SAT and Hybrid models of the Car-Sequencing problem

Christian Artigues, Emmanuel Hebrard, Valentin Mayer-Eichberger, Mohamed Siala, and Toby Walsh

LAAS-CNRS NICTA UNSW AUSTRALIA

Cork, Ireland

CP & SAT Solving

- to encode into SAT or to use global constraints?
- Can we get the best from both approaches?

CP & SAT Solving

- to encode into SAT or to use global constraints?
- Can we get the best from both approaches?
- Hybridization!

CP & SAT Solving

- to encode into SAT or to use global constraints?
- Can we get the best from both approaches?
- Hybridization!
  →A key concept in hybrid solvers (*Lazy Clause Generation*): explaining constraints

An explanation is a set of atomic constraints triggering a failure/filtering.

CP & SAT Solving

example

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

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

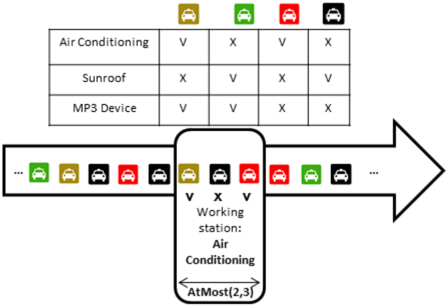
CP & SAT Solving

example

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

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

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.

Modelling in CP

Variables:

- n integer variables $\{x_1, \ldots, x_n\}$ taking values in $\{1, \ldots, k\}$
- nm Boolean variables $\{y_1^1, \ldots, y_n^m\}$

Constraints:

1. *Demand constraints*: for each class $c \in \{1..k\}$

$$|\{i \mid x_i = c\}| = D_c^{class}.$$
$$\rightarrow \textsc{Gcc}$$

2. *Capacity constraints*: for each option $j \in \{1..m\}$, for each slot $i \in \{1, \ldots, n - q_j + 1\}$.

$$\sum_{l=i}^{i+q_j-1} y_l^j \leq u_j.$$

$\rightarrow$ Gsc, AtMostSeqCard or AtMostSeqCard$\oplus$ Gsc

.

Modelling in CP

Variables:

- n integer variables $\{x_1, \ldots, x_n\}$ taking values in $\{1, \ldots, k\}$
- nm Boolean variables $\{y_1^1, \ldots, y_n^m\}$

Constraints:

1. *Demand constraints*: for each class $c \in \{1..k\}$

$$|\{i \mid x_i = c\}| = D_c^{class}.$$

$\rightarrow$ Gcc

2. *Capacity constraints*: for each option $j \in \{1..m\}$, for each slot $i \in \{1, \ldots, n - q_j + 1\}$.

$$\sum_{l=i}^{i+q_j-1} y_l^j \leq u_j.$$

$\rightarrow$ Gsc, AtMostSeqCard or AtMostSeqCard$\oplus$ Gsc

.

Definition

Definition

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

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

Definition

Definition

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

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

Example $\textsc{AtMostSeqCard}(2, 5, 4, [x_1, \ldots, x_9])$

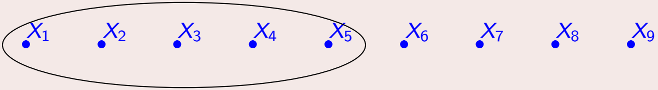
Definition

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$(2, 5, 4, [x_1, \ldots, x_9])$



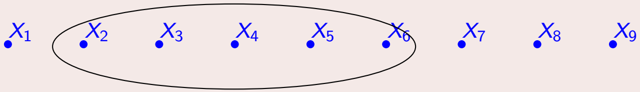
Definition

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$(2, 5, 4, [x_1, \ldots, x_9])$



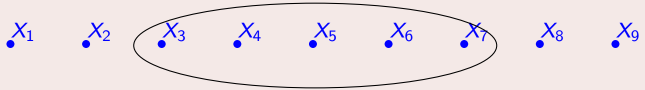
Definition

Definition

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

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

Example $\textsc{AtMostSeqCard}(2, 5, 4, [x_1, \dots, x_9])$



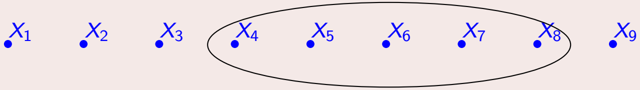
Definition

Definition

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

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

Example $\textsc{AtMostSeqCard}(2, 5, 4, [x_1, \ldots, x_9])$



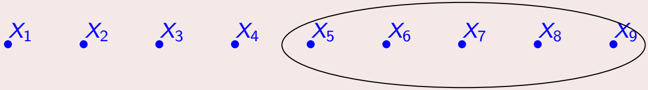
Definition

Definition

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

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

Example $\textsc{AtMostSeqCard}(2, 5, 4, [x_1, \ldots, x_9])$



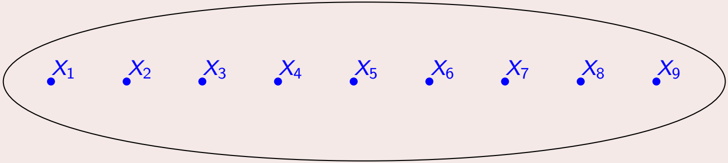
Definition

Definition

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

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

Example AtMostSeqCard$(2, 5, 4, [x_1, \dots, x_9])$



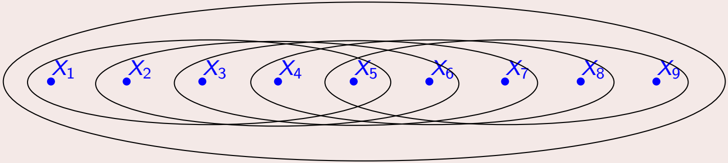
Definition

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$(2, 5, 4, [x_1, \ldots, x_9])$



The propagator

- `leftmost`: computes an assignment w maximizing the cardinality of the sequence with respect to the AtMost constraints.
- $max[i]$: maximum cardinality for each sub-sequence involving x_i
- $Left[i] = \sum_{j=1}^{j=i} \texttt{leftmost}[j]$.
- $Right[i]$: same as $Left$ but in the reverse order, i.e. $[x_n, .., x_1]$.

LAAS-CNRS, NICTA, UNSW

The propagator

- `leftmost`: computes an assignment w maximizing the cardinality of the sequence with respect to the AtMost constraints.
- $max[i]$: maximum cardinality for each sub-sequence involving x_i
- $Left[i] = \sum_{j=1}^{j=i}$ `leftmost`$[j]$.
- $Right[i]$: same as $Left$ but in the reverse order, i.e. $[x_n, .., x_1]$.

AtMostSeqCard$(u = 4, q = 8, d = 12)$

$\mathcal{D}(x_i)$		.	0	.	.	.	.	.	.	0	1	0	.	.	.	.	.	.	.	.	.	.	1	
`leftmost`$[i]$		**1**	0	**1**	**1**	**1**	0	0	0	0	1	0	**1**	**1**	**1**	0	0	0	**1**	0	**1**	**1**	1	
$Left[i]$	0	1	1	2	3	4	4	4	4	4	4	**4**	5	6	7	7	7	7	8	8	9	10	10	
$Right[i]$		10	9	9	9	8	7	6	6	6	6	6	6	**5**	4	3	3	3	3	3	2	1	0	0

The propagator

- `leftmost`: computes an assignment w maximizing the cardinality of the sequence with respect to the AtMost constraints.
- $max[i]$: maximum cardinality for each sub-sequence involving x_i
- $Left[i] = \sum_{j=1}^{j=i}$ `leftmost`$[j]$.
- $Right[i]$: same as $Left$ but in the reverse order, i.e. $[x_n, .., x_1]$.

AtMostSeqCard$(u = 4, q = 8, d = 12)$

$\mathcal{D}(x_i)$		.	0	.	.	.	.	.	.	0	1	0	**?**	.	.	.	.	.	.	.	.	.	1	
`leftmost`$[i]$		**1**	0	**1**	**1**	**1**	0	0	0	0	1	0	**1**	**1**	**1**	0	0	0	**1**	0	**1**	**1**	1	
$Left[i]$	0	1	1	2	3	4	4	4	4	4	4	**4**	5	6	7	7	7	7	8	8	9	10	10	
$Right[i]$		10	9	9	9	8	7	6	6	6	6	6	6	**5**	4	3	3	3	3	3	2	1	0	0

The propagator

- `leftmost`: computes an assignment w maximizing the cardinality of the sequence with respect to the ATMOST constraints.
- $max[i]$: maximum cardinality for each sub-sequence involving x_i
- $Left[i] = \sum_{j=1}^{j=i}$ `leftmost`$[j]$.
- $Right[i]$: same as $Left$ but in the reverse order, i.e. $[x_n, .., x_1]$.

ATMOSTSEQCARD$(u = 4, q = 8, d = 12)$

$\mathcal{D}(x_i)$		.	0	.	.	.	.	.	.	0	1	0	**?**	.	.	.	.	.	.	.	.	.	1	
												Remaining demand : 10												
`leftmost`$[i]$		**1**	0	**1**	**1**	**1**	0	0	0	0	1	0	**1**	**1**	**1**	0	0	0	**1**	0	**1**	**1**	1	
$Left[i]$	0	1	1	2	3	4	4	4	4	4	4	**4**	5	6	7	7	7	7	8	8	9	10	10	
$Right[i]$		10	9	9	9	8	7	6	6	6	6	6	6	**5**	4	3	3	3	3	3	2	1	0	0

LAAS-CNRS, NICTA, UNSW

The propagator

- `leftmost`: computes an assignment w maximizing the cardinality of the sequence with respect to the AtMost constraints.
- $max[i]$: maximum cardinality for each sub-sequence involving x_i
- $Left[i] = \sum_{j=1}^{j=i}$ `leftmost`$[j]$.
- $Right[i]$: same as $Left$ but in the reverse order, i.e. $[x_n, .., x_1]$.

AtMostSeqCard$(u = 4, q = 8, d = 12)$

$\mathcal{D}(x_i)$		.	0	.	.	.	.	.	.	0	1	0	**1**	.	.	.	.	.	.	.	.	.	1	
`leftmost`$[i]$		**1**	0	**1**	**1**	**1**	0	0	0	0	1	0	**1**	**1**	**1**	0	0	0	**1**	0	**1**	**1**	1	
$Left[i]$	0	1	1	2	3	4	4	4	4	4	4	**4**	5	6	7	7	7	7	8	8	9	10	10	
$Right[i]$		10	9	9	9	8	7	6	6	6	6	6	6	**5**	4	3	3	3	3	3	2	1	0	0
AC$(\mathcal{D}(x_i))$		**1**	0	.	.	.	.	**0**	**0**	0	1	0	**1**	**1**	**1**	**0**	**0**	**0**	.	.	**1**	**1**	1	

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.

Some observations

Let S: 1 1 0 0 . subject to AtMost(2/5).
→`leftmost` on S gives 1 1 0 0 0

Consider the sequence S_0: 1 1 . 0 .
→`leftmost` on S_0 gives 1 1 0 0 0

Some observations

Let S: 1 1 0 0 . subject to AtMost(2/5).
→`leftmost` on S gives 1 1 0 0 0

Consider the sequence S_0: 1 1 . 0 .
→`leftmost` on S_0 gives 1 1 0 0 0

$$\{x_i \leftarrow 0 \mid max[i] = u\}$$

Some observations

Let S: 1 1 0 0 . subject to AtMost(2/5).
→`leftmost` on S gives 1 1 0 0 0

Consider the sequence S_0: 1 1 . 0 .
→`leftmost` on S_0 gives 1 1 0 0 0

$$\{x_i \leftarrow 0 \mid max[i] = u\}$$

Consider the sequence S_2: . 1 0 0 .
→`leftmost` on S_2 gives 1 1 0 0 0

Some observations

Let S: 1 1 0 0 . subject to AtMost(2/5).
→`leftmost` on S gives 1 1 0 0 0

Consider the sequence S_0: 1 1 . 0 .
→`leftmost` on S_0 gives 1 1 0 0 0

$$\{x_i \leftarrow 0 \mid max[i] = u\}$$

Consider the sequence S_2: . 1 0 0 .
→`leftmost` on S_2 gives 1 1 0 0 0

$$\{x_i \leftarrow 1 \mid max[i] \neq u\}$$

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` once.

Example: $AtMost(2, 5)$

S	1	0	1	0	0	.	.	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1
$\texttt{leftmost}(S(x_i))$	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
$max[i]$	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	.

Size of S is 20 while size of $S^{\cdot *}$ is 9.

Explaining pruning

explanation for $x \leftarrow k$?

1. Add $x \not\leftarrow k$ to the instantiation where the pruning was performed.
2. Use the previous procedure to explain the failure on the new instantiation.

PB & SAT Modelling

Variables:

- c_i^j: c_i^j is *true* iff the class of the ith slot is j.
- y_i^j: y_i^j is *true* iff the ith vehicle requires option j.

Constraints:

- Demand constraints: $\forall j \in [1..k]$, $\sum_i c_i^j = D_j$
- Capacity constraints: $\sum_{l=i}^{i+q_j-1} y_l^j \leq u_j$
- Channelling:
  - $\forall i \in [1..n]$, $\forall l \in [1..k]$, we have:
    - $\forall j \in \mathcal{O}_l$, $\overline{c_i^l} \vee y_i^j$
    - $\forall j \notin \mathcal{O}_l$, $\overline{c_i^l} \vee \overline{y_i^j}$
  - a redundant clause: $\forall i \in [1..n]$, $j \in [1..m]$, $\overline{y_i^j} \vee \bigvee_{l \in \mathcal{C}_j} c_i^l$
- $\forall i \in [1..n]$, $\sum_j c_i^j = 1$

PB & SAT Modelling

Variables:

- c_i^j: c_i^j is *true* iff the class of the ith slot is j.
- y_i^j: y_i^j is *true* iff the ith vehicle requires option j.

Constraints:

- Demand constraints: $\forall j \in [1..k]$, $\sum_i c_i^j = D_j$
- Capacity constraints: $\sum_{l=i}^{i+q_j-1} y_l^j \leq u_j$
- Channelling:
  - $\forall i \in [1..n]$, $\forall l \in [1..k]$, we have:
    - $\forall j \in \mathcal{O}_l$, $\overline{c_i^l} \vee y_i^j$
    - $\forall j \notin \mathcal{O}_l$, $\overline{c_i^l} \vee \overline{y_i^j}$
  - a redundant clause: $\forall i \in [1..n], j \in [1..m], \overline{y_i^j} \vee \bigvee_{l \in \mathcal{C}_j} c_i^l$
- $\forall i \in [1..n]$, $\sum_j c_i^j = 1$

SAT model? encode Cardinality constraints: Sequential counter, Cardinality Networks, Sorting network, etc.

PB & SAT Modelling

Variables:

- c_i^j: c_i^j is *true* iff the class of the ith slot is j.
- y_i^j: y_i^j is *true* iff the ith vehicle requires option j.

Constraints:

- Demand constraints: $\forall j \in [1..k], \sum_i c_i^j = D_j$
- Capacity constraints: $\sum_{l=i}^{i+q_j-1} y_l^j \leq u_j$
- Channelling:
  - $\forall i \in [1..n], \forall l \in [1..k]$, we have:
    - $\forall j \in \mathcal{O}_l, \overline{c_i^l} \vee y_i^j$
    - $\forall j \notin \mathcal{O}_l, \overline{c_i^l} \vee \overline{y_i^j}$
  - a redundant clause: $\forall i \in [1..n], j \in [1..m], \overline{y_i^j} \vee \bigvee_{l \in \mathcal{C}_j} c_i^l$
- $\forall i \in [1..n], \sum_j c_i^j = 1$

SAT model? encode Cardinality constraints: Sequential counter, Cardinality Networks, Sorting network, etc.

Sequential Counter C_C[Sin05]

Encoding $\sum_{i \in [1..n]} x_i = d$ to a CNF ?

- Variables:
  - $s_{i,j}$: $\forall i \in [0..n], \forall j \in [0..d+1]$, $s_{i,j}$ is *true* iff $\sum_{k \in [1..i]} x_k \geq j$
- Encoding: $\forall i \in [1..n]$
  - Clauses for restrictions on the same level: $\forall j \in [0..d+1]$
    1. $\neg s_{i-1,j} \vee s_{i,j}$
    2. $x_i \vee \neg s_{i,j} \vee s_{i-1,j}$
  - Clauses for increasing the counter, $\forall j \in [1..d+1]$
    3. $\neg s_{i,j} \vee s_{i-1,j-1}$
    4. $\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}$
  - Initial values for the bounds of the counter:
    5. $s_{0,0} \wedge \neg s_{0,1} \wedge s_{n,d} \wedge \neg s_{n,d+1}$

Sequential Counter C_C[Sin05]

Encoding $\sum_{i \in [1..n]} x_i = d$ to a CNF ?

- Variables:
  - $s_{i,j}$: $\forall i \in [0..n]$, $\forall j \in [0..d+1]$, $s_{i,j}$ is *true* iff $\sum_{k \in [1..i]} x_k \geq j$
- Encoding: $\forall i \in [1..n]$
  - Clauses for restrictions on the same level: $\forall j \in [0..d+1]$
    1. $\neg s_{i-1,j} \vee s_{i,j}$
    2. $x_i \vee \neg s_{i,j} \vee s_{i-1,j}$
  - Clauses for increasing the counter, $\forall j \in [1..d+1]$
    3. $\neg s_{i,j} \vee s_{i-1,j-1}$
    4. $\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}$
  - Initial values for the bounds of the counter:
    5. $s_{0,0} \wedge \neg s_{0,1} \wedge s_{n,d} \wedge \neg s_{n,d+1}$

Unit Propagation on this encoding enforces AC on $\sum_{i \in [1...n]} x_i = d$.

Extension to AtMostSeqCard

AtMostSeqCard: Cardinality⊕ AtMost→C_C on Cardinality and C_A on each AtMost.

Extension to AtMostSeqCard

AtMostSeqCard: Cardinality$\oplus$ AtMost$\rightarrow$C$_C$ on Cardinality and C$_A$ on each AtMost.

Other possibility: Using a similar encoding of the Gen-Sequence constraint [Bac07, BNQ+07] (C$_S$).

7. $\neg s_{i,j} \vee s_{i-q,j-u}$

Extension to AtMostSeqCard

AtMostSeqCard: Cardinality$\oplus$ AtMost$\rightarrow$$\mathrm{C}_C$ on Cardinality and C_A on each AtMost.

Other possibility: Using a similar encoding of the Gen-Sequence constraint [Bac07, BNQ+07] (C_S).

8 $\neg s_{i,j} \vee s_{i-q,j-u}$

Proposition

The level of pruning using C_S is incomparable with C_A.

GAC on AtMostSeqCard

Theorem

UP on $\mathrm{C}_C + \mathrm{C}_A + \mathrm{C}_S$ *enforces GAC on the* AtMostSeqCard *constraint.*

Configuration

- SAT:
  1. *SAT (1)* $\mathrm{C}_C \oplus \mathrm{C}_A$
  2. *SAT (2)* $\mathrm{C}_C \oplus \mathrm{C}_S$
  3. *SAT (3)* $\mathrm{C}_C \oplus \mathrm{C}_A \oplus \mathrm{C}_S$.
- Mistral as a hybrid CP/SAT solver:
  1. *hybrid (VSIDS)*
  2. *hybrid (Slot)*
  3. *hybrid (Slot/VSIDS)*
  4. *hybrid (VSIDS/Slot)*
- Baseline methods:
  1. *CP*: A pure CP approach
  2. *PBO-clauses*: SAT encoding [MiniSat+]
  3. *PBO-cutting planes*: [SAT4J]

Method	sat [easy] (74 × 5)			sat [hard] (7 × 5)			unsat* (28 × 5)		
	#suc	avg fails	time	#suc	avg fails	time	#suc	avg fails	time
SAT (1)	370	2073	1.71	28	337194	282.35	**85**	**249301**	**105.07**
SAT (2)	370	1114	0.87	31	60956	56.49	65	220658	197.03
SAT (3)	370	612	0.91	34	32711	36.52	77	190915	128.09
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
CP	**370**	**43**	**0.03**	35	57966	16.25	0	-	-
PBO-clauses	277	538743	236.94	0	-	-	43	175990	106.92
PBO-cutting planes	272	2149	52.62	0	-	-	1	5031	53.38

Table : Experimental results

Method	sat [easy] (74 × 5)			sat [hard] (7 × 5)			unsat* (28 × 5)		
	#suc	avg fails	time	#suc	avg fails	time	#suc	avg fails	time
SAT (1)	370	2073	1.71	28	337194	282.35	**85**	**249301**	**105.07**
SAT (2)	370	1114	0.87	31	60956	56.49	65	220658	197.03
SAT (3)	370	612	0.91	34	32711	36.52	77	190915	128.09
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
CP	**370**	**43**	**0.03**	35	57966	16.25	0	-	-
PBO-clauses	277	538743	236.94	0	-	-	43	175990	106.92
PBO-cutting planes	272	2149	52.62	0	-	-	1	5031	53.38

Table : Experimental results

Finding solutions quickly:

- Propagation is very important to find solutions quickly when they exist, by keeping the search "on track" and avoiding exploring large unsatisfiable subtrees.

Method	sat [easy] (74 × 5)			sat [hard] (7 × 5)			unsat* (28 × 5)		
	#suc	avg fails	time	#suc	avg fails	time	#suc	avg fails	time
SAT (1)	370	2073	1.71	28	337194	282.35	**85**	**249301**	**105.07**
SAT (2)	370	1114	0.87	31	60956	56.49	65	220658	197.03
SAT (3)	370	612	0.91	34	32711	36.52	77	190915	128.09
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
CP	**370**	**43**	**0.03**	35	57966	16.25	0	-	-
PBO-clauses	277	538743	236.94	0	-	-	43	175990	106.92
PBO-cutting planes	272	2149	52.62	0	-	-	1	5031	53.38

Table : Experimental results

For proving unsatisfiability

- Clause learning is by far the most critical factor.
- Surprisingly, the "lightest" encoding gave best results!

Conclusion

Contributions

- First non-trivial SAT encodings for the car-sequencing problem.
- A linear time explanation for ATMOSTSEQCARD
- A SAT encoding of ATMOSTSEQCARD maintaining *GAC*
- Closing 13 out of the 23 large open instances.

Thank you!

Fahiem Bacchus.
GAC Via Unit Propagation.
In *Proceedings of CP*, pages 133–147, 2007.

Sebastian Brand, Nina Narodytska, Claude-Guy Quimper, Peter J. Stuckey, and Toby Walsh.
Encodings of the Sequence Constraint.
In *Proceedings of CP*, pages 210–224, 2007.

Carsten Sinz.
Towards an Optimal CNF Encoding of Boolean Cardinality Constraints.
In *Proceedings of CP*, pages 827–831, 2005.