Explaining the AtMostSeqCard constraint

Mohamed Siala, Christian Artigues, and Emmanuel Hebrard

LAAS-CNRS

Uppsala, Sweden

Outline

Context

SAT & CP : Can we get the best from both approaches?
→A key concept : explaining constraints
An explanation is a set of assignments/prunings triggering a failure/pruning.

example

Cardinality Constraint : $\sum_{i=1}^{n} x_i \leq k$; $D(x_i) = \{0, 1\}$.
$x_i \leftarrow 1$ is pruned if we already have k appearances of the value 1.

$$\{x_j \leftarrow 1 | D(x_j) = \{1\}\} \rightarrow x_i \not\leftarrow 1 .$$

Context

SAT & CP : Can we get the best from both approaches?
$\rightarrow$A key concept : explaining constraints
An explanation is a set of assignments/prunings triggering a failure/pruning.

example

Cardinality Constraint : $\sum_{i=1}^{n} x_i \leq k$; $D(x_i) = \{0, 1\}$.
$x_i \leftarrow 1$ is pruned if we already have k appearances of the value 1.

$$\{x_j \leftarrow 1 | D(x_j) = \{1\}\} \quad \rightarrow x_i \not\leftarrow 1 \ .$$

Definition

Definition

$\textsc{AtMostSeqCard}(u, q, d, [x_1, \ldots, x_n]) \Leftrightarrow$

$$\bigwedge_{i=0}^{n-q} \left(\sum_{l=1}^{q} x_{i+l} \leq u\right) \wedge \left(\sum_{i=1}^{n} x_i = d\right)$$

Definition

Definition

AtMostSeqCard$(u, q, d, [x_1, \ldots, x_n]) \Leftrightarrow$

$$\bigwedge_{i=0}^{n-q} \left(\sum_{l=1}^{q} x_{i+l} \leq u\right) \wedge \left(\sum_{i=1}^{n} x_i = d\right)$$

Example AtMostSeqCard$(2, 4, 4, [x_1, \ldots, x_7])$

0 1 1 0 1 1 0

1 1 0 0 1 0 1

The propagator

- `leftmost`: computes an assignment w maximizing the cardinality of the sequence with respect to the AtMost constraints.
- Let $max(i)$ be the maximum cardinality of the q subsequences involving x_i when computing `leftmost`[i].

0 1 0 . 0 1 0

— — — | $cardinality = 1$

— — — | $cardinality = 0$

— — — | $cardinality = 1$

$max(4) = 1$

- $Left[i] = \sum_{j=1}^{j=i} \texttt{leftmost}[j]$.
- $Right[i]$: same as *Left* but in the reverse sense, i.e. $[x_n, .., x_1]$.
- Example : with ATMOST(2,5):

$\mathcal{D}(x_i)$	0	.	.	.	.	1	.	.	
$max(i)$	0	1	2	2	2	2	1	2	2
leftmost$[i]$	0	1	0	0	0	1	1	0	0
$Left[i]$	0	1	1	1	1	1	2	2	2

Domain consistency

- DC on each AtMost: $(\sum_{l=1}^{q} x_{i+l} \leq u)$
- DC on $\sum_{i=1}^{n} x_i = d$
- If $Left[n] < d$ Then *fail*
- If $Left[n] = d$ and $Left[i] + Right[n-i+1] \leq d$ Then $\mathcal{D}(x_i) \leftarrow \{0\}$
- If $Left[n] = d$ and $Left[i-1] + Right[n-i] < d$ Then $\mathcal{D}(x_i) \leftarrow \{1\}$

Explaining AtMostSeqCard: the key idea

Explaining Failure

1. If a failure is triggered by a cardinality constraint (i.e. $(\sum_{l=1}^{q} x_{i+l} \leq u)$ or $\sum_{i=1}^{n} x_i = d)$, then it is easy to generate an explanation.
2. If a failure triggered by $Left[n] < d$, a naive explanation would be the set of all assignments in the sequence.

Theorem

Theorem

Let S be the set of all assignments,
$S^{\cdot *} = S \setminus (\{x_i \leftarrow 0 \mid max(i) = u\} \cup \{x_i \leftarrow 1 \mid max(i) \neq u\})$*, then*
$S^{\cdot *}$ *is a valid explanation.*

→runs in $O(n)$ since we call `leftmost_count` once.

Example : AtMostSeqCard$(2, 5, 8, [x_1, ..x_{22}])$

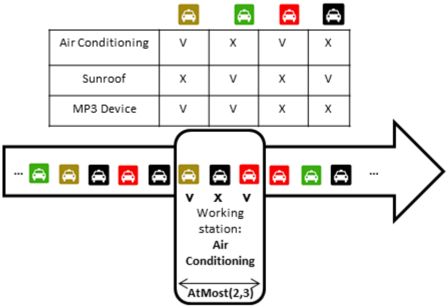
S	1	0	1	0	0	.	.	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1
leftmost$(S(x_i))$	1	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1
$Left[i]$	1	1	2	2	2	3	3	3	3	3	4	5	5	5	5	5	6	6	6	6	6	7

$Left[22] = 7 < 8$: FAILURE

$max(i)$	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
$S^{\cdot *}$	1	.	1	.	.	.	.	.	.	.	1	1	.	.	.	0	.	0	0	0	0	.

The final explanation size $|S^{\cdot *}|$ is 9 while the naive one ($|S|$) is 20.

Car-sequencing



Constraints

- Each class c is associated with a demand D_c.
- For each option j, each sub-sequence of size q_j must contain at most u_j cars requiring the option j.

Configuration

- Mistral as a hybrid CP/SAT solver
  1. *hybrid (VSIDS)* uses VSIDS;
  2. *hybrid (Slot)* uses a cp heuristic based on the usage rate.
  3. *hybrid (Slot → VSIDS)* first uses *hybrid (Slot)* then switches after 100 non-improving restarts to VSIDS.
  4. *hybrid (VSIDS → Slot)* uses VSIDS and switches after 100 non-improving restarts to *hybrid (Slot)*.
- *pure-CP*: Mistral without clause learning using the *Slot* branching.

Table: Experimental Evaluation

Method	sat [easy] (74 × 5)			sat [hard] (7 × 5)			unsat (28 × 5)		
	#suc	avg fails	time	#suc	avg fails	time	#suc	avg fails	time
hybrid (VSIDS)	370	903	0.23	16	207211	286.32	35	177806	224.78
hybrid (VSIDS → Slot)	370	739	0.23	35	76256	64.52	**37**	**204858**	**248.24**
hybrid (Slot → VSIDS)	370	132	0.04	34	4568	2.50	37	234800	287.61
hybrid (Slot)	370	132	0.04	**35**	**6304**	**3.75**	23	174097	299.24
pure-CP	**370**	**43.06**	**0.03**	35	57966	16.25	0	-	-

Conclusion & Future research

Contributions & Analysis

- A linear time explanation for the AtMostSeqCard constraint
- The experimental results emphasize the importance of using a hybrid approach instead of pure-CP!

Future research

- Can we generate optimal explanations?
- Is it worthy to use a 'sophisticated' [explanation + propagator] instead of decomposing to simpler constraints?
- To encode into SAT or to propagate?

Thank you!

Questions?