The Polytope of Context-Free Grammar Constraints

Gilles Pesant[1], Claude-Guy Quimper[2], Louis-Martin Rousseau[1], Meinolf Sellmann[3]

[1] Ecole Polytechnique de Montreal
Montreal, Canada
{Gilles.Pesant,Louis-Martin.Rousseau}@polymtl.ca
[2] Google Inc.
Waterloo, Canada
cquimper@gmail.com
[3] Brown University
Department of Computer Science
115 Waterman Street, P.O. Box 1910
Providence, RI 02912
sello@cs.brown.edu

Abstract. Context-free grammar constraints enforce that a sequence of variables forms a word in a language defined by a context-free grammar. The constraint has received a lot of attention in the last few years as it represents an effective and highly expressive modeling entity. Its application has been studied in the field of Constraint Programming, Mixed Integer Programming, and SAT to solve complex decision problems such as shift scheduling. In this theoretical study we demonstrate how the constraint can be linearized efficiently. In particular, we propose a lifted polytope which has only integer extreme points. Based on this result, for shift scheduling problems we prove the equivalence of Dantzig's original set covering model and a lately introduced grammar-based model.

Keywords: grammar constraints, polytope

1 Introduction

Global constraints capture natural substructures of common combinatorial optimization or satisfaction problems. They facilitate the modeling process and, at the same time, offer the solver awareness of structures that it can exploit to boost performance. In constraint programming, global constraints allow greater filtering effectiveness. In integer programming, they offer the possibility to linearize specific structures more effectively than a non-expert is able to achieve. Systems like SCIP [1] and SIMPL [12] linearize entire substructures automatically and effectively.

In this short theoretical study, we show how context-free grammar constraints can be linearized effectively. We prove that the linearization proposed in [3] results in a polytope which has integer feasible corners only, thus giving us the convex hull of all integer feasible points and allowing us to obtain tight linear relaxation models when grammar constraints are used in combination with other constraints.

2 Basic Concepts

We start by reviewing some well-known definitions from the theory of formal languages and the existing algorithm for filtering context-free grammar constraints. For a full introduction, we refer the interested reader to [5] and [11] where all the proofs that are omitted in this paper can be found.

Definition 1 (Alphabet and Words). *Given sets Z, Z_1, and Z_2, with Z_1Z_2 we denote the set of all* sequences *or* strings *$z = z_1z_2$ with $z_1 \in Z_1$ and $z_2 \in Z_2$, and we call Z_1Z_2 the* concatenation *of Z_1 and Z_2. Then, for all $n \in \mathbb{N}$ we denote by Z^n the set of all sequences $z = z_1z_2 \dots z_n$ with $z_i \in Z$ for all $1 \leq i \leq n$. We call z a* word *of length n, and Z is called an* alphabet *or set of* letters. *The empty word has length 0 and is denoted by ε. It is the only member of Z^0. We denote the set of all words over the alphabet Z by $Z^* := \bigcup_{n\in\mathbb{N}} Z^n$. In case that we wish to exclude the empty word, we write $Z^+ := \bigcup_{n\geq 1} Z^n$.*

Definition 2 (Context-Free Grammars). *A* grammar *is a tuple $G = (\Sigma, N, P, S_0)$ where Σ is the alphabet, N is a finite set of* non-terminals, *$P \subseteq (N \cup \Sigma)^*N(N \cup \Sigma)^* \times (N \cup \Sigma)^*$ is the set of* productions, *and $S_0 \in N$ is the start non-terminal. We will always assume that $N \cap \Sigma = \emptyset$. Given a grammar $G = (\Sigma, N, P, S_0)$ such that $P \subseteq N \times (N \cup \Sigma)^*$, we say that the grammar G and the language L_G are* context-free. *A context-free grammar $G = (\Sigma, N, P, S_0)$ is said to be in* Chomsky Normal Form *if and only if for all productions $(A \to \alpha) \in P$ we have that $\alpha \in \Sigma^1 \cup N^2$. Without loss of generality, we will then assume that each literal $a \in \Sigma$ is associated with exactly one unique non-literal $A_a \in N$ such that $(B \to a) \in P$ implies that $B = A_a$ and $(A_a \to b) \in P$ implies that $a = b$.*

Remark 1. Throughout the paper, we will use the following convention: Capital letters A, B, C, D, and E denote non-terminals, lower case letters a, b, c, d, and e denote letters in Σ, Y and Z denote symbols that can either be letters or non-terminals, u, v, w, x, y, and z denote strings of letters, and α, β, and γ denote strings of letters and non-terminals. Moreover, productions (α, β) in P can also be written as $\alpha \to \beta$.

Definition 3 (Derivation and Language).

- *Given a grammar $G = (\Sigma, N, P, S_0)$, we write $\alpha\beta_1\gamma \underset{G}{\Rightarrow} \alpha\beta_2\gamma$ if and only if there exists a production $(\beta_1 \to \beta_2) \in P$. We write $\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_m$ if and only if there exists a sequence of strings $\alpha_2, \dots, \alpha_{m-1}$ such that $\alpha_i \underset{G}{\Rightarrow} \alpha_{i+1}$ for all $1 \leq i < m$. Then, we say that α_m can be* derived *from α_1.*
- *The language given by G is $L_G := \{w \in \Sigma^* \mid S_0 \overset{*}{\underset{G}{\Rightarrow}} w\}$.*

2.1 Context-Free Grammar Constraints

Based on the concepts above, we review the definition of context-free grammar constraints introduced in [11]:

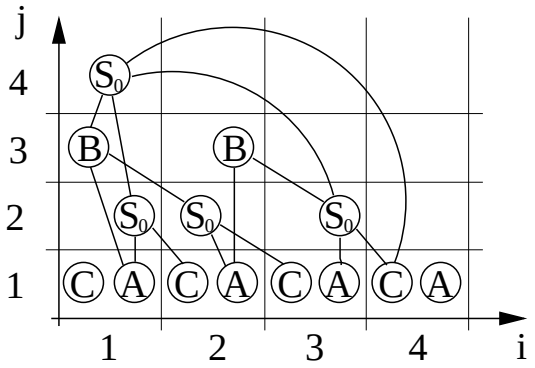

Fig. 1. The picture shows sets S_{ij} in Algorithm 1.

Definition 4 (Grammar Constraint). *For a given grammar $G = (\Sigma, N, P, S_0)$ and variables $X_1, \ldots, X_n$ with domains $D_1 := D(X_1), \ldots, D_n := D(X_n) \subseteq \Sigma$, we say that $Grammar_G(X_1, \ldots, X_n)$ is true for an instantiation $X_1 \leftarrow w_1, \ldots, X_n \leftarrow w_n$ if and only if it holds that $w = w_1 \ldots w_n \in L_G \cap D_1 \times \cdots \times D_n$. We denote a given grammar constraint $Grammar_G(X_1, \ldots, X_n)$ over a context-free grammar G in Chomsky Normal Form by $CFGC_G(X_1, \ldots, X_n)$.*

The filtering algorithm for $CFGC_G$ presented in [11] is based on the parsing algorithm from Cooke, Younger, and Kasami (CYK). CYK works as follows: Given a word $w \in \Sigma^n$, let us denote the subsequence of letters starting at position i with length j (that is, $w_i w_{i+1} \ldots w_{i+j-1}$) by w_{ij}. Based on a grammar $G = (\Sigma, N, P, S_0)$ in Chomsky Normal Form, CYK determines iteratively the set of all non-terminals from which we can derive w_{ij}, i.e. $S_{ij} := \{A \in N \mid A \overset{*}{\underset{G}{\Rightarrow}} w_{ij}\}$ for all $1 \leq i \leq n$ and $1 \leq j \leq n - i$. It is easy to initialize the sets S_{i1} just based on w_i and all productions $(A \to w_i) \in P$.

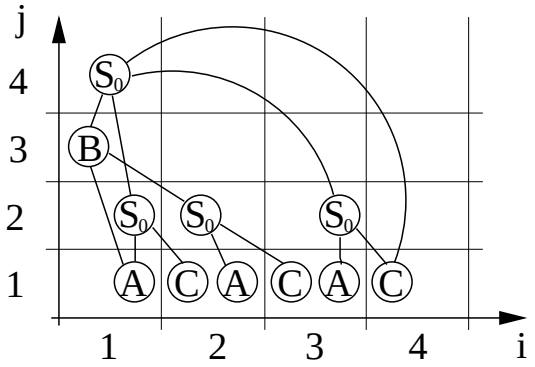

Fig. 2. The left picture shows the sets S'_{ij} in Algorithm 1. We see that the constraint filtering algorithm determines that the word may not start with a closing, nor end with an opening bracket.

Algorithm 1 CFGC Filtering Algorithm

1. We run the dynamic program based on recursion equation (1) with initial sets $S_{i1} := \{A \mid (A \to v) \in P, v \in D_i\}$.
2. We define the directed graph $Q = (V, E)$ with node set $V := \{v_{ijA} \mid A \in S_{ij}\}$ and arc set $E := E_1 \cup E_2$ with $E_1 := \{(v_{ijA}, v_{ikB}) \mid \exists\ C \in S_{i+k,j-k} :\ (A \to BC) \in P\}$ and $E_2 := \{(v_{ijA}, v_{i+k,j-k,C}) \mid \exists\ B \in S_{ik} :\ (A \to BC) \in P\}$ (see Figure 1).
3. Now, we remove all nodes and arcs from Q that cannot be reached from v_{1nS_0} and denote the resulting graph by $Q' := (V', E')$.
4. We define $S'_{ij} := \{A \mid v_{ijA} \in V'\} \subseteq S_{ij}$, and set $D'_i := \{v \mid \exists\ A \in S'_{i1} :\ (A \to v) \in P\}$.

Then, for j from 2 to n and i from 1 to $n - j + 1$, we have that

$$S_{ij} = \bigcup_{k=1}^{j-1} \{A \mid (A \to BC) \in P \text{ with } B \in S_{ik} \text{ and } C \in S_{i+k,j-k}\}. \qquad (1)$$

Then, $w \in L_G$ if and only if $S_0 \in S_{1n}$. From the recursion equation it is simple to derive that CYK can be implemented to run in time $O(n^3|G|) = O(n^3)$ when we treat the size of the grammar as a constant.

The filtering algorithm for $CFGC_G$ that we sketch in Algorithm 1 works bottom-up by computing the sets S_{ij} for increasing j after initializing S_{i1} with all non-terminals that can produce in one step a terminal in the domains of X_i. Then, the algorithm works top-down by removing all non-terminals from each set S_{ij} which cannot be reached from $S_0 \in S_{1n}$. In [11], we showed:

Theorem 1. *Algorithm 1 achieves generalized arc-consistency for the $CFGC$ and requires time and space cubic in the number of variables.*

Example 1. Assume we are given the following context-free, normal-form grammar $G = (\{], [\}, \{A, B, C, S_0\}, \{S_0 \to AC, S_0 \to S_0S_0, S_0 \to BC, B \to AS_0, A \to [\ , C \to]\ \}, S_0)$ that gives the language L_G of all correctly bracketed expressions (like, for example, "[[][]]" or "][[]]"). In Figures 1 and 2, we illustrate how Algorithm 1 works when the initial domain of all domains are $D_1 = \cdots = D_4 = \{[,]\}$: First, we work bottom-up, adding non-terminals to the sets S_{ij} if they allow to generate a word in $D_i \times \cdots \times D_{i+j-1}$. Then, in the second step, we work top-down and remove all non-terminals that cannot be reached from $S_0 \in S_{1n}$.

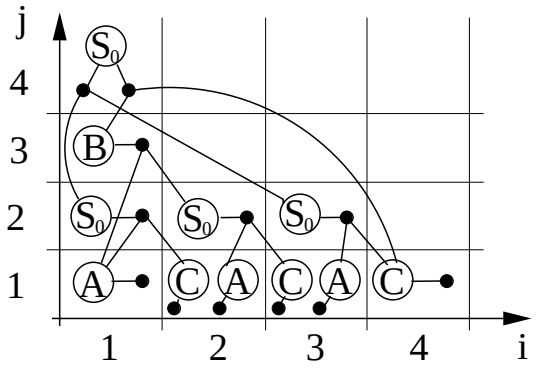

Fig. 3. The picture shows the modified And/Or-graph. The solid nodes denote And-nodes.

3 Linearization of Context-Free Grammar Constraints

In [9] an And/Or representation of the graph constructed in Algorithm 1 was given. The idea is to split each node v_{ijA^1} in the original graph into one *Or-node* $N^{or}_{ijA^1}$ which receives all incoming arcs of the original node. This Or-node then connects to *And-nodes* $N^{and}_{ijA^1A^2A^3k}$ for all productions $(A^1 \rightarrow A^2A^3) \in P$ and $k < j$. Each $N^{and}_{ijA^1A^2A^3k}$ then connects to $N^{or}_{ikA^2}$ and $N^{or}_{i+k,j-k,A^3}$. For Or-nodes $N^{or}_{i1A_a}$ on the lowest level, we add one And-node N^{and}_{i1a}, whereby we assume that the only production in P with non-terminal A_a on the left-hand side and a on the right is $(A_a \rightarrow a)$. In Figure 3 we show how the graph in Figure 2 is transformed in this way.

Based on this And/Or representation, in [3] a linearization of context-free grammar constraints was introduced which works as follows. A boolean variable is introduced for each node (AND/OR) of the graph, taking a value of 1 when the associated node is *true*. In the following, N^{or} and N^{and} respectively refer to the sets of all Or-nodes and And-nodes. Letting $u \in N^{or}$ (resp. $v \in N^{and}$) be an Or-node (resp. And-node) then o_u (resp. a_v) refers to its associated 0-1 variable. We also define $p(u)$ the set of u's parent nodes and, when not considering a leaf node, $c(u)$ the set of u's children nodes. The constraint in the associated MIP model was formulated as:

$$\sum_{v \in c(u)} a_v = o_u \quad \forall u \in N^{or} \tag{2}$$

$$\sum_{v \in p(u)} a_v = o_u \quad \forall u \in N^{or} \tag{3}$$

$$a_{1,n,S_0} = 1 \tag{4}$$

$$o_u, a_v \in \{0,1\} \qquad \forall u \in N^{or}, v \in N^{and} \tag{5}$$

The main difference between the MIP model and the AND/OR graph is that if 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 only 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.

This model can however be simplified: we can remove the variable associated with Or-nodes (since (2) and (3) can be combined) and use non-negative variables instead of binary ones (as (4) imposes an upper bound on all variables). These changes will also significantly simplify the proof that all extreme points of the defined polytope are integer. The new model becomes:

$$\sum_{v \in c(u)} a_v = \sum_{v \in p(u)} a_v \quad \forall u \in N^{or} \tag{6}$$

$$a_{1,n,S_0} = 1 \tag{7}$$

$$a_v \geq 0, \qquad \forall v \in N^{and} \tag{8}$$

It is also possible to define the grammar polytope directly from the signature of the constraint, that is without referring to the AND/OR graph. The polytope is then defined as follows:

$$\text{IP} - \text{CF} = \min \sum c_{ijABCk} a_{ijABCk} + \sum c_{i1a} a_{i1a} \tag{9}$$

s.t.

$$\sum_{\substack{(A \to BC) \in P \\ 1 \leq k < j}} a_{ijABCk} = \sum_{\substack{(B \to AC) \in P \\ j < k < n}} a_{ikBACk} + \sum_{\substack{(B \to CA) \in P \\ j < k < n}} a_{i+j-k,k,BCAk} \qquad \begin{array}{l} \forall\, i, j \geq 1 \\ \forall\, A \in S_{ij} \\ \quad i + j \leq n + 1 \end{array} \tag{10}$$

$$\sum_{(A_a \to a) \in P} a_{i1a} = \sum_{\substack{(B \to A_a C) \in P \\ 1 < k < n}} a_{ikBACk} + \sum_{\substack{(B \to C A_a) \in P \\ 1 < k < n}} a_{i+1-k,k,BCAk} \qquad \begin{array}{l} \forall\, i \in 1..n \\ \forall\, A_a \in S_{i1} \end{array} \tag{11}$$

$$1 = \sum_{\substack{(S_0 \to BC) \in P \\ 1 \leq k < n}} a_{1nS_0BCk} \tag{12}$$

$$0 \leq a_{ijABCk}, a_{i1a} \qquad \begin{array}{l} \forall\, A_a \in S_{i1} \\ \forall\, i, j \geq 1 \\ \quad i + j \leq n \end{array} \tag{13}$$

Example 2. Continuing with the bracketing example (Example 1), we illustrate the linearization of the corresponding `grammar` constraint. For the graph depicted in Figure 3, among other constraints, according to Equation 6, we enforce

$$a_{14S_0BC3} + a_{32S_0AC1} = a_{41]},$$

and according to Equation 4, we enforce

$$a_{14S_0BC3} + a_{14S_0S_0S_02} = 1.$$

4 The Context-Free Grammar Polytope

We now state and prove the main result of this theoretical study. Given a grammar constraint, the polytope defined by (6)-(8) has the following property:

Theorem 2. *The linearization IP-CF of a given context-free grammar constraint has integer-feasible corners only.*

Proof. We can write IP-CF in the form minimize $c^T a$ such that $Aa = b$, $a \geq 0$. Let a^1 denote an optimal solution to IP-CF, and u^1 the corresponding optimal dual solution. According to the well-known complementary slackness conditions, we know that all a with $Aa = b$ and $a^T(A^T u^1 - c) = 0$ are optimal. The complementary slackness conditions ensure that at optimality, we have $a_i^T = 0$ or $A_i^T u^1 - c_i = 0$ for every i. It is therefore sufficient to construct an integer-feasible solution a^0 to $Aa = b$ for which $A_i^T u^1 < c_i$ implies $a_i^0 = 0$. In particular, we want to construct an integer-feasible solution a^0 for which a_i^0 is greater than zero only if the non-integer solution satisfies $a_i^1 > 0$ or equivalently $A_i^T u^1 = c_i$. That is, it is sufficient to construct an integer-feasible solution a^0 to $Aa = b$ whose support (the set of variables that take non-zero values) is a subset of the support of a^1.

Let us consider any node in cell $S_1 n$ for which $a^1_{1nS_0BCk} > 0$ (at least one such node must exist according to Equation 12), and set $a^0_{1nS_0BCk} = 1$. Now, since the Or-nodes o_{1kB} and $o_{k,n-k,C}$ received a positive "flow" according to a^1, according to Equations 10 there must exist some And-nodes for which $a^1_{1kBDEh}, a^1_{k,n-k,CFGl} > 0$. Again, we set $a^1_{1kBDEh} = 1$ and $a^1_{k,n-k,CFGl} = 1$ and continue until we fade out at the bottom level. All other variables are set to zero. The tree of And-nodes which we constructed obeys all constraints in IP-CF. Moreover, we only utilized And-nodes that were already used in the fractional solution a^1. Consequently, a^0 obeys the complementary slackness conditions with respect to u^1 and is thus optimal.

□

The result applies to models where costs are associated with variables taking specific values as studied in [6], but also models where using certain productions incurs specific costs as studied in [7].

5 Implication for Shift Scheduling Problems

We illustrate the use of grammar constraints in the domain of shift scheduling. 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 in assigning one activity to each employee in each period in such a way that the demands are met. In this context, a *shift* is a sequence of activities corresponding to a continuous presence at work (that may include lunch and break, but not rest periods). The original objective (introduced by Dantzig in [2]) is to build a set of shifts that minimize labor costs while meeting labor regulations.

Given that Ω is the set of legal shifts, T the set of all time periods, d_i the required number of employees at time i, c_s the cost of shift s (equal to a weighted sum of the number of periods it covers), a_{is} an indicator specifying whether s covers time period i, and x_s is an integer variable that represents the number of employees assigned to s, we can state the original model of [2] as follows:

$$SC = \min \sum_{s \in \Omega} c_s x_s \tag{14}$$

$$s.t. \sum_{s \in \Omega} a_{is} x_s \geq d_i \quad \forall i \in T \tag{15}$$

$$x_s \geq 0, \text{integer} \quad \forall s \in \Omega \tag{16}$$

The most common methods used to solve this problem ([4]) are various set covering heuristics, which select from a set of *potentially* good shifts, the ones that together generate the best schedule. In order to generate an optimal solution the set covering model has to be defined over the *entire* set of possible shifts and solved through branch and price when Ω is too large.

For the purpose of comparison (and also many other practical issues that involve the need to track individual employees) this model can be disaggregated to identify the shift performed by each employee in set E. The boolean decision variable x_{se} indicates whether employee e performs shift s.

$$SCe = \min \sum_{s \in \Omega, e \in E} c_s x_{se} \tag{17}$$

$$s.t. \sum_{s \in \Omega, e \in E} a_{is} x_{se} \geq d_i \quad \forall i \in T \tag{18}$$

$$\sum_{s \in \Omega} x_{se} = 1 \quad \forall e \in E \tag{19}$$

$$x_{se} \in \{0, 1\} \quad \forall s \in \Omega, e \in E \tag{20}$$

In [3] Coté et al. showed that one could express implicitly the set of all shifts using a simple context-free grammar imposed on the sequence of fine grained decisions y_{ie} (employee e works at time i). At a high level, the model was:

$$SCg = \min \sum_{i \in T, e \in E} c_i y_{ie} \tag{21}$$

$$s.t. \sum_{e \in E} y_{ie} \geq d_i \ \ \forall i \in T \tag{22}$$

$$\texttt{grammar}(G, y_{i \in T, e}) \quad \forall e \in E \tag{23}$$

$$y_{ie} \in \{0,1\} \, \forall i \in T, e \in E \tag{24}$$

where c_i is the cost of having an employee working at time i and G is the grammar defining how the shifts in Ω can be assembled. Note that even though Models SC and SCe could encapsulate more sophisticated costs than the weighted sum of the worked periods, it would also be possible to impose costs on the variables associated to the And-nodes of the grammar constraint formulations allowing for substantial flexibility in modeling.

Corollary 1. *Given that the $c_s = \sum_{i \in T} a_{is} c_i \ \forall s \in \Omega$ then models SCe and SCg are equivalent.*

Proof. From Theorem 2, the linearization of (23) yields a polytope that admits only integer extreme points which are, by definition of G, all the pairs (valid shifts $s \in \Omega$, employee $e \in E$). Thus any feasible solution $y_{.e}$ associated to one employee can be written as a convex combination of the extreme points (s,e). If we associate a Boolean variable to each of these extreme points, say x_{se}, we can convert any solution vector given in terms of the extreme points x back to the original variable y by applying

$$y_{ie} = \sum_{s \in \Omega} a_{is} x_{se} \ \ \forall i \in T, \forall e \in E \tag{25}$$

Using (25), we can rewrite the y_{ie} variables in SCg (where (23) is no longer necessary) as a convex combination of the extreme point variables x_{se}:

$$SCg = \min \sum_{i \in T, e \in E} c_i \sum_{s \in \Omega} a_{is} x_{se} \tag{26}$$

$$s.t. \sum_{e \in E} \sum_{s \in \Omega} a_{is} x_{se} \geq d_i \ \ \forall i \in T \tag{27}$$

$$\sum_{s \in \Omega} a_{is} x_{se} = 1 \, \forall i \in T, e \in E \tag{28}$$

$$x_{se} \in \{0,1\} \quad \forall s \in \Omega, e \in E \tag{29}$$

Given that $c_s = \sum_{i \in T} a_{is} c_i \ \forall s \in \Omega$ and that (28) is derived from (12), this is exactly SCe. □

6 Conclusion

Grammar constraints have received much attention in last few years, as they have been studied in the context of Constraint Programming [6, 9, 11], SAT [10], Mixed Integer Programming [3], and Large Neighborhood Search [8]. In this paper we studied the linearization of this global constraint and showed that it is possible to generate a polytope that possesses only integer feasible extreme points. We believe this result is fundamental as it means that the use of grammar constraints in the context of Mixed Integer Programming does not introduce any integrality gap. We know that the grammar constraint can be very useful to solve shift scheduling problems [8], and we plan to investigate other areas where these structures can be useful.

References

1. T. Achterberg, T. Berthold, T. Koch, K. Wolter Constraint Integer Programming: a New Approach to Integrate CP and MIP *Proceedings of Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, LNCS 5015, 2008
2. G. Dantzig. A comment on edies traffic delay at toll booths. *Operations Research 2*, pages 339–341, 1954.
3. M.C. Coté, B. Gendron, C.-G. Quimper and L.-M. Rousseau. Formal Languages for Integer Programming Modeling of Shift Scheduling Problems. *Cirrelt Technical Report #CIRRELT-2007-64*, 2007.
4. 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.
5. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*, Addison Wesley, 1979.
6. S. Kadioglu and M. Sellmann. Efficient Context-Free Grammar Constraints. *23rd National Conference on Artificial Intelligence (AAAI)*, pp. 310–316, 2008.
7. G. Katsirelos, N. Narodytska, T. Walsh. The Weighted CfgConstraint. *Fifth International COnference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR)*, pp. 323–327, 2008.
8. C.-G. Quimper and L.-M. Rousseau. Language Based Operators for Solving Shift Scheduling Problems. Seventh Metaheuristics International Conference, 2007.
9. 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.
10. 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.
11. M. Sellmann. The Theory of Grammar Constraints. 12th intern. Conference on the Principles and Practice of Constraint Programming (CP), Springer LNCS 4204:530–544, 2006.
12. T. Yunes, I. Aron, and J. Hooker An Integrated Solver for Optimization Problems *Working Paper WPS-MAS-08-01, School of Business Administration, University of Miami*, 2008.