**Optimal N-Queens**

|  |  |  |
| --- | --- | --- |
| **Search** | **Number of Failures** | **Objective Value** |
| **def** | 5,735,329 | 796 |
| **dWd-rand** | 11,617,849 | 790 |
| **dWd-rand + restart** | 5,949,412 | 676 |
| **dWd-rand + restart + LNS** | 1,853,636 | 650 |

The table above shows the number of failures and the corresponding objective values related to the type of search performed.

We started with a "default" search and gradually added elements to improve the objective value.

Let's go through each step specifically:

1. **def -> dWd-rand:** the dWd strategy manages to achieve a better result because it identifies in advance the queens that are more likely to fail and places them first. It is worth noting that in the second case, we have a higher number of failures. This is justified by the fact that dWd-rand allows failures to occur as early as possible without necessarily reaching blind spots in the tree. This way, it can explore the tree more rapidly, evaluating more branches and therefore experiencing more failures. The reason for this increase is that by imposing a maximum time limit, the second strategy evaluates more branches of the tree within that time, finding more failures.
2. **dWd**-**rand -> dwd-rand+restart:** Conducting the search by restarting the solver leads to an improvement due to the retention of some information obtained during the previous executions, then executing with a different order of variables.
3. **dwd**-**rand+restart -> dWd-rand + restart + LNS:** in this case we can observe a clear improvement given by the high percentage of fixed variables. This makes the remaining variables simple to explore, reducing the number of failures and simplifying constraint propagation.

In conclusion, as we could expect, by using the random heuristic with restart and applying the Large Neighbourhood Search (fixing 85% of the variables) we obtain a better result both in terms of optimality and a lower number of failures.