SAT Encodings for the AtMostSeqCard Constraint with Application on Car Sequencing

Valentin Mayer-Eichberger

NICTA
University of New South Wales
valentin.mayer-eichberger@nicta.com.au

1 Introduction

We will present several encodings of the car sequencing constraints and discuss their differences. First we start with a rather simple and direct translation of the description of the problem and then show gradually how to come up with a better encoding. In the constructions we will show how to use a partial sum and order encoding, how to compactly translate cardinality constraints and strengthening propagation by focusing on the auxiliary variables.

2 Motivation

We are seeking an compact encoding that enforces GAC on the AtMostSeqCard constraint ([Siala et al., 2012]). This constraint treats a special case of the Sequence constraint ([van Hoeve et al., 2006]) where the lower bound for the Amongs is 0. Here we will show that there are compact CNF encodings that shows good results on benchmarks in the CSPLIB [**?**].

3 Two simple Encoding

1. Encoding by boolean variables $x_{i,k}$ for class k in position i and a cardinality constraint on the total number of classes on the sequence and cardinality constraints on each sub window restricting the capacity on options.
2. As 1) but introducing auxiliary variables $x_{i,o}$ for option o in position i and converting the capacity constraints to use these variables.

4 Encoding of a single AtMostSeqCard Constraint

We will first show how to encode a cardinality constraint with a counter encoding (ref , [Eén and Sörensson, 2006] ?) and then integrate the AtMostSeq by reusing the auxiliary variables of the counter encoding.

Over this whole section we will work with the following notation. Given a set of consecutive positions $P = \{1 \dots n\}$ and a property that holds at a position $i \in P$ iff the boolean variable x_i is true.

4.1 Encoding of Counters

We want to encode the following cardinality constraint

$$\sum_{i \in \{1 \ldots n\}} x_i = d$$

where d is a fixed value. The encoding given here is named a counter encoding. The idea is to encode cumulative sums and an order encoding ([Tamura et al., 2009]) on the auxiliary variables. Note that the given encoding is not the most compact for a single cardinality constraint. However, the auxiliary variables are subsequently reused to encode the sequence of AtMost constraints.

We introduce the following boolean variables (representing cumulative sums):

- $y_{i,j}$ is true iff in the positions $1 \ldots i$ the property holds at least j times.

The following formula clarifies the relationship between x and y.

$$y_{i,j} \iff (j \leq \sum_{l=0}^{i} x_l)$$

Fig. 1. The variables $y_{i,j}$ with an upper bound d of two over a sequence of 10. By preprocessing the variables corresponding to the cells containing $U(L)$ and above(below) are set to false (true). The question mark identifies yet unassigned variables.

3			U	U	U	U	U	U	U	U	U
2		U	?	?	?	?	?	?	?	?	L
1	U	?	?	?	?	?	?	?	?	L	
0	L	L	L	L	L	L	L	L	L		
j/i	0	1	2	3	4	5	6	7	8	9	10

The following binary clauses relate the variables y among each other:

$$\bigwedge_{i \in \{0 \ldots n-1\}} \bigwedge_{j \in \{0..d+1\}} \neg y_{i,j} \vee y_{i+1,j} \tag{1}$$

$$\bigwedge_{i \in \{1..n\}} \bigwedge_{j \in \{1..d+1\}} \neg y_{i,j} \vee y_{i-1,j-1} \tag{2}$$

These clauses restrict assignments of the auxiliary variables to represent a counter.

Now we need to relate these variables to x. First we restrict the counter not to increase if x_i is false:

$$\bigwedge_{i\in\{1\ldots n\}} \bigwedge_{j\in\{0..d+1\}} x_i \vee \neg y_{i,j} \vee y_{i-1,j} \tag{3}$$

Second we define clauses that push the counter up if x_i is true.

$$\bigwedge_{i\in\{0\ldots n-1\}} \bigwedge_{j\in\{0..d\}} \neg x_{i+1} \vee \neg y_{i,j} \vee y_{i+1,j+1} \tag{4}$$

Now we finally have to "initialize" the counter.

$$y_{n,d} \wedge \left(\bigwedge_{i\in\{0\ldots n\}} y_{i,0} \right) \wedge \neg y_{0,1} \wedge \left(\bigwedge_{i\in\{0\ldots n\}} \neg y_{i,d+1} \right) \tag{5}$$

Fig. 2. Taking the previous example and let x_i be true for position 2 and 7. Then the resulting assignment to the counter variable is given in this table.

3			U	U	U	U	U	U	U	U	U
2		U	0	0	0	0	0	1	1	1	L
1	U	0	1	1	1	1	1	1	1	L	
0	L	L	L	L	L	L	L	L	L		
j/i	0	1	2	3	4	5	6	7	8	9	10
x_i	0	0	1	0	0	0	0	1	0	0	0

It should be mentioned that with (6) we instruct the counter to measure exactly d occurences. It is easy to change the intializer to at most d or at least d occurences.

4.2 Extending to AtMostSeqCard

Given a sequence of boolean variables among exactly d have to be true and each window of size q cannot contain more than u true variables. This is the AtMostSeqCard constraint:

$$\text{AtMostSeqCard } (u, q, d, [x_1, \ldots, x_n]) \iff (\sum_{i=1}^{n} x_i = d) \wedge \bigwedge_{i=1}^{n-q} (\sum_{l=1}^{q} x_{i+l} \leq u)$$

To archive GAC on this constraint we need to take the counter encoding of the previous section and add the following binary clauses:

$$\bigwedge_{\substack{i \in \{1 \dots n\} \\ i-q \geq 0}} \bigwedge_{\substack{j \in \{1 \dots d+1\} \\ j-u \geq 0}} \neg y_{i,j} \vee y_{i-q,j-u} \tag{6}$$

This seems suprising!

Theorem 1. *The clauses of counter encoding (1) to (5) with the clauses of (6) are logically equivalent to the AtMostSeqCard . Moreover they detect disentailment with UP and thus enforce GAC with failed litenforce GAC with UP and failed literal test.*

Proof. The proof uses ideas from the decompositions of the Sequence Constraint [**?**] and the encoding by cumulative sums, see [Brand et al., 2007]

4.3 Size of Encoding

Let $s = n \cdot d -$ upper and lower triangle, then we generate s auxiliary variables and $3 \cdot s$ binary clauses and $2 \cdot s$ tenery clauses. The term $n \cdot d$ dominates. The precise number of variables can be computed by the hight d and the slope u/q and a little bit of algebra, which leads to $d \cdot (u - q + \frac{d \cdot q}{u})$. So the number of variables s are

$$s = d \cdot n - d \cdot (u - q + \frac{d \cdot q}{u})$$

Thus the size of this encoding lies in $O(nd)$, but can be more compact if q and u are rather strict and/or d is close to n. For example AtMostSeqCard ($u = 4, q = 8, d = 12, n = 22$) would have
$12 \cdot 22 - 12 \cdot (4 - 8 + \frac{12 \cdot 8}{4}) = 24$ variables.

Conjecture 1. The clauses (1) to (6) enforce GAC on the AtMostSeqCard on every partial assignment with the failed literal test.

The power of the binary clauses are best shown in the example [Siala et al., 2012].

5 Encoding of Carsequencing

We need to relate the cars and options as in the problem specification. This can be done in two ways. First the straight forward way.

Fig. 3. Here we analyse the initial state of variables for a constraint with $u = 4, q = 8, d = 12, n = 22$. In this example we see that x_7, x_8, x_{15} and x_{16} should be false, this is detected by the failed literal test, e.g. the encoding detects dis-entailment.

13																					U	U	U
12																				U	?	?	L
11																			U	?	?	L	
10																		U	?	?	L		
9													U	U	U	U	U	?	?	L			
8												U	?	?	L	L	L	L	L				
7											U	?	?	L									
6										U	?	?	L										
5					U	U	U	U	U	?	?	L											
4				U	?	?	L	L	L	L	L												
3			U	?	?	L																	
2		U	?	?	L																		
1	U	?	?	L																			
0	L	L	L																				
j/i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
x_i								**0**	**0**							**0**	**0**						

Fig. 4. State of the auxiliary variables for $u = 4, q = 8, d = 12, n = 22$ and choices x_1 and x_{13} to true and x_{12}, x_{14} and x_{21} to false. Notice the amount of propagation due to the clauses of the AtMostSeqCard constraint, also notice that variable x_1 was a redundant choice. Normal font= choice, bold = propagated, ()= dis-entailment detected, green arrow positve implication, red arrow negative implication by (6).

13																					U	U	U
12																				U	0	0	L
11																			U	?	1	L	
10																		U	?	1	L		
9													U	U	U	U	U	?	1	L			
8												U	0	1	L	L	L	L	L				
7											U	1	1	L									
6										U	1	1	L										
5					U	U	U	U	U	1	1	L											
4				U	?	1	L	L	L	L	L												
3			U	1	1	L																	
2		U	1	1	L																		
1	U	1	1	L																			
0	L	L	L																				
j/i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
x_i		1	**1**	**1**			**0**	(0)	(0)	**1**	**1**	**1**	0	1	0	(0)	(0)					0	**1**

5.1 Relating Cars and Option directly

Let C be the index set identifying a class and O be the index set for options. The instance for a car sequencing problem is given by a mapping $m : C \to 2^O$, relating to each class a set of options. For each class we have a cardinality constraint and for each option a AtMostSeq constraint. From this we can construct for each option and for each class a AtMostSeqCard constraint (since all cars have to be assigned).

Each such AtMostSeqCard constraint is encoded into SAT. In addition we need to relate classes and options on each position. This is done by the following clauses. Let $m' : O \to 2^C$ be the mapping relating to each option the corresponding classes.

$$\bigwedge_{i \in \{1 \ldots n\}} \bigwedge_{\substack{c \in C \\ o \in m(c)}} \neg x_{i,c} \vee x_{i,o} \tag{7}$$

and the reverse

$$\bigwedge_{i \in \{1 \ldots n\}} \bigwedge_{o \in O} \left(\neg x_{i,o} \vee \bigvee_{c \in m'(o)} x_{i,c} \right) \tag{8}$$

Notice in an ASP encoding this would be modelled by one rule and the completion semantics covers for the reverse case.

5.2 Alternative: The purest encoding only relating the auxiliary variables

Here we will show that there is an encoding of the car sequencing problem that does not use at all the variables $x_{i,k}$. The encoding builds entirely on the auxiliary variables $y_{i,j,k}$ and their relationship. This is rather surprising.

This encoding will take care of a propagation that is not covered by the previous encodings and as such has stronger properties taking the auxiliary variables into account! give example which propagation is missing

Here the idea:

For each position in the sequence, the sum of true y for all cars that have option o need to be the same size as the number of true y for o. As a formula, for all positions i and options $o \in O$ it has to hold:

$$\sum_j y_{i,j,o} = \sum_{c \in m'(o)} \sum_j y_{i,j,c}$$

Encoding this restriction should be enough to fully grasp the car sequencing problem. The construction can be done by a sorting network (e.g. [Batcher, 1968]) and the size should be $n \cdot log^2(d)$ in the hight d of the counter. For pure cardinality constraints this can be improved in a CNF encoding as in [Asín et al., 2011,Codish and Zazon-Ivry, 2010], but here we are faced with an equvialence which needs a full sorter. Maybe the recursive structure of the cummulative sum can be exploited in this construction, since in position $i-1$ we have already almost have sorted the sequence with up to one difference.

Another interesting aspect of the equality above and its encoding is its relation to UTVPI enconstraints (see [Seshia et al., 2007] for a recent treatment). Sorting Networks give an interesting application for a similar types of constraint (and even a generalisation of them).

6 Evaluation

Best of results that can be robustly (standard heuristics) archived by current sat solvers. We compared newest version of minisat, lingeling, cryptominisat, glucose and clasp and they all consistently find solutions within 1h runtime.

Table 1.

	set1	**set2**	**set3**	**set4**
sat	70	4	0	7
unsat	0	0	4	13
unknown	0	0	1	10

This is by far better than most papers evaluating the car sequencing problem on some specialized algorithm (e.g. branch and bound) or special constraint (CP) or optimization (IP).

For the set 4 a more detailed view is interesting as the benchmark targets the optimization version of the car sequencing problem.

We can solve 7 satisfiable instances and prove 13/23 instances to be unsatisfiable.

7 Extensions

- Optimizations: there are two definitions of the cost function for the car sequencing problem. First is to allow arbitrary cars without any options and minimize the number of cars with options. And second is to minimize the number of windows that exceed the capacity constraint on their options. It would be interesting to compare both definition and to evaluate against published results in the literature. There are still gaps between known upper and lower bounds.
- There is a natural extension of the AtMostSeqCard constraint that to a cyclic version and in the same and natural way we can extend the encoding given above. It would be interesting to find good benchmarks.
- The Sequence constraint consists of a sequence of among constraints and we should compare this encoding to the known CNF encodings and filtering algorithms in the literature.

Table 2. Solutions to the benchmark proposed in [Gravel et al., 2005] with minimum violations found on the target function (violated capacity of options per window) by a local search method and compared to solutions on the decision version SAT encoding with lingeling (LING).

name	**min**	**LING**	**sec**
200-01	0	SAT	189.9
300-01	0	SAT	315.7
400-01	1	?	-
200-02	2	?	-
300-02	12	?	-
400-02	16	?	-
200-03	4	UNSAT	70.2
300-03	13	UNSAT	873.0
400-03	9	UNSAT	88.1
200-04	7	UNSAT	19.7
300-04	7	UNSAT	33.6
400-04	19	UNSAT	83.2
200-05	6	UNSAT	543.7
300-05	29	UNSAT	9.7
400-05	0	SAT	2146.1
200-06	6	?	-
300-06	2	?	-
400-06	0	SAT	605.4
200-07	0	SAT	30.2
300-07	0	SAT	122.6
400-07	4	?	-
200-08	8	?	-
300-08	8	UNSAT	65.9
400-08	4	?	-
200-09	10	UNSAT	350.4
300-09	7	?	-
400-09	5	UNSAT	220.9
200-10	19	UNSAT	9.9
300-10	21	UNSAT	18.3
400-10	0	SAT	468.1

– Analyzing the quality of SAT encodings our results suggests to evaluate the consisty archived on the auxiliary variable introduced. E.g. in the pure encoding we even had a stronger notion of propagation, as also on all partial assignment on $y_{i,j,k}$ we archive GAC.

References

[Asín et al., 2011] Asín, R., Nieuwenhuis, R., Oliveras, A., and Rodríguez-Carbonell, E. (2011). Cardinality Networks: a theoretical and empirical study. *Constraints*, 16(2):195–221.

[Batcher, 1968] Batcher, K. E. (1968). Sorting Networks and Their Applications. In *AFIPS Spring Joint Computing Conference*, pages 307–314.

[Brand et al., 2007] Brand, S., Narodytska, N., Quimper, C.-G., Stuckey, P. J., and Walsh, T. (2007). Encodings of the Sequence Constraint. In *CP*, pages 210–224.

[Codish and Zazon-Ivry, 2010] Codish, M. and Zazon-Ivry, M. (2010). Pairwise Cardinality Networks. In *LPAR (Dakar)*, pages 154–172.

[Eén and Sörensson, 2006] Eén, N. and Sörensson, N. (2006). Translating Pseudo-Boolean Constraints into SAT. *JSAT*, 2(1-4):1–26.

[Gravel et al., 2005] Gravel, M., Gagné, C., and Price, W. L. (2005). Review and Comparison of Three Methods for the Solution of the Car Sequencing Problem. *The Journal of the Operational Research Society*, 56(11):1287–1295.

[Seshia et al., 2007] Seshia, S. A., Subramani, K., and Bryant, R. E. (2007). On Solving Boolean Combinations of UTVPI Constraints. *JSAT*, 3(1-2):67–90.

[Siala et al., 2012] Siala, M., Hebrard, E., and Huguet, M.-J. (2012). An Optimal Arc Consistency Algorithm for a Chain of Atmost Constraints with Cardinality. In *CP*, pages 55–69.

[Tamura et al., 2009] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M. (2009). Compiling finite linear CSP into SAT. *Constraints*, 14(2):254–272.

[van Hoeve et al., 2006] van Hoeve, W. J., Pesant, G., Rousseau, L.-M., and Sabharwal, A. (2006). Revisiting the Sequence Constraint. In *CP*, pages 620–634.