5.192 indexed_sum

DESCRIPTION LINKS GRAPH

Origin N. Beldiceanu

Constraint indexed_sum(ITEMS, TABLE)

Arguments

ITEMS : collection(index−dvar, weight−dvar)
TABLE : collection(index−int, summation−dvar)

Restrictions

|ITEMS| > 0
|TABLE| > 0
required(ITEMS, [index, weight])
ITEMS.index ≥ 1
ITEMS.index ≤ |TABLE|
required(TABLE, [index, summation])
TABLE.index ≥ 1
TABLE.index ≤ |TABLE|
increasing_seq(TABLE, index)

Purpose

Given several items of the collection ITEMS (each of them having a specific fixed index as well as a weight that may be negative or positive), and a table TABLE (each entry of TABLE corresponding to a summation variable), assign each item to an entry of TABLE so that the sum of the weights of the items assigned to that entry is equal to the corresponding summation variable.

Example

$$\left(\begin{array}{l}\left\langle\begin{array}{ll}\texttt{index}-3 & \texttt{weight}-{-4},\\ \texttt{index}-1 & \texttt{weight}-6,\\ \texttt{index}-3 & \texttt{weight}-1\end{array}\right\rangle,\\ \left\langle\begin{array}{ll}\texttt{index}-1 & \texttt{summation}-6,\\ \texttt{index}-2 & \texttt{summation}-0,\\ \texttt{index}-3 & \texttt{summation}-{-3}\end{array}\right\rangle\end{array}\right)$$

The indexed_sum constraint holds since the summation variables associated with each entry of TABLE are equal to the sum of the weights of the items assigned to the corresponding entry:

- TABLE[1].summation = ITEMS[2].weight = 6 (since TABLE[1].index = ITEMS[2].index = 1),
- TABLE[2].summation = 0 (since TABLE[2].index = 2 does not occur as a value of the index attribute of an item of ITEMS),
- TABLE[3].summation = ITEMS[1].weight + ITEMS[3].weight = −4 + 1 = −3 (since TABLE[3].index = ITEMS[1].index = ITEMS[3].index = 3).

Typical

|ITEMS| > 1
range(ITEMS.index) > 1
|TABLE| > 1
range(TABLE.summation) > 1

Symmetries

- Items of ITEMS are permutable.
- Items of TABLE are permutable.

Reformulation

The indexed_sum(ITEMS, TABLE) constraint can be expressed in term of a set of reified constraints and of |TABLE| arithmetic constraints (i.e., scalar_product constraints).

1. For each item ITEMS$[i]$ ($i \in [1, |\texttt{ITEMS}|]$) and for each table entry j ($j \in [1, |\texttt{TABLE}|]$) of TABLE we create a 0-1 variable B_{ij} that will be set to 1 if and only if ITEMS$[i]$.index is fixed to j (i.e., $B_{ij} \Leftrightarrow \texttt{ITEMS}[i].\texttt{index} = j$).
2. For each entry j of the table TABLE, we impose the sum $\texttt{ITEMS}[1].\texttt{weight} \cdot B_{1j} + \texttt{ITEMS}[2].\texttt{weight} \cdot B_{2j} + \cdots + \texttt{ITEMS}[|\texttt{ITEMS}|].\texttt{weight} \cdot B_{|\texttt{ITEMS}|j}$ to be equal to TABLE$[j]$.summation.

See also

implied by: elements_alldifferent.

specialisation: bin_packing *(negative contribution not allowed, effective use variable for each bin replaced by an overall fixed capacity)*, bin_packing_capa *(negative contribution not allowed, effective use variable for each bin replaced by a fixed capacity for each bin)*.

used in graph description: sum_ctr.

Keywords

application area: assignment.

modelling: variable indexing, variable subscript.

For all items of TABLE:

Arc input(s)	ITEMS TABLE
Arc generator	$PRODUCT \mapsto$ collection(items, table)
Arc arity	2
Arc constraint(s)	items.index = table.index
Sets	SUCC $\mapsto$ [source, variables − col(VARIABLES−collection(var−dvar), [item(var − ITEMS.weight)])]
Constraint(s) on sets	sum_ctr(variables, =, TABLE.summation)

Graph model We enforce the sum_ctr constraint on the weight of the items that are assigned to the same entry. Within the context of the **Example** slot, part (A) of Figure 5.427 shows the initial graphs associated with entries 1, 2 and 3 (i.e., one initial graph for each item of the TABLE collection). Part (B) of Figure 5.427 shows the corresponding final graphs associated with entries 1 and 3. Each source vertex of the final graph can be interpreted as an item assigned to a specific entry of TABLE.

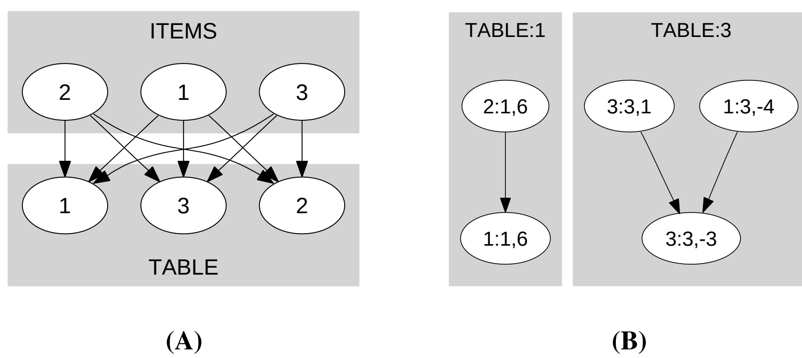

Figure 5.427: Initial and final graph of the indexed_sum constraint