Solving hard sequencing problems via the AtMostSeqCard constraint

Mohamed Siala, Christian Artigues, Eric Hébrard, Marie-José Huguet

Solving hard sequencing problems via the AtMostSeqCard constraint

Mohamed Siala[1,2], Christian Artigues[1,3], Emmanuel Hebrard[1,3], and Marie-José Huguet[1,2]

[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2] Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
[3] Univ de Toulouse, LAAS, F-31400 Toulouse, France
{siala, artigues, hebrard, huguet}@laas.fr

Keywords : Sequencing Problems, Constraint Propagation, Sequence Constraints, Hybrid CP/SAT.

1 Introduction

Sequence constraints are useful in a number of applications. Constraints of this class enforce upper and/or lower bounds on all sub-sequences of variables of a given length within a main sequence. For instance, in *Crew-Rostering* problems, we may want to have an upper bound on the number of worked days in every sub-sequence to meet working regulations. Several constraints of this class have been studied in the *Constraint Programming* (CP) literature such as GenSequence, AmongSeq, etc. An even more general constraint, Regular, can be used to enforce arbitrary patterns on all sub-sequences. However, the more general a constraint is, the higher is the complexity of reasoning about it. We therefore investigate the AtMostSeqCard constraint, a particular case where the sequence of variables is subject to a chain of AtMost constraints (all subsequences of size q have at most u values in a set v) along with a global cardinality [6]. This constraint is useful in *Car-Sequencing* and *Crew-Rostering* problems. In [7], two algorithms designed for the AmongSeq constraint were adapted to this constraint with an $O(2^q n)$ and $O(n^3)$ worst case time complexity, respectively. In [2], another algorithm similarly adaptable to filter the AtMostSeqCard constraint with a time complexity of $O(n^2)$ was proposed. Our contributions start with a an algorithm achieving arc consistency on this constraint with an $O(n)$ (hence optimal) worst case time complexity. Next, we showed that this algorithm can be easily modified to achieve arc consistency on some extensions of this constraint. In particular, the conjunction of a set of m AtMostSeqCard constraints sharing the same scope can be filtered in $O(nm)$. Finally, in [1], we showed how to use this constraint in a hybrid CP/SAT framework in order to solve efficiently car-sequencing problems via powerful clause learning mechanisms. Empirical experiments show outstanding results on *Car-Sequencing* and *Crew-Rostering* benchmarks.

2 Achieving GAC in an optimal worst case time complexity

We consider in this paper tree search procedures in a *Constraint Programming* (CP) framework, where each constraint is associated to a propagator (or filtering algorithm) that prunes unconsistent values w.r.t to the current domains at each node of the search tree. A constraint C is *Generalized Arc-Consistent* (GAC) iff for every variable x in its scope and every value v in the current domain of x, we can extend the current assignment to be consistent while assigning v to x.

Let u, q, d be three integers and $[x_1, \ldots, x_n]$ be a sequence of Boolean variables. We define the AtMostSeqCard constraint as follows :

Definition 1. ATMOSTSEQCARD$(u, q, d, [x_1, \ldots, x_n]) \Leftrightarrow$

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

While the best filtering algorithm for this constraint runs in $O(n^2)$ [2], we showed in [6] that GAC can be achieved in linear time in the worst case. For a lack of space, we only outline the main idea. Let `leftmost` be a greedy rule taking a sequence of Boolean variables as a parameter and returning an assignment maximizing the cardinality of the sequencing w.r.t. the ATMOST constraints (i.e. constraints of type $\sum_{l=1}^{q} x_{i+l} \leq u$). `leftmost` starts from x_1 towards x_n by trying to assign the unassigned values to 1 without violating any of the related ATMOST constraints. The outcome is an assignment with the maximum possible cardinality. We denote $\overrightarrow{w}$ the result of `leftmost` from left to right (i.e. on $[x_1, .., x_n]$) and $\overleftarrow{w}$ its result from right to left (i.e. on $[x_n, \ldots, x_1]$). We also define two vectors L and R such that $L[i] = \sum_{k=1}^{i} \overrightarrow{w}[k]$ and $R[i] = \sum_{k=1}^{i} \overleftarrow{w}[k]$. After computing L and R, the propagator executes the following filtering rules :

- If $L[i] + R[n - i + 1] \leq d$, then assign x_i to 0
- If $L[i - 1] + R[n - i] < d$, then assign x_i to 1

Applying the two rules above in addition to achieving GAC on each ATMOST and on the global cardinality enforces the constraint to be GAC [6]. This can be done in $O(n)$ (hence optimal).

3 Extension

We presented in [6] two possible extensions of this constraint using similar propagation. In particular, take the example of a *Crew-Rostering* problem where an employee cannot work more than one 8-h shift per day and no more than 5 days in any 7 days period. If we consider a boolean variable per shift and employee, then the sequence corresponding to a given employee is subject to (simultaneously) ATMOST(1,3) and ATMOST(5,21) in addition to the global work demand (i.e. a cardinality constraint). In fact, it is very common to have different ATMOST constraints in the same sequence, hence the MULTIATMOSTSEQCARD constraint. Let $u_1, .., u_m, q_1, .., q_m$ be integers.

Definition 2. MULTIATMOSTSEQCARD$(u_1, .., u_m, q_1, .., q_m, d, [x_1, \ldots, x_n]) \Leftrightarrow$

$$\bigwedge_{k=1}^{m} \bigwedge_{i=0}^{n-q_k} (\sum_{l=1}^{q_k} x_{i+l} \leq u_k) \wedge (\sum_{i=1}^{n} x_i = d)$$

Observe first that using m ATMOSTSEQCARD does not enforce GAC on their conjunction [6]. However, we were able to show a complete filtering on MULTIATMOSTSEQCARD in $O(m.n)$ based on an extension of the `leftmost` procedure.

4 Learning

We went one step further with ATMOSTSEQCARD by considering learning in a hybrid CP/SAT context. The most general architecture for that is, *Lazy Clause Generation* [4]. The key point for such frameworks is to explain constraints that can be later used to extract a nogood [3]. Propagators are augmented to give clausal

representation of their actions (i.e. pruning and failure triggering). We outline the main idea for explaining a failure on ATMOSTSEQCARD. A similar procedure is used when explaining pruning. Let $\max(i)$ be the maximum cardinality of any of the q subsequences involving x_i at the the beginning of the ith iteration in `leftmost`. It is computed alongside $\overrightarrow{w}$ and will be useful to explain failure.

The explanation procedure that we propose works as follows : when a failure is triggered, we first consider the set of all assignments in the sequence. Note that this is, by default, a valid explanation of this failure. Then we try to reduce it according to the following rules :

- If x_i is assigned to 0 and $max(i) = u$, then remove $x_i \leftarrow 0$
- If x_i is assigned to 1 and $max(i) \neq u$, then remove $x_i \leftarrow 1$

Note that all the values $\max(i)$ can be generated using one call of `leftmost`, hence the whole procedure runs in $O(n)$.

5 Experimental results

We first evaluated the new propagator embedded in tree search on *Car-Sequencing* and *Crew-Rostering* benchmarks [6]. Tables 1 and 2 summarise these results where *sum* is the default model, *gsc* uses the *IloSequence* constraint of Ilog Solver (GSC) [5], *amsc* uses the ATMOSTSEQCARD propagator, *gsc+amsc* combines GSC with ATMOSTSEQCARD and finally *mamsc* uses the MULTIATMOSTSEQCARD propagator.

Table 1: *Car-Sequencing* results (averaged over 42 heuristics)

propagation	*set1* (70×5)			*set2* (4×5)			*set3* (5×5)			*set4* (7×5)		
	#sol	avg bts	time	#sol	avg bts	time	#sol	avg bts	time	#sol	avg bts	time
sum	11270	174017	10.49	124	1101723	58.75	0	-	> 1200	99	378475	30.83
gsc	14008	1408	3.16	425	131062	109.45	31	55365	276.06	195	23897	53.61
amsc	13497	33600	3.79	**470**	**665205**	**70.56**	16	40326	8.62	**214**	**215349**	**38.45**
gsc+amsc	**14033**	**1007**	**3.03**	439	104823	99.71	**32**	**57725**	**285.43**	202	22974	61.61

Table 2: *Crew-Rostering* results

Heuristic	Most constrained employee						Most constrained shift					
Model	satisfiable (1140)			unsatisfiable (385)			satisfiable (1140)			unsatisfiable (385)		
	#sol	time	avg bts	#sol	time	avg bts	#sol	time	avg bts	#sol	time	avg bts
sum	772	21.93	205087	165	0.06	0	987	20.76	169964	352	19.74	180161
gsc	746	65.75	14133	175	0.98	0	1006	33.30	8875	335	15.97	5145
amsc	818	20.51	147479	215	0.13	330	1061	10.07	90247	362	12.19	108797
mamsc	**842**	**20.78**	**125886**	**270**	**0.05**	**0**	**1074**	**10.94**	**91222**	**377**	**14.63**	**110244**

Note first that the the two filtering (ATMOSTSEQCARD and GSC) are totally different on *Car-Sequencing*. Hence, their combination is worth trying. In *Car-Sequencing*, best results were obtained by ATMOSTSEQCARD or by the new combination. In crew-restoring, however, ATMOSTSEQCARD is by far better than GSC. In addition, one can see the impact of MULTIATMOSTSEQCARD compared to a simple conjunction of ATMOSTSEQCARD. Furthermore, in both benchmarks, we observed that the new extra filtering was faster (in terms of nodes explored per second) by at least an order of magnitude compared to GSC.

Regarding learning, we evaluated the new explanation procedure with the *Car-Sequencing* problem against a pure CP model (details regarding the experiments can be found in [1]). In table 3, the lines *hybrid (VSIDS)*

and *hybrid (Slot)* use a hybrid model with respectively a SAT generic heuristic, and a CP dedicated heuristic whereas *pure-CP* is a state-of-the-art CP model. Note also that *pure-CP* and *hybrid (Slot)* use the same heuristic.

Table 3: Learning 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	739	0.23	35	76256	64.52	**37**	**204858**	**248.24**
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	-	-

Using AtMostSeqCard alone already outperform all the other CP models with a 100% resolution of the satisfiable instances from the CSPLib in a very short time (few seconds). However, CP-Solvers suffer when trying to prove unfeasability (i.e. with unsatisfiable instances). Using the new explanation, however, greatly improve this behaviour and outperforms all CP models. In addition, explanations do not hinder solver's capacity to find solutions quickly. It is even better sometimes (for instance *hybrid (Slot)* on `sat[hard]` compared to *pure-CP*).

6 Conclusion

We considered the AtMostSeqCard constraint for solving hard sequencing problems. While the state-of-the-art filtering algorithm for this constraint runs in $O(n^2)$, we showed how to enforce arc-consistency in an optimal worst case time complexity. Then we presented practical extensions of this constraint and showed how the propagator can be adapted accordingly. We finally proposed a novel linear time mechanism for explaining this constraint in a hybrid CP/SAT context. Our new models outperform previous systematic approaches in *Car-Sequencing* and *Crew-Rostering* benchmarks.

References

1. C. Artigues, E. Hebrard, V. Mayer-Eichberger, M Siala, and T. Walsh. SAT and Hybrid models of the Car-Sequencing problem. In *CSPSAT Workshop, in conjunction with CP 2013*, Uppsala, Sweden, September 2013.
2. M. J. Maher, N. Narodytska, C.-G. Quimper, and T. Walsh. Flow-Based Propagators for the Sequence and Related Global Constraints. In *CP*, pages 159–174, 2008.
3. J.P. Marques-Silva and K.A. Sakallah. Grasp: a search algorithm for propositional satisfiability. *Computers, IEEE Transactions on*, 48(5):506–521, 1999.
4. O. Ohrimenko, P J. Stuckey, and M Codish. Propagation via Lazy Clause Generation. *Constraints*, 14(3):357–391, 2009.
5. J.-C. Régin and J.-F. Puget. A Filtering Algorithm for Global Sequencing Constraints. In *CP*, pages 32–46, 1997.
6. M. Siala, E. Hebrard, and M-J. Huguet. An optimal arc consistency algorithm for a particular case of sequence constraint. *Constraints*, pages 1–27, 2013.
7. W. J. van Hoeve, G. Pesant, L.-M.Rousseau, and A. Sabharwal. New Filtering Algorithms for Combinations of Among Constraints. *Constraints*, 14(2):273–292, June 2009.