CP Fourth Assignment

Students: Massimo Rondelli e Gianmarco Gabrielli

RCPSP Problem

Looking at the results obtained from the computation, we can draw some conclusions. Starting in order from the data files we can observe that with "Data 1" both searches, "Default Search" and "Indomain Min" found the same makespan of 90. Both take about the same amount of time, differing by about 20 milliseconds. With "Data 2," however, we can see a big difference between the two search methods With "Default search" we get a makespan of 53 in 362 milliseconds. As for "Indomain Min," on the other hand, we get a makespan of 54 however we go into timeout, with a timelimit of 5 minutes Which means that in 5 minutes it cannot find a better makespan than the "Default search" finds in 362 milliseconds. With "Data 3" we see that both solutions fail to find the optimal solution in the 5-minute time limit However, the "Indomain Min" search finds the makespan better than 75, compared to 81 of the "Default Search."

	Default Search		Indomain Min	
	Makespan	**Time**	**Makespan**	**Time**
Data 1	**90**	**80ms**	90	106ms
Data 2	**53**	**362ms**	54	-
Data 3	81	-	**75**	**-**

Table 1: RCPSP Problem statistics

The results indicate that the default search appears to function quite effectively. We have identified the global optimum in "Data 1" and "Data 2". The smallest start time search is a basic greedy search that, as "Data 3" shows, produces good but less than ideal results in complex data that are also unsolvable with the default search. ”Data 2” demonstrates that, although the default search makes finding a solution simple, the greedy search make things more difficult. This is because it may take us in the wrong direction within the search tree, and if the search space is complex, it would take too long to find a global optimum. However, since the makespan is better, it might be useful to use the greedy search to find a suboptimal when the data is complex, as shown in ”Data 3”, since the default search does not produce an optimum or better solution than the greedy one.

JSP Problem

By looking at the data that showed up from the JSP mdoel, we can make similar assumptions to those we covered in the previous model. Default search is the optimal choice. We can only find the optimal result using the default search method. Using the smallest search method, we get worse results. It’s not possible to find optimal solutions within 5 minutes. Looking at the problem we can say that EST is good for an initial solution. It give us a greedy approach in a real small amount of time. Of course that cannot be the best and optimal solution because we need to backtrack but before do io we must apply a bounding constraint that give us some more propagation. The bounding constraint is a particular constraint that we add in order to find only better solution that we can find respect the best possible solution we found until that. Often the problem depends from some decision mistake at the first decisions, so avoid to be stuck for to much in a subtree we have to choose some heuristics different from che first fail, like the postpone search strategy. In the postpone we can often find a better solution due to the nature that give them effective branching choices. Postpone to be effective needs an initial solution that is the reason that we combine EST with postpone in order to have an initial solution in a quick way and postpone to find the best optimal solution.

	Default Search		Smallest	
	Makespan	**Time**	**Makespan**	**Time**
Data 1	**663**	80ms	669	-
Data 2	**826**	27s 411ms	921	-

Table 2: JSP Problem statistics