

5.294 sliding_sum

	DESCRIPTION	LINKS	GRAPH
Origin	CHIP		
Constraint	<code>sliding_sum(LOW, UP, SEQ, VARIABLES)</code>		
Arguments	LOW : <code>int</code> UP : <code>int</code> SEQ : <code>int</code> VARIABLES : <code>collection(var-dvar)</code>		
Restrictions	$UP \geq LOW$ $SEQ > 0$ $SEQ \leq VARIABLES $ <code>required(VARIABLES, var)</code>		
Purpose	<div style="border: 1px solid pink; padding: 5px;"> Constrains all sequences of SEQ consecutive variables of the collection VARIABLES so that the sum of the variables belongs to interval [LOW, UP]. </div>		
Example	<div style="border: 1px solid blue; padding: 10px; display: inline-block;"> $\left(\begin{array}{c} \text{var} - 1, \\ \text{var} - 4, \\ 3, 7, 4, \left\langle \begin{array}{c} \text{var} - 2, \\ \text{var} - 0, \\ \text{var} - 0, \\ \text{var} - 3, \\ \text{var} - 4 \end{array} \right\rangle \end{array} \right)$ </div>		
	The example considers all sliding sequences of $SEQ = 4$ consecutive values of $\langle 1, 4, 2, 0, 0, 3, 4 \rangle$ collection and constraints the sum to be in $[LOW, UP] = [3, 7]$. The <code>sliding_sum</code> constraint holds since the sum associated with the corresponding subsequences 1 4 2 0, 4 2 0 0, 2 0 0 3, and 0 0 3 4 are respectively 7, 6, 5 and 7.		
Symmetry	Items of VARIABLES can be reversed .		
Algorithm	Beldiceanu and Carlsson [29] have proposed a first incomplete filtering algorithm for the <code>sliding_sum</code> constraint. In 2008, Maher <i>et al.</i> showed in [244] that the <code>sliding_sum</code> constraint has a solution “if and only there are no negative cycles in the flow graph associated with the dual linear program” that encodes the conjunction of inequalities. They derive a bound-consistency filtering algorithm from this fact.		
See also	common keyword: sliding_distribution (<i>sliding sequence constraint</i>). part of system of constraints: sum_ctr . soft variant: relaxed_sliding_sum . used in graph description: sum_ctr .		

Keywords

characteristic of a constraint: hypergraph, sum.

combinatorial object: sequence.

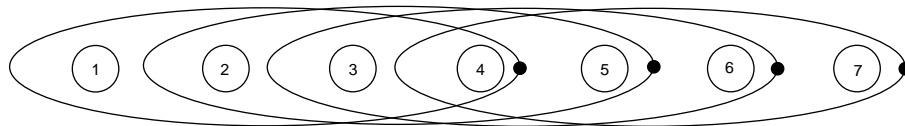
constraint type: decomposition, sliding sequence constraint, system of constraints.

filtering: linear programming, flow, bound-consistency.

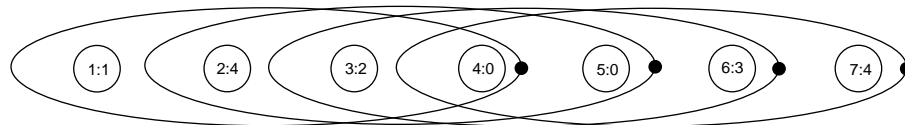
Arc input(s)	VARIABLES
Arc generator	$\text{PATH} \mapsto \text{collection}$
Arc arity	SEQ
Arc constraint(s)	<ul style="list-style-type: none"> • $\text{sum_ctr}(\text{collection}, \geq, \text{LOW})$ • $\text{sum_ctr}(\text{collection}, \leq, \text{UP})$
Graph property(ies)	$\overline{\text{NARC}} = \text{VARIABLES} - \text{SEQ} + 1$

Graph model We use sum_ctr as an arc constraint. sum_ctr takes a collection of domain variables as its first argument.

Parts (A) and (B) of Figure 5.530 respectively show the initial and final graph associated with the **Example** slot. Since all arc constraints hold (i.e., because of the graph property $\overline{\text{NARC}} = |\text{VARIABLES}| - \text{SEQ} + 1$) the final graph corresponds to the initial graph.



(A)



(B)

Figure 5.530: Initial and final graph of the `sliding_sum` constraint

Signature

Since we use the PATH arc generator with an arity of SEQ on the items of the VARIABLES collection, the expression $|\text{VARIABLES}| - \text{SEQ} + 1$ corresponds to the maximum number of arcs of the final graph. Therefore we can rewrite the graph property $\overline{\text{NARC}} = |\text{VARIABLES}| - \text{SEQ} + 1$ to $\overline{\text{NARC}} \geq |\text{VARIABLES}| - \text{SEQ} + 1$ and simplify $\overline{\text{NARC}}$ to $\overline{\text{NARC}}$.

20000128

1535