Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

Constraint Programming for Data Mining and Machine Learning

Luc De Raedt and **Tias Guns** and **Siegfried Nijssen**
Departement Computerwetenschappen, Katholieke Univeristeit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

Machine learning and data mining have become aware that using constraints when learning patterns and rules can be very useful. To this end, a large number of special purpose systems and techniques have been developed for solving such constraint-based mining and learning problems. These techniques have, so far, been developed independently of the general purpose tools and principles of constraint programming known within the field of artificial intelligence. This paper shows that off-the-shelf constraint programming techniques can be applied to various pattern mining and rule learning problems (cf. also (De Raedt, Guns, and Nijssen 2008; Nijssen, Guns, and De Raedt 2009)). This does not only lead to methodologies that are more general and flexible, but also provides new insights into the underlying mining problems that allow us to improve the state-of-the-art in data mining. Such a combination of constraint programming and data mining raises a number of interesting new questions and challenges.

Introduction

Constraint satisfaction problems are well-known and have extensively been studied in artificial intelligence. Today, effective and general purpose constraint solvers exist in the area of constraint programming (Rossi, van Beek, and Walsh 2006). The main principles in constraint programming are:

- users specify a problem declaratively by providing constraints on variables with domains;
- solvers find solutions by constraint propagation and search.

Constraint propagation is the process of using a constraint to reduce the domain of a variable based on the domains of other variables. Constraint programming systems have been used in a wide variety of applications and often lead to effective solutions.

Even though constraints have been used within machine learning and data mining, the use of principled constraint programming techniques in these fields is uncommon. One exception in this regard is the recent work of (De Raedt, Guns, and Nijssen 2008; Nijssen, Guns, and De Raedt 2009) on applying constraint programming to the mining and learning of rules in databases. This paper reviews these results.

An example of a rule targeted by these techniques is:

if income=high and debt=low **then** accept_loan=yes.

These rules can be used both in a descriptive setting, such as discovering *association rules* (Agrawal et al. 1996), and in a predictive setting, such as rule learning (Fürnkranz and Flach 2005). In the machine learning community, such rules are typically learned one at a time using a heuristic method. In the data mining community, the focus has been on finding all rules satisfying certain constraints using an exact method.

An essential question in both cases is when a rule is to be *accepted* as part of the final solution. Most approaches differ in their definition of the *constraints* that a rule should satisfy to be accepted. Particularly in exact approaches, which may yield many rules if the given constraints are not restrictive enough, the use of constraints is essential in order to make the computation both feasible and useful.

The interest in constraints in data mining led to the development of solvers for problems known as *closed itemset mining*, *maximal frequent itemset mining*, *discriminative itemset mining*, and so on. These problems are often related to well-studied problems in other areas. For instance, closed itemset mining is an instance of formal concept analysis (Ganter, Stumme, and Wille 2005), an area related to artificial intelligence; maximal itemset mining is related to the discovery of borders of version spaces; and discriminative itemset mining is related to rule-based classification and subgroup discovery. In all cases, *specialized* solvers were developed to find the proposed type of itemset efficiently, but almost no attention was given to the development of a general framework in which these solvers could be encompassed.

The key contribution of this paper is that we show how constraint programming can be applied to pattern mining and rule learning problems, two important applications, that have, so far, not really been addressed within the constraint programming community. This does not only lead to promising results – in some cases the constraint programming system outperforms the state-of-the-art in data mining – but also raises a number of new challenges for constraint programming, and new opportunities for data mining and machine learning.

This paper is organized as follows: we first briefly summarize the principles of constraint programming and introduce the problem of itemset mining; subsequently, we present the main contribution and provide an outlook on the future of the combination of constraint programming, machine learning and data mining.

Constraint Programming

Constraint programming (Rossi, van Beek, and Walsh 2006) is a declarative programming paradigm: instead of specifying how to solve a problem, the user only has to specify the problem itself. The constraint programming system is then responsible for solving it. Constraint programming systems solve constraint satisfaction problems (CSP). A CSP $\mathcal{P} = (\mathcal{V}, D, \mathcal{C})$ is specified by

- a finite set of variables $\mathcal{V}$;
- an initial domain D, which maps every variable $v \in \mathcal{V}$ to a finite set of values $D(v)$;
- a finite set of constraints $\mathcal{C}$.

The aim is to find an assignment of the variables which satisfies all constraints.

Constraint programming systems are typically based on exhaustive depth-first search. The main concept used to speed-up the search is constraint propagation. Constraint propagation reduces the domains of individual variables such that the domain of all variables remains *locally consistent*. In general, in a locally consistent domain, a variable x does not have value d in its domain if it can be determined that there is no solution D' for which $D'(x) = \{d\}$. In this way, the propagation prevents the search from visiting a part of the search tree that does not contain a solution.

Every constraint is implemented by a propagator. Such a propagator takes as input a domain and outputs a stronger, locally consistent domain. For example, if $D(X) = D(Y) = \{0, 1, 2\}$ and $X < Y$, then we can derive that $X \neq 2$ and $Y \neq 0$, so $D(X) = \{0, 1\}, D(Y) = \{1, 2\}$. The repeated application of propagators can lead to increasingly stronger domains. Propagation continues until a *fixed point* is reached in which the domain does not change any more. At this point, the search assigns a variable one of its values. Whenever the domain of one of the variables becomes empty, the search backtracks to explore alternatives.

Constraint programming systems differ in the constraints that they support. Most systems support arithmetic constraints, such as $X_1 + X_2 + X_3 \geq 2$, and reified arithmetic constraints such as $Y = 1 \leftrightarrow X_1 + X_2 + X_3 \geq 2$. This sets them apart from satisfiability (SAT) solvers and Integer Linear Programming (ILP) systems, in which it is hard to formulate reified arithmetic constraints. As we will see, such constraints are very important in machine learning and data mining, motivating the choice for constraint programming.

Frequent Itemset Mining

Let $\mathcal{I} = \{1, \ldots, m\}$ be a set of items, and $\mathcal{T} = \{1, \ldots, n\}$ a set of transactions. Then an itemset database $\mathcal{D}$ is a binary matrix of size $n \times m$; an example database is given in Figure 1. Furthermore, $\varphi : 2^{\mathcal{I}} \rightarrow 2^{\mathcal{T}}$ is a function that maps an itemset I to the set T of transactions from $\mathcal{T}$ in which all its items occur, that is,

$$\varphi(I) = \{t \in \mathcal{T} | \forall i \in I : \mathcal{D}_{ti} = 1\}$$

Tid	Itemset	Tid	A	B	C	D	E	Class
T_1	{C}	1	0	0	1	0	0	+
T_2	{A,B}	2	1	1	0	0	0	+
T_3	{A,E}	3	1	0	0	0	1	+
T_4	{B,C}	4	0	1	1	0	0	+
T_5	{C,D,E}	5	0	0	1	1	1	-
T_6	{A,B,C}	6	1	1	1	0	0	-
T_7	{A,B,E}	7	1	1	0	0	1	-
T_8	{A,B,C,E}	8	1	1	1	0	1	-

Figure 1: A small example of an itemset database, in multiset notation and in binary matrix notation. In discriminative itemset mining, every transaction has a class label; in other itemset mining settings no class label is used.

In our example database, we have for $I = \{A, B\}$ that $T = \varphi(\{A, B\}) = \{2, 6, 7, 8\}$. Dually, $\psi : 2^{\mathcal{T}} \rightarrow 2^{\mathcal{I}}$ is a function that maps a transaction set T to the set of all items from $\mathcal{I}$ shared by all transactions in T, that is,

$$\psi(T) = \{i \in \mathcal{I} | \forall t \in T : \mathcal{D}_{ti} = 1\}$$

In our example database in Figure 1, for $T = \{6, 7, 8\}$ we have that $\psi(\{6, 7, 8\}) = \{A, B\}$.

In the remainder of this section, we introduce several well-known itemset mining problems. We will show that they can be formulated as a constrained search for pairs (I, T), where I is an itemset and T a transaction set.

Frequent Itemsets Our first example are the traditional frequent itemsets (Agrawal et al. 1996). The search for these itemsets can be seen as a search for pairs (I, T), such that

$$T = \varphi(I) \quad (1)$$

$$|T| \geq \theta \quad (2)$$

where θ is a frequency threshold. The first constraint specifies that T must equal the set of transactions in which I occurs; the next constraint is the minimum frequency requirement: the absolute number of transactions in which I occurs must be at least θ. In our example database, if we impose a threshold of $\theta = 2$, the set of frequent itemsets includes $\{A, B\}$ and $\{A, B, E\}$; it does not include $\{A, B, C, E\}$.

Anti-Monotonic and Monotonic Constraints In pattern mining it is well-known that the frequency constraint $|T| \geq \theta$ is *anti-monotonic* with respect to the subset relation of itemsets: every subset of an itemset that satisfies the constraint, also satisfies it. For instance, for patterns (I, T) and (I', T') satisfying the coverage and support constraints,

$$I' \subseteq I \Rightarrow |T'| \geq |T| \geq \theta.$$

Other examples of anti-monotonic constraints are maximum itemset size and maximum total itemset cost (Bucila et al. 2003; Bonchi and Lucchese 2007). Assume that every item has a cost c_i (for a size constraint, $c_i = 1$). Then a maximum cost constraint with $c(I) = \sum_{i \in I} c_i$, is satisfied if

$$c(I) \quad \leq \quad \gamma. \tag{3}$$

Given that $I' \subseteq I \rightarrow c(I') \leq c(I) \leq \gamma$ we can see that this constraint is anti-monotonic too. Dual to this anti-monotonic constraints are monotonic constraints such as maximum frequency, minimum size and minimum cost. They can be expressed by replacing in the above constraints the $\geq$ by $\leq$, and vice versa.

Closed Itemsets Closed Itemsets are a popular *condensed* representation for the set of all frequent itemsets and their frequencies (Pasquier et al. 1999). Itemsets are called closed when they satisfy

$$I \quad = \quad \psi(T) \tag{4}$$

in addition to constraint (1); alternatively this can be formulated as $I = \psi(\varphi(I))$. Closed itemsets are closely related to the topic of formal concept analysis. Another type of condensed representation is that of maximal itemsets, which corresponds to the border of a version space as known from version space theory in machine learning.

Discriminative Itemset Mining

In discriminative or correlated itemset mining (Morishita and Sese 2000) we assume that the transactions or examples belong to two classes $\mathcal{T}^+$ and $\mathcal{T}^-$. Furthermore, a function $f(p, n)$ is assumed given which, for a given number of positive (p) and negative (n) examples covered, computes a score that measures how well the itemset discriminates the two classes. Often, the information gain measure or a correlation measure such as χ^2 is used (Cheng et al. 2008).

For example, assume that the first four transactions in Figure 1 are positive, and the last four are negative examples: $\mathcal{T}^+ = \{T_1, T_2, T_3, T_4\}$ and $\mathcal{T}^- = \{T_5, T_6, T_7, T_8\}$. Then the itemset $\{A, B\}$, which covers 1 positive example (T_2) and 3 negative examples (T_6, T_7, T_8), receives the following score when χ^2 is used as scoring function: $f(1, 3) = 2$. Likewise, for itemset $\{B, E\}$: $f(0, 2) = 2.666$.

The discriminative itemset mining task is to find all itemsets for which

$$T = \varphi(I) \tag{5}$$

$$f(|T \cap \mathcal{T}^+|, |T \cap \mathcal{T}^-|) \geq \theta \tag{6}$$

An alternative setting is to find the top-k itemsets which score the highest according to the given measure. This setting is strongly related to that of rule discovery in machine learning (Fürnkranz and Flach 2005). For instance, a rule learning algorithm such as FOIL (Quinlan 1990) also performs an iterative search for a rule which maximizes a certain measure; the main difference is that in FOIL this search is performed heuristically while in itemset mining this is done exhaustively.

Itemset Mining as Constraint Programming

In (De Raedt, Guns, and Nijssen 2008) we showed that we can formulate the problem of frequent itemset mining in CP by introducing a boolean variable I_i for each $i \in \mathcal{I}$ and a boolean variable T_t for each $t \in \mathcal{T}$. The frequent itemset constraints (1) and (2) are then formulated as:

$$\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} I_i(1 - \mathcal{D}_{ti}) = 0 \tag{7}$$

$$\forall i \in \mathcal{I} : I_i = 1 \rightarrow \sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti} \geq \theta. \tag{8}$$

Here $\mathcal{D}_{ti} = 1$ if $(t, i) \in \mathcal{D}$ and $\mathcal{D}_{ti} = 0$ if $(t, i) \notin \mathcal{D}$. It is straightforward to formulate these constraints in a CP system, as is demonstrated for frequent itemset mining using the Essence' language (Algorithm 1). In a constraint programming system such as Gecode (Schulte and Stuckey 2008), the solver uses the readily available propagators to search for all itemsets satisfying these constraints. Interestingly, the resulting search strategy is similar to that of known itemset mining systems (De Raedt, Guns, and Nijssen 2008).

Algorithm 1 Essence' model of frequent itemset mining

```
given NrT, NrI : int
given TDB : matrix indexed by [int(1..NrT),int(1..NrI)] of int(0..1)
given Freq : int
find Items : matrix indexed by [int(1..NrI)] of bool
find Trans : matrix indexed by [int(1..NrT)] of bool
such that
forall t: int(1..NrT).
  $Trans[t]$ <=> ((sum i: int(1..NrI). (1-TDB[t,i])* $Items[i]$) <= 0),
forall i: int(1..NrI).
  $Items[i]$ <=> ((sum t: int(1..NrT). TDB[t,i]* $Trans[t]$) >= Freq.
```

We implemented our model in the state-of-the-art constraint programming system Gecode (Schulte and Stuckey 2008). Figure 2 shows an experimental comparison of the resulting mining system, FIM_CP, to other mining systems. The first task is standard frequent itemset mining. This simple task has been studied for many years, amounting to a competition focused on efficient frequent itemset mining implementations (Goethals and Zaki 2004). The mining systems LCM, Eclat and Mafia were obtained from the competition website. They are still considered to be among the fastest implementations available. PATTERNIST is a constraint-based itemset mining system (Bonchi and Lucchese 2007). Given the highly optimized nature of these competitor systems, our constraint programming implementation in an out-of-the-box solver does not achieve faster runtimes. However, we can observe that the runtime of our method behaves in the same way as the other miners, indicating that the difference could be attributed to implementation issues.

The second setting for which we show results here is itemset mining with a monotonic minimum size constraint. The gray line in Figure 2 indicates the time needed to mine the patterns without a minimum size constraint. The experiment demonstrates that for tightly constrained problems, for instance, with a high minimum size constraint, our CP formulation is able to outperform the specialized PATTERNIST mining system. In general we have observed that the use

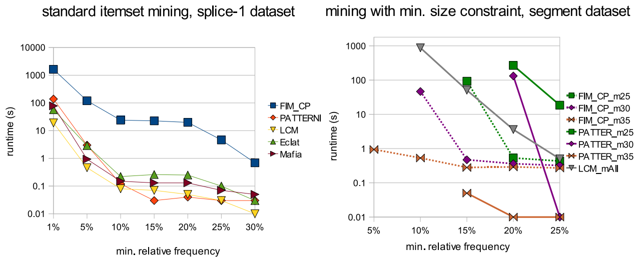

Figure 2: Comparison of mining systems for 2 different tasks. $\mathtt{m}x$ indicates a minimum size constraint of x.

of a constraint programming system is beneficial in settings with multiple and tight constraints. These are the cases in which constraint propagation is able to significantly reduce the size of the search tree.

Discriminative itemset mining The frequent itemset mining model can be adapted for the problem of discriminative itemset mining. Reusing constraint (7), we showed in (Nijssen, Guns, and De Raedt 2009) that for functions such as information gain and χ^2, we can model the correlation constraint (6) as:

$$\forall i \in \mathcal{I} : I_i = 1 \rightarrow f(\sum_{t \in \mathcal{T}^+} T_t \mathcal{D}_{ti}, \sum_{t \in \mathcal{T}^-} T_t \mathcal{D}_{ti}) \geq \theta. \qquad (9)$$

For this constraint, a propagator is needed that can calculate an upper bound on the correlation scores that can still be reached given current domains of items and transactions. We showed in (Nijssen, Guns, and De Raedt 2009) how to do this effectively, based on ROC analysis known in the machine learning community.

Table 1 illustrates how the discriminative itemset mining approach, when implemented in Gecode, compares to existing algorithms in the data mining community. In this experiment we are searching for the best scoring itemset. These experiments show convincingly that the constraint programming approach can outperform existing data mining systems on this task. The prime reason for this is that the constraint programming principle of maintaining local consistencies allowed us to device a more effective propagation rule than was employed in known itemset mining algorithms. Combined with the reified formulation, this leads to a large increase in efficiency.

These experiments show convincingly that the constraint programming approach can outperform existing data mining systems on some tasks. A further and perhaps even more important advantage is that CP systems are general purpose systems supporting many different types of constraints. Therefore, it is straightforward to incorporate many other well-known constraints, such as cost constraints, closedness or discriminative measures as defined above, as well as their combinations in the constraint programming system. This is unlike the typically itemset mining system which require substantial and non-trivial modifications in order to accommodate new constraints or their combination. The CP approach thus leads to a much more flexible and extensible system than state-of-the-art solutions in the data mining community, which in turn indicates that the relationship between machine learning, data mining and constraint programming deserves further study.

Further Research

The results summarized above consider two types of mining problems, namely frequent and discriminative itemset mining. Frequent itemset mining is the prototypical pattern mining problem; discriminative itemset mining is closely related to rule learning and subgroup discovery. These results show that constraint programming is promising as a method for solving these mining and learning problems.

Continuing this research, we are currently studying the application of our approach to problems arising in bioinformatics. For instance, itemset mining has commonly been applied to the analysis of microarray data; constraint programming may offer a more general and more flexible approach to analyze such data.

Whereas the above work is still restricted to the discovery of patterns in binary data, the use of constraint programming in other pattern mining related problems is also a promising direction of future research. A problem closely related to pattern mining is that of pattern set mining (De Raedt and Zimmermann 2007), where one does not only impose constraints on individual patterns, but also on the overall set of patterns constituting a solution. Constraints that can be imposed include, for instance, the requirement that patterns do not overlap too much, or that they cover the complete set of transactions together. Another related problem is that of finding patterns in continuous data. This requirement is in particular relevant to deal with problems in bioinformatics. Likewise, there are many approaches to mining structured data, such as sequences, trees and graphs. It is an interesting open question as to whether it is possible to represent such problems using constraint programming too. One of the challenges here is that such structured data can no longer be represented using a fixed number of features or items.

Next to pattern mining, other areas of machine learning and data mining may also profit from a closer study of constraint programming techniques. One such area is statistical machine learning, where problems are typically formulated using mathematical programming. Recently some results in the use of other types of solvers have already been obtained for certain probabilistic models (Chang et al. 2008; Cussens 2008). In these approaches, however, Integer Linear Programming (ILP) or satisfiability (SAT) solvers were used. CP solvers address a more general class of problems than ILP and SAT solvers, but this generality sometimes comes at a computational cost. Current developments in CP which aim at combining ILP and SAT with CP may also help in addressing these machine learning problems. In this it is important that the CP system deals effectively not only with satisfaction, but also with optimization problems.

Other topics of interest are constraint-based clustering and constraint-based classifier induction. In constraint-based clustering the challenge is to cluster examples when additional knowledge is available about them, for instance,

Dataset	CP	(Cheng et al. 2008)	(Morishita and Sese 2000)
anneal	0.22	22.46	24.09
australian-credit	0.30	3.40	0.30
breast-wisconsin	0.28	96.75	0.28
diabetes	2.45	–	128.04
heart-cleveland	0.19	9.49	2.15
hypothyroid	0.71	–	10.91
ionosphere	1.44	–	>
kr-vs-kp	0.92	125.60	46.20
letter	52.66	–	>
mushroom	14.11	0.09	13.48
primary-tumor	0.03	0.26	0.13
segment	1.45	–	>
soybean	0.05	0.05	0.07
splice-1	30.41	1.86	31.11
vehicle	0.85	–	>
yeast	5.67	–	781.63

Table 1: Run times, in seconds, of 3 top-1 discriminative itemset miners, on an Intel Core 2 Duo E6600 and 4GB of RAM; >: experiments timed out after 900s. −: experiments failing due to memory overflow. Results for (Cheng et al. 2008) were obtained using the original author's implementation; we re-implemented the algorithm of Morishita and Sese (2000).

prohibiting certain examples from being clustered together (so-called cannot-link constraints). Similarly, in constraint-based classifier induction, one may wish to find a decision tree that satisfies size and cost-constraints. A first study applying CP on this problem was recently performed (Bessiere, Hebrard, and O'Sullivan 2009). In data mining, the relationship between itemset mining and constraint-based decision tree learning was studied (Nijssen and Fromont 2007). It is an open question whether this relation can also be exploited in a constraint programming setting.

Whereas the previous cases study how machine learning and data mining could profit from constraint programming, the opposite direction is also a topic of interest: how can constraint programming systems be extended using techniques from machine learning and data mining? In our experiments we found that certain propagators are evaluated in a rather inefficient way. The use of matrix representations common in data mining may alleviate some of these problems. In conclusion, we believe that further integration of machine learning, data mining and constraint programming may contribute to the advancement of all these areas.

Acknowledgments

This work was supported by a Postdoc and a project grant from the Research Foundation—Flanders, project "Principles of Patternset Mining", as well as a grant from the Institute for the Promotion and Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. I. 1996. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, 307–328. AAAI Press.

Bessiere, C.; Hebrard, E.; and O'Sullivan, B. 2009. Minimising decision tree size as combinatorial optimisation. In Gent, I. P., ed., *CP*, volume 5732 of *Lecture Notes in Computer Science*, 173–187. Springer.

Bonchi, F., and Lucchese, C. 2007. Extending the state-of-the-art of constraint-based pattern discovery. *Data Knowl. Eng.* 60(2):377–399.

Bucila, C.; Gehrke, J.; Kifer, D.; and White, W. M. 2003. Dualminer: A dual-pruning algorithm for itemsets with constraints. *Data Min. Knowl. Discov.* 7(3):241–272.

Chang, M.-W.; Ratinov, L.-A.; Rizzolo, N.; and Roth, D. 2008. Learning and inference with constraints. In Fox, D., and Gomes, C. P., eds., *AAAI*, 1513–1518. AAAI Press.

Cheng, H.; Yan, X.; Han, J.; and Yu, P. 2008. Direct discriminative pattern mining for effective classification. In *ICDE*, 169–178.

Cussens, J. 2008. Bayesian network learning by compiling to weighted max-sat. In McAllester, D. A., and Myllymäki, P., eds., *UAI*, 105–112. AUAI Press.

De Raedt, L., and Zimmermann, A. 2007. Constraint-based pattern set mining. In *SDM*. SIAM.

De Raedt, L.; Guns, T.; and Nijssen, S. 2008. Constraint programming for itemset mining. In *KDD*, 204–212.

Fürnkranz, J., and Flach, P. A. 2005. ROC 'n' rule learning – towards a better understanding of covering algorithms. *Machine Learning* 58(1):39–77.

Ganter, B.; Stumme, G.; and Wille, R. 2005. *Formal Concept Analysis: Foundations and Applications*, volume 3626 of *Lecture Notes in Artificial Intelligence*.

Goethals, B., and Zaki, M. J. 2004. Advances in frequent itemset mining implementations: report on FIMI'03. In *SIGKDD Explorations Newsletter*, volume 6, 109–117.

Morishita, S., and Sese, J. 2000. Traversing itemset lattice with statistical metric pruning. In *PODS*, 226–236.

Nijssen, S., and Fromont, E. 2007. Mining optimal decision trees from itemset lattices. In *KDD*, 530–539.

Nijssen, S.; Guns, T.; and De Raedt, L. 2009. Correlated itemset mining in ROC space: A constraint programming approach. In *KDD*, 647–656.

Pasquier, N.; Bastide, Y.; Taouil, R.; and Lakhal, L. 1999. Discovering frequent closed itemsets for association rules. In *ICDT*, 398–416.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5:239–266.

Rossi, F.; van Beek, P.; and Walsh, T. 2006. *Handbook of Constraint Programming (Foundations of Artificial Intelligence).* Elsevier Science Inc.

Schulte, C., and Stuckey, P. 2008. Efficient constraint propagation engines. *ACM Trans. Program. Lang. Syst.* 31(1).