CIRRELT

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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Formal Languages for Integer Programming Modeling of Shift Scheduling Problems

Marie-Claude Côté
Bernard Gendron
Claude-Guy Quimper
Louis-Martin Rousseau

December 2007

CIRRELT-2007-64

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

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

Formal Languages for Integer Programming Modeling of Shift Scheduling Problems

Marie-Claude Côté[1,2], Bernard Gendron[2,3], Claude-Guy Quimper[4], Louis-Martin Rousseau[1,2,*]

[1] Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

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

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

[4] Omega Optimisation, 4200 St-Laurent, suite 301, Montréal, Canada H2W 2R2

Abstract. This paper approaches the problem of modeling optimization problems containing substructures involving constraints on sequences of decision variables. Such constraints can be very complex to express with Mixed Integer Programming (MIP). We suggest an approach inspired by global constraints used in Constraint Programming (CP) to use formal languages for the modeling of such substructures with MIP. More precisely, we first suggest a way to use automata, as the CP regular constraint does, to express allowed patterns for the values taken by the constrained sequence of variables. Secondly, we present how context free grammars can contribute to formulate constraints on sequences of variables in a MIP model. Experimental results on both approaches show that they can not only facilitate the modeling, but also give models easier to solve by MIP solvers compared to classical MIP formulations.

Keywords. Constraint programming, integer programming, reformulations, formal languages.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: Louis-Martin.Rousseau@cirrelt.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2007

© Copyright Côté, Gendron, Quimper, Rousseau and CIRRELT, 2007

1. Introduction

Given a sequence of n decision variables X_i, each with a finite domain D_i, $i = 1, \ldots, n$, a constraint on such a sequence is a set of n-tuples $L \subseteq D_1 \times \ldots \times D_n$ called a *language*. The constraint over the sequence is satisfied when the tuple $\langle X_1, \ldots, X_n \rangle \in L$ belongs to the language. Such constraints arise in many optimization and satisfaction problems. In this paper, we focus on shift scheduling problems, where a sequence of activities (work activities, break, lunch, rest) must be assigned to a set of employees. In these problems, the difficulty lies in building shifts that comply with work regulations such legal placement of breaks and lunches, and transitions between activities.

In this paper, we study how to model constraints on sequences of decision variables using a Mixed Integer Programming (MIP) framework. Our approach is inspired by global constraints in Constraint Programming (CP) that use formal languages. First, we suggest using automata to represent constraints on sequences of decision variables, as the CP `regular` [22] constraint does. From the automaton, we automatically generate a network flow model that can be included into any MIP model involving constraints on sequences of decision variables. Second, we propose a way to use context-free grammars instead of automata to describe the constraints on sequences of decision variables. To apply this to MIP, we use an and/or graph structure associated to the CP `grammar` constraint [24, 28, 25] and derive the associated linear constraints.

These approaches allow MIP to benefit from CP expressiveness in modeling, by automatically generating MIP models from intuitive modeling tools, such as automata and context-free grammars. Furthermore, our experimental results on a shift scheduling model show that they can not only facilitate the modeling, but also give models easier to solve by MIP solvers compared to classical MIP formulations.

The paper is organized as follows. In Section 2, we present a literature review on shift scheduling problems. Section 3 presents some background material on formal languages and their use in CP. In Sections 4 and 5, we introduce our two approaches to model constrained sequences of decision variables: the `MIP regular` and the `MIP grammar` constraints respectively. Finally, in Section 7, we compare different formulations on a shift scheduling problem.

2. Shift Scheduling Problems

Given a planning horizon divided into periods of equal length, a set of employees and a demand for different activities (work activities, lunch, break, rest) at each period, the shift scheduling problem consists of assigning an activity to each employee at each period in such a way that the demands are met, while optimizing an objective and satisfying several rules (including some that can be expressed as constraints on sequences of decision variables). In this context, a *shift* is a sequence of activities corresponding to a countinuous presence at work (that may include lunch and break, but not rest periods). A *schedule* (also called a *tour*) is a sequence of shifts and rest periods, over the whole planning horizon, that satisfies all the rules associated to an employee. A *pattern* is a sequence of activities that respects some of the rules over a subset of the planning horizon.

Mathematical programming models for shift scheduling problems can be divided into three categories (see [15, 14] for recent surveys on shift scheduling and related problems): the compact assignment formulations, the explicit set covering formulations and the implicit set covering formulations. Compact assignment formulations [18, 4, 5] use decision variables to assign activities to each employee at each period. In the explicit set covering formulations, the decision variables represent all possible shifts and the problem is to select a subset of them which covers the demands. The number of shifts being potentially large, different methods were proposed to select good subsets. Most notably, the column generation method efficiently solves this kind of problems (see for instance [19, 8, 12]).

Implicit set covering formulations were introduced and developed by Moondra [21], Bechtold and Jacobs [6, 7], Thompson [30], Aykin [2, 3], and Rekik et al. [27, 26]. In these models, shift types, specified by starting and ending times, are not directly associated with break positions at first. For instance, one can independently decide how many employees are going to work from 8am to 4pm and how many employees are going to be on break at 10am. Additional constraints, named forward and backward constraints, are necessary to guarantee the existence of a valid schedule which can later be reconstructed with a polynomial-time algorithm. The main advantage of this approach is that the number of decision variables is significantly reduced compared to explicit set covering formulations.

Network flow formulations were used for different generalizations of shift scheduling problems. Çezik et al. [9] propose a MIP formulation for the Weekly Tour Scheduling Problem. It handles the weekly horizon by combining seven daily shift scheduling models in a net-

work flow framework, which handles the demands for each day. Millar and Kiragu [20] and Ernst et al. [13] use a layered network to represent allowed transitions between a set of a priori patterns (series of nights for instance) to develop complete schedules. Sodhi [29] studies the problem of assigning a type of shift (day, evening, night) to each day of a planning horizon of several weeks. The model combines predefined weekly patterns to create a complete schedule by using a directed graph with nodes representing allowed weekly patterns and arcs corresponding to allowed week-to-week transitions between these patterns. A MIP model is then used to find an optimal cyclic path to cover all the weeks of the schedule.

In this paper, we present two different generic ways to capture and model a large set of rules that can be expressed as constraints on sequences of decision variables. We compare the resulting formulations to a compact assignment model for a shift scheduling problem.

3. Background Material

Before we define our modeling approaches, we introduce important definitions related to formal languages theory (for more details on the subject, see Hopcroft et al. [16]).

3.1. Automata and the CP Regular Constraint

A deterministic finite automaton (DFA) is described by a 5-tuple $\Pi = \langle Q, \Sigma, \delta, q_0, F \rangle$ where:

- Q is a finite set of states;
- Σ is an alphabet;
- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function;
- $q_0 \in Q$ is the initial state;
- $F \subseteq Q$ is a set of final states.

An alphabet is a finite set of symbols. A language is a set of words, formed by symbols over a given alphabet. Regular languages are languages recognized by a DFA. A word is recognized by a DFA if by processing its symbols one by one from the initial state using the transitions, we find ourselves in a final state after we process the last symbol.

A non-deterministic finite automaton (NFA) distinguishes itself from a DFA by its set of transitions δ. In fact, a transition is no longer a function but a set of triplets : $\delta \subseteq Q \times \Sigma \times Q$.

A transition $\langle q_1, j, q_2 \rangle \in \delta$ indicates that reading the symbol j from state q_1 can lead to state q_2. However, it is possible that another transition from q_1, $\langle q_1, j, q_3 \rangle \in \delta$, leads to another state q_3, hence the non-determinism of the automaton. DFAs and NFAs strictly encode the same languages. However, NFAs can encode some languages with exponentially fewer states than DFAs.

Example 1 *Let $\Sigma = \{a, b, c\}$ be an alphabet. Π, represented in Figure 1, is a DFA recognizing a regular language over this alphabet. This DFA recognizes, for instance, the words c, cccc, aba, aabba, but does not recognize ac and ab.*

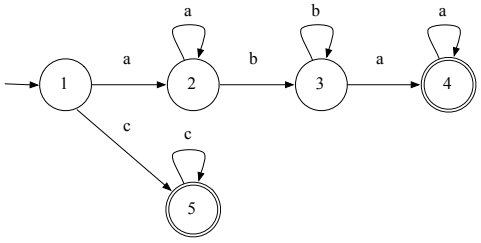

Figure 1: DFA Π with each state shown as a circle, each final state as double circle, and each transition as an arc.

Pesant [22] introduced the constraint $\texttt{Regular}([X_1, \ldots, X_n], \Pi)$ which is satisfied if the automaton Π recognizes the sequence of decision variables $X_1, \ldots, X_n$.

3.2. Context-free Grammars and the CP Grammar Constraint

A *context-free grammar* G is a tuple $\langle \Sigma, N, S, P \rangle$ where Σ is the alphabet of characters also called the *terminal symbols*, N is a set of *non-terminal symbols*, $S \in N$ is the starting symbol, and P is a set of productions of the form $A \rightarrow w$ where $A \in N$ is a non-terminal symbol and w is a sequence of terminal and non-terminal symbols. We use capital letters for non-terminal symbols and lower case letters for terminal symbols. A *parsing tree* is a tree where each leaf is labeled with a terminal and each inner-node is labeled with a non-terminal. The root is labeled with the starting symbol S. The children of a node A, when listed from left to right, form a sequence w such that the production $A \rightarrow w$ belongs to the grammar. A grammar recognizes a sequence if and only if there exists a parsing tree where the leaves, when listed from left to right, reproduce this sequence. Any grammar can be written in its *Chomsky normal form* i.e., any production either generates two non-terminals

or one terminal. A *context-free language* is the set of sequences accepted by a context-free grammar.

Context-free grammars are more expressive than automata since any regular language can be encoded with a context-free grammar but not every context-free language can be encoded with an automaton [16].

Quimper and Walsh [24], and Sellmann [28] introduced the constraint $\texttt{grammar}([X_1, \ldots, X_n], G)$ which is satisfied if the grammar G recognizes the sequence of decision variables $X_1, \ldots, X_n$.

Given a context-free grammar G, Quimper and Walsh [25] build a Boolean formula that returns *true* for every sequence recognized by the grammar and *false* for any other sequence. This Boolean formula is encoded in an and/or graph where each leaf corresponds to an assignment $X_i = t$ that can either be *true* or *false*. An or-node is *true* if one of its children is *true*. An and-node is *true* if all its children are *true*. The root is *true* if the grammar G accepts the sequence encoded by the leaves. Their algorithm (see Algorithm 1) builds the and/or graph and is based on the CYK parser (see [10, 17, 31]) that takes as input a grammar written in its normal Chomsky form. The and/or graph embeds every possible parsing tree of a grammar. Each or-node $N(A, i, j)$ in the graph is assigned to *true* if the non-terminal A produces the sub-sequence $X_i, \ldots, X_{i+j-1}$. The nodes set to *true* in a solution form a parsing tree. Example 2 describes the and/or graph of a simple grammar.

Example 2 *Consider the following simple grammar taken from [25].*

$$S \rightarrow AB \qquad A \rightarrow aA \mid a \qquad B \rightarrow bB \mid b$$

Algorithm 1 builds the graph depicted in Figure 2.

Quimper and Walsh [25] show how context-free grammars can be enhanced by imposing some constraints on a production $A \rightarrow BC$. For instance, a non-literal can be constrained to produce a sequence of a given length or only be produced at given positions. Such constraints simply remove some nodes in the and/or graph.

4. MIP Regular

The use of automata to express constraints on values taken by sequences of variables is very useful in CP. Equivalent constraints can be very complex to formulate in a MIP 0-1 compact model. The aim of our work in this section is precisely to propose a way to formulate

for *all non-terminals* A **do**
  for $i \in [1, n]$ **do**
1 Create or-node $N(A, i, 1)$
    for $A \rightarrow t \in G$ **do**
      Create the leaf node $N(t, i, 1)$ if it does not already exist.
      Create an and-node with child $N(t, i, 1)$ and parent $N(A, i, 1)$.

for $j \in [2, n]$ **do**
  for $i \in [1, n - j + 1]$ **do**
    for *all non-terminals* A **do**
      Create or-node $N(A, i, j)$
      // Create a list of children composed of and-nodes
2 $Children(N(A, i, j)) \leftarrow \{N(B, i, k) \wedge N(C, i + k, j - k) \mid$
          $k \in [1, j), \quad A \rightarrow BC \ \in \ G,$
          $Children(N(B, i, k)) \neq \emptyset,$
3 $Children(N(C, i + k, j - k)) \neq \emptyset\}$

4 Delete any node that does not have $N(S, 1, n)$ for ancestor.

Algorithm 1: This algorithm based on the CYK parser [10, 17, 31] constructs a graph embedding all possible parsing trees of sequences of length n recognized by the grammar G.

MIP 0-1 compact models by using automata. Our approach is inspired by the CP `regular` constraint [22].

First, we introduce the following 0-1 decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if position } i \in I \text{ of the sequence is assigned to value } j \in D_i, \\ 0, & \text{otherwise.} \end{cases}$$

where $I = \{1, 2, \ldots, n\}$ represents the set of positions in the sequence and D_i, the set of values that can be assigned to this position.

To obtain a graph structure representing all sequences of length n recognized by an automaton, we use the following property of regular languages:

- Let L_1 and L_2 be two regular languages. Then $L_1 \cap L_2$ is a regular language.

Given this property, if we have an automaton A_1 that encodes a set of constraints on the values taken by a sequence of variables and an automaton A_2 that encodes the language specifying all sequences of length n on the same set of values, the conjunction of A_1 and A_2 results in an automaton A recognizing all sequences of length n recognized by A_1. Automaton

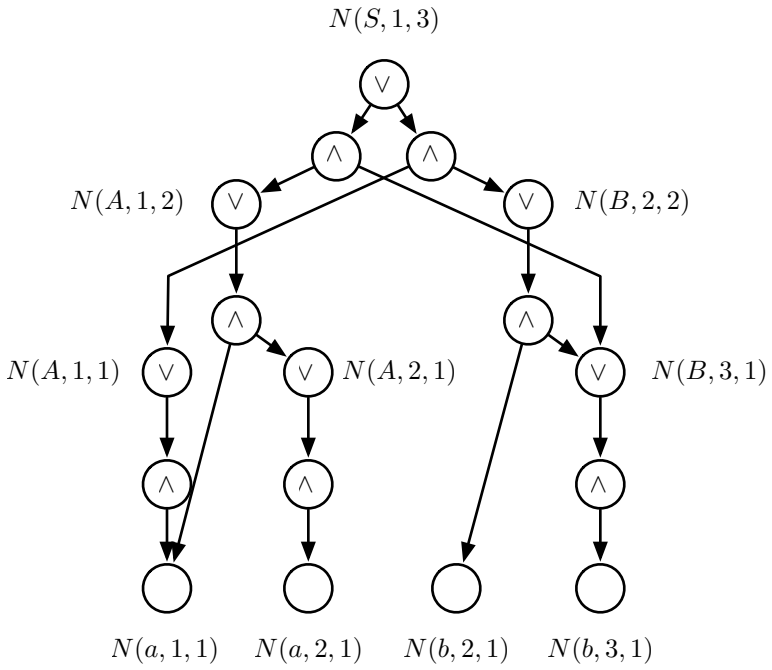

Figure 2: And/or tree constructed by Algorithm 1 on the grammar of Example 2 and a sequence of length $n = 3$.

A has a special structure. It is a directed layered graph, with $n + 1$ layers and no cycles. Each layer potentially contains all states of A_1. Let $N^1, N^2, \ldots, N^{n+1}$ be the sets of states of each layer. We note that N^1 has a single element, the initial state of A_1, and that N^{n+1} is a subset of the set of final states of A_1. Pesant [22] shows how to build A.

Since A recognizes all sequences of length n recognized by A_1, our modeling approach uses this structure to derive a network flow formulation. The correspondence between the automaton A and the graph G used for the network flow model is direct. First, a state $k \in N^i$, $1 \leq i \leq n + 1$, is a node in G and a transition in A is an arc in G. A transition between a state $k \in N^i$ and a state $l \in N^{i+1}$ labeled with symbol j defines a unique arc in G representing the value j assigned to position i in the sequence. For all such arcs in G, we have a *flow variable* f_{ijkl} (see [1] for details on network flow theory). Notice that if A_1 is a DFA, the index l is not needed, but our approach also applies to an NFA. Finally, we identify s, the unique element of N^1, as the *source* node, and we link each node $k \in N^{n+1}$ to a *sink* node t with an arc labeled with the flow variable $f_{(n+1)kt}$. We also define a 0-1 variable w that specifies if the constraint is active or not. The value of w corresponds to the amount of flow (0 or 1) entering and leaving the graph.

Example 3 *Let* Π *be the automaton of Example 1 represented in Figure 1. Let* π_5 *be the automaton depicted in Figure 3 representing all sequences of length* $n = 5$ *on alphabet* $\Sigma = \{a, b, c\}$*. Then, Figure 4 presents automaton* $A = \Pi \cap \pi_5$ *and Figure 5, the associated graph* G*.*

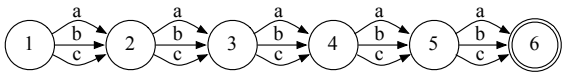

Figure 3: Automaton π_5 recognizing all sequences of length 5 on alphabet $\Sigma = \{a, b, c\}$

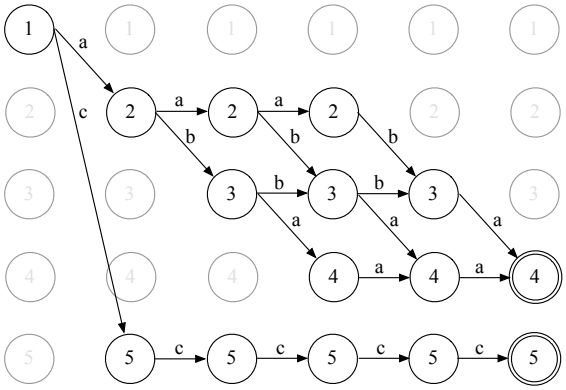

Figure 4: Automaton $A = \Pi \cap \pi_5$

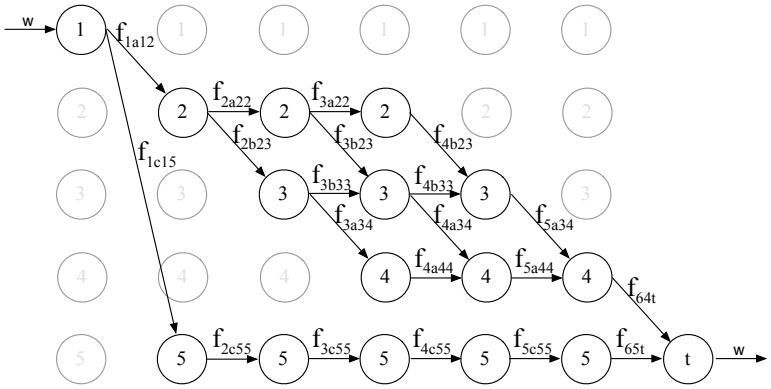

Figure 5: Graph G associated to automaton A

The network flow problem on G is a set of linear constraints, the flow conservation equations, ensuring that for each node in the graph, the amount of flow entering and leaving

the node is the same. An arc from $k \in N^i$ to $l \in N^{i+1}$ with label j is defined as a quadruplet (i, j, k, l). For each node $k \in N^i$, we introduce the sets of outgoing and incoming arcs:

$$\Delta_{ik}^{+} = \{(i, j, k, l) | l \in N^{i+1} \text{ and } < k, j, l > \in \delta_i\},$$

$$\Delta_{ik}^{-} = \{(i-1, j, l, k) | l \in N^{i-1} \text{ and } < l, j, k > \in \delta_{i-1}\},$$

where δ_i is the set of transitions at layer i, $1 \leq i \leq n+1$. The MIP formulation of the `regular` constraint is then written as follows:

$$\sum_{(j,l)|(1,j,s,l)\in\Delta_{1s}^{+}} f_{1jsl} = w, \tag{1}$$

$$\sum_{(j,l)|(i-1,j,l,k)\in\Delta_{ik}^{-}} f_{(i-1)jlk} = \sum_{(j,l)|(i,j,k,l)\in\Delta_{ik}^{+}} f_{ijkl}, \qquad \forall i \in \{2, \ldots, n\}, k \in N^i, \tag{2}$$

$$\sum_{(j,l)|(n,j,l,k)\in\Delta_{(n+1)k}^{-}} f_{njlk} = f_{(n+1)kt}, \qquad \forall k \in N^{n+1}, \tag{3}$$

$$\sum_{k\in N^{n+1}} f_{(n+1)kt} = w, \tag{4}$$

$$x_{ij} = \sum_{(k,l)|<k,j,l>\in\delta_i} f_{ijkl}, \qquad \forall i \in \{1, \ldots, n\}, j \in D_i, \tag{5}$$

$$f_{ijkl} \in \{0, 1\} \qquad \forall i \in \{1, \ldots, n\}, < k, j, l > \in \delta_i, \tag{6}$$

$$f_{(n+1)kt} \in \{0, 1\} \qquad \forall k \in N^{n+1}. \tag{7}$$

Constraints (5) link the *decision variables* x with the *flow variables*. Note that in the case where the `MIP regular` constraint is the only constraint in the model, the decision variables x and constraints (5) are not needed in the model. Without these, the resulting model is a pure network flow formulation that reduces to the determination of a path between s and t in an acyclic network, which can be solved very efficiently by a specialized algorithm [1]. In the case where the `MIP regular` constraint is part of a larger model, constraints (5) allow to formulate the rest of the model using the decision variables x.

Thus, introducing a `MIP regular` constraint to a MIP model induces the addition of a set of flow conservation linear constraints (1)-(4) and linking constraints (5) to the model. We use a procedure with the following signature:

$$\texttt{AddMIPRegular}(\Pi(Q,\Sigma,\delta,q_0,\, F),\, n,\, x,\, w,\, M),$$

to add the linear constraints associated with a `MIP regular` constraint to a model M, given a DFA Π, the decision variables x subject to the constraint, the length of the sequence n formed by these variables and the amount of flow w entering the graph.

5. MIP Grammar

As we did with regular languages, we derive a MIP that accepts any sequence belonging to a context-free language. The model directly comes from the and/or graph presented in Section 3.2. Each node $N(A, i, j)$ in the and/or graph corresponds to a MIP 0-1 variable $X(A, i, j)$. A leaf node $N(t, i, 1)$ is associated to the MIP 0-1 decision variable x_{it}. The variable x_{it} is equal to one if and only if the node $N(t, i, 1)$ is *true*. A leaf node is considered as an or-node in the graph. In the following, the notations N_{or} and N_{and} refer to general or-node and and-node, while the notations X_{or} and X_{and} refer to their associated 0-1 variables. As we did for the `MIP regular` constraint, we introduce a 0-1 variable w that specifies if the constraint is active. When $w = 0$, every variable x_{it} must be assigned to zero.

The constraints on the variables of the MIP depend on the relationship between the corresponding nodes in the and/or graph. There is one significant difference between the MIP and the and/or graph. When there exist more than one parsing tree for a sequence, all nodes in the and/or graph that belong to at least one parsing tree are set to *true*, while for the MIP, one parsing tree is arbitrarily selected and all its variables are set to one. All other variables, including those that belong to other parsing trees, are set to zero. Choosing an arbitrary parsing tree simplifies the MIP without changing the solution space. Indeed, only one parsing tree is necessary to prove that a sequence belongs to a context-free language. We now present the constraints representing a MIP of a `grammar`.

Let N_{or} be an or-node other than a leaf node. Let $c(N_{or})$ be its children's label. The following constraint forces X_{or} to be equal to one if exactly one of the variables associated to the children of N_{or} is equal to one:

$$X_{or} = \sum_{n \in c(N_{or})} X_{and,n} \tag{8}$$

A node belongs to a parsing tree only if exactly one of its parent belongs to the parsing tree. Let N_{or} be an or-node and $p(N_{or})$ be its parents label. We have the following equality:

$$X_{or} \quad = \quad \sum_{n \in p(N_{or})} X_{and,n} \tag{9}$$

Finally, we force the root of the directed acyclic graph (DAG) to be assigned to one if and only if the constraint is active, i.e. when $w = 1$:

$$X(S, 1, n) \quad = \quad w \tag{10}$$

.

Lemma 1 *Constraints (8), (9), and (10) are satisfied if and only if the and/or graph evaluates to* true.

Proof: Observe that all parents and children of an or-node are and-nodes and all parents and children of an or-node are and-nodes. And-nodes have a unique parent.

($\Longrightarrow$) Suppose there exists a valid assignment of every node in the and/or graph such that the root is assigned to *true*. Quimper and Walsh [25] showed that there exists at least one parsing tree whose nodes in the graph are assigned to *true*. We arbitrarily select one such parsing tree and set to one every variable whose corresponding or-node belongs to the parsing-tree. Consider an or-node in the parsing tree. Among all its children assigned to *true*, we arbitrarily select one and-node and set its corresponding variable to one. All unassigned variables remaining in the MIP are set to zero. The constraint (8) is satisfied since an or-node is set to one if exactly one child is set to one. The constraint (9) is also satisfied since in the parsing tree, each node has only one parent (except for root). Finally, constraint (10) is satisfied since the root node is set to one.

($\Longleftarrow$) Suppose the MIP is feasible. For every variable assigned to one, we assign the corresponding node to *true*. By constraints (8), every or-node has one child set to *true*. By constraint (9), every or-node has one parent set to *true*. This parent is an an-node that has either one or two children. In either case, constraint (9) ensures that a node that has a parent set to *true* is also set to *true*. Therefore, the children of an and-node set to *true* are also set to *true*. Finally, notice that the root node of the tree is set to *true* thanks to constraint (10). We proved that every variable set to one in the MIP have its corresponding node set to *true* in the graph. We now prove that values set to zero in the MIP can be assigned to Boolean values in the graph. We set to *false* every leaf node whose corresponding MIP variable is

set to zero. Every unassigned node is evaluated using a bottom-up approach, i.e., and-nodes are assigned to *true* if both children are *true* and or-nodes are assigned to *true* if at least one child is *true*. Therefore, we obtain a valid assignment of Boolean values in the and-or graph. ■

Since Quimper and Walsh [25] proved that the grammar recognizes a sequence if and only if the and/or graph evaluates to *true*, we conclude that the grammar accepts only the sequences satisfying the MIP.

Theorem 1 *The constraints (8), (9), and (10) are satisfied if and only if the grammar recognizes the sequence s for which $N(s[i], i, 1) = 1$.*

6. Comparison Between MIP Regular and MIP Grammar

It is interesting to compare `MIP regular` obtained from an automaton Π with the `MIP grammar` obtained from a context-free grammar G that encodes the same language recognized by Π. We recall how to automatically generate the grammar G from the transitions of the automaton Π. The states of the automaton form the set of non-terminals of the grammar and the alphabet symbols form the set of terminals. The starting non-terminal of the grammar is the initial state of the automaton. Each transition in the automaton is associated with a production in the grammar as follows. If S_1 is the state at the beginning of a transition, S_2 is the state at the end of this transition, and α is the associated alphabet symbol, then the production $S_1 \rightarrow \alpha S_2$ is added to the grammar. If S_2 is a final state then the production $S_1 \rightarrow \alpha$ is also added to the grammar.

The and/or graph produced with such a grammar has the following properties. The only productions having two literals on their right-hand side have the form $S_1 \rightarrow \alpha S_2$, i.e. a non-terminal produces a terminal followed by a non-terminal. The parsing trees produced by this grammar are therefore unbalanced trees where the left child of a node is necessarily a leaf labeled with a terminal symbol. For instance, the production $S_1 \rightarrow \alpha S_2$ creates in a parsing tree a node S_1 with a left-child α and a right-child S_2. The or-nodes in the graph created from a sequence of n characters are either of the form $N(P, t, n - t + 1)$ for the inner-nodes or the form $N(\alpha, t, 1)$ for the leaf nodes.

The and-or graph associated to a regular grammar has a similar structure to the layered graph used to model the `MIP regular`. The or-node $N(P, t, n - t + 1)$ in the and/or graph

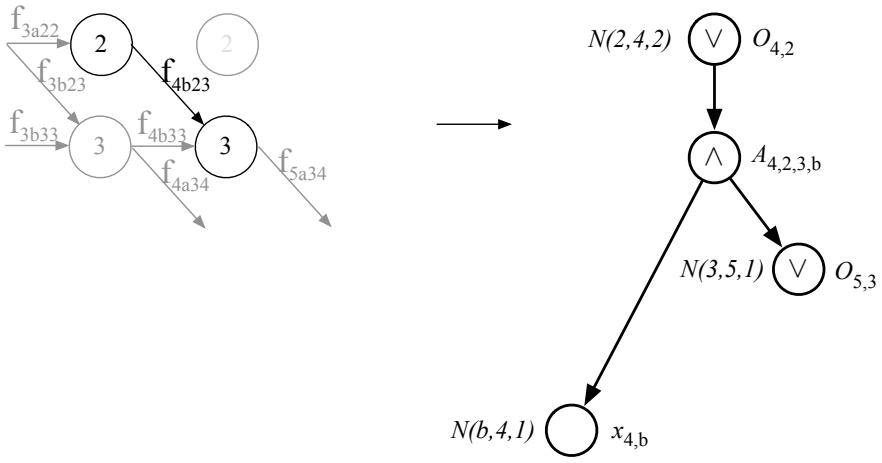

Figure 6: The relation between the arc $(4, b, 2, 3)$ from the layered graph of `MIP regular` depicted in Figure 5 and the corresponding nodes in the and/or graph. The label of each node is written at the left and the associated 0-1 variable at the right.

corresponds to the node $P \in N^t$ in the layered graph of the `MIP regular`. An and-node with left child $N(\alpha, t, 1)$, right child $N(Q, t+1, n-t)$ and parent $N(P, t, n-t+1)$ corresponds to the arc (t, α, P, Q) in the layered graph of the `MIP regular`. Figure 6 shows the relation between the arc $(4, b, 2, 3)$ from the layered graph of `MIP regular` depicted in Figure 5 and the corresponding nodes in the and/or graph.

The similarities between both graphs lead to similarities in the corresponding MIPs. Let $O_{t,A}$ be the 0-1 variable associated to the inner node $N(P, t, n-t+1)$ and $x_{t,\alpha}$ be the 0-1 variable associated to the leaf node $N(\alpha, t, 1)$. Let $A_{t,P,Q,\alpha}$ be the 0-1 variable associated to an and-node whose parent is $N(P, t, n-t+1)$, whose left child is the leaf node $N(\alpha, t, 1)$, and whose right child is the node $N(Q, t+1, n-t)$. Figure 6 shows some nodes in an and/or graph and their corresponding variables.

Equations (8) and (9), when applied on the inner node $N(P, t, n-t+1)$, lead to the two following equations:

$$O_{t,P} = \sum_{Q,\alpha} A_{t,P,Q,\alpha} \tag{11}$$

$$O_{t,P} = \sum_{Q,\alpha} A_{t+1,Q,P,\alpha} \tag{12}$$

These two equations lead to the following one, which is strictly equivalent to the flow conservation constraint (2) in the `MIP regular`:

$$\sum_{Q,\alpha} A_{t,P,Q,\alpha} \;=\; \sum_{Q,\alpha} A_{t+1,Q,P,\alpha} \tag{13}$$

Equation (9), when applied on the leaf node $N(P,t,1)$, gives the following equation, which is equivalent to constraint (5) in the `MIP regular`:

$$x_{t,\alpha} \;=\; \sum_{P,Q} A_{t+1,P,Q,\alpha} \tag{14}$$

Constraints (8) and (10) on the root node $N(S,1,n)$ lead to the following equations, which are equivalent to constraint (1):

$$w = \;\; O_{1,S} \;= \sum_{Q,\alpha} A_{1,S,Q,\alpha} \tag{15}$$

The `MIP grammar` is therefore equivalent to the `MIP regular` when the grammar encodes a regular language. However, notice that we have a 0-1 variable for each or-node in the `MIP grammar`, while there is no such variable associated to the states of the layered graph in the `MIP regular`.

7. Case study

7.1. Problem Definition

To evaluate the quality of our modeling approaches for constrained sequences of decision variables, we present computational results on complete shift scheduling problems described in [11]. The benchmarks are randomly generated, but are based on rules from a real-world shift scheduling problem. The demand curves come from a retail store. The objective is to create an optimal employee schedule for one day that satisfies the work regulation rules and the demands for each work activity.

The one day planning horizon is decomposed into 96 periods of 15 minutes each. We introduce the following notations before we define the problem:

- E : set of available employees.
- W : set of work activities.

- J : set of all activities ($J = W \cup \{l, p, o\}$)

  where l = lunch, p = break, o = rest.

- $I = \{1, 2, \ldots, n\}$: set of periods. $I' = I \setminus \{1\}$.

- $F_i \subseteq J$: set of activities that are not allowed to be performed at period $i \in I$.

- c_{ij}: cost for an employee to cover an activity $j \in W \setminus F_i$ at period $i \in I$.

Work Regulation Rules:

1. Activities $j \in F_i$ are not allowed to be performed at period $i \in I$.
2. If an employee is working, he must cover between 3 hours and 8 hours of work activities.
3. If a working employee covers at least 6 hours of work activities, he must have two 15 minute-breaks and a lunch break of 1 hour.
4. If a working employee covers less than 6 hours of work activities, he must have a 15 minute break, but no lunch.
5. If performed, the duration of any activity $j \in W$ is at least 1 hour (4 consecutive periods).
6. A break (or lunch) is necessary between two different work activities.
7. Work activities must be inserted between breaks, lunch and rest stretches.
8. Rest activities have to be assigned at the beginning and at the end of the day.

Demand Covering

1. The required number of employees for activity $j \in W \setminus F_i$ at period $i \in I$ is d_{ij}. Undercovering and overcovering are allowed. The cost of undercovering activity $j \in W \setminus F_i$ at period $i \in I$ is c_{ij}^- and the cost of overcovering activity $j \in W \setminus F_i$ at period $i \in I$ is c_{ij}^+.

The following sections present four ways of modeling this problem. The first model is a compact assignment MIP formulation that does not exploit the `MIP regular` constraint nor the `MIP grammar` constraint. The second model uses the `MIP regular` constraint and the third and fourth, the `MIP grammar` constraint.

7.2. A Compact Assignment MIP Model

Decision Variables

$$x_{eij} = \begin{cases} 1, & \text{if employee } e \in E \text{ covers activity } j \in J \text{ at period } i \in I, \\ 0, & \text{otherwise.} \end{cases}$$

Work Regulation Rules:

Rule 1:

$$x_{eij} = 0, \qquad e \in E, i \in I, j \in F_i. \tag{16}$$

Rule 2:

$$w_e = \begin{cases} 1, & \text{if employee } e \in E \text{ is working,} \\ 0, & \text{otherwise.} \end{cases}$$

$$\sum_{j \in J} x_{eij} = w_e, \qquad e \in E, i \in I, \tag{17}$$

$$12 w_e \leq \sum_{i \in I} \sum_{j \in W \setminus F_i} x_{eij} \leq 32 w_e, \qquad e \in E. \tag{18}$$

Rules 3 and 4:

$$u_e = \begin{cases} 1, & \text{if employee } e \text{ covers at least 6 hours of work activities,} \\ 0, & \text{otherwise.} \end{cases}$$

$$\sum_{i \in I} \sum_{j \in W \setminus F_i} x_{eij} - 8 u_e \leq 24, \qquad e \in E, \tag{19}$$

$$\sum_{i \in I} \sum_{j \in W \setminus F_i} x_{eij} \geq 23 u_e, \qquad e \in E, \tag{20}$$

$$\sum_{i \in I} x_{eip} = u_e + w_e, \qquad e \in E, \tag{21}$$

$$\sum_{i \in I} x_{eil} = 4 u_e, \qquad e \in E. \tag{22}$$

Rule 5:

$$v_{eij} = \begin{cases} 1, & \text{if employee } e \in E \text{ starts activity } j \in J \text{ at period } i \in I, \\ 0, & \text{otherwise.} \end{cases}$$

$$v_{eij} \geq x_{eij} - x_{e(i-1)j}, \qquad e \in E, i \in I, j \in W \setminus F_i \cup \{o\}, \tag{23}$$

$$v_{eij} \leq x_{eij}, \qquad e \in E, i \in I, j \in W \setminus F_i \cup \{o\}, \tag{24}$$

$$v_{eij} \leq 1 - x_{e(i-1)j}, \qquad e \in E, i \in I, j \in W \setminus F_i \cup \{o\}, \tag{25}$$

$$x_{ei'l} \geq v_{eil}, \qquad e \in E, i \in I, i' = i, i+1, i+2, i+3, \tag{26}$$

$$x_{ei'j} \geq v_{eij}, \qquad e \in E, i \in I, i' = i, i+1, i+2, i+3, j \in W \setminus F_i. \tag{27}$$

Rule 6:

$$v_{eij} \leq 1 - \sum_{j' \in W \setminus F_{i-1}} x_{e(t-1)j'}, \qquad e \in E, i \in I', j \in W \setminus F_i. \tag{28}$$

Rule 7:

$$x_{eip} \leq 1 - x_{e(i-1)j}, \qquad e \in E, i \in I', \tag{29}$$

$$x_{eip} \leq \sum_{j \in W \setminus F_{i-1}} x_{e(i-1)j}, \qquad e \in E, i \in I', \tag{30}$$

$$x_{eip} \leq \sum_{j \in W \setminus F_{i+1}} x_{e(i+1)j}, \qquad e \in E, i \in I', \tag{31}$$

$$v_{eil} \leq 1 - x_{e(i-1)p}, \qquad e \in E, i \in I', \tag{32}$$

$$v_{eil} \leq \sum_{j \in W \setminus F_{i-1}} x_{e(i-1)j}, \qquad e \in E, i \in I', \tag{33}$$

$$v_{eil} \leq \sum_{j \in W \setminus F_{i+1}} x_{e(i+1)j}, \qquad e \in E, i \in I'. \tag{34}$$

Rule 8:

$$v_{ei}^{-} = \begin{cases} 1, & \text{if employee } e \in E \text{ covers at least one working activity} \\ & \text{beginning before period } i \in I; \\ 0, & \text{otherwise.} \end{cases}$$

$$v_{ei}^{+} = \begin{cases} 1, & \text{if employee } e \in E \text{ covers at least one working activity} \\ & \text{beginning after period } i \in I; \\ 0, & \text{otherwise.} \end{cases}$$

$$v^-_{ei} \leq \sum_{i^- < i} \sum_{j \in W \setminus F_{i^-}} v_{ei^-j}, \qquad e \in E, i \in I', \tag{35}$$

$$v^-_{ei} \geq \sum_{j \in W \setminus F_{i^-}} v_{ei^-j}, \qquad e \in E, i \in I, i^- < i, \tag{36}$$

$$v^+_{ei} \leq \sum_{i^+ > t} \sum_{j \in W \setminus F_{i^+}} v_{ei^+j}, \qquad e \in E, i \in I', \tag{37}$$

$$v^+_{ei} \geq \sum_{j \in W \setminus F_{i^+}} v_{ei^+j}, \qquad e \in E, i \in I, i^+ > i, \tag{38}$$

$$x_{eio} \leq (1 - v^-_{ei}) + (1 - v^+_{ei}), \qquad e \in E, i \in I. \tag{39}$$

Demand Covering

$$\sum_{e \in E} x_{eij} - s^+_{ij} + s^-_{ij} = d_{at}, \qquad t \in T, a \in W \setminus F_t. \tag{40}$$

Objective Function

$$\min \qquad \sum_{i \in I} \sum_{j \in W \setminus F_i} (\sum_{e \in E} c_{ij} x_{eij} + c^+_{ij} s^+_{ij} + c^-_{ij} s^-_{ij}). \tag{41}$$

7.3. A MIP Regular Model

To observe the impact of modeling with the `MIP regular` constraint, we include several rules of the problem in a DFA and we formulate the other rules and the objective function as stated in the previous section. Work regulation rules 1 to 4 and demand covering constraints are formulated as in the compact assignment model, and work regulation rules 5 to 8 are included in the DFA. We use the DFA suggested by Demassey et al. [12] for the same problem. The DFA presented in Figure 7 is for the problem with two work activities (a and b on the figure). It is easily generalized for any number of work activities. Let us denote Π_n the DFA for the problem with n work activities.

We insert a `MIP regular` constraint for each employee $e \in E$ to the model. This constraint ensures that the covering of the activities $a \in A$ for each $t \in T$ for any employee $e \in E$ is a word recognized by the DFA $\Pi_{|W|}$. To add this constraint, we use the procedure presented in Section 4:

$$\textbf{AddMIPRegular}(\Pi_{|W|}, |T|, x_e, w_e, M), \qquad \forall e \in E. \tag{42}$$

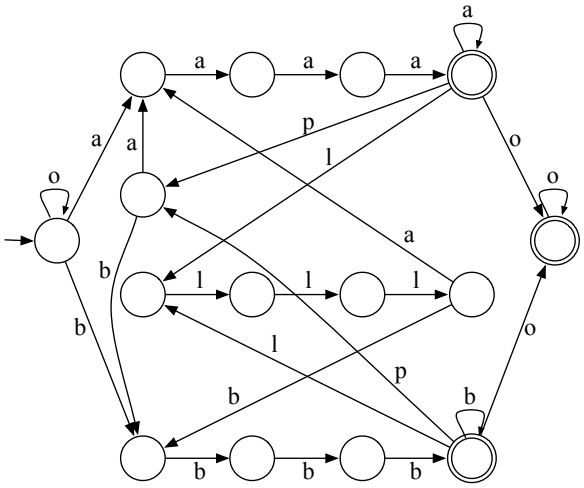

Figure 7: DFA Π_2 for two activities

where M is the model presented in the previous section without work regulation constraints 5 to 8, and the variables x_e and w_e have the same interpretation as in this model.

7.4. MIP Grammar Models

To test the `MIP Grammar` constraint on this problem, we tried two different grammars.

First, we used a grammar encoding the DFA presented in Section 7.3 with the same linear constraints as in the two previous models for work regulation rules 1 to 4 and the demand constraints. We call this model the *partial* `MIP grammar` *model*. This grammar is obtained from the automaton Π as described in Section 6.

Then, we used a context-free grammar presented in [25] that encodes all work regulation rules. It only uses the demand constraints as side constraints. This second grammar, leading to the *complete* `MIP grammar` *model*, can be expressed as follows:

$$R \rightarrow O \mid o \qquad L \rightarrow lL \mid l$$
$$A \rightarrow aA \mid a \qquad G \rightarrow A$$
$$P \rightarrow GpG \qquad Q \rightarrow PpG$$
$$F \rightarrow PLP \mid QLG \mid GLQ \qquad S \rightarrow RPR \mid RFR$$

where R, A, P, F, L, G, Q are non-terminals, S is the starting non-terminal. Terminals o, p, l represent rest, break and lunch periods respectively. Terminal a represents a work-activity.

Several rules of the model are handled with a set of restrictions on the productions of the grammar. These rules are dealt within the construction of the and/or graph.

As for the `MIP regular` model, a `MIP grammar` constraint is posted for each available employee on a sequence length of $|T|$ in both models.

7.5. Computational Results

Experiments were run on a 2.4 GHz Dual AMD Opteron Processor 250 (where only one processor was used) with 3 GB of RAM, using the MIP solver CPLEX 10.0. Tables 1 to 4 present the results for the four formulations presented in the last sections for the problem with one work activity and 12 available employees (10 instances). In the tables, the $Time$ and $Value$ on the LP refer to the results for the LP relaxation of the models. These results were obtained without CPLEX presolve to allow comparing the LP relaxation lower bounds of the different models. The symbol ">" means that CPLEX could not find the solution of the LP relaxation within a time limit of 3600 seconds. The results on the MIP were obtained with CPLEX default parameters on the given models. In particular, the branch-and-bound algorithm is stopped when the objective value is within 1% of optimality, which explains the differences in the MIP objective values of two models for which CPLEX stops within the 3600 seconds elapsed time limit. The symbol ">" means that no integer solution was found within this time limit. $|C|$ and $|V|$ are, respectively, the number of variables and the number of constraints in the MIP model after CPLEX presolve.

Note that the three models using formal languages lead to a faster computation than the compact assignment MIP model on all instances in our benchmarks. The LP relaxations for the `MIP regular` and the partial `MIP grammar` models are the same since they contain the same constraints, as explained in Section 6. We observed in our experiments that CPLEX presolve does not eliminate the variables associated to the or-nodes. Therefore, there are more variables in the partial `MIP grammar` model than in the `MIP regular` model. Observe that the LP relaxation bounds of these two formulations are never worse than the LP bounds of the compact assignment model, and are stronger on 4 out of the 10 instances.

We did not observe any advantage to include all the constraints in a single grammar as we did for the complete `MIP grammar` model. Indeed, the partial `MIP grammar` model and the `MIP regular` model lead to a faster computation. However, this complete grammar has been proved successful in a local search framework [23].

Table 1: Results for the classical MIP model

Id	LP		MIP			
	LP Value	*LP Time*	$\|C\|$	$\|V\|$	*MIP Value*	*MIP Time*
1	138,8952	36,53	29871	4040	172,6670	2142,17
2	162,9396	104,73	56743	5104	>	>
3	168,8223	89,06	56743	5104	>	>
4	131,6560	32,95	45983	4704	152,2240	3610,00
5	143,7443	63,61	40447	4360	171,9930	3610,01
6	129,0947	36,65	40447	4360	137,5180	3616,57
7	148,4681	38,26	45887	4608	>	>
8	147,2002	90,49	56743	5104	>	>
9	142,4836	27,15	36175	4156	182,5370	3607,76
10	145,9563	36,89	45983	4704	149,1810	3611,72

Table 2: Results for the `MIP regular` model

Id	LP		MIP			
	LP Value	*LP Time*	$\|C\|$	$\|V\|$	*MIP Value*	*MIP Time*
1	138,8952	15,95	1491	1856	172,6670	1,03
2	162,9396	13,32	2719	3976	164,1370	40,09
3	168,8223	15,70	2719	3976	169,0120	64,64
4	131,6560	19,15	2183	3144	133,3830	46,39
5	144,4182	30,81	1915	2728	145,4640	14,03
6	133,0766	12,34	1915	2728	135,2180	3,28
7	149,2739	26,55	2183	3144	150,6810	5,99
8	147,2002	12,81	2719	3976	148,0470	131,77
9	142,4836	11,84	1759	2416	182,5370	16,14
10	146,2410	23,66	2183	3144	147,5030	20,22

Table 3: Results for the partial `MIP grammar` model

Id	LP		MIP			
	LP Value	*LP Time*	$\|C\|$	$\|V\|$	*MIP Value*	*MIP Time*
1	138,8952	34,63	1683	2060	172,6665	1,16
2	162,9396	31,67	3040	4285	163,9210	66,26
3	168,8223	41,62	3040	4285	170,5663	46,74
4	131,6560	42,89	2459	3408	133,1414	98,00
5	144,4182	52,52	2143	2956	145,2850	21,83
6	133,0766	57,56	2143	2956	135,1286	1,56
7	149,2739	44,03	2456	3405	150,7675	53,82
8	147,2002	47,63	3040	4285	148,0467	263,43
9	142,4836	27,46	1957	2614	182,4833	12,91
10	146,2410	37,19	2458	3408	147,6853	19,28

Table 4: Results for the complete `MIP grammar` model

Id	LP		MIP			
	LP Value	*LP Time*	$\|C\|$	$\|V\|$	*MIP Value*	*MIP Time*
1	>	3612,35	2727	6356	172,6665	7,42
2	>	3610,68	35419	190480	>	>
3	>	3615,64	35419	190480	>	>
4	>	3614,92	19163	84672	133,0630	1850,38
5	>	3609,11	10387	41464	145,8830	322,57
6	>	3618,18	10387	41464	134,8178	130,21
7	>	3618,83	19163	84672	151,2082	1662,75
8	>	3611,39	35419	190480	>	>
9	>	3614,71	5503	21580	182,5370	1015,10
10	>	3610,10	19163	84672	147,0870	1313,28

8. Conclusion

We presented two new modeling approaches to express constraints on sequences of decision variables in MIP. These approaches are inspired by two CP constraints using formal languages: the `regular` constraint and the `grammar` constraint. The MIP version of the `regular` constraint uses an automaton to model the rules on the sequence of decision variables and transform it into a set of linear constraints representing a network flow problem in the graph. The MIP version of the `grammar` constraint uses an and/or decomposition of the parsing tree of all the sequences accepted by a context-free grammar and translates the logical clauses associated with the tree into linear constraints on 0-1 variables.

From a modeling point of view, both approaches allow the design of complex rules on sequences of variables to be handled with formal languages tools (an automaton or a context-free grammar) instead of directly into linear constraints. This process generates automatically a set of linear constraints that can be managed by any MIP solver. With this approach, the modeling of many complex rules is simplified and experimental results show that the resulting formulations can be strong.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows.* Prentice Hall, 1993.

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

[3] T. Aykin. A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. *Journal of the Operational Research Society*, 49(6):603–615, 1998.

[4] A. Balakrishan and R. Wong. Model for the rotating workforce scheduling problem. *Networks*, 20:25–42, 1990.

[5] H. Beaulieu, J. A. Ferland, B. Gendron, and P. Michelon. A mathematical programming approach for scheduling physicians in the emergency room. *Health Care Management Science*, 3:193–200, 2000.

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

[7] S. Bechtold and L. Jacobs. The equivalence of general set-covering and implicit integer programming formulations for shift scheduling. *Naval Research Logistics*, 43(2):233–249, 1996.

[8] M. Bouchard. Optimisation des pauses dans le problème de fabrication des horaires avec quarts de travail. Memoire de maitrise, Ecole Polytechnique de Montreal, 2004.

[9] T. Çezik, O. Günlük, and Luss. H. An integer programming model for the weekly tour scheduling problem. *Naval Research Logistic*, 48(7), 1999.

[10] J. Cocke and J. T. Schwartz. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, 1970.

[11] S. Demassey, G. Pesant, and L.-M. Rousseau. Constraint programming based column generation for employee timetabling. In *Proceedings of the Seconc International Conference on Intergretion of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2005)*, volume 3524, pages 140–154, 2005.

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

[13] A. Ernst, P. Hourigan, M. Krishnamoorthy, G. Mills, h Nott, and D. Sier. Rostering ambulance officers. In *Proceedings of the Fifteenth National Conference of the Australian Society for Operations Research, Gold Coast*, pages 470–481, 1999.

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

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

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

[17] T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical Report Scientific report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA, 1965.

[18] G. Laporte, Y. Nobert, and J. Biron. Rotating schedules. *European Journal of Operational Research*, 4(1):24–30, 1980.

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

[20] H. Millar and M. Kiragu. Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming. *European Journal of Operational Research*, 104(3):582–592, 1998.

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

[22] G. Pesant. A regular language membership constraint for finite sequences of variables. In *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming (CP 2004)*, pages 482–495, 2004.

[23] C.-G. Quimper and L.-M. Rousseau. Language based operators for solving shift scheduling problems. *Working Paper*, 2007.

[24] C.-G. Quimper and T. Walsh. Global grammar constraints. In *Proceedings of the Twelfth International Conference on Principles and Practice of Constraint Programming (CP 2006*, pages 751–755, 2006.

[25] C.-G. Quimper and T. Walsh. Decomposing global grammar constraints. In *Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming (CP 2007)*, pages 590–604, 2007.

[26] M. Rekik. *Construction d'horaires de travail dans des environnements hautement flexibles.* PhD thesis, École Polytechnique de Montréal, 2006.

[27] M. Rekik, J.-F. Cordeau, and F. Soumis. Using benders decomposition to implicitly model tour scheduling. *Annals of Operations Research*, 118:111–133, 2004.

[28] M. Sellmann. The theory of grammar constraints. In *Proceedings of the Twelfth International Conference on Principles and Practice of Constraint Programming (CP 2006*, pages 530–544, 2007.

[29] M.S. Sodhi. A flexible, fast, and optimal modeling approach applied to crew rostering at London Underground. *Annals of Operations Research*, 127:259–281, 2003.

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

[31] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.