Report on

OFFLINE-2 (CSP)

Course No: CSE 318

Course Name: Artificial Intelligence Sessional

Submitted By

Name: Ishika Tarin Ime

Student ID: 1805092

Section: B2

Level 3 Term 2

Latin Square:

A Latin Square is a n x n grid filled by n distinct numbers each appearing exactly once in each row and column.

For example, if given n= 3. Then a latin square would be:

[1 ,2 ,3

  2 ,3 ,1

  3 ,1 ,2]

Here each value appears exactly once in each row and each column.

CSP:

Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem. Such type of technique leads to a deeper understanding of the problem structure as well as its complexity.

Constraint satisfaction depends on three components, namely:

- **X:** It is a set of variables.
- **D:** It is a set of domains where the variables reside. There is a specific domain for each variable.
- **C:** It is a set of constraints which are followed by the set of variables.
-

Data Table

Problem	Solver	VAH	#Node	#BT	Time(ms)
d-10-01	BT	VAH1	339777	55152	48
	BT	VAH2	9563222	1339909	349
	BT	VAH3	2363403	365381	130
	BT	VAH4	80993	13411	4986
	BT	VAH5	1937356906	315048460	83994
	FC	VAH1	339777	55152	38
	FC	VAH2	9563222	1339909	330
	FC	VAH3	2363403	365381	196
	FC	VAH4	80993	13411	29
	FC	VAH5	57416534	57407666	21728
d-10-06	BT	VAH1	2027857	334404	103
	BT	VAH2	309595	53936	165
	BT	VAH3	30402525	5248033	2813
	BT	VAH4	6635601	1125632	573
	BT	VAH5	1374653	1374595	1460
	FC	VAH1	2027857	334404	197
	FC	VAH2	309595	53936	96
	FC	VAH3	30402525	5248033	1895
	FC	VAH4	6635601	1125632	522
	FC	VAH5	1401353	1401295	3390
d-10-07	BT	VAH1	780426	120546	52
	BT	VAH2	2538242	340837	112
	BT	VAH3	7085866	1151038	375
	BT	VAH4	2235894	344047	149
	BT	VAH5	1093722653	1093722595	1223448
	FC	VAH1	780426	120546	73
	FC	VAH2	2538242	340837	165
	FC	VAH3	7085866	1151038	409
	FC	VAH4	2235894	344047	178
	FC	VAH5	148261343	24936556	11991
	BT	VAH1	2071270	285817	80

d-10-08	BT	VAH2	6220849	1114822	176
	BT	VAH3	16212994	3028658	658
	BT	VAH4	39530500	6824897	2670
	BT	VAH5	17960381	242892	45146
	FC	VAH1	2071270	285817	107
	FC	VAH2	6220849	1114822	262
	FC	VAH3	16212994	3028658	987
	FC	VAH4	39530500	6824897	2005
	FC	VAH5	17960381	242892	48299
d-10-09	BT	VAH1	1848154633	285666735	48276
	BT	VAH2	3088664073	445652699	74400
	BT	VAH3	2166220852	361766545	107517
	BT	VAH4	7331882386	1164272533	428297
	BT	VAH5	84798141	89324659	94748
	FC	VAH1	1848154633	285666735	35416
	FC	VAH2	3088664073	445652699	53445
	FC	VAH3	2166220852	361766545	85355
	FC	VAH4	7331882386	1164272533	403748
	FC	VAH5	84798141	89324659	77748
d-15-01	BT	VAH1	1361353596	8083539	24024471
	BT	VAH2	*	*	*
	BT	VAH3	71074429	71062472	209334
	BT	VAH4	16727626	16726525	698753
	BT	VAH5	*	*	*
	FC	VAH1	1361353596	8083539	10033230
	FC	VAH2	*	*	*
	FC	VAH3	71074429	71062472	154324
	FC	VAH4	16727626	16726525	334033
	FC	VAH5	*	*	*

Analysis:

There are 5 Value ordering heuristics used here.

They are:

VAH1 : The variable chosen is the one with the smallest domain

VAH2 : The variable chosen is the one with the maximum degree to unassigned variables

VAH3 : The variable chosen by VAH1, Ties are broken by VAH

VAH4 : The variable chosen is the one that minimizes the VAH1 / VAH2

VAH5: A random unassigned variable is chosen

Two solvers are used here.

1. Simple Backtracking
2. Forward checking

Among the above VAH's , VAH 1 needed less nodes and backtracks. So it made the program faster.

And in my implementation, though the runtime was a bit close, still forward checking was faster than backtracking.