Alfonso Esposito 1057171
Isabella Marasco 1040993

# Decision Making with Constraint Programming: Exercises 1

## To Do - 1.

***Objective:*** Resolve the N-Queens problem and observe how the variation of the model changes the search for solutions.

***Comments:*** We wrote 5 different models for the problem. Given the same input each model returns the same number of solutions, but with a different number of failures.

The first model obtains the worst results because it has simpler constraints that impose only different values on columns/rows and diagonal.

The models rc1, rc2 and rc2 obtain the same number of solution but with fewer failures than the simplest model r. The rc1 model gets the best results among the three models, it is made by two models: one that use decision variables for rows and another with variables for columns, both search in the same space but in different ways and with the channeling to maintain consistency between the variables of the two models.

The rc2 model removes some constraint, but this change does not affect the number of solutions found, since the removed constraints are already implied by the other. The search is influenced by the removal of those implied constraint and in this model, we have a higher number of failures compared to rc1, from which rc2 is derived.

The rc3 model reduces the number of constraints even more, leaving only one constraint on the rows. This makes the results of the model very similar to the model r, but the use of the channeling constraint allows to obtain a slightly better result.

The Alldifferent model use global constraints that can find solutions with fewer attempts and faster. In fact, the global constraints capture complex, non-binary, and recurring combinatorial substructures.

The first four models obtain the same number of solutions, although with different time, but, since symmetry was not broken, we have a lot of solutions that are symmetrical to each other.

The fifth model, that use the lexicographical ordering constraint, obtains lower number of failures. It also obtains fewer solutions, compared to the other, because it can break symmetries by imposing an ordering to avoid permutations.

| **N** | **#sols** | **r** | **rc1** | **rc2** | **rc3** | **alldiff** |
|---|---|---|---|---|---|---|
| **8** | 92 | 891 | 500 | 593 | 864 | 254 |
| **9** | 352 | 4262 | 2656 | 2772 | 4603 | 849 |
| **10** | 724 | 23291 | 13996 | 13593 | 23195 | 3722 |
| **12** | 14200 | 773550 | 355041 | 380595 | 820127 | 75823 |

.

| **N** | **#sols** | **alldiffsym** |
|---|---|---|
| **8** | 12 | 78 |
| **9** | 46 | 296 |
| **10** | 92 | 965 |
| **12** | 1787 | 16343 |

## To Do - 2.

***Objective:*** Resolve a puzzle problem and observe how the variation of the model changes the search for solutions.

***Comment:*** We wrote 2 different models to solve the puzzle. The use of implied constraints reduced the total time for solution and lowered the number of failures. This happens because implied constraints introduce constraints that cannot be deduced by the solver, and this reduce the search space making the solver faster.

Even with the use of implicit constraints, the total time with n=1000 is high, this difference is more relevant in this case, where, as seen in statistic note, the initTime is 26.4364 while the solveTime is 2.23487, so most of the time it is busy doing the compilation.

| **N** | **Base T** | **Base F** | **Implied T** | **Implied F** |
|---|---|---|---|---|
| **500** | 37s 390msec | 617 | 25s 158msec | 495 |
| **1000** | 3m 50s | 1247 | 1m 51s | 995 |