that of Dantzig's set covering model. When solved by a state-of-the-art IP solver on problem instances with a small number of shifts, our model, the set covering formulation and a typical implicit model from the literature yield comparable solution times. On instances with a large number of shifts, our formulation shows superior performance and can model a wider variety of constraints. In particular, multi-activity cases, which cannot be modeled by existing implicit formulations, can easily be handled with grammars. We present comparative experimental results on a large set of instances involving one work-activity, as well as on problems dealing with up to ten work-activities.

Keywords*: Shift Scheduling, Implicit Models, Mixed Integer Programming, Context-Free Grammars*

1 Introduction

In this paper, we consider *shift scheduling problems* defined over a planning horizon of one day, divided into multiple periods. In this context, a shift is defined by its starting and ending times and by the activities or breaks to be performed at each period. The assignment of activities **and breaks** to a shift is constrained by different rules mainly arising from work regulation agreements and ergonomic considerations. In a *single-activity* shift scheduling problem, one only specifies, at each period, if an employee is working or **taking a break**. In a *multi-activity* shift scheduling problem, there are several work-activities and whenever an employee is working at a given period, it is further necessary to specify which work-activity is assigned to that employee. In this paper, we deal with multi-activity shift scheduling problems in which all employees are identical.

The problem we consider is defined as follows. Given a planning horizon I divided into periods of equal length, a set of work-activities J, the set of all feasible shifts Ω, and the number of employees b_{ij} required at each period $i \in I$ for each work-activity $j \in J$, one must select from Ω a subset and multiplicities for each shift in this subset that covers the required number of employees at minimum cost. Each feasible shift $s \in \Omega$ has an associated cost $c_s \geq 0$, which we assume to be decomposable by period and by work-activity as follows: $c_s = \sum_{i \in I} \sum_{j \in J} \delta_{ijs} c_{ij}$, where $c_{ij} \geq 0$, for each $i \in I$ and $j \in J$, and $\delta_{ijs} = 1$ if work-activity $j \in J$ is assigned to period $i \in I$ in shift $s \in \Omega$.

We will consider two types of integer programming (IP) models for this problem, explicit and implicit, a terminology that is also used to characterize formulations for single-activity shift scheduling problems. In an *explicit* model, one obtains the schedule for each employee simply by scanning the optimal solution (i.e., in a time linear to the model size), while in an *implicit* model, a post-processing algorithm must be called upon in order to derive the schedule for each employee. This algorithm is typically efficient **(i.e. polynomial in the model size)**, especially when compared to solving the model, but its running time is generally not linear in the model size.

The following IP model, denoted D, extends in a straightforward manner the original set covering formulation proposed in Dantzig (1954) for the shift scheduling problem first described in Edie (1954). This model uses a variable x_s for each shift $s \in \Omega$ **corresponding to the number of employees** assigned to shift s:

$$
\begin{aligned}
f(D) = \min \sum_{s \in \Omega} c_s x_s & & \\
\sum_{s \in \Omega} \delta_{ijs} x_s \geq b_{ij}, & \qquad \forall i \in I, j \in J, & (1)\\
x_s \geq 0 \text{ and integer}, & \qquad \forall s \in \Omega. & (2)
\end{aligned}
$$

Model D explicitly enumerates all feasible shifts; therefore, we will call it the *shift-based explicit model.* Note that model D allows the formulation of problems with any cost structures that decompose by shift, not only by period and by work-activity, as we assume in our problem definition (see the Conclusion for a discussion on more general cost structures).

In practice, such explicit models can only be solved when Ω is relatively small, or else, by using column generation approaches as in Demassey et al. (2006), Mehrotra et al. (2000). In this paper, we present a new implicit formulation based on assignment variables y_{ij} indicating the number of employees assigned to activity $j \in J$ at period $i \in I$. These variables are related to the variables in the shift-based explicit model by the simple equations $y_{ij} = \sum_{s \in \Omega} \delta_{ijs} x_s$. More precisely, the nonnegative integer variables x_s defined over feasible shifts $s \in \Omega$ in model D are represented equivalently by using an additional set of integer variables $v \geq 0$ such that $(y, v) \in H$, where H is a bounded polyhedron. The implicit model that we propose, denoted Q, has therefore the following form:

$$
\begin{aligned}
f(Q) = \min \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} & & \\
y_{ij} \geq b_{ij} \text{ and integer}, & \qquad \forall i \in I, j \in J, & (3)\\
(y, v) \in H, & & (4)\\
v \geq 0 \text{ and integer}. & & (5)
\end{aligned}
$$

Côté et al. (2007) and Côté et al. (2009) exploit automata and context-free grammars to formulate similar IP models that represent all feasible shifts for any single employee. Since they explicitly represent the assignment of work-activities to each employee, these formulations belong to the class of explicit models, but unlike model D, they do not use the set of shifts; hence, we will call them *employee-based explicit models.* When the number of employees or activities increase, these employee-based explicit models do not scale well as performance degrades rapidly, mainly due to symmetry issues (see Section 5.2 for experimental results). In this paper, we show how to derive grammar-based models with tractable size, allowing us to handle large-scale problems with multiple activities. Assuming that any feasible shift can be represented by a word in a context-free language, we will show how to derive polyhedron H from the context-free grammar G defining the language. Moreover, we will show that polyhedron H is integral, and therefore that Q and D have equivalent linear programming (LP) relaxations. To the best of our knowledge, our approach is the first implicit modeling technique that is able to accurately formulate and efficiently solve *multi-activity* shift scheduling problems.

The remainder of the paper is organized as follows. In the next section, we present a literature review on shift scheduling problems and we introduce formal languages and grammar theory. In Section 3, we describe our modeling methodology using grammars to formulate shift scheduling problems. In Section 4, we present theoretical results relevant to our model; in particular, we demonstrate that our model has the same LP relaxation as Dantzig's set covering model. In Section 5, we present comparative computational results on classical shift scheduling problems from Mehrotra et al. (2000) and on a set of large-scale problem instances with multiple activities.

2 Background Material

This section reviews the literature on shift scheduling problems and presents basic notions of context-free grammars which are relevant to our study.

2.1 Shift Scheduling

For many organizations, finding the best schedule satisfying all their requirements and constraints is an important, but difficult task. Consequently, several studies **have been dedicated** to this problem. Ernst et al. (2004a,b) present an exhaustive overview of models and methods for problems related to staff scheduling and rostering.

Implicit formulations provide an interesting alternative to the explicit model D. **While the explicit model uses one variable by shift (a shift is defined by its starting and**

ending times, and the placement of the breaks within the shift), existing implicit formulations introduce the notion of *shift types*, which are characterized only by starting and ending times, giving no details about how breaks are assigned within the shifts. Typically, contrarily to explicit models, these models capture the number of employees assigned to each shift type and to each break with different sets of variables. From an optimal solution to such an implicit model, one can retrieve the number of employees assigned to each shift type and each break, and construct an optimal set of shifts through a polynomial-time procedure.

Rekik et al. (2004) give such an implicit model based on a transportation problem to assign breaks to shifts. They show that the LP relaxation of their model is equivalent to the LP relaxations of two other classical implicit formulations, namely those proposed in Aykin (1996) and in Bechtolds and Jacobs (1990). Since Dantzig's set covering model has the same integrality gap as Aykin's model, the LP relaxations of these four models are equivalent. Rekik et al. (2005) propose extensions to allow more flexibility in the definition of breaks. However, to this date, we are not aware of any implicit formulation that can accurately represent *multi-activity* shift scheduling problems.

An alternative to existing explicit and implicit models is to use formal languages to model work regulations. Côté et al. (2007) propose an IP model based on a regular language, represented by a finite deterministic automaton, to formulate the constraints defining a shift, and to represent all feasible shifts using a network flow formulation. Côté et al. (2009) extend these results by using context-free grammars in modeling shift scheduling problems. From a grammar describing work regulations, they generate an IP model based on assignment variables y_{ije}, that describe all feasible shifts for each employee e. Since the number of employees is bounded from below by $\max_{i \in I} \sum_{j \in J} b_{ij}$, this model generates a large number of variables. Moreover, in the case where many employees are alike, this model has symmetry issues.

Multi-activity shift scheduling problems. With the use of formal languages, many constraints in the planning of shifts can be considered. In particular, these modeling methods can deal with contexts where multiple work-activities can be performed during the same shift, each activity having its own labor requirements. Compact models for multi-activity problems are not common in the literature. Among the few papers addressing this topic, Loucks and Jacobs (1991) and Ritzman et al. (1976) model the tour scheduling problem (shift scheduling over one week) with Boolean assignment variables specifying the number of employees assigned to a given task at any given time. Since such modeling approaches yield very large IP formulations, both papers propose heuristic methods to construct and improve the solutions. Moreover, they do not place breaks or meals during the shifts, nor do they handle regulations concerning the transition between activities. Approaches using column generation were suggested in Bouchard (2004), Vatri (2001) and Demassey et al. (2006). The first two propose approaches to schedule air traffic controllers. While Vatri (2001) uses a heuristic method to build the schedule without taking into account break placement, Bouchard (2004) extends his work to include break placement and solves the problem with a heuristic column generation approach. Demassey et al. (2006) propose a column generation procedure based on constraint programming, that solves efficiently the LP relaxation of **the problem stated below in Section 5.2 for up to 10 work-activities**. However, they report that branching to find integer solutions is difficult and succeeds only for the smallest instances. More recently, Lequy et al. (2009) address another type of multi-activity shift scheduling problem where shifts and breaks are fixed a priori for each employee and where work-activities must then be assigned to shifts. Each employee has a set of skills that restricts the set of activities he can perform. Lequy et al. (2009) present three integer programming models and show good computational results with a heuristic column generation method embedded within a rolling horizon procedure.

In the following, we study some basic properties of grammars and show how they can be used in the context of shift scheduling problems.

2.2 Grammars

A context-free grammar defines a language over a given alphabet by means of a set of rules called *productions*. A production is a rule that specifies a substitution of symbols. These symbols are of two types: the *terminal symbols* are letters of the alphabet, generally represented by lower case letters, and the *non-terminal symbols* designate a subsequence that could be rewritten using the associated productions, generally represented by upper case letters. More formally, a production is represented as follows: $\alpha \rightarrow \beta$, where α is a non-terminal symbol and β is a sequence of terminal and/or non-terminal symbols. The productions of a grammar can be used recursively to generate new symbol sequences until only terminal symbols are part of the sequence. A sequence of terminal symbols is called a *word*.

Definition. A *context-free grammar* G is characterized by a tuple (Σ, N, P, S) where:

- Σ is an alphabet;
- N is a set of non-terminal symbols;
- P is a set of productions;
- S is the starting non-terminal.

A word, or sequence of letters from alphabet Σ, is *recognized* by a grammar G if it can be generated by successive applications of productions from G, starting with non-terminal S.

In the following, we will use the term grammar to refer to a context-free grammar and we will assume that, except when specified otherwise, all grammars are in Chomsky normal form, meaning that all productions are of the form $X \rightarrow \beta$ where $X \in N$ and $\beta \in (N \times N) \cup \Sigma$. Note that this assumption is not restrictive since any context-free grammar can be converted to Chomsky normal form; see Hopcroft et al. (2001) for more information on formal languages.

Example 1 *The following grammar G defines all feasible shifts for a simple shift scheduling problem. A shift must have a duration equal to the planning horizon and contain one break of one period anywhere during the shift except at the first or the last period. Work and break periods are respectively represented by letters w and b.*
$G = (\Sigma = (w, b), N = (S, X, W, B), P, S)$, *where P is:*
$S \rightarrow XW, \qquad X \rightarrow WB, \qquad W \rightarrow WW \mid w, \qquad B \rightarrow b,$
*where the symbol $\mid$ specifies a choice of production. The shifts wbw, $wwwwwbw$ and $wbww$, among others, are recognized by G. $wbwb$ is not recognized by G. Word $wwbw$ is obtained by the derivation shown in Table 1, where **P** is the production used and **CS** is the current sequence, obtained from the previous sequence by applying the production on the left side.*

Table 1: Derivation of word *wwbw* from grammar G of Example 1

P	**CS**
$-$	S
$S \rightarrow XW$	XW
$X \rightarrow WB$	WBW
$W \rightarrow WW$	$WWBW$
$W \rightarrow w$	$wWBW$
$W \rightarrow w$	$wwBW$
$B \rightarrow b$	$wwbW$
$W \rightarrow w$	$wwbw$

A common way to illustrate the derivation of a word from a grammar is to use a tree, called *parse tree*, where the root node is the starting non-terminal S, the interior nodes are non-terminals and leaves are letters of the alphabet. A production $X \rightarrow YZ$ is represented by nodes Y and Z as left and right children of node X, while $X \rightarrow a$ is represented by node X and a unique child, leaf a. When listed from left to right, the leaves form a word recognized by the grammar. Figure 1 shows the two parse trees induced by grammar G from Example 1 on words of length four ($wwbw$ and $wbww$).

A parse tree representing a word ω of length n has the following properties:

- An interior node and its children represent a production in P.
- A leaf is associated with a position $i \in \{1, \ldots, n\}$ in ω and represents the letter from Σ taking place at position i.
- Any interior node is the root of a tree inducing a subsequence of ω, starting at position $i \in \{1, \ldots, n\}$ with length $l \in \{1, \ldots, n - i + 1\}$.

Using these observations, the next developments characterize a graph embedding all parse trees associated to words of a given length.

The DAG Γ. In the following, we describe a *directed acyclic graph* (DAG) Γ that encapsulates all parse trees associated to words of a given length n recognized by a grammar $G = (\Sigma, N, P, S)$. The DAG Γ has an and-or structure containing two types of nodes: nodes O (the or-nodes) represent non-terminals from N and letters from Σ, and nodes A (the

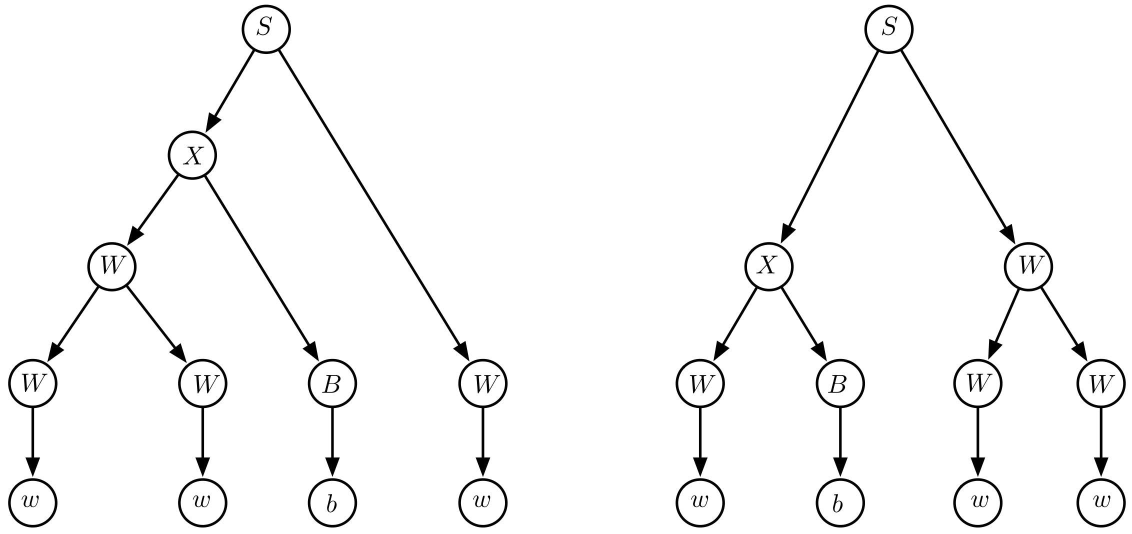

Figure 1: parse trees for grammar G from Example 1 on words of length 4.

and-nodes) represent productions from P. Each node is characterized by its symbol (non-terminal, letter, or production) and the position and length of the subsequence it generates. We define O^{π}_{il} the node associated with non-terminal or letter π that generates a subsequence at position i of length l. Note that if $\pi \in \Sigma$, the node is a leaf and l is equal to one. Also, Γ has a root node described by O^{S}_{1n}. Likewise, $A^{\Pi,t}_{il}$ is the t^{th} node representing production Π generating a subsequence from position i of length l. There are as many $A^{\Pi,t}_{il}$ nodes as there are ways of using Π to generate a sequence of length l from position i. We will refer to this set as the (potentially empty) set $A(\Pi, i, l)$.

The DAG Γ is built in such a way that a path from one node to any other node alternates between or-nodes O and and-nodes A. More precisely, the DAG Γ has the following properties:

- Children of an or-node O^{π}_{il} with $l > 1$, denoted $ch(O^{\pi}_{il})$, are all and-nodes $A^{\Pi,t}_{il}$ such that $\Pi : \pi \to \beta$, $\beta \in (N \times N) \cup \Sigma$ and $t \in A(\Pi, i, l)$.
- Each or-node O^{π}_{i1} where π is a non-terminal has only one child: $ch(O^{\pi}_{i1}) = A^{\Pi,1}_{i1}$ such that $\Pi : \pi \to a$, where $a \in \Sigma$.
- Parents of an or-node O^{π}_{il} where $\pi \neq S$ is a non-terminal, denoted $par(O^{\pi}_{il})$, are and-

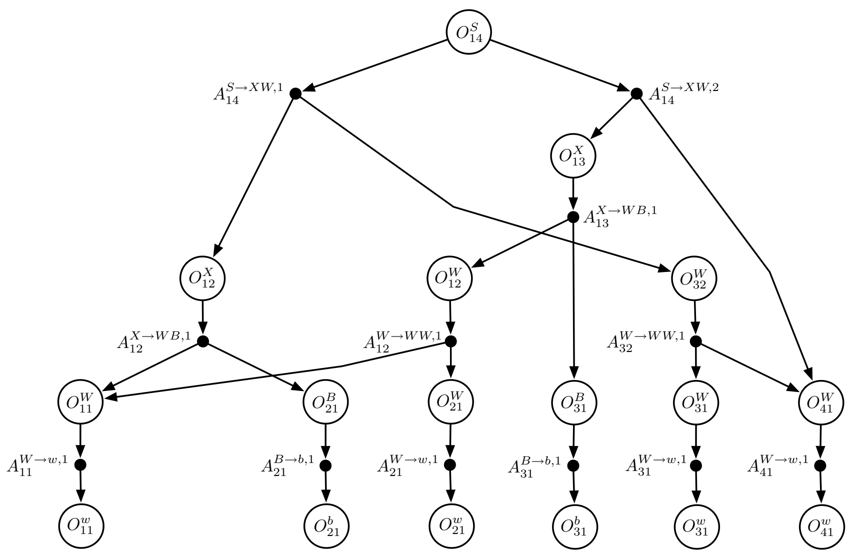

Figure 2: DAG Γ for grammar from Example 1 on a word of length 4.

nodes of the form $A_{jm}^{\Pi,t}$ such that $\Pi : X \to \pi Z$ or $\Pi : X \to Y\pi$, where $j \leq i$ and $m \geq l$.

- Each and-node $A_{il}^{\Pi,t}$ with $l > 1$ such that $\Pi : X \to YZ$ has exactly two children: O_{ik}^{Y} and $O_{i+k,l-k-1}^{Z}$, where $k < l - 1$.

- Each and-node $A_{i1}^{\Pi,1}$ such that $\Pi : X \to a$, where $a \in \Sigma$, has only one child: O_{i1}^{a}.

- Each and-node $A_{il}^{\Pi,t}$ has only one parent: O_{il}^{π} such that $\Pi : \pi \to \beta$, $\beta \in (N \times N) \cup \Sigma$, if $l > 1$, and $\Pi : \pi \to a$, $a \in \Sigma$, if $l = 1$.

Figure 2 presents the DAG Γ associated with grammar G from Example 1 on a word of length 4. It is easy to verify the above properties on this DAG.

To derive any parse tree from Γ, we start at the root O_{1n}^{S}. We visit an or-node O_{il}^{π} by selecting exactly one child, which is necessarily an and-node. We visit an and-node $A_{il}^{\Pi,t}$ by choosing all its children (exactly two if $l > 1$, one otherwise). By traversing Γ in this way until the only remaining unvisited nodes are leaves, we obtain a parse tree associated to the word defined by the leaves. Conversely, starting from a given word ω, we can traverse Γ backwards in a straightforward way to derive the parse tree associated to ω. In practice, Γ

is built by a procedure suggested in Quimper and Walsh (2007) inspired by an algorithm from Cooke, Younger, and Kasami (see Hopcroft et al. (2001)).

Grammar-based IP model. Using the structure of the DAG Γ, Côté et al. (2009) present a system of linear equations in 0-1 variables that allow the identification of any word recognized by a given grammar G. To each node O_{il}^{π} and $A_{il}^{\Pi,t}$ in Γ are associated 0-1 variables u_{il}^{π} and $v_{il}^{\Pi,t}$, respectively. If we denote by L, the set of leaves in Γ, these equations are as follows:

$$u_{il}^{\pi} = \sum_{A_{il}^{\pi,t} \in ch(O_{il}^{\pi})} v_{il}^{\Pi,t}, \qquad \forall O_{il}^{\pi} \in O \setminus L, \tag{6}$$

$$u_{il}^{\pi} = \sum_{A_{il}^{\Pi,t} \in par(O_{il}^{\Pi})} v_{il}^{\Pi,t}, \qquad \forall O_{il}^{\pi} \in O \setminus O_{1n}^{S}, \tag{7}$$

$$u_{il}^{\pi} \in \{0,1\}, \qquad \forall O_{il}^{\pi} \in O, \tag{8}$$

$$v_{il}^{\Pi,t} \in \{0,1\}, \qquad \forall A_{il}^{\Pi,t} \in A. \tag{9}$$

Constraints (6) ensure that if variable u_{il}^{π} is equal to one, exactly one of the variables associated with its children must be equal to one. Similarly, constraints (7) ensure that if variable u_{il}^{π} is equal to one, exactly one of the variables associated with its parents must be equal to one. Consequently, when we set $u_{1n}^{S} = 1$, if this system of equations has a solution, then, in any solution, the variables equal to one form a parse tree associated to a word of length n recognized by G. Conversely, let ω be a word of length n on alphabet Σ. If we set to one the u_{i1}^{j} variables that form ω when the letters j are listed from left to right, then, if this system of equations has a solution, ω is recognized by G and the variables of the solution set to one form a parse tree associated to word ω.

We can rewrite equations (6) and (7) as follows:

$$u_{1n}^{S} = \sum_{A_{1n}^{\Pi,t} \in ch(O_{1n}^{S})} v_{1n}^{\Pi,t}, \tag{10}$$

$$\sum_{A_{il}^{\Pi,t} \in par(O_{il}^{\pi})} v_{il}^{\Pi,t} = \sum_{A_{il}^{\Pi,t} \in ch(O_{il}^{\pi})} v_{il}^{\Pi,t}, \qquad \forall O_{il}^{\pi} \in O \setminus \left\{L \cup \{O_{1n}^{S}\}\right\}, \tag{11}$$

$$u_{i1}^{j} = \sum_{A_{i1}^{\Pi,t} \in par(O_{i1}^{j})} v_{i1}^{\Pi,t}, \qquad \forall O_{i1}^{j} \in L. \tag{12}$$

This system of equations presents a structure similar to network flow conservation equations, but the two systems are different. Indeed, a solution to equations (10)-(12) does not specify a path in a network, but rather a tree in the DAG Γ, since any variable associated to an and-node which is equal to one in a solution will also have its two children with variables equal to one. Hence, (10)-(12) are not flow conservation equations. Further, if we represent system (10)-(12) in matrix notation, we can easily show that the corresponding matrix is not totally unimodular, contrary to the incidence matrix of a network, which is used to represent flow conservation equations. In spite of this, Pesant et al. (2009) have shown (see Section 4) that the polyhedron defined by (10)-(12) is integral, like the polyhedron defined by flow conservation equations.

3 Grammar-Based Model for Shift Scheduling

As explained in Côté et al. (2009), the system of equations (10)-(12) can be used in the context of shift scheduling problems, where the constraints defining any feasible shift are represented by a grammar G, i.e., each word ω recognized by grammar G corresponds to a feasible shift $s \in \Omega$. In this context, the number of periods $|I|$ corresponds to n, the length of any given word recognized by G, while the set of activities J corresponds to Σ, the letters of the alphabet.

Côté et al. (2009) describe an IP model based on this correspondance, using assignment variables y_{ije}, that describe all feasible shifts for each employee e. But, as explained in Section 2.1, when employees are similar, this model exhibits a lot of symmetry, which makes it impractical to solve large-scale instances. Assuming all employees can be assigned to the same shifts, we introduce here a new grammar-based IP model, that will not suffer from the same performance issues.

In equations (10)-(12), each variable is binary and specifies whether or not its corresponding node is part of the parse tree selected to generate a word. In the new model, each variable is a nonnegative integer that specifies how many parse trees the associated node is part of. Since we minimize an objective function with nonnegative costs, the integer variables do not need to be bounded from above.

As in the Introduction, let y_{ij} denote the number of employees assigned to activity $j \in J$ at period $i \in I$. We can replace the leaf variables u_{i1}^{j} by the variables y_{ij}. Model Q presented in the Introduction can now be explicitly stated as follows:

$$f(Q) = \min \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij}$$

$$y_{ij} \geq b_{ij}, \qquad \forall i \in I, j \in J, \tag{13}$$

$$u_{1n}^{S} = \sum_{A_{1n}^{\Pi,t} \in ch(O_{1n}^{S})} v_{1n}^{\Pi,t}, \tag{14}$$

$$\sum_{A_{il}^{\Pi,t} \in par(O_{il}^{\pi})} v_{il}^{\Pi,t} = \sum_{A_{il}^{\Pi,t} \in ch(O_{il}^{\pi})} v_{il}^{\Pi,t}, \qquad \forall O_{il}^{\pi} \in O \setminus \left\{ L \cup \{O_{1n}^{S}\} \right\}, \tag{15}$$

$$y_{ij} = \sum_{A_{i1}^{\Pi,t} \in par(O_{i1}^{j})} v_{i1}^{\Pi,t}, \qquad \forall i \in I, j \in J, \tag{16}$$

$$u_{1n}^{S} \geq 0 \text{ and integer}, \tag{17}$$

$$v_{il}^{\Pi,t} \geq 0 \text{ and integer}, \qquad \forall A_{il}^{\Pi,t} \in A, \tag{18}$$

$$y_{ij} \geq 0 \text{ and integer}, \qquad \forall i \in I, j \in J. \tag{19}$$

Once Q is solved, an implicit solution is obtained. To find the individual schedules from this solution, we traverse the DAG Γ from the root to the leaves visiting the nodes with value greater than zero. Each time a node is evaluated, its value is decreased by one. When a leaf node is reached, its value is inserted to the current schedule at the right position (see Appendix for the detailed algorithm). Model Q ensures that u_{1n}^{S} words recognized by grammar G can be extracted from the implicit solution. In the context of shift scheduling, variable $u_{1n}^{S} = k$ thus represents the total number of employees needed to perform all required shifts. The complexity of the algorithm used to extract the explicit set of shifts from the optimal implicit solution is $O(kn^3|G|)$ where n is the sequence length and $|G|$ is the number of productions in grammar G. In practice, the running time to perform this algorithm is negligible compared to the time necessary to solve model Q..

4 Theoretical Properties of Grammar-Based Models

In this section, we study the polyhedral properties of the implicit grammar-based model Q and compare it with other models from the literature. First, using the notation from the Introduction, we denote by H the polyhedron defined by equations (14)-(16) along with nonnegativity constraints on all variables. Our first result states that this polyhedron is integral, thus extending the result derived in Pesant et al. (2009) for a similar polyhedron defined over 0-1 variables.

Theorem 2 *H is an integral polyhedron.*

Proof. To simplify the notation, we denote H using matrix notation as follows: $H = \{z \geq 0 | Mz = b\}$. Now, let d be any arbitrary costs associated to variables z. The result will follow if we can prove that there always exists an integer optimal solution to the linear program: $\min\{dz | z \in H\}$. For this, it suffices to construct an integer point z^I in H that satifies the complementary slackness conditions: $(\lambda^* M - d)z^I = 0$, where λ^* is an optimal solution to the dual $\max\{\lambda b | \lambda M \leq d\}$.
Let z^* be an optimal solution to the linear program and let λ^* be the corresponding dual solution. We assume that $m = \lceil u^S_{1n} \rceil > 0$ (otherwise, if $m = 0$, $z^I = 0$ is an integer point in H satisfying the complementary slackness conditions). Our objective is to construct an integer solution z^I such that for every k for which $\lambda^* M_k < d_k$, we have $z^I_k = 0$. This condition can be easily maintained by enforcing that $z^I_k = 0$ whenever $z^*_k = 0$ ..
First, set $u^S_{1n} = m$ in z^I. By definition of H, since $u^S_{1n} > 0$ in z^*, there exists at least one variable corresponding to a child of root-node O^S_{1n} that has a value greater than 0 in z^*, say z^*_k corresponding to node $A^{\Pi,t}_{1n}$, with $\Pi : X \rightarrow YZ$. We continue our construction by fixing $z^I_k = m$. From constraints (15), since $z^*_k > 0$, the two children of $A^{\Pi,t}_{1n}$, say O^X_{1k} and $O^Y_{k+1,n-k}$ have at least one child each with a corresponding value greater than 0, say $z^*_{k_1}$ and $z^*_{k_2}$. We then fix $z^I_{k_1} = m$ and $z^I_{k_2} = m$. We continue this process, following the children of the nodes and setting them to m in z^I, until we reach the leaves of the DAG Γ.
The variables set to m in z^I form a tree in the DAG Γ that satisfies constraints $Mz = b$ by construction. Furthermore, since we only used variables that were already set to a value greater than 0 in the LP optimal solution z^*, we know that z^I satisfies the complementary slackness conditions with respect to λ^*. Therefore, there always exists an optimal integer solution to the linear program $\min\{dz | z \in H\}$, with arbitrary d, i.e., polyhedron H is integral.. ■

From this result, we observe that integrality constraints (17) and (18) are redundant in model Q. However, in practice, we found that leaving these constraints in the formulation helps the IP solver to further presolve the model and, overall, speeds up the solution process. Consequently, the experimentions in Section 5 were performed by leaving the integrality constraints in the models.

The next theorem uses the previous result to establish the equivalence between the LP relaxations of models Q and D, the **Dantzig's set covering formulation** presented in the Introduction.

Theorem 3 *Q and D have equivalent LP relaxations.*

Proof. The proof is direct using Lagrangean duality arguments. Let $\gamma_{ij} \geq 0$ denote Lagrangean multipliers associated to the requirement constraints, (1) in model D and (3) in model Q. Also, let $f_{LP}(M)$ denote the optimal objective value of the LP relaxation of a model M. We then have:

$$\begin{aligned}
f_{LP}(Q) &= \max_{\gamma \geq 0} \left\{ \sum_{i \in I} \sum_{j \in J} \gamma_{ij} b_{ij} + \min \left\{ \sum_{i \in I} \sum_{j \in J} (c_{ij} - \gamma_{ij}) y_{ij} \,|\, (y,v) \in H \right\} \right\} \\
&= \max_{\gamma \geq 0} \left\{ \sum_{i \in I} \sum_{j \in J} \gamma_{ij} b_{ij} + \min \left\{ \sum_{i \in I} \sum_{j \in J} (c_{ij} - \gamma_{ij}) y_{ij} \,|\, (y,v) \in H, (y,v) \text{ integer} \right\} \right\} \\
&= \max_{\gamma \geq 0} \left\{ \sum_{i \in I} \sum_{j \in J} \gamma_{ij} b_{ij} + \min \left\{ \sum_{i \in I} \sum_{j \in J} (c_{ij} - \gamma_{ij}) (\sum_{s \in \Omega} \delta_{ijs} x_s) \,|\, x_s \geq 0 \text{ and integer} \right\} \right\} \\
&= \max_{\gamma \geq 0} \left\{ \sum_{i \in I} \sum_{j \in J} \gamma_{ij} b_{ij} + \min \left\{ \sum_{s \in \Omega} (c_s - \sum_{i \in I} \sum_{j \in J} \gamma_{ij} \delta_{ijs}) x_s \,|\, x_s \geq 0 \text{ and integer} \right\} \right\} \\
&= \max_{\gamma \geq 0} \left\{ \sum_{i \in I} \sum_{j \in J} \gamma_{ij} b_{ij} + \min \left\{ \sum_{s \in \Omega} (c_s - \sum_{i \in I} \sum_{j \in J} \gamma_{ij} \delta_{ijs}) x_s \,|\, x_s \geq 0 \right\} \right\} \\
&= f_{LP}(D).
\end{aligned}$$

■

Since Dantzig's set covering model yields the same integrality gap as the models suggested in Aykin (1996), Bechtolds and Jacobs (1990) and Rekik et al. (2004) (see Section 2), Theorem 3 implies that the LP relaxation of P is also equivalent to the LP relaxations of these other implicit models. Note however that, to the best of our knowledge, these models cannot be extended to the multi-activity case.

5 Computational Experiments

The objective of our computational experiments is to evaluate the efficiency of our new implicit grammar-based model, when processed by a state-of-the-art IP solver. For this purpose, we will first compare model Q to the shift-based explicit model D and to another implicit model, due to Aykin (1996), under the same conditions. In order to compare ourselves to these other modeling approaches, we will first use instances from the literature, all having one work-activity. Then, we will present computational results on large-scale instances with multiple work-activities and compare our approach to employee-based explicit models tested by Côté et al. (2009) on the same instances.

5.1 Shift Scheduling with Multiple Rest Breaks, Meal Breaks, and Break Windows

In this section, we compare our model with a state-of-the-art implicit model, proposed in Aykin (1996), and to the shift-based explicit model D on a large set of shift scheduling instances used in Mehrotra et al. (2000) from shift specifications and labor requirements reported in Aykin (1996), Henderson and Berry (1976), Segal (1974) and Thompson (1995). The problems differ from one another in the labor requirements, the set of allowed shifts, the planning horizon, the number of breaks, the break windows, the cost structures, and whether the problem is cyclic or not. We refer to Mehrotra et al. (2000) for details on shift generation rules. Here, we present a general description of the three classes of problems studied.

Thompson Set. Thompson (1995) presents two sets of non-cyclic problems. The first set are problems on 15-h demand patterns. Shifts either allow one break or none depending on their length. The second set are problems on 20-h demand patterns. Shifts allow one break of one hour. The planning horizons are divided into periods of 15 minutes and shifts can start at any period that allows them to finish within the planning horizon. The break windows depend on the duration of the shifts and the costs are proportional to the number of work hours in a shift.

Aykin Set. Aykin (1996) presents a set of cyclic problems, with shifts containing exactly three breaks and differing only in the length of the break windows. The planning horizon is

24-h divided into periods of 15 minutes. All shifts have the same length and must start on the hour or the half-hour. The cyclic case is handled in the same way in the three modeling approaches. The planning horizon is extended to allow shifts to start at any time in the original planning horizon.

Mehrotra Set. Mehrotra et al. (2000) use the same shift generation rules as Aykin, but allow the shifts to have different durations and to start at any period. The problems were tested as cyclic problems using the same labor requirements as in the Aykin Set.

Definition of the Grammars. The following presents the grammars used for each set of instances. For the sake of clarity, the grammars are not stated in Chomsky normal form. In the sets of productions P, $\rightarrow_{[min,max]}$ restricts the subsequences generated with a given production to have a length between min and max periods.

Grammar for Thompson Set. Let Φ be the set of feasible shift types. Let bws_l and bwe_l be the break window starting and ending periods for shift type $l \in \Phi$. Let sl_l and bl_l be the shift and break lengths for shift type $l \in \Phi$.

Then $G = (\Sigma = (w, b, r), N = (S, W, R, A_l, B_l, M_l \ \forall l \in \Phi), P, S)$,

where w is a period of work, b is a break period and r is a rest period. P is defined as follows:

$$\begin{array}{lll}
S \rightarrow RA_lR \mid A_lR \mid RA_l & \forall l \in \Phi, & W \rightarrow Ww \mid w, \\
A_l \rightarrow_{[sl_l, sl_l]} M_lW & \forall l \in \Phi, & R \rightarrow Rr \mid r, \\
M_l \rightarrow_{[bws_l + bl_l, bwe_l + bl_l]} WB_l & \forall l \in \Phi, & \\
B_l \rightarrow b^{bl_l}, & \forall l \in \Phi. &
\end{array}$$

Grammar for Aykin and Mehrotra Sets. Let Φ be the set of feasible shift types. Note that in the Aykin Set Φ contains only one element, since all shifts have the same length. Let bws_l^n and bwe_l^n be the break window starting and ending periods for shift type $l \in \Phi$ and break $n \in \{1, 2, 3\}$. Let sl_l and bl_l^n be the length of the shift and of the breaks $n \in \{1, 2, 3\}$, respectively, for shift type $l \in \Phi$.

Then $G = (\Sigma = (w, b, r), N = (S, W, R, B_l^n \ \forall n \in \{1, 2, 3\}, A_l, M_l^A, M_l^B, M_l^C \ \forall l \in \Phi), P, S)$,

where w is a period of work, b is a break period and r is a rest period. P is defined as follows:

Table 2: Number of instances solved to optimality within the four minutes time limit

Models/Sets	Thom. 15-h (21)	Thom. 20-h (80)	Aykin (16)	Mehrotra (16)
Implicit Grammar	21	75	16	16
Aykin	21	78	16	16
Dantzig	21	76	16	16

$$\begin{array}{llll}
S \rightarrow RA_lR \mid A_lR \mid RA_l & \forall l \in \Phi, & B_l^n \rightarrow b^{bl_l^n} & \forall l \in \Phi, n \in \{1,2,3\}\,, \\
A_l \rightarrow_{[sl_l, sl_l]} M_l^A W & \forall l \in \Phi, & W \rightarrow Ww \mid w, & \\
M_l^A \rightarrow_{[bws_l^3+bl_l^3, bwe_l^3+bl_l^3]} M_l^B W B_l^3 & \forall l \in \Phi, & R \rightarrow Rr \mid r, & \\
M_l^B \rightarrow_{[bws_l^2+bl_l^2, bwe_l^2+bl_l^2]} M_l^C W B_l^2 & \forall l \in \Phi, & & \\
M_l^C \rightarrow_{[bws_l^1+bl_l^1, bwe_l^1+bl_l^1]} W B_l^1 & \forall l \in \Phi. & &
\end{array}$$

Results. We compare our model on these instances to the shift-based explicit model D derived from Dantzig (1954) and to the implicit model from Aykin (1996). We generated the three IP models and **solved them with CPLEX 12.1 with the default parameters**. As in Mehrotra et al. (2000), we gave a four minute time limit to find the optimal solution. Experiments were run on a 2.3GHz AMD Opteron with 3GB of memory.

Table 2 shows the number of instances solved to optimality within the four-minute time limit by the three models on each set. The numbers in parentheses are the number of available instances in each set.

For the instances solved to optimality by all three models, Table 3 presents a summary of the model sizes and solution statistics. $|C|$, $|V|$ and $|NZ|$ are the number of constraints, variables and non-zeroes in the models. Nit, $Nnodes$ and $Time(s)$ give the average values of the number of simplex iterations, the number of nodes and the time (in seconds) needed to solve the instances to optimality. $Gap(\%)$ is the average IP gap for the instances that were not solved to optimality **by at least one of the three models**, i.e., $Gap = 100(Z_{IP} - Z_{LP})/Z_{IP}$ where Z_{IP} and Z_{LP} are, respectively, the best upper and lower bounds after the time limit has been reached.

Tables 2 and 3 show that our approach lead to comparable results when compared to models from Aykin and Dantzig. However, overall, our model appears less adapted for these instances. For instances in the Aykin and the Mehrotra Sets, our model is less efficient than Aykin's, although it finds the

Table 3: Summary for the instances solved to optimality

Model	$\lvert C\rvert$	$\lvert V\rvert$	$\lvert NZ\rvert$	Nit	$Nnodes$	$Time(s)$	$Gap(\%)$
			Thompson set for 15-h demand curves				
Implicit Grammar	6418	9686	28481	2511	42	0.72	–
Aykin	421	3673	27521	2994	17	0.63	–
Dantzig	60	3312	85977	528	51	0.34	–
			Thompson set for 20-h demand curves				
Implicit Grammar	13432	21513	63768	11865	139	7.14	0.108
Aykin	850	8914	66742	15465	293	7.65	0.012
Dantzig	80	8144	223549	1451	170	2.98	0.045
			Aykin set				
Implicit Grammar	5703	36184	106902	1485	5	1.67	–
Aykin	274	697	3396	259	2	0.09	–
Dantzig	130	4681	149760	423	0	0.70	–
			Mehrotra et al. set				
Implicit Grammar	11201	16488	47683	6695	14	3.97	–
Aykin	1572	6961	33955	1971	2	0.93	–
Dantzig	132	46801	1497113	617	7	8.93	–

optimal solution for all instances within the time limit, as the two other models do. On the Thompson Set, with Aykin's and Dantzig's model, more instances are solved to optimality than with our modeling approach. Note that the average gap for the instances that were not solved to optimality by at least one model is however quite small.

Mehrotra et al. (2000) present a branch-and-price approach involving specialized branching rules for solving Dantzig set covering formulation. They compare their method with Aykin's model solved with CPLEX 4.0 on the same instances stated above. The results show that their method is generally superior. Since CPLEX has evolved considerably since these experiments were performed, it is difficult to deduce from these results a fair comparison between their approach and our model solved with CPLEX 10.1.1.

5.2 Shift Scheduling with Multiple Rest and Meal Breaks, and Multiple Work Activities

This section presents a shift scheduling problem for a retail store, allowing up to ten different work activities. We present the specifications of the problem and first compare our model with Dantzig's model and an extentsion of Aykin's model suggested in Rekik et al. (2005), allowing to model work-stretch duration restrictions for instances with one work-activity. Then, we report solution times from Côté et al. (2009) on the instances with up to two work-activities and compare them with the results from our model.

Problem Definition

1. The planning horizon is 24 hours divided into 96 periods of 15 minutes.
2. A shift may start at any period of the day allowing enough time to complete its duration during the planning horizon.
3. A shift must cover between 3 hours and 8 hours of work activities.
4. If a shift covers at least 6 hours of work activities, it must have two 15-minute breaks and a lunch break of 1 hour.

5. If a shift covers less than 6 hours of work activities, it must have one 15-minute break, but no lunch.

6. If performed, the duration of a work-activity is at least 1 hour (4 consecutive periods).

7. A break (or lunch) is necessary between two different work activities.

8. Work activities must be inserted between breaks, lunch and rest stretches.

9. For each period of the planning horizon, labor requirements for every work-activity are available.

10. Overcovering and undercovering are allowed. Costs are associated with overcovering and undercovering the requirements of a work-activity at a given period.

11. The cost of a shift is the sum over every period of the costs of all work-activities performed in the shift.

Definition of the Grammar. The following presents the grammar used for this problem. For the sake of clarity, the grammar is not stated in Chomsky normal form.
$G = (\Sigma = (a_j \ \ \forall j \in A, b, l, r), N = (S, F, P, W, A_j \ \ \forall j \in A, B, L, R), P, S)$,
where A is the set of work-activities, a_j is a period of work on activity $j \in A$, b is a break period, l is a lunch period and r is a rest period. In P, $\rightarrow_{[min,max]}$ restricts the subsequences generated with a given production to have a length between min and max periods. P is defined as follows:

$$\begin{array}{ll}
S \rightarrow RFR \mid FR \mid RF \mid RPR \mid PR \mid RP, & B \rightarrow b, \\
F \rightarrow_{[30,38]} WBWLWBW \mid WLWBWBW \mid WBWBWLW, & L \rightarrow llll, \\
P \rightarrow_{[13,24]} WBW, & R \rightarrow Rr \mid r, \\
W \rightarrow_{[4,\infty)} A_j \quad \forall j \in A, & \\
A_j \rightarrow A_j a_j \mid a_j \quad \forall j \in A. &
\end{array}$$

Results. To compare the different models for this problem, we generated the IP models representing these rules and solved them with CPLEX 10.1.1 with the default parameters. We gave a one-hour time limit to find the optimal solution. Experiments were run on a 3.20 GHz Pentium 4.

First, we compare Dantzig's model and the extension of Aykin's model suggested in Rekik et al. (2005), called the Aykin/Rekik model, to our model on ten instances with one work-activity, which differ only in their labor requirements. Table 4 presents the results. $|C|$, $|V|$ and $|NZ|$ are the number of constraints, variables and non-zeroes in the models. Note that the differences in the number of variables between the instances for the same model come from the slack variables introduced to allow overcovering and undercovering of requirements constraints. For periods where no employees are required, we suppose that the retail store is closed and that no work should be scheduled. Nit, $Nnodes$ and $Time(s)$ give the number of simplex iterations, nodes and the solution times (in seconds) for the instances solved to optimality within the time limit; otherwise the sign "$>$" is used.

The comparison between the three models shows that Dantzig's model tends to be less competitive on problems with a large number of shifts. In the one-activity case, solving the entire model with an IP solver is still manageable, but both the Implicit Grammar models and Aykin/Rekik models are solved more rapidly. Our model succeeds in proving optimality for 9 out of 10 instances, as does the Aykin/Rekik model, but does so in less time for 6 out of these 9 instances. Note also that the number of variables in the Implicit Grammar model is smaller than in the two other models.

Table 5 shows our results on the multi-activity instances. We ran experiments on our model on instances ranging from two to ten work activities. For each instance, we tested ten different labor requirements. Column $NbShifts$ gives the number of feasible shifts for each of the problems, which would be the number of variables needed by Dantzig's set covering model. $Nopt$ gives the number of instances solved to optimality within the one-hour time limit. $|C|$, $|V|$ and $Time(s)$ are the average number of constraints and variables, and solution times (in seconds) for the instances solved to optimality.

To our knowlegde, no other implicit formulations are capable of modeling multi-activity instances. To solve Dantzig's model on these problems, one must consider column generation methods, since the number of feasible shifts is very large. Demassey et al. (2006) present a column generation approach for these problems. However, their method does not succeed in finding optimal solutions, even for the single-activity instances. As for our modeling

Table 4: Model comparison on the one-activity problem

No	$\|C\|$	$\|V\|$	$\|NZ\|$	Nit	$Nnodes$	$Time(s)$
			Implicit Grammar model			
1	16191	66621	198517	137	0	0.27
2	16191	66653	198549	2585	0	12.52
3	16191	66653	198549	817435	702	767.75
4	16191	66637	198533	61515	201	16.44
5	16191	66629	198525	1219	0	0.45
6	16191	66629	198525	975	0	0.36
7	16191	66637	198533	153688	544	35.13
8	16191	66653	198549	>	>	>
9	16191	63068	198525	9027	250	1.31
10	16191	66637	198533	1767	0	0.73
			Aykin/Rekik model			
1	50007	78247	930056	593	0	2.47
2	50007	78279	930088	248660	389	371.76
3	50007	78279	930088	324511	673	461.69
4	50007	78263	930072	13868	235	5.17
5	50007	78255	930064	647	0	2.22
6	50007	78255	930064	882	0	2.20
7	50007	78263	930072	6015	175	3.41
8	50007	78279	930088	>	>	>
9	50007	78255	930064	31460	360	12.38
10	50007	78263	930072	1943	0	2.59
			Dantzig model			
1	96	845176	24605722	23	0	24.58
2	96	845208	24605754	>	>	>
3	96	845208	24605754	11808	728	1824.11
4	96	845192	24605738	1180	141	52.06
5	96	845184	24605730	85	0	29.86
6	96	845184	24605730	30	0	29.44
7	96	845192	24605738	7455	1209	51.23
8	96	845208	24605754	>	>	>
9	96	845184	24605730	3769	280	35.68
10	96	845192	24605738	154	0	32.86

Table 5: Multi-activity problems with the Implicit Grammar model

$NbAct$	$NbShifts$	$\|C\|$	$\|V\|$	$Time(s)$	$Nopt(10)$
2	13404928	18068	69893	409.07	10
3	67752783	19945	73152	205.38	9
4	214010944	21822	76417	300.47	10
5	522350575	23699	79688	146.16	10
6	1082991744	25576	82961	213.79	10
7	2006203423	27453	86246	230.88	10
8	3422303488	29330	89492	257.06	10
9	5481658719	31207	92731	289.08	10
10	8354684800	33084	96026	516.74	10

approach, the multi-activity problems can easily be handled with a few more productions than in the one-activity case, and results show that they can rapidly be solved on almost all available instances. Note that the growth in the number of constraints and variables when increasing the number of work activities is much slower than the increase in the number of feasible shifts.

Comparison with existing IP formulations based on formal languages. Table 6 presents the times reported by Côté et al. (2009) to solve the one and two work-activities instances with two employee-based explicit formulations, the *IP Regular* model, based on a finite automaton, and the *IP Grammar* model, based on a context-free grammar. In both cases, 0-1 assignment variables for each employee are used, instead of the general integer variables used in model Q. These experiments were run on a 2.4 GHz Dual AMD Opteron Processor 250 with 3 GB of RAM, using the MIP solver CPLEX 10.0 and a time limit of 3600 seconds. **In this experiment, we also used CPLEX 10.0 to solve our implicit grammar models to make a more fair comparison.** The table reports the time in seconds to obtain an integer solution with a relative IP gap smaller or equal than 1%. The symbol “>” represents an instance for which that gap could not be reached within the time limit. The *Implicit Grammar* column shows the time needed by the implicit grammar model to reach the 1% relative IP gap limit.

Table 6 illustrates that the two employee-based explicit formulations suffer from scal-

Table 6: Comparison of solution times (seconds) between employee-based explicit formulations and the implicit grammar model on the one and two-activities instances to obtain a near-optimal solution ($Gap \leq 1\%$) in less than 3600 seconds

No	IP Regular	IP Grammar	Implicit Grammar
One-activity instances			
1	1.03	7.42	0.26
2	40.09	>	110.88
3	64.64	>	75.25
4	46.39	1850.38	2.75
5	14.03	322.57	0.48
6	3.28	130.21	0.34
7	5.99	1662.75	2.71
8	131.77	>	2642.12
9	16.14	1015.10	1.18
10	20.22	1313.28	0.80
Two-activities instances			
1	228.07	2826.40	1.27
2	2870.20	1952.58	4.12
3	1541.15	>	81.91
4	169.96	>	16.27
5	>	>	2.59
6	1288.56	>	51.16
7	29.94	>	0.60
8	>	325.08	36.20
9	>	>	>
10	1108.23	>	4.99

ability issues as the number of work-activities grows. Observe that the implicit grammar model shows comparable solution times for both classes of instances. For the one-activity problem, the *IP Regular* model is much faster than the implicit grammar model on 3 out of 10 instances. However, for the instances with two work-activities, the implicit grammar model solves all instances more rapidly than the two other models, and succeeds in solving more instances within the time limit. It is interesting to note that, on almost all instances, the *IP Regular* model has better solution times than the explicit *IP Grammar* model, from which we built the implicit model presented in this paper.

6 Conclusion

In this paper, we presented a new implicit IP model for solving multi-activity shift scheduling problems. This model differs significantly from the models proposed in the literature, as our modeling approach uses context-free grammars to represent the constraints defining feasible shifts. This model yields the same LP relaxation bound as the classical set covering model by Dantzig (1954) and the other well-known implicit models in the literature, i.e., Aykin (1996), Bechtolds and Jacobs (1990) and Rekik et al. (2004).

Our experiments showed that the solution times for our model are **comparable** with the solution times for Aykin's model on one-activity shift scheduling instances from the literature, and slightly superior to an extension of Aykin's model suggested in Rekik et al. (2004) on large-scale one-activity instances. We also showed that our model can be solved to optimality efficiently on instances with up to ten work activities. To the best of our knowledge, no other technique in the literature can solve multi-activity instances efficiently.

An interesting feature of our formulation is that the objective function allows many types of cost structures, contrary to classical implicit formulations. In model Q, costs are associated with activities and periods, but one can modify the objective function to have costs on productions ($\sum_{A_{il}^{\Pi,t} \in A} \text{cost}(v_{il}^{\Pi,t}) v_{il}^{\Pi,t}$), on the root-node (minimizing the root-node variable, would be equivalent to minimizing the number of employees needed), or even on subsequences. Indeed, $\sum_t v_{il}^{\Pi,t}$ corresponds to the number of words having a subsequence of length l starting in position i generated from production Π. A subsequence can be a shift or a part of a shift, such as a task. One could be interested in tracking a task corresponding to a consecutive assignment of work activities and do so by using the corresponding variables v from A. **The same idea can be used to model a switch-over or ramp-up (ramp-down) effects.** Therefore, the implicit grammar-based model can easily be extended to handle problems that require the simultaneous assignment of tasks and activities.

APPENDIX

Schedule Construction

In the following, $V(N)$ is initialized to the value of the variable associated to node N in the implicit solution. $c_l(N)$ and $c_r(N)$ are the left and right children of and-node N. $c(N)$ is the set of children of or-node N. *schedule* is the schedule resulting from the algorithm, i.e., the shift assigned to each employee. L is the set of leaves in the DAG.

```
Data: Solution from an implicit grammar model
Result: Detailed schedule
Stack $K = \emptyset$ ;
Array $schedule[V(O^S_{1n}), n]$ ;
$e = 0$ ;
while $V(O^S_{1n}) > 0$ do
    $V(O^S_{1n}) = V(O^S_{1n}) - 1$ ;
    Choose $N \in \{c(V(O^S_{1n})) \mid V(N) > 0\}$ ;
    Push $N$ on $K$;
    while $K \neq \emptyset$ do
        Pop $N$ from $K$;
        $V(N) = V(N) - 1$;
        if $c_l(N) \in L$, $c_l(N)$ corresponds to $O^j_{i1}$ then
            $schedule[e, i] = j$;
        else
            Choose $N_l \in \{c(c_l(N)) \mid V(N_l) > 0\}$ ;
            Push $N_l$ on $K$;
            Choose $N_r \in \{c(c_r(N)) \mid V(N_r) > 0\}$;
            Push $N_r$ on $K$;
        end
    end
    $e = e + 1$;
end
```

Algorithm 1: Extracting detailed schedules from an implicit grammar solution

Acknowledgments

This work was supported by a grant from the Fond québécois de recherche sur la nature et les technologies. We would like to thank Claude-Guy Quimper for his useful comments on our work. We would like to thank two anonymous referees whose constructive comments and questions helped us to improve our paper.

References

Aykin, T. 1996. Optimal shift scheduling with multiple break windows. *Management Science* **42** 591–602.

Bechtolds, S., L. Jacobs. 1990. Implicit optimal modeling of flexible break assigments. *Management Science* **36** 1339–1351.

Bouchard, M. 2004. Optimisation des pauses dans le problème de fabrication des horaires avec quarts de travail. M.Sc. Thesis, Ecole Polytechnique de Montréal.

Côté, M.-C., B. Gendron, C.-G. Quimper L.-M. Rousseau. 2009. Formal languages for integer programming modeling of shift scheduling problem. *Constraints* **doi:10.1007/s10601-009-9083-2**.

Côté, M.-C., B. Gendron, L.-M. Rousseau. 2007. Modeling the regular constraint with integer programming. *Proc. of CPAIOR'07, Springer-Verlag LNCS* **4510** 29–43.

Dantzig, G. 1954. A comment on Edie's traffic delay at toll booths. *Journal of the Operations Research Society of America* **2** 339–341.

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

Edie, L. 1954. Traffic delays at toll booths. *Journal of the Operations Research Society of America* **2** 107–138.

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

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

Henderson, W.B., W.J. Berry. 1976. Heuristic methods for telephone operator shift scheduling. *Management Science* **22** 1372–1380.

Hopcroft, J., R. Motwani, J. D. Ullman. 2001. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley.

Lequy, Q., M. Bouchard, G. Desaulniers, F. Soumis. 2009. Assigning multiple activities to work shifts. Tech. rep., Les Cahiers du GERAD G-2009-86, HEC Montreal, Montreal, Canada.

Loucks, J.S., F.R. Jacobs. 1991. Tour scheduling and task assignment of a heterogeneous work force: a heuristic approach. *Decision Sciences* **22** 719–739.

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

Pesant, Gilles, C.-G. Quimper, L.-M. Rousseau, M. Sellmann. 2009. The polytope of context-free grammar constraints. *Proc. of CPAIOR'09, Springer-Verlag LNCS* **5547** 223–232.

Quimper, C.-G., T. Walsh. 2007. Decomposing global grammar constraint. *Proc. of CP'07, Springer-Verlag LNCS* **4741** 590–604.

Rekik, M., J.-F. Cordeau, F. Soumis. 2004. Using benders decomposition to implicitly model tour scheduling. *Annals of Operations Research* **128** 111–133.

Rekik, M., J.-F. Cordeau, F. Soumis. 2005. Implicit shift scheduling with multiple breaks ans work stretch duration restrictions. *Publication GERAD-2005-15* .

Ritzman, L., L.J. Krajewski, M.J. Showalter. 1976. The disaggregation of aggregate manpower plans. *Management Science* **22** 1204–1214.

Segal, M. 1974. The operator-scheduling problem: A network-flow approach. *Operations Research* **22** 808–823.

Thompson, G. 1995. Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science* **41** 595–607.

Vatri, E. 2001. Integration de la génération de quart de travail et de l'attribution d'activités. M.Sc. Thesis, Ecole Polytechnique de Montréal.