CSE-318

(Artificial Intelligence Sessional)

Report on Offline-2 (CSP)

(Latin Square Completion Problem)

Submitted by:

Tanjeem Azwad Zaman

Roll: 1805006

Dept. of CSE, BUET

Date of Submission:

09.01.2023

1. Value Order Heuristic:

1.1 *Our Heuristic: Random Selection*

1.1.1 How it works:

- Picks a value from the domain of the selected variable
- Resetting a variable (for a backtrack) means re-entering domain values without ordering.
- Thus if the first domain value is picked every time, it still randomizes the choice to a certain extent

1.1.2 Reasoning:

- In our implementation, a node is basically a variable assignment (with no additional memory required). So, we are willing to trade "more nodes explored" for "runtime reduction"
- No additional overhead for any computation. Just picks a value from a domain
- Performs well and succeeds in minimizing runtime for smaller cases.

1.1.3 Disadvantages:

- Branch pruning is not as effective as a Least-Constraining Value (LCV) Heuristic.
- Number of nodes is higher compared to the LCV heuristic for larger N.
- In other implementations, where a node may contain an entire board as a field, this heuristic may result in memory overflow.

1.2 Other Options:

1.2.1 Least Constraining Value Heuristic:

- Involves sorting the domain each time a node is explored.
- May reduce nodes explored for larger N, but introduces O(nlogn) overhead for the sort.
- Actually reduces performance in most test cases for this particular problem

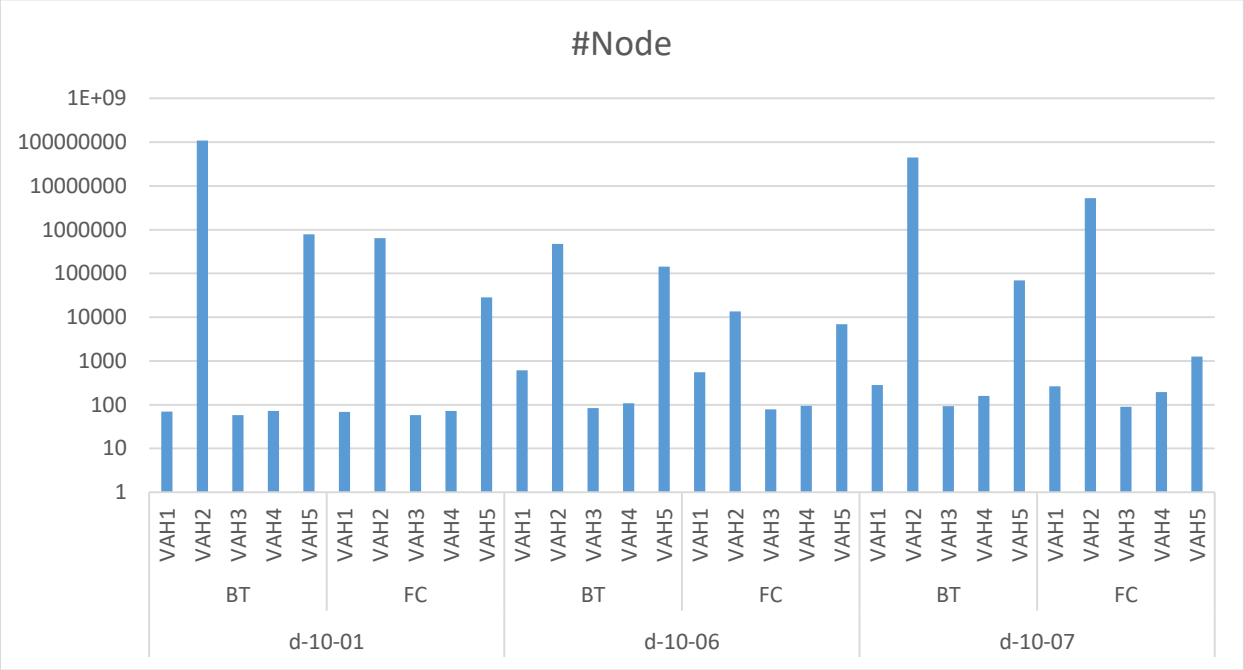
2.1 Table: Summary of Results

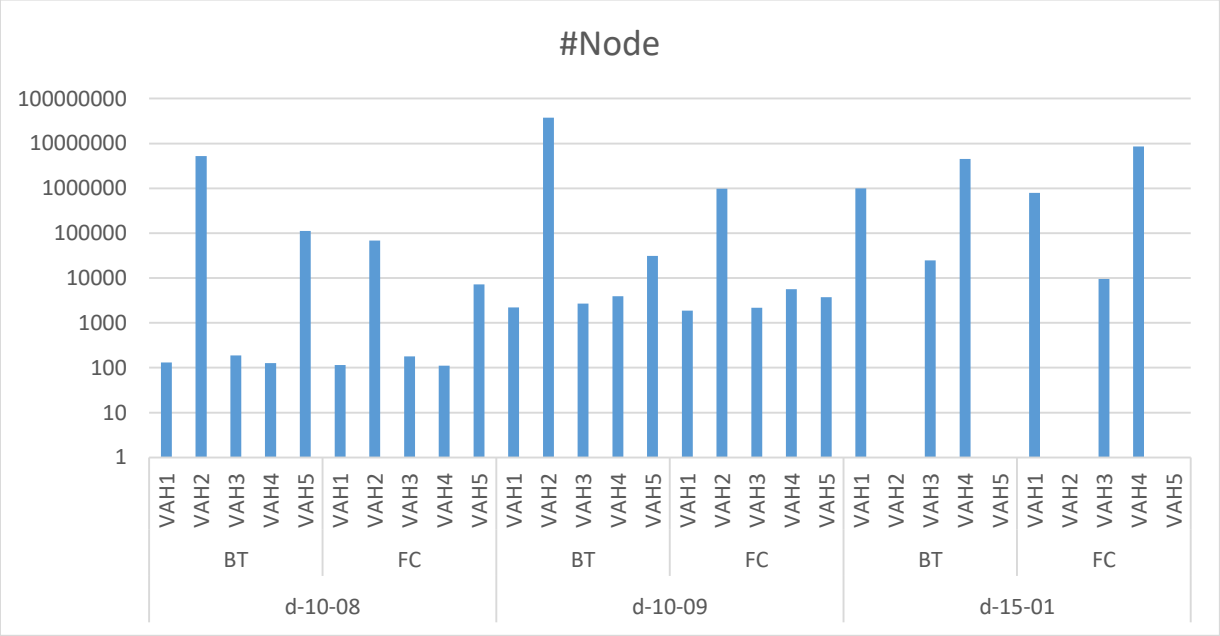
Problem	Solver	VAH	#Node	#BT	Runtime (ms)
d-10-01	BT	VAH1	70	12	0.1
		VAH2	109151296	109151238	43514
		VAH3	58	0	0.06
		VAH4	72	14	0.09
		VAH5	779965	779907	307
	FC	VAH1	68	10	0.3
		VAH2	646127	646067	804
		VAH3	58	0	0.1
		VAH4	72	14	0.2
		VAH5	28373	28315	94
d-10-06	BT	VAH1	607	549	0.1
		VAH2	469423	469365	229
		VAH3	84	26	0.1
		VAH4	107	49	0.08
		VAH5	142608	142550	54
	FC	VAH1	553	495	0.5
		VAH2	13546	13488	20
		VAH3	79	21	0.16
		VAH4	95	37	0.169
		VAH5	6881	6823	8
d-10-07	BT	VAH1	284	226	0.5
		VAH2	44809510	44809452	18795
		VAH3	93	35	3
		VAH4	160	102	5
		VAH5	69304	69246	150
	FC	VAH1	262	204	5
		VAH2	5252852	5252794	2283
		VAH3	90	32	1
		VAH4	193	135	1
		VAH5	1252	1194	6

2.2 Table: Continued

Problem	Solver	VAH	#Node	#BT	Runtime (ms)
d-10-08	BT	VAH1	131	73	0.71
		VAH2	5252852	5252794	2669
		VAH3	187	129	0.421
		VAH4	127	69	0.516
		VAH5	112602	112544	254
	FC	VAH1	114	56	0.7
		VAH2	68808	68750	261
		VAH3	179	121	0.3
		VAH4	111	53	0.3
		VAH5	7229	7171	9
d-10-09	BT	VAH1	2216	2158	19
		VAH2	37671374	37671316	15620
		VAH3	2681	2623	6
		VAH4	3903	3845	24
		VAH5	31345	31287	18
	FC	VAH1	1867	1809	4
		VAH2	974379	974321	714
		VAH3	2163	2105	5
		VAH4	5650	5592	12
		VAH5	3743	3685	3
d-15-01	BT	VAH1	992224	992117	1916
		VAH2	*	*	*
		VAH3	24572	24465	100
		VAH4	4537424	4537317	4915
		VAH5	*	*	*
	FC	VAH1	785160	785053	1808
		VAH2	*	*	*
		VAH3	9546	9439	54
		VAH4	8577634	8577527	10646
		VAH5	*	*	*

3. ***Charts:***


#Node
1E+09
100000000
10000000
1000000
100000
10000
1000
100
10
1
VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5
BT FC BT FC BT FC
d-10-01 d-10-06 d-10-07


#Node
100000000
10000000
1000000
100000
10000
1000
100
10
1
VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5 VAH1 VAH2 VAH3 VAH4 VAH5
BT FC BT FC BT FC
d-10-08 d-10-09 d-15-01

4. ***Observations and Conclusion:***

- Forward checking with VAH3 (VAH1 then tiebreak with VAH2) is the best combination, since it consistently outperforms other solvers (with 1 exceptional case), in terms of both nodes explored and runtime.

- The second-best heuristic depends on the test case, with VAH1 (minimal domain size) doing better in cases d-10-08 , d-10-09, d-15-01 and VAH4 (VAH1/VAH2) doing better in cases d-10-01 , d-10-06, d-10-07. VAH4 was the best in the dataset d-10-01.

- VAH5 (Random) performed better than VAH2 but worse than all others in most cases. Only in 1 case did it do better than VAH4 (d-10-9 FC).

- The worst heuristic by far was VAH2 (max-forward-degree) doing far worse in both the metrics of runtime and nodes explored. This heuristic on its own is not very effective. But if combined with VAH1 and used for tie-breaking, this gives much better results as seen from VAH3 performance.

- For smaller runtimes (≈ 1ms), the runtime comparisons are a bit skewed, and runtimes for the same solver with the same dataset vary quite a bit. This may be due to machine limitations and/or precision issues with the function System.nanoTime() used to record the timestamp.

- VAH2 and VAH5 could not finish on dataset d-15-01 for both BT and FC.

- FC performed better than BT in most cases. (Except easy ones, where overhead mattered more, then BT was better)

Overall, FC with VAH3 proved to be the most reliable in terms of performance.

5. ***NB :***

- The original "simple backtrack" algorithm, which did not involve domain reduction, was not yielding results for most cases even up to 20-30 minutes.
    - Thus a modified backtrack algorithm was used, that implemented domain reduction, but left out the condition checking of the Forward Checking
    - This resulted in a "Modified Backtracking" algorithm that performed worse than Forward Checking, but still yielded results
- ***"Backtrack count"*** was defined as *"the number of **false** from the **solve** function"*.
- ***"Node count"*** was defined as *"the total number of calls to **solve** function"*.
- ***Node Count = Backtrack Count + Initial Unassigned Cells***