**Introduction**

Let’s start describing the parameters for the heuristics, we have two categories of parameters:

1. The variable-value ordering heuristics, determining the way the variables are chosen during the search, we have:
   1. Input Order: The variables are chosen in order of input.
   2. First Fail: The variables are chosen based on the size of the domain, smallest size first.
   3. Weighted Degree: The variables are chosen based on quota of the of the domain size over weighted degree, implying that the variables with small domain and large weighted degree are chosen first.
2. Constraints on the variables:
   1. Minimum Value: Constraining the variables to take the smallest value in the domain.
   2. Random Value: Constraining the variables to take a random value from the domain.

**N-Queens**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **input\_order** | | **first\_fail** | | **dom\_w\_deg** | |
|  | **indomain\_min** | **indomain\_random** | **indomain\_min** | **indomain\_random** | **indomain\_min** | **indomain\_random** |
| **30x30** | 1,588,827 | 9 | 15 | **1** | 15 | **1** |
| **35x35** | 2,828,740 | 10 | 21 | **0** | 21 | **0** |
| **45x45** | - | 6 | 6 | **1** | 6 | **1** |
| **50x50** | - | 42 | 123 | **10** | 123 | **10** |

As we can see, **input\_order+indomain\_min** heuristics give the worst results, this is probably due to the fact that using the lowest value in the domain for the variables selected in the order of the input gives the solver no choice in the selection of the better variables to assign first, it also gives no choice in the values assigned to them, this implies that if, let’s suppose, the first choice is wrong and leads to no solutions, the solver is going to spend a lot of time and fail many times before exiting the subtree of the first decision. That’s probably the reason of the 45x45 and 50x50 instances not terminating at all (in under 5 minutes).

This is also magnified through the results of **input\_order+indomain\_random**: as we can see the failures are much less, this is because picking values in random order from the input let the solver explore different subtrees, increasing our chances of finding a solution. This works for every one of these heuristics.

When we try the **first\_fail** heuristic we essentially try to fit the variables with less possibilities of assignable values, thus eliminating from the search tree huge subtrees without even exploring them through propagation. The fails happen earlier in the search compared to the input\_order mode.

We gain the same numbers from the searches using **dom\_w\_deg** rule, this is because given that all the variables have the same number of constraints and the same weight associated to them making dom\_w\_deg and first\_fail the same.

**Poster**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***input\_order*** | | ***first\_fail*** | | ***dom\_w\_deg*** | |
|  | ***indomain\_min*** | ***indomain\_random*** | ***indomain\_min*** | ***indomain\_random*** | ***indomain\_min*** | ***indomain\_random*** |
| **19x19** | 1,362,457 fails  6.369s | - | **239,954 fails**  **1.385s** | 2,929,153 fails  13.360s | **236,024 fails**  **1.257s** | 2,929,030 fails  13.382s |
| **20x20** | - | - | **1,873 fails**  **120ms** | 5,797,312 fails  25.527s | **1,873 fails**  **129ms** | 5,797,456 fails  25.716s |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***input\_order*** | | ***first\_fail*** | | ***dom\_w\_deg*** | |
|  | ***indomain\_min*** | ***indomain\_random*** | ***indomain\_min*** | ***indomain\_random*** | ***indomain\_min*** | ***indomain\_random*** |
| **19x19** | 30 fails  112ms | - | **252,210 fails**  **1.326s** | 3,637,566 fails  16.626s | **245,441 fails**  **1.313s** | 3,457,753 fails  15.849s |
| **20x20** | 323 fails  116ms | - | **1,737 fails**  **120ms** | 4,402,830 fails  19.69s | **1,737 fails**  **121ms** | 4,402,770 fails  19.197s |

**Ordered Poster**

The first thing we can discuss is the difference between this problem and N-Queens, where the latter gains performances (smaller number of failures) from the random selection of values, while in this problem, trying random positions for the placement of the Posters is intuitively useless and counterproductive.

We can also observe that the results of **first\_fail** and **dom\_w\_deg** are close, even between the ordered and the unordered versions of the data. This is because only the **input\_order** heuristic is dependant on the order of the data, while of course the domain of a variable and is weighted degree aren’t related to the order of the data. **The order** given to the data is **decreasing on the area of the rectangles**, making the input\_order heuristic assign a value to the biggest posters first, which means selecting the hardest to place first. Given that **input\_order** can be classified as a **static VOH** and that in the instance with the data ordered it performs way better than the other heuristics, this is a practical example that dynamic heuristics aren’t always the best way to go.

**QuasiGroup Completation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ***default\_search*** | | ***dom\_w\_deg + random*** | | ***dom\_w\_deg + random + restart*** | |
| ***failures*** | ***time*** | ***failures*** | ***time*** | ***failures*** | ***time*** |
| **30-03** | - | | 1,061,184 | 1.20s | 642,427 | 184.87s |
| **30-05** | 657,955 | 54.687s | 5,885 | 690ms | 303,205 | 83.83s |
| **30-08** | 627 | 186ms | 6,403 | 738ms | 11,990 | 4.295s |
| **30-12** | 259,082 | 19.690s | 53,200 | 4.552s | 21,986 | 6.528s |
| **30-19** | 381,330 | 33.58s | **-** | | 48,244 | 14.61s |

We can’t really find a pattern here: the best result is the instance **[default\_search, 30-19]**, then we can notice that the worst results in terms of fail is **[dom\_w\_deg+random, 30-03]**, and the two are not noticeably related.

We have two instances not terminating without any noticeable (in under 5 minutes).

Also, we notice that for some instances of the problem, **the default search** performs better than the **dom\_w\_deg+random** and **dom\_w\_w\_deg+random+restart** heuristics.

We can notice that the **dom\_w\_deg+random** works very well, in terms of time and failures, **compared to the restarting model** in the second and third instances: There’s a huge difference of performances suggesting that restarting the searches have worsened the results maybe because of the randomization component of the searches that may have led to subsequent wrong choices and the restarting of a good guess.