Martina Daghia 0001097932 - **Martina Zauli** 0001097933

OPTIMAL N-QUEENS

The objective of this problem is to find the optimal solution to the **N-Queens problem** using the Gecode solver with the MiniZinc modeling language.

The solver configuration was adapted to obtain solution statistics and set a *time limit of 300 seconds* (5 minutes).

Four experiments were conducted to find the optimal solution to the **50-queens problem** using **different search strategies** with the Gecode solver:

1. Default search
2. DomWdeg-random heuristics
3. Search with restart (Luby strategy with L = 250) added to previous heuristic
4. Large Neighborhood Search (LNS) with 85% variable fixation added to previous model.

RESULTS AND ANALYSIS

search	f	obj
def	8,263,989	796
dWd-rand	10,332,081	790
dWd-rand + restart	7,260,121	676
dWd-rand + restart + lns	**1,802,557**	**650**

The results of each experiment were written in the table above with the number of **failures** (f) and **objective value** (obj) achieved within the time limit.

We start with the **default search** and notice that the performance gradually improves.

The **domWdeg heuristic** performs better than the default search because it considers the queens that are most likely to **fail first**. The solver then strives to place the most difficult queens first.

Implementing **restarts** in the solver leads to an overall improvement, probably due to better exploitation of prior information, such as considering a different order of variables.

The use of **Large Neighborhood Search** (LNS) results in the best objective value. The percentage of **fixed variables** is quite high, which may be why we have fewer failures. The few remaining variables are much easier to explore, as is the propagation of constraints.

<u>note well:</u> fixed variables **reduce domain** size and **propagation works better** when domains are small.

The **domWdeg+rand search** offers a better objective value than the default search (i.e., it finds about the same number of solutions as the default search) but more **failures** occur.

Thus, although the **objective value improves**, it is plausible that the different **search space** is responsible for more failures.

The **domWdeg** allows us to **fail as early as possible without getting stuck in some subtree**, and thus can explore the search tree much more intensively. If we force the time with an equal limit, the domWdeg search will fail more times because it may have explored more branches than the default search.