CSE-318 Artificial Intelligence Sessional

Jan 9, 2023

Report On Latin Square Problem Solver

Offline-2 On CSP

Abu Nowshed Sakib
1705107

Introduction

A Latin square is a square grid in which each cell contains a symbol, such that each row and each column contains exactly one of each symbol. The Latin Square problem is to determine whether it is possible to fill a given grid with a Latin square, and if so, to find one. This problem has applications in the fields of statistics, experimental design, and cryptography, among others.

One approach to solving the Latin Square problem is to use **Backtracking**, a method of trial and error that involves systematically searching through all possible solutions and eliminating those that do not satisfy the constraints of the problem. Another approach is to use **Forward Checking**, which can prune the search space by eliminating invalid assignments early on in the search process. This makes it more efficient than Backtracking.

There are a number of heuristics on selecting a variable and its value. Selecting one rather than the other can make the process faster. So, finding out a good heuristic is necessary. Here, in this report, we have tried to figure out exactly that.

Value Order Heuristics

We have used two Value Order Heuristics.

1. **Sequential Order:** The domain of the chosen variable is sorted in ascending order and value is chosen in that order one by one until a consistent solution is found.
2. **Least Constrained Value (LCV):** This heuristic selects the value which is present in the domains of least number of empty squares that are in the corresponding row and column of the selected square.

LCV heuristics choose a value of a square so that it is not present in most of the empty squares in the corresponding row or column. So, there is less conflict. As a result, LCV heuristic solves the latin square problem visiting less nodes or with lesser backtracks than Sequential order heuristic.

Variable Order Heuristics

We have used five variable order heuristics.

1. **Smallest Domain (VAH1):** The variable chosen is the one with the smallest domain.
2. **Maximum Forward Degree (VAH2):** The variable chosen is the one with the maximum degree to unassigned variables. Also, called max-forward-degree.
3. **Smallest Domain + Max Forward Degree (VAH3):** The variable chosen by VAH1; Ties are broken by VAH2.
4. **Minimum VAH1/VAH2 (VAH4):** The variable chosen is the one that minimizes the VAH1 / VAH2.
5. **Random Unassigned Variable (VAH5):** A random unassigned variable is chosen.

We have tried all possible combinations of Variable order and Value order heuristics with Backtracking and Forward Checking with our given datasets. The results are shown below:

- Green cells: BEST scheme for that dataset
- Red cells: WORST scheme for that dataset

Value Order Heuristic: Sequential

Dataset	Backtrack/Forward Check	Variable Order Heuristics	Time(ms)	No. of Nodes	No. of Backtracks
d-10-01	Backtrack	VAH1	30	2031	1802
		VAH2	*	*	*
		VAH3	12	321	263
		VAH4	16	341	281
		VAH5	*	*	*
	Forward Checking	VAH1	16	229	15
		VAH2	16416	8911853	4135325
		VAH3	8	58	0
		VAH4	14	60	1
		VAH5	747	362349	155857
d-10-06	Backtrack	VAH1	12	339	281
		VAH2	*	*	*
		VAH3	12	339	281
		VAH4	20	349	290
		VAH5	*	*	*
	Forward Checking	VAH1	6	58	0
		VAH2	30378	16391733	7337490
		VAH3	7	58	0
		VAH4	8	59	1
		VAH5	158	32622	14263
d-10-07	Backtrack	VAH1	54	5487	4912
		VAH2	*	*	*
		VAH3	53	3507	3130
		VAH4	54	5997	5371
		VAH5	456703	333440257	300096205
	Forward Checking	VAH1	38	575	37
		VAH2	9970	5040372	2110532
		VAH3	34	377	23
		VAH4	35	588	58
		VAH5	679	351164	145734

Dataset	Backtrack/Forward Check	Variable Order Heuristics	Time(ms)	No. of Nodes	No. of Backtracks
d-10-08	Backtrack	VAH1	39	2758	2457
		VAH2	*	*	*
		VAH3	18	568	486
		VAH4	15	448	378
		VAH5	*	*	*
	Forward Checking	VAH1	26	301	22
		VAH2	8071	4292669	2026948
		VAH3	17	82	3
		VAH4	12	70	2
		VAH5	3323	1881341	804636
d-10-09	Backtrack	VAH1	9	305	247
		VAH2	*	*	*
		VAH3	384	55635	50044
		VAH4	325	81355	73192
		VAH5	*	*	*
	Forward Checking	VAH1	15	58	0
		VAH2	20617	12096135	5698521
		VAH3	304	5591	613
		VAH4	228	7855	1051
		VAH5	2708	1587366	673156
d-15-01	Backtrack	VAH1	226236	27423484	25595199
		VAH2	*	*	*
		VAH3	95005	6275554	5857131
		VAH4	121283	14811394	13823915
		VAH5	*	*	*
	Forward Checking	VAH1	169456	1828285	204202
		VAH2	*	*	*
		VAH3	87344	418423	39615
		VAH4	98952	972335	109416
		VAH5	*	*	*

Value Order Heuristic: LCV

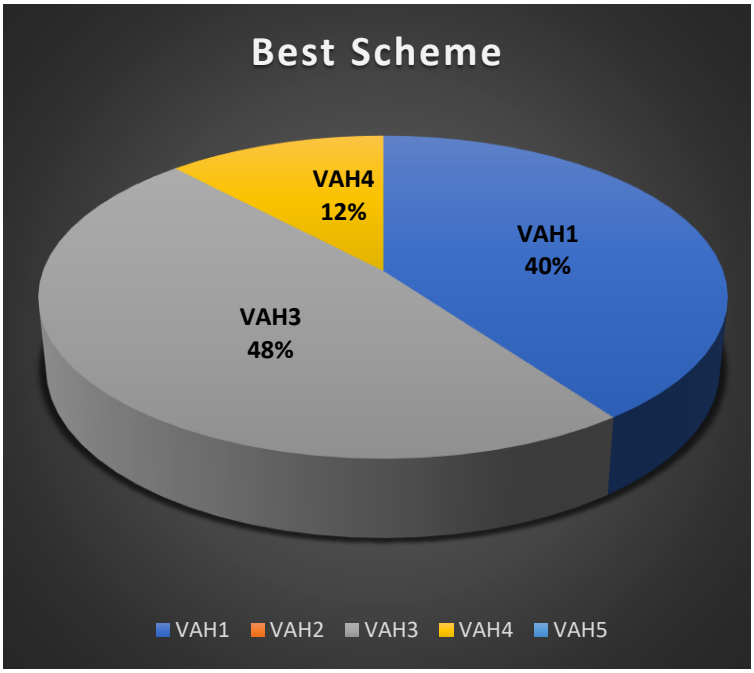
Dataset	Backtrack/Forward Check	Variable Order Heuristics	Time(ms)	No. of Nodes	No. of Backtracks
d-10-01	Backtrack	VAH1	88	3264	2929
		VAH2	*	*	*
		VAH3	36	700	623
		VAH4	60	1844	1653
		VAH5	*	*	*
	Forward Checking	VAH1	57	291	20
		VAH2	27732	1308289	569400
		VAH3	40	77	1
		VAH4	47	188	15
		VAH5	26878	1244373	534383
d-10-06	Backtrack	VAH1	32	495	437
		VAH2	*	*	*
		VAH3	38	510	452
		VAH4	35	522	463
		VAH5	*	*	*
	Forward Checking	VAH1	15	58	0
		VAH2	135853	5892112	2217685
		VAH3	18	58	0
		VAH4	23	59	1
		VAH5	3235	138691	60494
d-10-07	Backtrack	VAH1	39	500	442
		VAH2	180123	44690170	40221143
		VAH3	44	606	538
		VAH4	40	500	442
		VAH5	951230	3434502672	3010563051
	Forward Checking	VAH1	45	268	22
		VAH2	1410	54773	21802
		VAH3	13	58	0
		VAH4	13	58	0
		VAH5	46620	2160964	905983

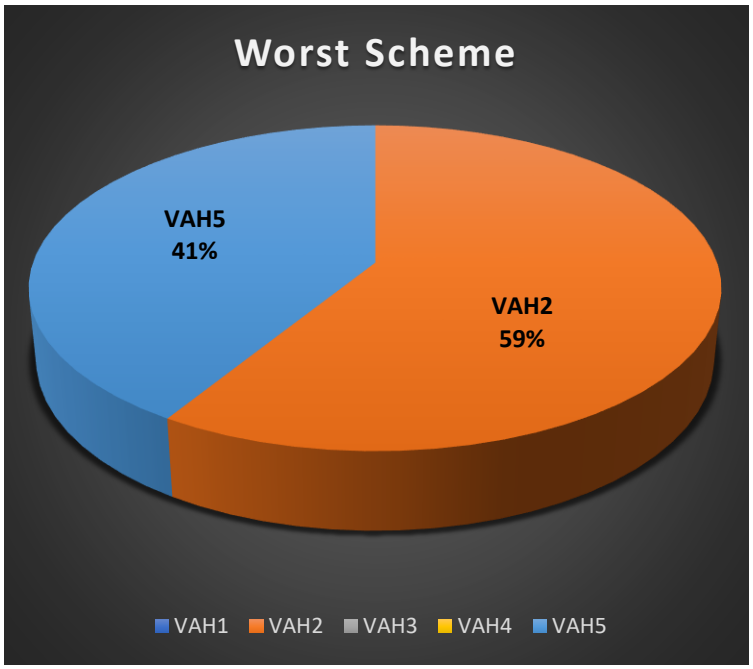
Dataset	Backtrack/Forward Check	Variable Order Heuristics	Time(ms)	No. of Nodes	No. of Backtracks
d-10-08	Backtrack	VAH1	32	946	844
		VAH2	*	*	*
		VAH3	101	3744	3362
		VAH4	133	7169	6445
		VAH5	51756	16970237	15273204
	Forward Checking	VAH1	26	102	3
		VAH2	1573363	81793354	36349776
		VAH3	82	382	30
		VAH4	94	718	83
		VAH5	1540	68643	28652
d-10-09	Backtrack	VAH1	21	514	456
		VAH2	115051	33621130	30259008
		VAH3	34	607	537
		VAH4	31	1495	1337
		VAH5	*	*	*
	Forward Checking	VAH1	20	67	1
		VAH2	3622	173460	77289
		VAH3	19	58	0
		VAH4	19	71	2
		VAH5	3190390	155009810	66374263
d-15-01	Backtrack	VAH1	38903	2302416	2148908
		VAH2	*	*	*
		VAH3	31517	1215235	1134208
		VAH4	50139	2856931	2666459
		VAH5	*	*	*
	Forward Checking	VAH1	17106	81883	7857
		VAH2	*	*	*
		VAH3	26855	81027	8412
		VAH4	38898	184865	21785
		VAH5	*	*	*

Conclusion

For Value Order Heuristics, we can clearly see that LCV uses lesser nodes and lesser backtracks than Sequential Order. So, **LCV is better than Sequential Order for choosing value.**

For Variable Order Heuristics, the Performance of a scheme depends vastly on the dataset. So, for further analysis we will use pie chart.





Best Scheme – VAH3

Second Best Scheme – VAH1

Worst Scheme – VAH2

From table, we can clearly see that Forward Checking Algo needs much lesser nodes and backtracks than Backtracking Algo as we prune the search space by eliminating values that are inconsistent with the current assignment of squares, thereby reducing the amount of search that needs to be done.

So, from all the above discussion we can come to the conclusion- **Forward Checking Algo** with **VAH3** (as Variable Ordering Heuristics) and **LCV**- Least Constraining Value (as Value Ordering Heuristics) is the best scheme to solve Latin Square Problem.