# RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM (RCPSP)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *rcpspData1* | | *rcpspData2* | | *rcpspData3* | |
|  | *Objective value* | *Time* | *Objective value* | *Time* | *Objective value* | *Time* |
| **Default search** | 90 | 394msec | 53 | 1s 544msec | 82 | 5m 0s |
| **Search on the smallest (earliest) start times** | 90 | 562msec | 54 | 5m 0s | 75 | 5m 0s |

Table above reports the objective value and the execution time of three instances for the Resource Constrained Project Scheduling Problem.

The resource constrained scheduling problem (RCPSP) was addressed by the following model in MiniZinc. The main objective is to minimize makespan by considering precedence constraints between tasks and limits on cumulative resources.

ANALYSIS AND CONCLUSIONS

We note from the analysis of the results shown in the table that some instances must be stopped within the time limit of **5 minutes**.

Thus, some target values are probably not the best the solver can find.

Although the results of the **default search** and the **smaller start time search** are very similar, we must consider that the instances that reach the time limit probably could have found a better makespan.

As in the **Job Scheduling exercise**, when the solver finds a solution and wants to improve it, it must return to the **root node** completely.

This is because the goal is to minimize the makespan and reassigning some last variables might only increase the makespan.

Thus, even if we guide the second type of search by saying to take the **smallest variable first**, this information cannot prevent the entire backtracking of the solver.

Note that the approach that provides a better objective value is considered a better approach. In this sense, in the third instance, the **early start time search** is better than the **default search**.

However, it is worse in the second instance (it ends up with a slightly worse solution). We cannot conclude that it is better than the **default search**, given the **JSP results** as well.

This approach is good for finding a solution, but not for proving optimality.

*In conclusion*, always looking for the *smallest start times is not always a good idea*. Although this approach improves the objective value in some instances, it may lead to slightly worse solutions in others. This variability in results suggests that searching based on the **smallest start times is not a universally optimal** **strategy** and may depend on the specific problem instance.

# JOB SHOP SCHEDULING PROBLEM (JSP)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Jobshop 1* | | *Jobshop 2* | |
|  | *Objective value* | *Time* | *Objective value* | *Time* |
| **Default search** | 663 | 415msec | 826 | 1m 25s |
| **Search on the smallest (earliest) start times** | 669 | 5m 0s | 921 | 5m 0s |

The table shows the results of the Job Shop problem, specifically the objective value and time calculated with two different data files. The goal is to find an optimal or near-optimal solution by minimizing the makespan.

From the results, it is easily observed that the results obtained indicate that the default search produced slightly better makespan solutions than the search on the smallest (earliest) start times for both instances. However, the search on the smallest (earliest) start times reached the time limit of 5 minutes in both instances.

In conclusion, it is evident from the table that the default search produces a more optimal makespan than the other search.

One thing that influences the execution of the search on the smallest (earliest) start times could be the weakness in the propagation of constraints: this makes the capabilities of this method poor compared to the default search.

*What do you observe? Is looking for the smallest (earliest) start times always a good idea? Justify your answer.*

What we want in the Job Shop Scheduling Problem (JSP) is to minimise the makespan and speed up the resolution of the problem, and this can be done through constraint propagation. We have seen, however, that search on the smallest (earliest) start times, on the contrary, does not help in this and has weak propagation and, for Jobshop 2, does not help in minimising the makespan much either.

In conclusion, it seems that search on the smallest (earliest) start times may not be the optimal choice for this specific problem and the default search is a better choice, seeing the performance of each.

You are doing well and is almost there. Can you explain better:

- what do you mean by "weak propagation" in searching on EST?

Weak propagation' refers to the limited ability of the search method based on earliest start times (EST) to fully propagate and exploit constraint information to improve search. In the context of the Job Shop Scheduling (JSP) problem, constraint propagation is very important to reduce the space of possible solutions. However, EST search may not be effective enough in exploiting constraints and, as a result, fails to narrow the search space efficiently.

- why does searching on EST take longer to find better solutions after having backtracked to the root node?

Searching EST may take longer to find better solutions after backtracking to the root node due to its sequential nature and lack of global information. When the search process backtracks to the root node, it may lose the information acquired during the previous stages of the search, making it necessary to repeat some of the work already done.

Furthermore, the EST search may not optimally utilise the global information obtained from different subtrees, thus preventing more efficient guidance of the search towards promising solutions.

PS: Searching on EST in JSP instance 1 should have returned the makespan 669, not 660 (otherwise it would be better than the default search but you are saying in the text that it is always worse than the default search).

After running the programme again without modifying anything, we realised that we had simply misspelled the value in the table. The result our programme returns is in fact the correct one, namely 669.

I couldn't really find the answers to my questions. Please study better this topic and try to find the right answers.

NUOVA RISPOSTA:

***“- what do you mean by "weak propagation" in searching on EST?”***

The concept of 'weak propagation' in the context of EST (Earliest Start Time) search refers to the inefficiencyof constraint propagation during the exploration of the solution space. When employing the Earliest Start Time search method (EST), constraint propagation is weak, particularly upon returning to the root node. This occurs because, during backtracking, new assignments to variables are made, but the propagation of constraints associated with these new assignments is ineffective in creating new branches leading to improved solutions.

This weakness in constraint propagation results in non-optimal branching decisions, hindering the algorithm's ability to efficiently explore and identify optimal solutions.

In our context, "weak propagation" in searching on EST refers to the ineffective transmission of information about the earliest start times of activities. Weak propagation occurs when constraints and restrictions on the start times of activities are not robustly communicated from one activity to another during the problem-solving process.

***“- why does searching on EST take longer to find better solutions after having backtracked to the root node?”***

The EST-based search requires more time to find improved solutions after backtracking to the root node due to the intrinsic inefficiency of the smallest search method. Delayed scheduling decisions based on the smallest EST lead to non-optimal branches, making it challenging for the algorithm to navigate efficiently through the solution space (poor exploration of the search subtree). Consequently, the EST-based search takes more time to converge towards better solutions, especially when compared to the default search method.

In our case, after backtracking, when EST search attempts to improve the solution, it needs to reset variables to their original state and then propagate constraints again. However, this propagation can be weak, especially when attempting to enforce constraints like "S1 ≠ v" (where v is the smallest value in the domain of Si). This can lead to suboptimal subtrees that require further exploration.

**What exactly do you mean by "Delayed scheduling decisions based on the smallest EST lead to non-optimal branches"? The description I am after is here (first, do you understand why the solver backtracks here to the root node, while in a regular branch & bound, backtracking would be to the last decisions in chronological order, as I showed via the graph colouring example?)**

RISPOSTA – DAL REPORT DI ANNA:

The solver returns to the root node to ensure improvement in the result by adding a constraint that limits the size of the new makespan and allows for the search to be carried out again. We know that the objective value instructs us to finish as early as possible. Therefore, if each variable has already been assigned the minimum value, modifying one or more last variables could only increase the result. The only way to improve the result is to return to the root of the search tree, add a new constraint limiting the size of the new makespan, and then proceed with the search once again.