A SAT Encoding for the AtMostSeqCard Constraint

Valentin Mayer-Eichberger

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

1 Introduction

Give description of the car sequencing problem and the straight forward encoding in IP/CNF.

The naive CNF and IP encoding of the car sequencing benchmark is far from optimal. In this paper we will show gradually how to come up with a better encoding.

2 Motivation

We are seeking an encoding that enforces GAC on the recently proposed AtMost-SeqCard constraint (ref). This constraint is not as expressive as the Sequence constraint but is more suited for some benchmark problems and has a linear filtering algorithm. Here we will show that there is a compact CNF encoding that shows good results in the benchmark set of the CSPLIB. Furthermore we will try to improve the bounds on the set of hard instance.

3 Encoding of one AtMostSeqCard

We will first show how to encode a cardinality constraint with a counter encoding (ref) 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.

3.1 Encoding of Counters

We want to encode the following cardinality constraint

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

where d is a fixed value. We call the encoding for such a cardinality constraint a counter encoding because we count exactly d occurrences over positions $\{1 \ldots n\}$.

We introduce the following variables:

- $y_{i,j}$ 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 unassigned variables of the counter encoding

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

There are two types of binary clauses that relate the variables y among each other and two types of ternary clauses that coordinate y with the variables x.

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

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

These clauses restrict the structure of the auxiliary variables to consist of 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.

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

Now we finally have to "initialize" the counter. Here we decided whether the cardinality constraints is $=$, $\leq$ or $\geq$. In our case we are only interested in equality.

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

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

3.2 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:

$$\text{AtMostSeqCard}(u, q, d, [x_1, \ldots, x_n]) \iff \bigwedge_{i=0}^{n-q} (\sum_{i=1}^{q} x_{i+l} \leq u) \wedge (\sum_{i=1}^{n} x_i = d)$$

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

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

Theorem 1. *The clauses of counter encoding (1) to (5) with the clauses of (6) enforce GAC on the AtMostSeqCard for all partial assignment on variables x_i.*

We even have a stronger notion of propagation, because also on all partial assignment on $y_{i,j}$ we archive GAC.

Find precise size of the encoding: Let $s = n * d -$ upper and lower triangle, then we generate s auxiliary variables and $3 \cdot s$ binary clauses and $2 \cdot s$ tenery clauses.

The power of the binary clauses are best shown in the example taken from (ref)

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 already x_7, x_8, x_{15} and x_{16} need to be set to false. Notice that for this particular case we only need 24 boolean variables to create the GAC encoding.

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**					

4 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. 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, ()= earlier propagated, 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**

4.1 Relating Cars and Option directly

Let C be the index set identifying a class and O be the index for options. The instance for a car sequencing problem gives us a mapping $m : C \rightarrow 2^O$, relating to each class a set of options.

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

and the reverse

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

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

4.2 The Purist's Way

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 and can consistently identify all solutions to this problem. This is rather surprising. Here the idea:

The following formula can be encoded as a counter; For each position, the sum of true y for all cars that have option o need to be the same size as the counter for o.

5 Evaluation

Best of results that can be robustly (standard heuristics) archived by current sat solvers. I 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 all 7 satisfiable instances and prove 13/23 instances to be unsatisfiable.

Table 2. Solutions to the benchmark proposed in ref with lower and upper bounds on the target function (min,max) and this compared to solutions on the decision version SAT encoding with lingeling (LING).

name	min	max	LING	sec
200-01	0	3	SAT	189.9
300-01	0	4	SAT	315.7
400-01	1	4	?	-
200-02	2	3	?	-
300-02	12	13	?	-
400-02	16	20	?	-
200-03	4	8	UNSAT	70.2
300-03	13	14	UNSAT	873.0
400-03	9	11	UNSAT	88.1
200-04	7	9	UNSAT	19.7
300-04	7	10	UNSAT	33.6
400-04	19	20	UNSAT	83.2
200-05	6	8	UNSAT	543.7
300-05	29	35	UNSAT	9.7
400-05	0	1	SAT	2146.1
200-06	6	6	?	-
300-06	2	7	?	-
400-06	0	2	SAT	605.4
200-07	0	0	SAT	30.2
300-07	0	2	SAT	122.6
400-07	4	7	?	-
200-08	8	8	?	-
300-08	8	8	UNSAT	65.9
400-08	4	7	?	-
200-09	10	10	UNSAT	350.4
300-09	7	9	?	-
400-09	5	10	UNSAT	220.9
200-10	19	20	UNSAT	9.9
300-10	21	25	UNSAT	18.3
400-10	0	3	SAT	468.1

6 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.