Alfonso Esposito 1057171
Isabella Marasco 1040993

Decision Making with Constraint Programming: Exercises 3

To Do - 1.

Objective: Using the alldifferent model (without symmetry breaking), search for a solution for N = 30, 35, 45, 50, using the following 6 variable - value ordering heuristics of Gecode.
Comments: For this exercise we must observe how the choice of a heuristic can improve the performance of a solver. As we can see in the results, to search a solution, a good heuristic can decrease the number of total failures. Despite the good results, in some case the heuristics can't give the best performance for every input N. In fact, we can see that with Input order we have worst results than domWdeg, where we obtain 0 failures with input N = 35 against 2,828,740 failures in input order.
Analyzing every heuristic, we can also see that adding randomized parameter in search we can improve our results with less failures. That's because given the same random seed, the solver will never explore the same subproblems, even if we explore the same subtree. Finally, we can also see that the improvement is almost true in every heuristic, excepts for the input order that give us, with min value and random value, same results. In this case, the solver must choose from the domain following the input order, and for this reason it can make mistake and explore the whole sub-tree before backtracking from the infeasible problem that causes more time and more failures.

N	**Input order – min value**		**Input order - random value**	
	Fails	**Time**	**Fails**	**Time**
30	**1.588.827**	41s 806msec	**1.588.827**	29s 300msec
35	**2.828.740**	1m 29s	**2.828.740**	1m 1s
45	-	4m Limit time	-	4m Limit time
50	-	4m Limit time	-	4m Limit time

N	**Smallest domain - min value**		**Smallest domain - random value**	
	Fails	**Time**	**Fails**	**Time**
30	**15**	184msec	**1**	172msec
35	**21**	173msec	**0**	178msec
45	**6**	184msec	**1**	183msec
50	**123**	192msec	**10**	178msec

N	**DomWdeg - min value**		**DomWdeg - random value**	
	Fails	**Time**	**Fails**	**Time**
30	**15**	167msec	**1**	177msec
35	**21**	180msec	**0**	172msec
45	**6**	178msec	**1**	182msec
50	**123**	184msec	**10**	179msec

To Do - 2.

Objective: Implement model using a solver search on Xi and Yi for all I in [1..n] and repeat everthing after ordering the rectangles in decreasing order by their perimeter.

Comment:

The first thing we observed, using the heuristic, is a better performance for both failures and time in the case of the 19x19 poster but, this doesn't happen to 20x20 poster. This happens because heuristics can, sometime, improve the performance of the model.

Then, with the ordered rectangles, we obtained better results. That's because a bigger rectangular is more difficult to place, as if it is more constrained compared to a smaller one. Since the decision variables are ordered by their decreasing perimeter, the solver is forced to search solutions like in a deg heuristic, giving us better performance.

	Default Order		**Decreasing order**	
	Fails	**Time**	**Fails**	**Time**
19 x 19	2.037.842.	8s 315msec	30	257 msec
20 x 20	-	> 5 LIMIT TIME	323	260 msec

To Do - 3.

Objective: Implement model using all different constraints and search for a solution to the instances given in data files using Gecode. The experiment was done with default search, domWdeg -random value and domWdeg -random Value + restarting

Comment: In this exercise we tried to solve the problem with different heuristics.

As first we can say from the results that the qc30-08 with default search has obtained better result than the domWdeg version, even with restarting luby. For all the other the default search had worst results. With these results we note that the performance of a search could depend, not only from the model, but also from the dataset used. For some of them, heuristics can make more mistakes that make complexity of execution higher than the other and this could depend only on the dataset used, as seen for in heavy tail behaviour.

Qc30-03	**Default search**		**domWdeg - random value**		**domWdeg – random value + restarting luby**	
	Fails	**Time**	**Fails**	**Time**	**Fails**	**Time**
	-	> 5 LIMIT TIME	1.061.184	2m 23s	642.427	1m 34s

Qc30-05	**Default search**		**domWdeg - random value**		**domWdeg – random value + restarting luby**	
	Fails	**Time**	**Fails**	**Time**	**Fails**	**Time**
	657.955	1m 32s	5.885	1s 306msec	303.205	44s 990msec

Qc30-08	**Default search**		**domWdeg - random value**		**domWdeg – random value + restarting luby**	
	Fails	**Time**	**Fails**	**Time**	**Fails**	**Time**
	627	436msec	6.403	1s 315msec	11990	2s 90msec.

Qc30-12	Default search		domWdeg - random value		domWdeg – random value + restarting luby	
	Fails	**Time**	**Fails**	**Time**	**Fails**	**Time**
	259.082	33s 201msec	53.200	7s 500msec	21.986	3s 486msec

Qc30-19	Default search		domWdeg - random value		domWdeg – random value + restarting luby	
	Fails	**Time**	**Fails**	**Time**	**Fails**	**Time**
	381.330	56s 845msec	-	>5 LIMIT TIME	48.244	7s 519msec