CSE 318

OFFLINE -2

On

Constraint Satisfaction Problems

Submitted By:

Sk. Sabit Bin Mosaddek

Department: CSE

Section: B2

Student ID: **S201805106**

Value Ordering Heuristic

In this offline, we used two value ordering heuristics and compared them with each other to find the efficient one. First, we chose Least Constraining Value. Here, we prefer the value which has the least amount of frequency in the neighboring unassigned variable's domain. Here the neighboring variables are those which share the column or row with the chosen variable. This heuristic is theoretically better as it follows the best strategy to find a solution. It removes less values from others' domain which reduces the chance of failure. But everytime sorting the current variable's domain with this heuristics costs runtime. Hence, we introduced another heuristics to make it a bit faster. Here, we select the value which has the least frequency in the whole matrix. As it tries to balance the frequency of a value in every step, it works well. Maintaining the least frequent value in the matrix is easy enough which doesn't cause us with extra runtime overhead. Thus, in terms of node count, Least Constraining Value works well but sometimes Least Frequent Value in the matrix works faster in terms of runtime.

Collected Data

Input-1 (10x10):

Solver	VAH	Node Count	Backtrack Count	Runtime (ms)
Forward Check	VAH1	205	8	0.086
	VAH2	24899151	11186712	3168.012
	VAH3	539	36	0.213
	VAH4	329	24	0.111
	VAH5	526803	228837	146.089
Backtrack	VAH1	325381	292807	12.382
	VAH2	11441485654	2703508706	4702573.767
	VAH3	6779585649	1767750393	2841746.254
	VAH4	11441485654	2703508706	4669055.160
	VAH5	434141195	127299371	302179.798

Input-2 (10x10):

Solver	VAH	Node Count	Backtrack Count	Runtime (ms)
Forward Check	VAH1	609	56	0.185
	VAH2	60111578	27322113	7621.495
	VAH3	148	5	0.076
	VAH4	775	56	0.253

	VAH5	120846	52584	33.391
Backtrack	VAH1	7551	6760	0.35
	VAH2	*	*	*
	VAH3	7621927063	2681924753	3594566.158
	VAH4	*	*	*
	VAH5	*	*	*

Input-3 (10x10):

Solver	VAH	Node Count	Backtrack Count	Runtime (ms)
Forward Check	VAH1	92	1	0.055
	VAH2	56297990	24117880	7401.345
	VAH3	92	1	0.058
	VAH4	89	0	0.056
	VAH5	260919	111031	70.935
Backtrack	VAH1	2416	2113	0.544
	VAH2	*	*	*
	VAH3	4826545236	1254786320	2035419.174
	VAH4	*	*	*
	VAH5	*	*	*

Input-4 (10x10):

Solver	VAH	Node Count	Backtrack Count	Runtime (ms)
Forward Check	VAH1	284	29	0.100
	VAH2	57276	25377	20.559
	VAH3	291	14	0.116
	VAH4	3305	349	0.891
	VAH5	172457	71748	48.064
Backtrack	VAH1	2422	2144	0.149
	VAH2	*	*	*
	VAH3	5124536874	1724536875	2427851.124
	VAH4	*	*	*
	VAH5	*	*	*

Input-5 (10x10):

Solver	VAH	Node Count	Backtrack Count	Runtime (ms)
Forward Check	VAH1	74	2	0.044
	VAH2	151542334	49123564	35041.536
	VAH3	151	9	0.078

	VAH4	96	5	0.049
	VAH5	5418995	2313209	1451.263
Backtrack	VAH1	193291	173921	6.590
	VAH2	*	*	*
	VAH3	*	*	*
	VAH4	*	*	*
	VAH5	*	*	*

Input-6 (15x15):

Solver	VAH	Node Count	Backtrack Count	Runtime (ms)
Forward Check	VAH1	987	73	1.412
	VAH2	*	*	*
	VAH3	2019	186	3.038
	VAH4	645153	103413	185.015
	VAH5	*	*	*
Backtrack	VAH1	12542	11621	1.982
	VAH2	*	*	*
	VAH3	*	*	*
	VAH4	*	*	*
	VAH5	*	*	*

Conclusion

From the collected data, we can deduce that Forward checking works better than Backtracking by a large margin. Forward checking reduces the number of nodes by determining invalid paths early by looking at the domains. On the other hand, backtracking keeps going until it fails with an invalid value assignment. Now, in terms of variable assignment heuristics, VAH1 seems to work slightly better than VAH3 and VAH4 in forward checking. But in backtracking, VAH1 becomes a lot more efficient than the others. Hence, we can conclude that Forward Checking is better than Backtracking and VAH1 is a good variable assignment heuristic among the given 5 heuristics.