5.122 disj

DESCRIPTION LINKS GRAPH

Origin [287]

Constraint disj(TASKS)

Argument

$$\texttt{TASKS} \ : \ \texttt{collection}\left(\begin{array}{l}\texttt{start}-\texttt{dvar},\\ \texttt{duration}-\texttt{dvar},\\ \texttt{before}-\texttt{svar},\\ \texttt{position}-\texttt{dvar}\end{array}\right)$$

Restrictions

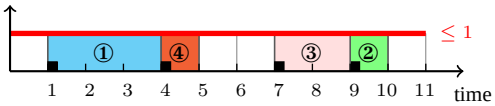
required(TASKS, [start, duration, before, position])
TASKS.duration ≥ 1
TASKS.position ≥ 0
TASKS.position $<$ |TASKS|

Purpose All the tasks of the collection TASKS should not overlap. For a given task t the attributes before and position respectively correspond to the set of tasks starting before task t (assuming that the first task is labelled by 1) and to the position of task t (assuming that the first task has position 0).

Example

$$\left(\left\langle\begin{array}{llll}\texttt{start}-1 & \texttt{duration}-3 & \texttt{before}-\emptyset & \texttt{position}-0,\\ \texttt{start}-9 & \texttt{duration}-1 & \texttt{before}-\{1,3,4\} & \texttt{position}-3,\\ \texttt{start}-7 & \texttt{duration}-2 & \texttt{before}-\{1,4\} & \texttt{position}-2,\\ \texttt{start}-4 & \texttt{duration}-1 & \texttt{before}-\{1\} & \texttt{position}-1\end{array}\right\rangle\right)$$

Figure 5.290 shows the tasks of the example. Since these tasks do not overlap the disj constraint holds.



TASKS

①	s − 1	d − 3	b − ∅	p − 0
②	s − 9	d − 1	b − {1, 3, 4}	p − 3
③	s − 7	d − 2	b − {1, 4}	p − 2
④	s − 4	d − 1	b − {1}	p − 1

(s for start, d for duration, b for before, p for position)

Figure 5.290: Tasks corresponding to the **Example** slot

Typical |TASKS| > 1

Symmetries

- One and the same constant can be added to the start attribute of all items of TASKS.
- TASKS.duration can be decreased to any value ≥ 1.

Usage The `disj` constraint was originally applied [287] to solve the *open-shop* problem.

Remark This constraint is similar to the `disjunctive` constraint. In addition to the `start` and the `duration` attributes of a task t, the `disj` constraint introduces a set variable `before` that represents the set of tasks that end before the start of task t as well as a domain variable `position` that gives the absolute order of task t in the resource. Since it assumes that the first task has position 0 we have that, for a given task t, the number of elements of its `before` attribute is equal to the value of its `position` attribute.

Algorithm The main idea of the algorithm is to apply in a systematic way shaving on the `position` attribute of a task. It is implemented in **Gecode** [374].

See also **common keyword:** `disjunctive` *(scheduling constraint)*.

used in graph description: `in_set`.

Keywords **complexity:** sequencing with release times and deadlines.

constraint arguments: constraint involving set variables.

constraint type: scheduling constraint, resource constraint, decomposition.

Arc input(s)	TASKS
Arc generator	$CLIQUE(\neq) \mapsto$ collection(tasks1, tasks2)
Arc arity	2
Arc constraint(s)	• $\bigvee \begin{pmatrix} \texttt{tasks1.start} + \texttt{tasks1.duration} \leq \texttt{tasks2.start}, \\ \texttt{tasks2.start} + \texttt{tasks2.duration} \leq \texttt{tasks1.start} \end{pmatrix}$ • tasks1.start + tasks1.duration ≤ tasks2.start ⇔ in_set(tasks1.key, tasks2.before) • tasks1.start + tasks1.duration ≤ tasks2.start ⇔ tasks1.position < tasks2.position
Graph property(ies)	NARC= |TASKS| ∗ |TASKS| − |TASKS|

Graph model

We generate a *clique* with a non-overlapping constraint between each pair of distinct tasks and state that the number of arcs of the final graph should be equal to the number of arcs of the initial graph. For two tasks t_1 and t_2, the three conditions of the arc constraint respectively correspond to:

- The fact that t_1 ends before the start of t_2 or that t_2 ends before the start of t_1.
- The equivalence between the fact that t_1 ends before the start of t_2 and the fact that the identifier of task t_1 belongs to the `before` attribute of task t_2.
- The equivalence between the fact that t_1 ends before the start of t_2 and the fact that the `position` attribute of task t_1 is strictly less than the `position` attribute of task t_2.

Parts (A) and (B) of Figure 5.291 respectively show the initial and final graph associated with the **Example** slot. The `disj` constraint holds since all the arcs of the initial graph belong to the final graph: all the non-overlapping constraints holds.

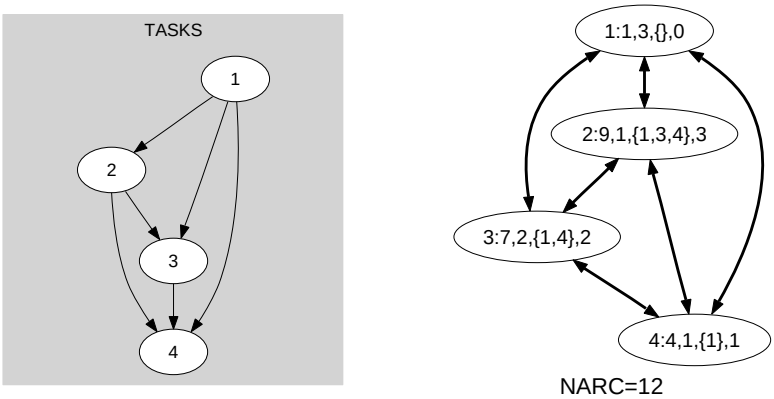


Figure 5.291: Initial and final graph of the `disj` constraint