RCPSP

The table below shows the total time needed for the scheduling of every task named as “**Obj value**” and the time required by the solver to find the optimal solution.

When an instance has “**max time**” as value for time, what we are implying is that the solver was going to exceed the time limit of **5 minutes**, has been stopped and in that time, found as the best solution **(total duration** of the execution **of all the tasks for a certain order** of execution), the one written under the corresponding “Obj. value” column.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Default search | | Earlier start time | |
| Obj value | Time | Obj value | Time |
| Data 1 | **90** | **210ms** | 90 | 222ms |
| Data 2 | **53** | **607ms** | 54 | max time |
| Data 3 | 81 | max time | **75** | **max time** |

First, we must consider that our “Obj value” is the real optimal value, only when the “Time” value is not “max time”.

For the first data the default search (minimizing the maximum of **start + duration** foreach task) and the **earlier start time search** perform the same way.

For the second one, the earliest start time search found a worst “Obj value” and it also required a huge amount of additional time by the solver, which doesn’t make it worth it at all.

We can see that the difference in terms of optimal “Objective value” is significative only for the third data, with both search heuristics taking the maximum time available for the solver.

So no, searching for the earliest start times isn’t always a good idea. This is probably because improving a solution means backtracking on the shallower levels of the search tree, because, as we know, rescheduling one of the latest scheduled tasks cannot improve much the “Obj value”.

JSP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Default search | | Earlier start time | |
| Obj value | Time | Obj value | Time |
| Data 1 | **663** | **430ms** | 669 | max time |
| Data 2 | **826** | **38.106s** | 921 | max time |

Reported above in the table shows the best makespan and the time needed by each instance of Job Shop Scheduling Problem, two set of data for each solving method: the default one and the earlier start time, as we did for the RCPSP, the results here show that the default search works better than the earlier start time. Given that the objective here is to produce the smallest makespan for the completion of all the tasks, this means that when we assign the smallest value in the domain to a variable early in the search, the solutions we find are heavily dependent on the firsts assigned values, so if the solver wants to improve the solution it has to backtrack to the first level of the tree. We can see that this behavior is wrong for our problem since this backtrack leads to great execution time, in some cases this time is so huge and the backtrack so expensive that the solver can’t even reach the solutions found by the default search in the 5 minutes limit we gave.