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 AtMostSeqCard 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 (class or option) i. Let $x_{i,j}$ be true iff property i is true at position j. In this section we embedd one counter for one property i, so i is fixed over the whole section. The interaction with other properties is shown in the subsequent section.

3.1 Encoding of Counters

We want to encode the following cardinality constraint

$$\sum_{j \in P} x_{i,j} = d$$

where d is a fixed value. We call the encoding for such a cardinality constraint a counter encoding because we count exactly d occurences of property i over positions P.

So given an option or a class identified by subscript $i \in I$ that needs to be counted over sequence of positions P. We introduce the following variables:

- $y_{i,j,k}$ is true iff at least k times holds property i in the positions $0 \dots j$.

Notice that we assume the set P to have a total ordering. We count along this ordering. The following formula clearifies the relationship between x and y.

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

Fig. 1. The variables $y_{i,j,k}$ for a countable property i with an upper bound of two over a sequence of 10. The variables corresponding to the cells containing $U(L)$ are set to false (true). The question mark identifies a unassigned variable 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		
k/j	0	1	2	3	4	5	6	7	8	9	10

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

$$\bigwedge_{\substack{j \in P \\ j+1 \in P}} \bigwedge_{k \in \{1..d\}} \neg y_{i,j,k} \vee y_{i,j+1,k} \tag{1}$$

$$\bigwedge_{\substack{j \in P \\ j-1 \in P}} \bigwedge_{k,k-1 \in \{1..d\}} \neg y_{i,j,k} \vee y_{i,j-1,k-1} \tag{2}$$

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 define clauses that consist of pushing the counter up.

$$\bigwedge_{\substack{j \in P \\ j+1 \in P}} \bigwedge_{k,k+1 \in \{0..d+1\}} \neg x_{i,j} \vee \neg y_{i,j,k} \vee y_{i,j+1,k+1} \tag{3}$$

And now we have to restrict the counter to be pushed up if $x_{i,j}$ is false.

$$\bigwedge_{\substack{j\in P\\ j+1\in P}} \bigwedge_{k,k+1\in\{1..d+1\}} \neg x_{i,j} \vee \neg y_{i,j,k} \vee y_{i,j+1,k+1} \tag{4}$$

Fig. 2. Taking the previous example and let $x_{i,j}$ 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		
k/j	0	1	2	3	4	5	6	7	8	9	10
$x_{i,j}$	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:

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

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

$$\bigwedge_{\substack{j\in P\\ j-q\in P}} \bigwedge_{\substack{k\in\{1..d\}\\ k-u\in\{1..d\}}} \neg y_{i,j,k} \vee y_{i,j-q,k-u} \tag{5}$$

Theorem 1. *The clauses of (1) to (5) enforce GAC on the AtMostSeqCard for all partial assignment on variables $x_{i,j,}$.*

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

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. 3. Taking the inital state of the propagator for a constraint with $u = 4, q = 8, d = 12, n = 22$. In this example we see that already $x_{i,7}$, $x_{i,8}$, $x_{i,15}$ and $x_{i,16}$ 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																				
k/j	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 auxiliray variables for $u = 4, q = 8, d = 12, n = 22$ and choices $x_{i,1}$ and $x_{i,13}$ to true and $x_{i,12}$, $x_{i,14}$ and $x_{i,21}$ to false. Notice the amount of propagation due to the clauses of the AtMostSeqCard constraint, also notice that variable $x_{i,1}$ was a redundant choice.

13																				U	U	U	
12																			U	0	0	L	
11																		U	?	?	L		
10																	U	?	?	L			
9												U	U	U	U	U	?	?	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																				
k/j	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 $i \in C$ be the index identifying a class and $j \in O$ be the index for options. The problem instance gives us a mapping $m : C \to 2^O$, relating to each class a set of options.

$$\bigwedge_{p \in P} \bigwedge_{\substack{i \in C \\ j \in m(i)}} \neg x_{i,p} \vee x_{j,p} \tag{6}$$

and the reverse

$$\bigwedge_{p \in P} \bigwedge_{j \in O} (\neg x_{i,p} \vee \bigvee_{\substack{i \in C \\ j \in m(i)}} x_{i,p}) \tag{7}$$

Notice in an ASP encoding this would be modelled by one rule and the completion semantics is equivalent to this encoding.

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 $x_{i,j}$. The encoding builds entirely on the auxiliary variables and can consistently identify all solutions to this problem. This is rather surprising. Here the idea:

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 proposed hard benchmark on the 2004 paper (IP) and solutions on the decision version on the SAT encoding (SAT).

Instance	IP	SAT
pb-200-01	0	SAT
pb-300-01	0	SAT
pb-400-01	1	UNKNOWN
pb-200-02	2	UNKNOWN
pb-300-02	12	UNKNOWN
pb-400-02	16	UNKNOWN
pb-200-03	4	UNSAT
pb-300-03	13	UNSAT
pb-400-03	9	UNSAT
pb-200-04	7	UNSAT
pb-300-04	7	UNSAT
pb-400-04	19	UNSAT
pb-200-05	6	UNSAT
pb-300-05	29	UNSAT
pb-400-05	0	SAT
pb-200-06	6	UNKNOWN
pb-300-06	2	UNKNOWN
pb-400-06	0	SAT
pb-200-07	0	SAT
pb-300-07	0	SAT
pb-400-07	4	UNKNOWN
pb-200-08	8	UNKNOWN
pb-300-08	8	UNSAT
pb-400-08	4	UNKNOWN
pb-200-09	10	UNSAT
pb-300-09	7	UNKNOWN
pb-400-09	5	UNSAT
pb-200-10	19	UNSAT
pb-300-10	21	UNSAT
pb-400-10	0	SAT

6 Extensions

- Optimizations: there are two definitions of the cost function for the car sequencing problem. First is to allow arbirary 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 algorithmsin the literature.