5.190 increasing_sum

DESCRIPTION **LINKS**

Origin Conjoin `increasing` and `sum_ctr`.

Constraint `increasing_sum(VARIABLES, S)`

Synonyms `increasing_sum_ctr`, `increasing_sum_eq`.

Arguments

VARIABLES	:	collection(var−dvar)
S	:	dvar

Restrictions

required(VARIABLES, var)
increasing(VARIABLES)

Purpose The variables of the collection VARIABLES are increasing. In addition, S is the sum of the variables of the collection VARIABLES.

Example $(\langle 3, 3, 6, 8\rangle, 20)$

The `increasing_sum` constraint holds since:

- The values of the collection $\langle 3, 3, 6, 8\rangle$ are sorted in increasing order.
- $\mathtt{S} = 20$ is set to the sum $\langle 3 + 3 + 6 + 8\rangle$.

Typical

$|\mathtt{VARIABLES}| > 1$
$\mathtt{range}(\mathtt{VARIABLES.var}) > 1$

Arg. properties Functional dependency: S determined by VARIABLES.

Usage The `increasing_sum` constraint can be used for breaking some symmetries in bin packing problems. Given a set of n bins with the same maximum capacity, and a set of items each of them with a specific height, the problem is to pack all items in the bins. To break symmetry we order bins by increasing use. This is done by introducing a variable x_i $(0 \leq i < n)$ for each bin i giving its use, i.e., the sum of items heights assigned to bin i, and by posting the following `increasing_sum`$(\langle x_0, x_1, \ldots, x_{n-1}\rangle, s)$ where s denotes the sum of the heights of all the items to pack.

Algorithm A linear time filtering algorithm achieving bound-consistency for the `increasing_sum` constraint is described in [313]. This algorithm was motivated by the fact that achieving bound-consistency on the inequality constraints and on the sum constraint independently hinders propagation, as illustrated by the following small example, where the maximum value of x_1 is not reduced to 2: $x_1 \in [1, 3]$, $x_2 \in [2, 5]$, $s \in [5, 6]$, $x_1 < x_2$, $x_1 + x_2 = s$.

Given an `increasing_sum`$(\langle x_0, x_1, \ldots, x_{n-1}\rangle, s)$ constraint, the bound-consistency algorithm consists of three phases:

1. A normalisation phase adjusts the minimum and maximum value of variables $x_0, x_1, \ldots, x_{n-1}$ with respect to the chain of inequalities $x_0 \leq x_1 \leq \cdots \leq x_{n-1}$. A forward phase adjusts the minimum value of $x_1, x_2, \ldots, x_{n-1}$ (i.e., $\underline{x_{i+1}} \geq \underline{x_i}$), while a backward phase adjusts the maximum value of $x_{n-2}, x_{n-1}, \ldots, x_0$ (i.e., $\overline{x_{i-1}} \leq \overline{x_i}$).

2. A phase restricts the minimum and maximum value of the sum variable s with respect to the chain of inequalities $x_0 \leq x_1 \leq \cdots \leq x_{n-1}$ (i.e., $\underline{s} \geq \sum_{0 \leq i < n} \underline{x_i}$ and $\overline{s} \leq \sum_{0 \leq i < n} \overline{x_i}$).

3. A final phase reduces the minimum and maximum value of variables $x_0, x_1, \ldots, x_{n-1}$ both from the bounds of s and from the chain of inequalities. Without loss of generality we now focus on the pruning of the maximum value of variables $x_0, x_1, \ldots, x_{n-1}$. For this purpose we first need to introduce the notion of *last intersecting index of a variable* x_i, denoted by $last_i$. This corresponds to the greatest index in $[i+1, n-1]$ such that $\overline{x_i} > \underline{x_{last_i}}$, or i if no such integer exists. Then the increase of the minimum value of s when x_i is equal to $\overline{x_i}$ is equal to $\sum_{k \in [i, last_i]} (\overline{x_i} - \underline{x_k})$. When this increase exceeds the available margin, i.e. $\overline{s} - \sum_{0 \leq i < n} \underline{x_i}$, we update the maximum value of x_i.

We illustrate a part of the final phase on the following example `increasing_sum`$(\langle x_0, x_1, x_2, x_3, x_4, x_5 \rangle, s)$, where $x_0 \in [2, 6]$, $x_1 \in [4, 7]$, $x_2 \in [4, 7]$, $x_3 \in [5, 7]$, $x_4 \in [6, 9]$, $x_5 \in [7, 9]$ and $s \in [28, 29]$. Observe that the domains are consistent with the first two phases of the algorithm, since,

1. the minimum (and maximum) values of variables $x_0, x_1, x_2, x_3, x_4, x_5$ are increasing,

2. the sum of the minimum of the variables $x_0, x_1, x_2, x_3, x_4, x_5$, i.e., 28 is less than or equal to the maximum value of s,

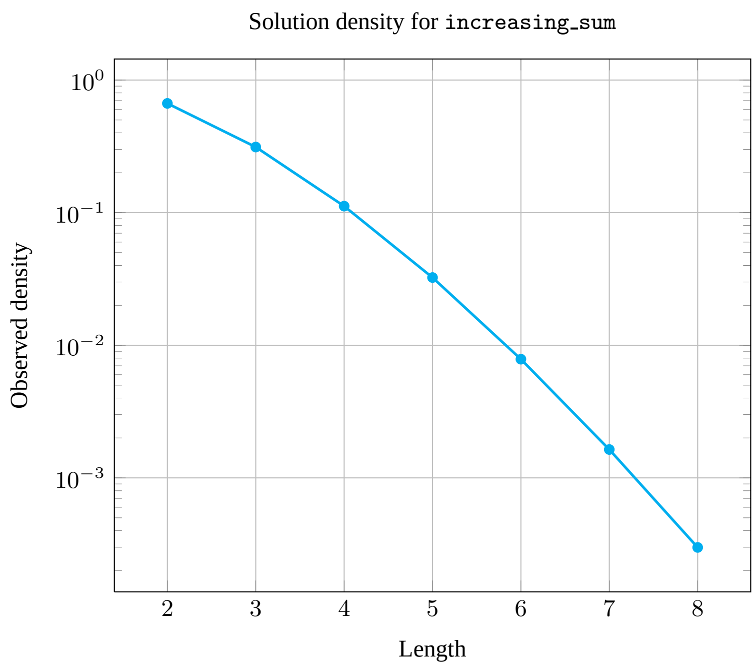
3. the sum of the maximum of the variables $x_0, x_1, x_2, x_3, x_4, x_5$, i.e., 45 is greater than or equal to the minimum value of s.

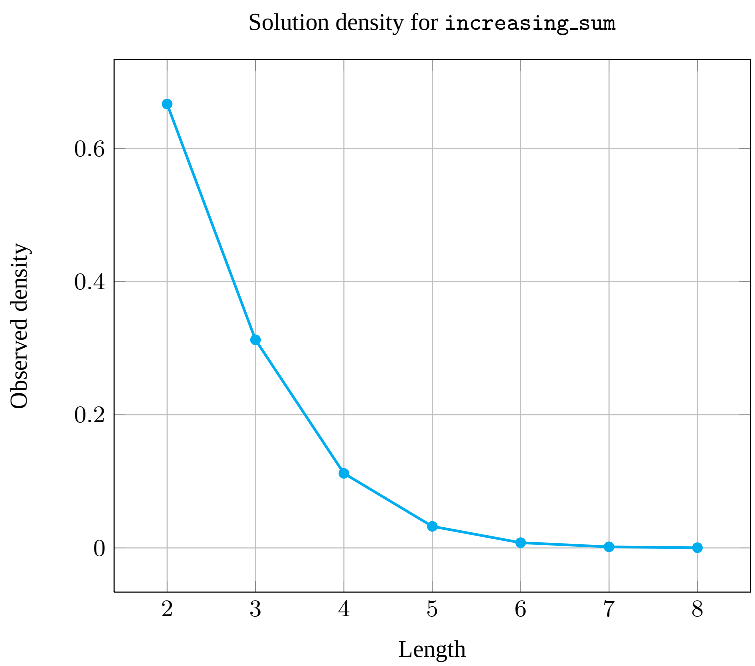
Now, assume we want to know the increase of the minimum value of s when x_0 is set to its maximum value 6. First we compute the last intersecting index of variable x_0. Since x_4 is the last variable for which the minimum value is less than or equal to maximum value of x_0 we have $last_0 = 4$. The increase is equal to $\sum_{k \in [0,4]} (\overline{x_0} - \underline{x_k}) = (6-2)+(6-4)+(6-4)+(6-5)+(6-6) = 9$. Since it exceeds the margin $29-(2+4+4+5+6+7) = 1$ we have to reduce the maximum value of x_0. How to do this incrementally is described in [313].

Counting

Length (n)	2	3	4	5	6	7	8
Solutions	6	20	70	252	924	3432	12870

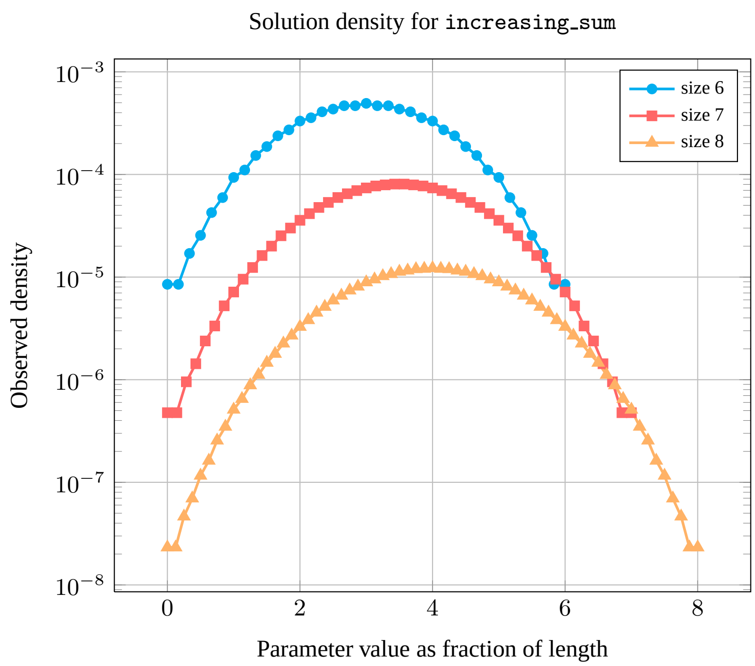
Number of solutions for `increasing_sum`: domains $0..n$

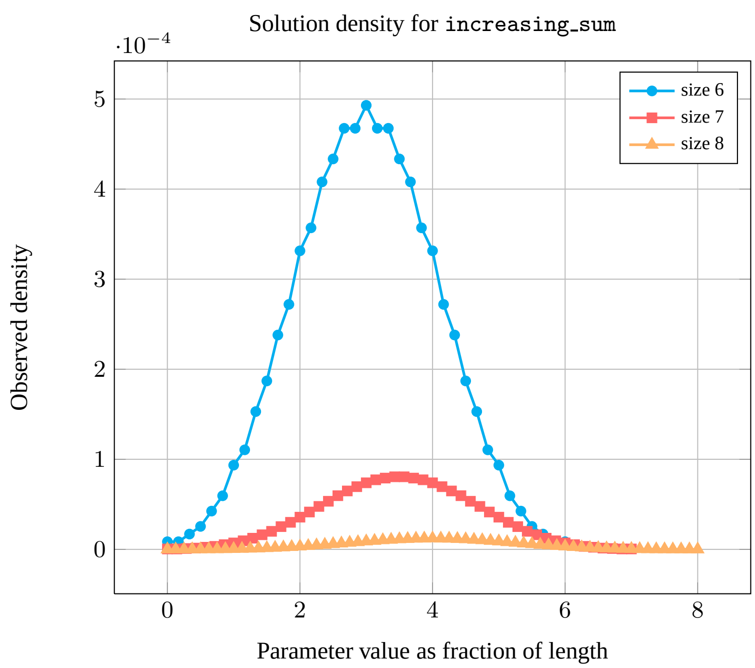

Solution density for increasing_sum
Observed density
10^{0}
10^{-1}
10^{-2}
10^{-3}
2
3
4
5
6
7
8
Length


Solution density for increasing_sum
Observed density
0.6
0.4
0.2
0
2
3
4
5
6
7
8
Length

Length (n)		2	3	4	5	6	7	8
Total		6	20	70	252	924	3432	12870
Parameter value	0	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2
	3	1	3	3	3	3	3	3
	4	1	3	5	5	5	5	5
	5	-	3	5	7	7	7	7
	6	-	3	7	9	11	11	11
	7	-	2	7	11	13	15	15
	8	-	1	8	14	18	20	22
	9	-	1	7	16	22	26	28
	10	-	-	7	18	28	34	38
	11	-	-	5	19	32	42	48
	12	-	-	5	20	39	53	63
	13	-	-	3	20	42	63	77
	14	-	-	2	19	48	75	97
	15	-	-	1	18	51	87	116
	16	-	-	1	16	55	100	141
	17	-	-	-	14	55	112	164
	18	-	-	-	11	58	125	194
	19	-	-	-	9	55	136	221
	20	-	-	-	7	55	146	255
	21	-	-	-	5	51	155	284
	22	-	-	-	3	48	162	319
	23	-	-	-	2	42	166	348
	24	-	-	-	1	39	169	383
	25	-	-	-	1	32	169	409
	26	-	-	-	-	28	166	440
	27	-	-	-	-	22	162	461
	28	-	-	-	-	18	155	486
	29	-	-	-	-	13	146	499
	30	-	-	-	-	11	136	515
	31	-	-	-	-	7	125	519
	32	-	-	-	-	5	112	526
	33	-	-	-	-	3	100	519
	34	-	-	-	-	2	87	515
	35	-	-	-	-	1	75	499
	36	-	-	-	-	1	63	486
	37	-	-	-	-	-	53	461
	38	-	-	-	-	-	42	440
	39	-	-	-	-	-	34	409
	40	-	-	-	-	-	26	383
	41	-	-	-	-	-	20	348
	42	-	-	-	-	-	15	319
	43	-	-	-	-	-	11	284
	44	-	-	-	-	-	7	255
	45	-	-	-	-	-	5	221
	46	-	-	-	-	-	3	194
	47	-	-	-	-	-	2	164
	48	-	-	-	-	-	1	141
	49	-	-	-	-	-	1	116
	50	-	-	-	-	-	-	97
	51	-	-	-	-	-	-	77
	52	-	-	-	-	-	-	63
	53	-	-	-	-	-	-	48
	54	-	-	-	-	-	-	38
	55	-	-	-	-	-	-	28
	56	-	-	-	-	-	-	22
	57	-	-	-	-	-	-	15
	58	-	-	-	-	-	-	11
	59	-	-	-	-	-	-	7
	60	-	-	-	-	-	-	5
	61	-	-	-	-	-	-	3
	62	-	-	-	-	-	-	2
	63	-	-	-	-	-	-	1
	64	-	-	-	-	-	-	1

Solution count for `increasing_sum`: domains $0..n$





See also **common keyword:** `sum_ctr` *(sum)*.

implies: `increasing`.

Keywords **characteristic of a constraint:** sum.

constraint type: predefined constraint, order constraint, arithmetic constraint.

filtering: bound-consistency.

modelling: functional dependency.

symmetry: symmetry.

Cond. implications

- $\texttt{increasing\_sum}(\texttt{VARIABLES}, \texttt{S})$
  with $\texttt{minval}(\texttt{VARIABLES.var}) > 0$
  implies $\texttt{atmost\_nvalue}(\texttt{S}, \texttt{VARIABLES})$.

- $\texttt{increasing\_sum}(\texttt{VARIABLES}, \texttt{S})$
  with $\texttt{minval}(\texttt{VARIABLES.var}) > 0$
  implies $\texttt{sum\_of\_increments}(\texttt{VARIABLES}, \texttt{LIMIT})$.