CSE318 Artificial Intelligence
Offline-2

# CONSTRAINT SATISFACTION PROBLEMS

**Value Order Heuristic:**

For value ordering, least-constraining-value (LCV) heuristic is used. It is used to sort the list of values during backtracking in the order of the least constraining to most constraining. This property is measured by how many values a given value "rules out" among other variables.

The intuition behind this heuristic is that we don't want an empty domain for a variable, because that means no solution exists in this path and we have to backtrack. So we want to rule out minimum number of values among other variables with an assignment. That's why LCV can optimize CSP.

**Table:**

| DataSet | Algorithm | VAH | Time (s) | No. of Nodes | No. of Backtracks |
|---|---|---|---|---|---|
| d-10-01 | Backtracking | VAH1 | 0.008 | 291 | 233 |
| | | VAH2 | * | * | * |
| | | VAH3 | 0.001 | 77 | 19 |
| | | VAH4 | 0.004 | 190 | 132 |
| | | VAH5 | 49.852 | 2288835 | 2288777 |
| | Forward Checking | VAH1 | 0.015 | 271 | 213 |
| | | VAH2 | 99.787 | 739521 | 739463 |
| | | VAH3 | 0.001 | 76 | 18 |
| | | VAH4 | 0.015 | 173 | 115 |
| | | VAH5 | 19.544 | 394594 | 394536 |
| d-10-06 | Backtracking | VAH1 | 0 | 58 | 0 |
| | | VAH2 | * | * | * |
| | | VAH3 | 0 | 58 | 0 |
| | | VAH4 | 0 | 59 | 1 |
| | | VAH5 | 63.853 | 2541251 | 2541193 |
| | Forward Checking | VAH1 | 0 | 58 | 0 |
| | | VAH2 | 151.529 | 3674427 | 3674369 |
| | | VAH3 | 0 | 58 | 0 |
| | | VAH4 | 0 | 58 | 0 |
| | | VAH5 | 190.846 | 3547472 | 3547414 |
| d-10-07 | Backtracking | VAH1 | 0.015 | 268 | 210 |
| | | VAH2 | * | * | * |
| | | VAH3 | 0 | 58 | 0 |
| | | VAH4 | 0 | 58 | 0 |
| | | VAH5 | 4.154 | 168443 | 168385 |
| | Forward Checking | VAH1 | 0.016 | 246 | 188 |
| | | VAH2 | 1.437 | 32971 | 32913 |
| | | VAH3 | 0 | 58 | 0 |
| | | VAH4 | 0 | 58 | 0 |
| | | VAH5 | 62.905 | 1496865 | 1496807 |
| d-10-08 | Backtracking | VAH1 | 0 | 102 | 44 |
| | | VAH2 | * | * | * |
| | | VAH3 | 0.015 | 382 | 324 |
| | | VAH4 | 0.016 | 718 | 660 |
| | | VAH5 | 28.508 | 1666732 | 1666674 |
| | Forward Checking | VAH1 | 0 | 99 | 41 |
| | | VAH2 | * | * | * |
| | | VAH3 | 0.015 | 352 | 294 |
| | | VAH4 | 0.031 | 635 | 577 |
| | | VAH5 | 0.375 | 7584 | 7526 |
| d-10-09 | Backtracking | VAH1 | 0.003 | 67 | 9 |
| | | VAH2 | * | * | * |
| | | VAH3 | 0 | 58 | 0 |
| | | VAH4 | 0.002 | 71 | 13 |
| | | VAH5 | 32.438 | 1274819 | 1274761 |
| | Forward Checking | VAH1 | 0.001 | 66 | 8 |
| | | VAH2 | 5.45 | 96171 | 96113 |
| | | VAH3 | 0 | 58 | 0 |
| | | VAH4 | 0.001 | 69 | 11 |
| | | VAH5 | 43.724 | 502431 | 502373 |

| | | | | | |
|---|---|---|---|---|---|
| d-15-01 | Backtracking | VAH1 | 3.297 | 81883 | 81776 |
| | | VAH2 | * | * | * |
| | | VAH3 | 3.491 | 81027 | 80920 |
| | | VAH4 | 15.042 | 190437 | 190330 |
| | | VAH5 | * | * | * |
| | Forward Checking | VAH1 | 2.724 | 74026 | 73919 |
| | | VAH2 | * | * | * |
| | | VAH3 | 2.958 | 72615 | 72508 |
| | | VAH4 | 7.056 | 163080 | 162973 |
| | | VAH5 | * | * | * |

**Conclusion:**

Overall, VAH3 performed the best among the variable order heuristics whereas VAH4 performed 2nd best.

In VAH1, we choose the variable with the smallest domain. It minimizes the number of backtracks. It works pretty well, but here we break ties randomly so there is room for improvement.

In VAH3 we use VAH1 and break ties using VAH2, and in VAH4 we choose the variable where VAH1/VAH2 is minimized. In both the cases, we are combining two separate heuristics and hence getting better performance than using them separately.

In terms of Backtracking (BT) vs Forward Checking (FC), almost all the time forward checking performed better as it minimizes number of backtracks by recognising failure earlier. But in VAH5 sometimes backtracking performs better than forward checking for the same data because of random ordering. But if we guarantee the same variable order for BT and FC, FC will perform better.