Report on

**“Offline 2: Solving Latin Square”**

**Course Code:** CSE 318

**Course Title:** Artificial Intelligence Sessional

**Submitted By:**

Md. Azizur Rahman Anik Student ID: 1805115

**Date of Submission:**

05-01-2023

**Value Order Heuristic: Least-Constraining-Value**

After a variable has been selected using a Variable-Order-Heuristic, to decide on the order in which to examine its values, **“Least-Constraining-Value”** heuristic has been chosen. It prefers the value that rules out the fewest choices for the neighboring variables (**in this problem, the neighboring variables are the ones that are either in same row or same column**) in the constraint graph. As it tries to leave the **maximum flexibility for subsequent variable assignments**, it is more likely to go through the path which will provide a solution faster. As we need only one solution in this assignment, this heuristic provides better performance than choosing the values from the domain of a variable randomly or serially. If we need to enumerate all solutions rather than just fine one, then value ordering would be irrelevant.

**Data Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem** | **Solver** | **VAH** | **#Node** | **#BT** | **Time(ms)** |
| d-10-01 | BT | VAH1 | 58 | 0 | 0 |
| BT | VAH2 | 416178375 | 416178317 | 962465 =16.01 mins |
| BT | VAH3 | 62 | 4 | 16 |
| BT | VAH4 | 64 | 6 | 0 |
| BT | VAH5 | 449500996 | 449500938 | 531218 |
| FC | VAH1 | 58 | 0 | 0 |
| FC | VAH2 | 6947264 | 6947206 | 37371 |
| FC | VAH3 | 61 | 3 | 0 |
| FC | VAH4 | 62 | 4 | 16 |
| FC | VAH5 | 574134 | 574076 | 2172 |
| d-10-06 | BT | VAH1 | 245 | 187 | 15 |
| BT | VAH2 | 240377219 | 240377161 | 484354 = 8.07 mins |
| BT | VAH3 | 58 | 0 | 0 |
| BT | VAH4 | 59 | 1 | 0 |
| BT | VAH5 | 1374653 | 1374595 | 1460 |
| FC | VAH1 | 227 | 169 | 0 |
| FC | VAH2 | 5266806 | 5266748 | 23158 |
| **FC** | **VAH3** | **58** | **0** | **0** |
| FC | VAH4 | 58 | 0 | 3 |
| FC | VAH5 | 1401353 | 1401295 | 3390 |
| d-10-07 | **BT** | **VAH1** | **58** | **0** | **0** |
| BT | VAH2 | 17948366 | 17948308 | 37233 |
| BT | VAH3 | 104 | 46 | 16 |
| BT | VAH4 | 59 | 1 | 0 |
| BT | VAH5 | 1093722653 | 1093722595 | 1223448 = 20.4 mins |
| **FC** | **VAH1** | **58** | **0** | **0** |
| FC | VAH2 | 336402 | 336344 | 1297 |
| FC | VAH3 | 99 | 41 | 16 |
| FC | VAH4 | 58 | 0 | 3 |
| FC | VAH5 | 483171 | 483113 | 1140 |
| d-10-08 | **BT** | **VAH1** | **73** | **15** | **0** |
| BT | VAH2 | \* | \* | \* |
| BT | VAH3 | 388 | 330 | 0 |
| BT | VAH4 | 795 | 737 | 15 |
| BT | VAH5 | 167833 | 167775 | 161 |
| **FC** | **VAH1** | **72** | **14** | **0** |
| FC | VAH2 | 75343876 | 75343818 | 306366 |
| FC | VAH3 | 361 | 303 | 0 |
| FC | VAH4 | 706 | 648 | 19 |
| FC | VAH5 | 132618 | 132560 | 297 |
| d-10-09 | BT | VAH1 | 3275 | 3217 | 16 |
| BT | VAH2 | 896016 | 895958 | 3515 |
| BT | VAH3 | 70 | 12 | 0 |
| **BT** | **VAH4** | **67** | **9** | **0** |
| BT | VAH5 | 84798141 | 84798083 | 94748 |
| FC | VAH1 | 2785 | 2727 | 16 |
| FC | VAH2 | 34063 | 34005 | 125 |
| FC | VAH3 | 67 | 9 | 16 |
| **FC** | **VAH4** | **63** | **5** | **2** |
| FC | VAH5 | 89324717 | 89324659 | 240271 |
| d-15-01 | **BT** | **VAH1** | **136196** | **136089** | **281** |
| BT | VAH2 | \* | \* | \* |
| BT | VAH3 | 710729 | 710622 | 2093 |
| BT | VAH4 | 1672762 | 1672655 | 6987 |
| BT | VAH5 | \* | \* | \* |
| **FC** | **VAH1** | **123922** | **123815** | **297** |
| FC | VAH2 | \* | \* | \* |
| FC | VAH3 | 644132 | 644025 | 1999 |
| FC | VAH4 | 1480838 | 1480731 | 9340 |
| FC | VAH5 | \* | \* | \* |

**Analysis:**

To apply the heuristics given, **a modified Backtrack** is used where the modification is: When a value of a variable is successfully assigned, then the domain of its neighboring variables have been updated.

**The difference between the Forward Checking and the modified Backtrack** is: in forward checking, after updating the domain of neighboring variables, if any domain becomes empty, then immediately backtracked from that node, whereas no such decision is made based on the size of the domain in the modified backtrack.

**Reason to use a Modified Backtrack:** If a pure backtrack solver is used, then the heuristics will not get the updated domain. Therefore it will work on the initial domain every time and this will be able to provide the power of the heuristics.

**Observations:** From the data table, it is clear that the Forward checking scheme performs better than Backtrack for all the test cases as it detects failure earlier and reduces the number of nodes in the search tree by pruning larger parts of the tree earlier. For example, for test case “d-10-09”, the BT with heuristic VAH2 searches total 896016 nodes where the FC searches only 34063 nodes (A factor of 26 improvement).

**Performance Comparison among the 5 Variable-Order-Heuristics:** The “Minimum Remaining Values (MRV)”, that is, VAH1 heuristic seems to be the best from the data table as it gives the best performance (considering number of nodes, backtracks and time to find the solution) in most of the test cases. The reason is, it picks a variable that is most likely to cause a failure soon, thereby pruning the search tree- avoiding pointless searches through other variables.

After VAH1, the VAH3 gives better performance, and then VAH4. The **VAH3** is a extended version of VAH1, where the tie is broken using the **degree heuristic (selecting the variable that is involved in the largest number of constraints on other unassigned variables).** Therefore some extra work has to be done to get the degree of the variables that make a tie.

On the other hand, **VAH4** tries to combine heuristic VAH1 and VAH2 by minimizing the ratio. It gives similar performance to VAH3 in some test cases.

The **VAH2** heuristic is mainly used to break tie of VAH1 heuristic. It alone does not provide much improvement in performance which is evident from the data table.

Finally the **VAH5** heuristic picks variables randomly and its performance changes in different runs but on average, it provides poor performance than VAH1, VAH3 and VAH4.

**Conclusion:**

Forward checking combining with VAH1 seems the best scheme. The reason is, it picks a variable that is most likely to cause a failure soon and checking the size of the domain of neighboring variables, it detects failure early, thereby pruning the search tree- avoiding pointless searches through other variables.