Bangladesh University of Engineering and Technology

Department of Computer Science and Engineering

Academic Year 2022 - 2023

CSE 318
-Artificial Intelligence Sessional-

Offline No. 2
Constraint Satisfaction Problem (Latin Square)

Name: Anwarul Bashir Shuaib

Roll: 1805010

Section: A1

Date of Submission: January 09, 2023

Evaluating Heuristics for Solving the Latin Square Problem

Abstract

In this report, we present a constraint satisfaction approach to solving the Latin square problem. We implemented simple backtracking and backtracking with forward checking, and tested five different heuristics to guide the search. We also included the least constraining value ordering heuristic in our comparison. The performance of the different heuristics was measured and compared. The best performance was achieved with backtracking and forward checking using the heuristic of selecting variables by minimum domain size.

Problem Description

The Latin square problem is a combinatorial problem where the goal is to fill in a grid of size n×n with n different symbols (usually the numbers 1 to n), such that no symbol appears more than once in any row or column. It is a generalization of the Sudoku puzzle, and is often used as a test case for constraint satisfaction algorithms. The problem can be represented as a constraint satisfaction problem, where each cell in the grid is a variable, and the constraints specify that each row and column must contain each symbol exactly once.

$$\begin{bmatrix} 5 & 2 & 3 & 4 & 1 \\ 1 & 4 & 5 & 3 & 2 \\ 3 & 1 & 4 & 2 & 5 \\ 2 & 5 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 & 5 \end{bmatrix}$$

Heuristics

1. `VAH1`: Select the variable with the smallest domain. This heuristic aims to choose variables that are most constrained, as they have fewer options for legal values.

2. `VAH2`: Select the variable with the maximum degree to unassigned variables. This heuristic aims to choose variables that are connected to the largest number of unassigned variables, as this may allow for more forward pruning.

3. `VAH3`: Select the variable using VAH1, with ties broken using VAH2. This heuristic combines the two previous heuristics, prioritizing variables with small domains but also considering the number of connections to unassigned variables.

4. `VAH4`: Select the variable that minimizes the ratio of VAH1 to VAH2. This heuristic aims to balance the conflicting goals of selecting variables with small domains and selecting variables with many connections to unassigned variables.

5. `VAH5`: Select a random unassigned variable. This heuristic does not consider any specific properties of the variables or the state of the search. It simply chooses a variable at random.

Simulation Report

We conducted several tests to compare the performance of different heuristics and solvers. The results are summarized in the following table, which shows the number of nodes explored, the number of backtracks, the time taken, and the solver used for each test case.

Table 1: Summary of the Simulation

Test Case	Solver	Heuristic	Explored	Backtracks	Time (ms)
d-10-01	FC	VAH1	100	5	14
		VAH2	978767	249547	3180
		VAH3	92	2	2
		VAH4	134	10	5
		VAH5	11671	2846	76
	BT	VAH1	450473	131339	273
		VAH2	2928097	845029	1909
		VAH3	217384	53004	146
		VAH4	217384	53004	160
		VAH5	286127551	84634490	221826
d-10-06	FC	VAH1	123	8	1
		VAH2	4120914	1119832	11878
		VAH3	57	0	1
		VAH4	57	0	1
		VAH5	843388	223702	3358
	BT	VAH1	119067	32945	77
		VAH2	200637	53439	133
		VAH3	20774	5711	16
		VAH4	20769	5711	19
		VAH5	195008006	59595357	152881
d-10-08	FC	VAH1	57	0	1
		VAH2	1071007	269721	3229
		VAH3	72	3	1
		VAH4	96	7	1
		VAH5	695000	175957	2827
	BT	VAH1	100189	25873	75
		VAH2	3467200	1015808	2322
		VAH3	48868	14604	32
		VAH4	48868	14604	30
		VAH5	173049739	49552889	131637

Table 2: Summary of the Simulation (Continued)

Test Case	Solver	Heuristic	Explored	Backtracks	Time
d-10-08	FC	VAH1	198	9	1
		VAH2	1965	518	6
		VAH3	938	89	3
		VAH4	1591	196	7
		VAH5	16332	4223	67
	BT	VAH1	424230	118604	276
		VAH2	270678388	71398699	162507
		VAH3	30045815	8386044	21104
		VAH4	30045815	8386044	21735
		VAH5	10159	2807	8
d-10-09	FC	VAH1	57	0	1
		VAH2	10224	2941	28
		VAH3	57	0	1
		VAH4	74	6	1
		VAH5	5053449	1326099	19340
	BT	VAH1	1225880	387908	773
		VAH2	203984	54536	132
		VAH3	16177286	4759592	10808
		VAH4	16287885	4782504	11236
		VAH5	69177541	19735367	55427
d-15-01	FC	VAH1	100638	11410	706
		VAH3	178029	20273	1120
		VAH4	762076	105612	3712

Machine Configuration

- `CPU`: Intel(R) Xeon(R) Platinum 8272CL CPU @ 2.60GHz
- `Cache Size`: 36608 KB
- `Java JDK`: 11.0.14.1
- `Operating System`: Ubuntu 20.04.5 LTS (Focal Fossa)
- `System Memory`: 8GB

Visualization

In order to visualize the performance of the different heuristics, we plotted three clustered stacked bar charts for the time taken, explored nodes count, and backtracked nodes count for the simulations. These charts provide a clear comparison of the relative effectiveness of the heuristics across different metrics. **Due to the large variations in the values for these metrics, we took the logarithm of the values before plotting the charts**. By analyzing the results of these charts, we can gain insights into the strengths and weaknesses of each heuristic and how they compare to one another.

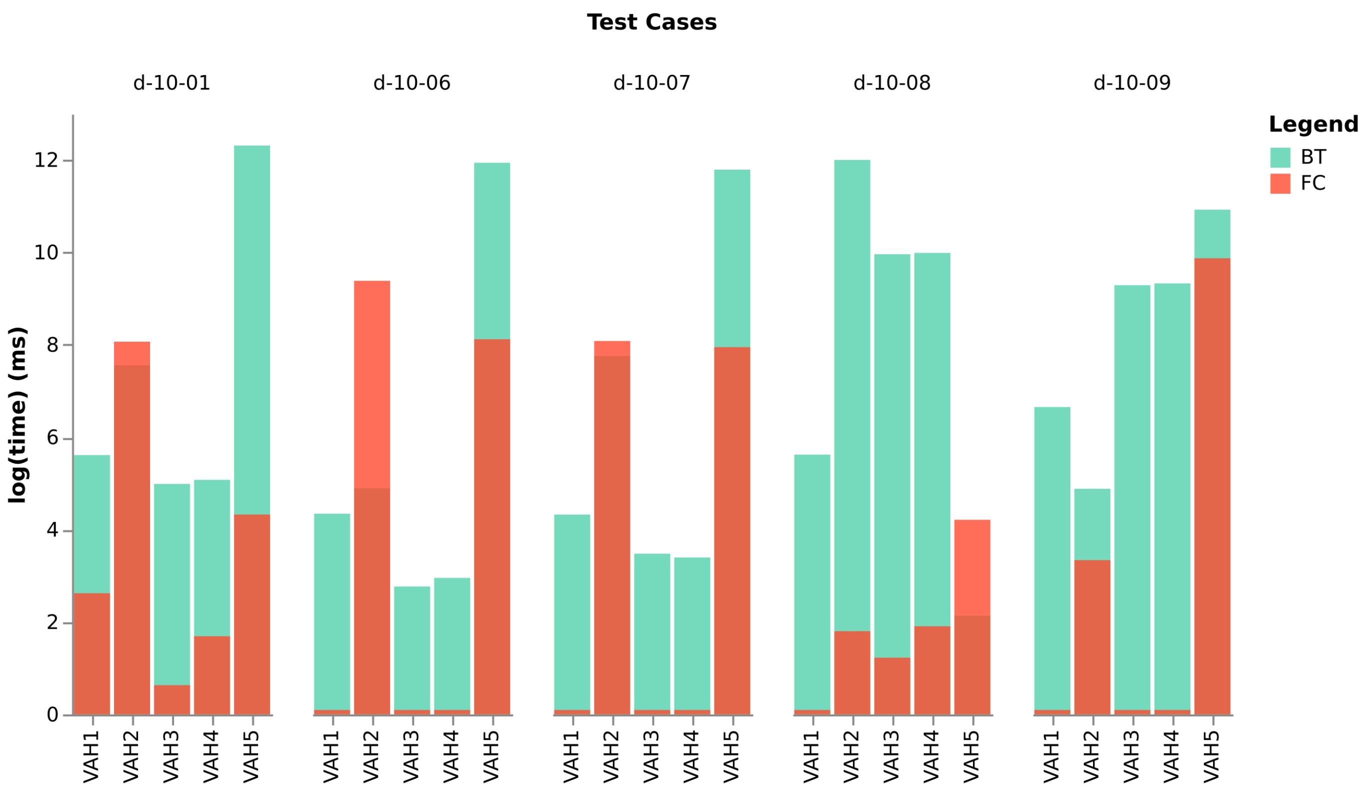

Figure 1: Comparison of the time taken using different heuristics

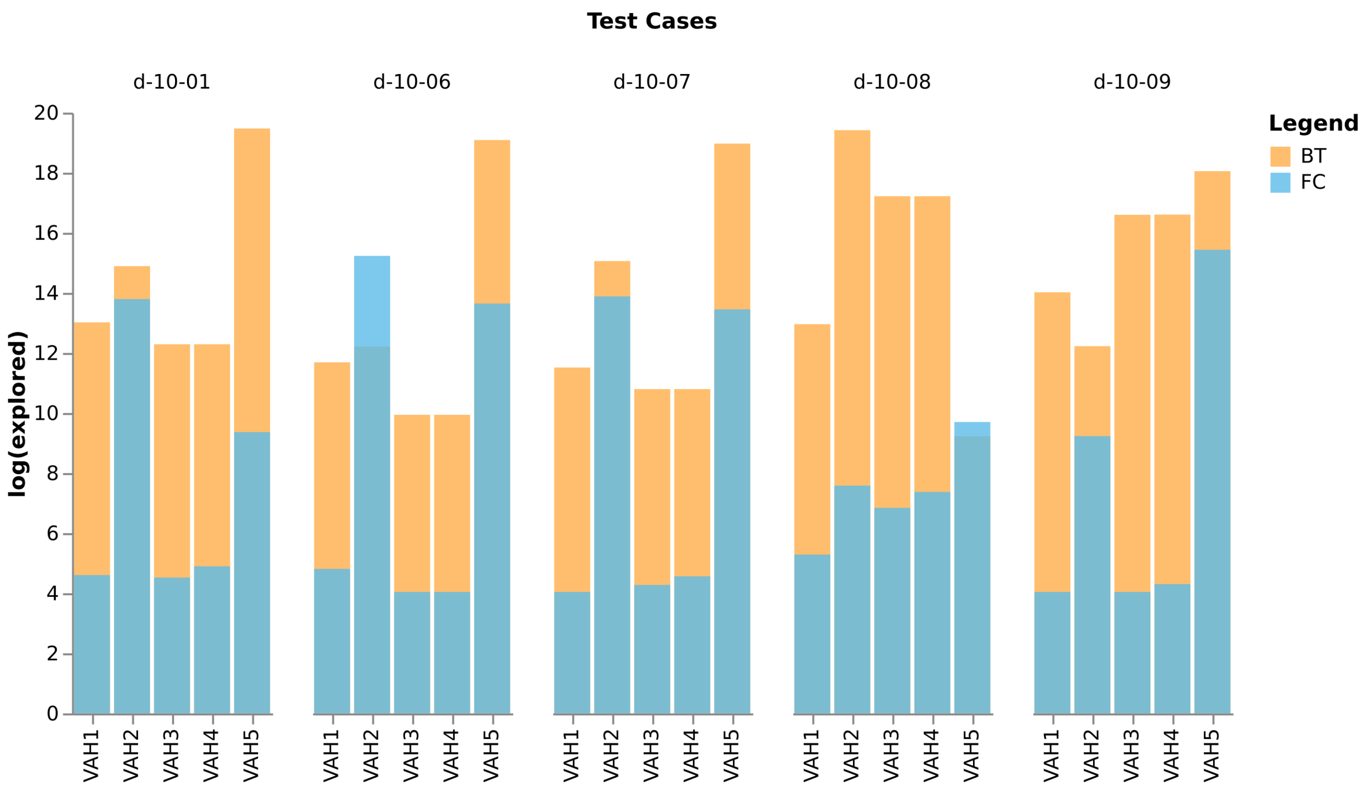

Figure 2: Comparison of the number of explored nodes using different heuristics

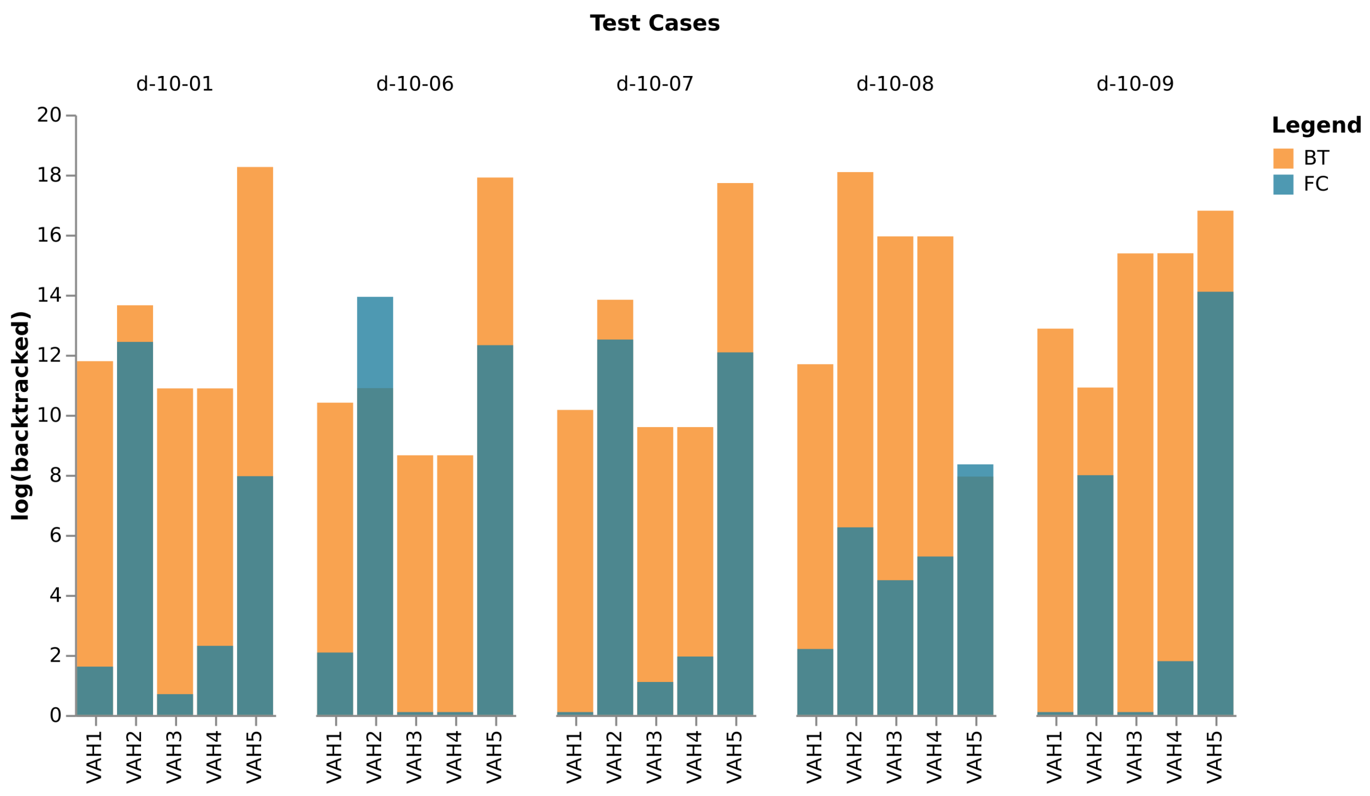

Figure 3: Comparison of the number of backtracked nodes using different heuristics

Justification for Variable Ordering Heuristics

Based on the results of our simulations, it appears that the VAH3 heuristic performed the best, followed by VAH1 for the forward checking scheme. On the other hand, VAH1 performed relatively better than VAH3 for simple backtracking scheme. For both schemes, the VAH5 heuristic performed the worst, with VAH2 also performing poorly.

The VAH3 heuristic is able to effectively balance the conflicting goals of selecting variables with small domains (as in VAH1) and selecting variables with many connections to unassigned variables (as in VAH2). By combining these two factors, VAH3 is able to make more informed choices about which variables to assign first, leading to faster convergence and fewer explored and backtracked nodes. This may not always be the best choice, as it can lead to more backtracking and a slower convergence to a solution.
The VAH1 heuristic was outperformed by VAH3 in forward checking scheme. This suggests that while choosing variables with small domains can be effective, additional considerations such as the number of connections to unassigned variables can further improve performance. Since simple backtracking scheme only worked with the initial degree information, VAH3 did not see any improvement over VAH1.
On the other hand, the VAH5 heuristic, which selects a random unassigned variable, performed poorly compared to the other heuristics. This is likely because it does not consider any specific properties of the variables or the state of the search, leading to suboptimal choices and slower convergence.

Overall, the results of our simulations suggest that the VAH3 heuristic is the most effective for solving the Latin square problem using constraint satisfaction techniques, followed by VAH1 for the forward checking scheme, and vice versa for the simple backtracking scheme. The VAH2 and VAH5 heuristics performed relatively poorly in comparison for both schemes.

Justification for Value Ordering Heuristic

The least constraining value heuristic aims to select values for variables that will have the least impact on the remaining variables and constraints in the problem. There are several benefits to using the least constraining value heuristic.

- This heuristic can reduce the number of backtracks and improve search efficiency by choosing values less likely to cause conflicts with other variables or constraints.
- It can also improve solution quality by choosing values less likely to cause conflicts, making it more likely to find a solution that satisfies all constraints.

Overall,the least constraining value heuristic is particularly useful in problems with many variables and constraints, where the search space is large and the risk of backtracking is high.

Improving VAH2

The VAH2 heuristic prioritizes variables that are connected to the largest number of unassigned variables. However, this strategy may not always be the most effective, as it can lead to more backtracking and a slower convergence to a solution. Here are the key points about how the VAH2 heuristic could be improved by selecting the variable with the minimum degree to unassigned variables:

- This approach would prioritize variables that are connected to the fewest number of unassigned variables.
- It may allow for more forward pruning, leading to faster convergence and fewer explored and backtracked nodes.
- This approach may offer improved performance in terms of both speed and solution quality compared to the VAH2 heuristic.

Conclusion

In conclusion, the Latin square problem was solved using constraint satisfaction techniques, with two algorithms (simple backtracking and backtracking with forward checking) implemented and tested using five different heuristics. The best performance was achieved with backtracking with forward checking. The least constraining value ordering heuristic was also utilized. The results of the simulations showed that the VAH3 heuristic was the most effective, followed by VAH1. The VAH2 and VAH5 heuristics performed relatively poorly in comparison. These findings suggest that carefully selecting variables and values using informed heuristics can significantly improve the efficiency and effectiveness of constraint satisfaction algorithms for solving the Latin square problem.