Report on the Performance of Different Variable Ordering Heuristics for Solving The Latin Square Problem

HAZ Sameen Shahgir
Student ID: 1805053, Department of CSE, BUET

January 2023

1 Introduction

A constraint satisfaction problem (CSP) is a problem in which variables representing objects or values in a system must be assigned values in such a way that all constraints or restrictions are satisfied. A common example of a CSP is a map coloring problem, in which the variables represent different regions on a map and the constraints specify that no two adjacent regions can have the same color.

Heuristics are methods or strategies used to find a solution to a problem that may not guarantee the optimal solution, but can often find a good solution in a reasonable amount of time. Heuristics can be used to solve CSPs by guiding the search for a solution through the space of possible assignments of values to variables. Some common heuristics for solving CSPs include backtracking search, which involves systematically trying different assignments of values to variables and backtracking when a conflict is encountered, and local search algorithms, which involve iteratively improving a solution by making small changes to the assignment of values to variables.

Overall, heuristics are useful for solving CSPs because they allow for the efficient exploration of the space of possible solutions and can often find good solutions even when the problem is too large or complex to be solved optimally.

2 Problem Statement

A Latin square is a matrix filled with distinct symbols such that each symbol appears exactly once in each row and each column. The Latin square problem is a constraint satisfaction problem in which the goal is to find a valid Latin square for a given size, or to determine that no such square exists.

3 Solvers and Heuristics

3.1 Solvers

3.1.1 Simple Backtracking

Backtracking is a general algorithmic technique for finding all (or some) solutions to a problem that incrementally builds candidates to the solutions, and abandons a candidate ("backtracks") as soon as it determines that the candidate cannot possibly be completed to a valid solution.

Backtracking can be used to solve constraint satisfaction problems by systematically trying different assignments of values to variables and backtracking when a conflict is encountered. The process begins with an empty assignment of values to variables and then adds values one at a time, checking

at each step to ensure that the assignment remains consistent with the constraints of the problem. If a conflict is encountered, the algorithm backtracks to the previous step and tries a different value for the variable. This process continues until a solution is found or it is determined that no solution exists.

3.1.2 Forward Checking

Forward checking is a heuristic for solving constraint satisfaction problems that involves maintaining a record of the possible values that each variable in the problem can take on, and eliminating values from this record as the search for a solution progresses.

Forward checking can be used to prune the search space of a constraint satisfaction problem by eliminating values that cannot possibly be part of a valid solution. This can significantly reduce the time required to find a solution, especially in problems with many variables and constraints.

3.2 Heuristics

3.2.1 Minimum Remaining Value

When selecting a new variable for assignment, the strategy of choosing the one with the smallest domain is called the minimum remaining-values (MRV) heuristic. It also has been called the "most constrained variable" "fail-first" heuristic, the latter because it picks a variable that is most likely to cause a failure soon, thereby pruning the search tree. If some variable X has no legal values left, the MRV heuristic will select X and failure will be detected immediately—avoiding pointless searches through other variables. The MRV heuristic usually performs better than a random or static ordering, sometimes by a factor of 1,000 or more, although the results vary widely depending on the problem.

3.2.2 Maximum Forward Degree

When selecting a new variable for assignment, the strategy of choosing the one with the most number of constraints attached to it is called the maximum forward degree heuristic. It attempts to reduce the branching factor on future choices by selecting the variable that is involved in the largest number of constraints on other unassigned variables.

4 Experimental Results

4.1 Heuristic Performance Comparison

In the given problem set, each 10*10 Latin Squares had 57 unassigned variables while the 15*15 Latin Square had 106. Minimum Remaining Value (MRV), Minimum Remaining Value with Greater Forward Degree Tie Breaking (TieBreaks) and minimizing Minimum Remaining Value by Forward Degree (MRV/Degree) are the three most effective heuristics. Maximum Degree and Random Selection heuristics performed significantly worse, to the point that they could not compute some cases. Specifically, Maximum Forward Degree heuristic failed on all simple Backtracking experiments and both Maximum Forward Degree and Random Selection failed on 15*15 Latin square.

4.2 Best Performer: Minimum Remaining Value with Maximum Degree Tie Break

For all cases, it was observed that Forward Checking is strictly better than Simple Backtracking. Using Forward Checking, it was observed that MRV, TieBreaks and MRV/Degree yield comparable performance, roughly in the same order of magnitude. Among them, TieBreaks is the fastest for the 15*15 case, exploring roughly 200K nodes compared to MRV's 900K, and MRV/Degree's 1.6M. This performance gain is significant especially because run-time becomes a greater concern as the

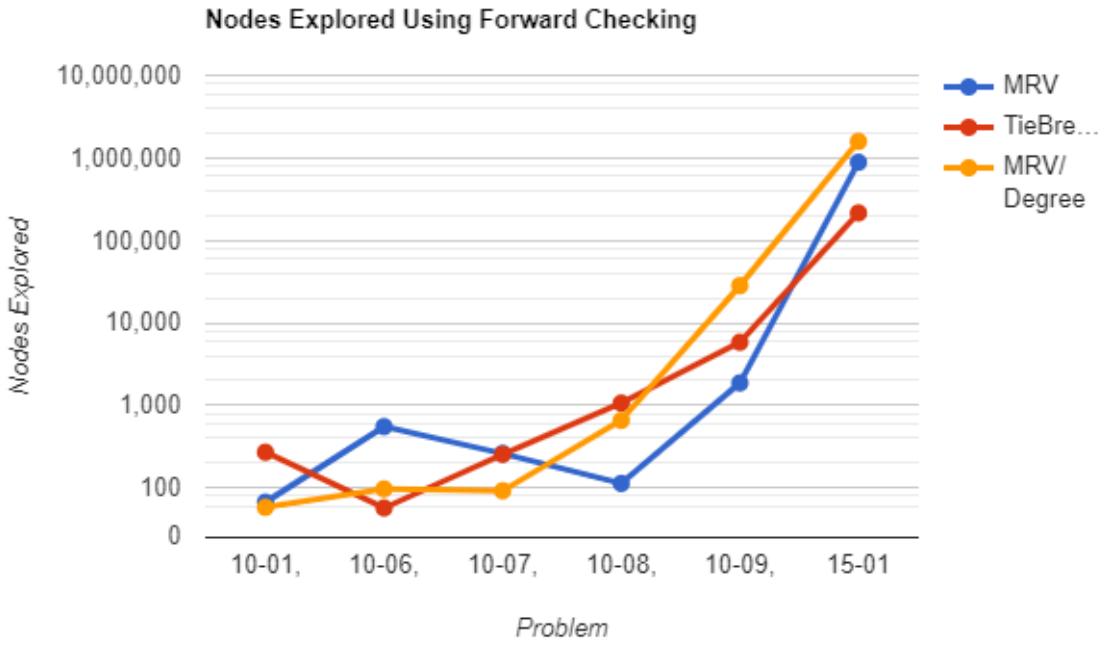

Figure 1: Nodes Explored by Heuristics

problem size increases. As a result, we believe Minimum Remaining Value with Maximum Degree Tie Break to be the best performing heuristic.

5 Conclusion

Based on the results of our experiment, it appears that the heuristics tested significantly improved the performance of the CSP solver on the problem instances used. The heuristic that performed the best overall was Minimum Remaining Value with Maximum Degree Tie Break, which was able to find solutions more quickly and with fewer search steps than the other heuristics in most cases. On the other hand, Maximum Degree heuristic did not perform as well and was often outperformed by the other heuristics.

It is worth noting that the specific heuristics that work best for a given CSP problem can depend on the structure of the problem and the constraints involved. In this experiment, we only tested a limited set of heuristics on a specific type of problem, so it is possible that other heuristics or combinations of heuristics might perform even better on this problem or on other types of CSPs. Further experimentation and analysis would be needed to fully understand the relative strengths and weaknesses of the different heuristics in different contexts.

Problem	Method	Heuristic	Time(ms)	Nodes Explored	Backtracks
10.01	BT	**VAH1: MRV**	**0.57589**	**69**	**2**
		VAH2: MaxDegree	*	*	*
		VAH3: TieBreaks	0.39746	286	16
		VAH4: MRV/Degree	0.18974	107	13
		VAH5: Random	58.81654	780052	238608
	FC	VAH1: MRV	0.0539	67	2
		VAH2: MaxDegree	1809.715480	12134184	2719243
		VAH3: TieBreaks	0.12402	270	16
		VAH4: **MRV/Degree**	**0.12063**	**58**	**1**
		VAH5: Random	2.8107	28789	6396
10.06	BT	VAH1: MRV	0.14224	606	54
		VAH2: MaxDegree	9598.34723	94814132	22381393
		VAH3: TieBreaks	**0.022460**	**57**	**0**
		VAH4: MRV/Degree	0.039630	109	7
		VAH5: Random	8.92699	142804	43316
	FC	VAH1: MRV	0.12774	552	54
		VAH2: MaxDegree	71.549990	366943	102544
		VAH3: TieBreaks	**0.02232**	**57**	**0**
		VAH4: MRV/Degree	0.0751	101	6
		VAH5: Random	0.62226	6855	1499
10.07	BT	VAH1: MRV	0.04809	283	21
		VAH2: MaxDegree	*	*	*
		VAH3: TieBreaks	0.066530	273	17
		VAH4: MRV/Degree	**0.053280**	**97**	**5**
		VAH5: Random	4.372210	69349	20372
	FC	VAH1: MRV	0.04583	261	21
		VAH2: MaxDegree	1505.181200	9387220	2442970
		VAH3: TieBreaks	0.061060	256	17
		VAH4: MRV/Degree	**0.051530**	**92**	**5**
		VAH5: Random	0.130850	1256	251
10.08	BT	**VAH1: MRV**	**0.026270**	**130**	**17**
		VAH2: MaxDegree	*	*	*
		VAH3: TieBreaks	0.192480	1167	102
		VAH4: MRV/Degree	0.269620	974	166
		VAH5: Random	7.114290	112784	37095
	FC	**VAH1: MRV**	**0.027110**	**113**	**17**
		VAH2: MaxDegree	324.577840	2063498	439617
		VAH3: TieBreaks	0.170820	1065	102
		VAH4: MRV/Degree	0.239420	657	122
		VAH5: Random	0.677680	7233	1682
10.09	BT	**VAH1: MRV**	**0.293150**	**2215**	**343**
		VAH2: MaxDegree	*	*	*
		VAH3: TieBreaks	0.978270	6521	700
		VAH4: MRV/Degree	8.111840	46429	8041
		VAH5: Random	2.241390	31416	9244
	FC	**VAH1: MRV**	**0.263900**	**1868**	**343**
		VAH2: MaxDegree	5008.149830	31754251	7809082
		VAH3: TieBreaks	0.904900	5819	700
		VAH4: MRV/Degree	5.754960	28235	4224
		VAH5: Random	0.352160	3742	907
15.01	BT	VAH1: MRV	194.901790	992221	107878
		VAH2: MaxDegree	*	*	*
		VAH3: TieBreaks	**58.972170**	**238948**	**23871**
		VAH4: MRV/Degree	1633.91562	5755066	1218009
		VAH5: Random	*	*	*
	FC	VAH1: MRV	171.575060	884296	107876
		VAH2: MaxDegree	*	*	*
		VAH3: TieBreaks	**57.6289**	**215075**	**23871**
		VAH4: MRV/Degree	680.53287	1583927	205108
		VAH5: Random	*	*	*

Table 1: Comparison of Simple Backtracking and Forward Checking with Different Heuristics