CS 537: Artificial Intelligence
HW#3 Sudoku Report

Kumar Anupam (110614410)
Muhammad Ali Ejaz (110559131)

This report is on the HW#3: Sudoku and highlights the 5 different algorithms used to solve the Sudoku board. It also shows the performance of the 5 algorithms and also gives an insight on which of the 5 implementation methods should be used when. The 5 implementation methods are:

1. Backtracking
2. Backtracking + MRV heuristic
3. Backtracking + MRV + Forward Checking
4. Backtracking + MRV + Constraint Propagation
5. Min-conflicts Heuristic

1. Backtracking

This algorithm chooses the first empty cell in the board, finds all the valid numbers possible for that cell, iterates on that list and keeps setting numbers in a Depth First manner. Recursion is used to implement the DFS nature. The algorithm unsets the values that do not result in a valid Sudoku board.

2. Backtracking + MRV heuristic

This algorithm implements the same backtracking algorithm along with Minimum Remaining Values and Least Constraining Values heuristics. This heuristic helps in solving the complex boards faster.

3. Backtracking + MRV + Forward Checking

This algorithm implements the Backtracking + MRV heuristic algorithm along with Forward Checking Heuristic. In this heuristic we check if there are no valid values remaining for the neighboring cells. This heuristic helps in pruning the tree and lessen the execution time.

4. Backtracking + MRV + Constraint Propagation

This algorithm implements the Backtracking + MRV heuristic algorithm along with 3-ARC constraint propagation. This also helps in pruning the tree and lessen the execution time.

5. Min-conflicts Heuristic

This algorithm requires a completely filled board. If the board is not completely filled, then this algorithm fills the empty cells randomly. The following steps are done in an iteration which runs for a counter = 10000. Next it finds the set of conflicting cells, randomly chooses a cell from the set, finds a number that that minimizes the conflict for that cell, sets that number to the cell and proceeds with the iteration. The number that minimizes the conflict is calculated by finding the frequency of numbers present in the domain of the cell and choosing a number that has the least frequency. There is a lot of randomization in this heuristic and thus this algorithm sometimes fails to find a solution for a board.

Performance of the 5 implementation methods

	BT	BT+MRV	BT+MRV+FC	BT+MRV+CP	Min Conflicts
Empty 4*4	0.000237 16	0.001836 16	0.002091 16	0.007693 16	0.00347 50
Empty 9*9	0.007189 407	0.055347 81	0.052324 81	0.321908 81	No Solution
Empty 12*12	0.131471 7080	0.198321 144	0.196934 144	1.340661 144	No Solution
Wrongly filled 12*12	No Solution	No Solution	No Solution	No Solution	0.320524 262
Board 1	0.004765 184	0.041713 76	0.038924 76	0.048022 76	No Solution
Board 2	3.06689 130315	2.011561 2746	1.947563 2746	1.972546 2746	No Solution
Board 3	11.671281 416165	0.212882 247	0.206673 247	0.233329 247	No Solution

The table above shows the execution time and number of consistency checks for each of the 5 implementation methods. The boards used for testing are listed at the end of this report.

Conclusion

- We see that for simpler boards, the Backtracking algorithm takes lesser execution time than the algorithms which use heuristics to prune the valid numbers. This is because for uncomplicated boards, the heuristics are not required and BT does a fair job in finding the solution board by DFS method.
- For larger complex boards such as Board 2 and Board 3, the Backtracking algorithm takes a lot of time to build the large DFS tree. The heuristics perform way better as they prune the tree and the numbers which may lead to an invalid solution board are not used from the start itself. The performance of BT+MRV+FC and BT+MRV+CP is better than the performance of BT+MRV as the former two heuristics do a 1 step forward check of numbers and 3-ARC consistency check respectively that may lead to invalid boards and prune the tree accordingly.

- Min Conflict heuristic involves a lot of randomization and thus produces results for smaller boards and larger boards which are partially filled. We cannot let the Min Conflict algorithm run for long for obvious performance constraints.

The boards used for testing mentioned in the performance table are as follows:

Wrongly filled 12*12:

```
12,3,4
1,1,3,4,5,6,7,8,9,10,12,12;
5,5,7,8,9,10,11,12,1,2,3,4;
9,10,10,12,1,2,3,4,8,5,6,7;
2,11,4,1,1,9,8,3,10,7,12,5;
8,7,5,6,2,1,1,10,11,9,4,3;
10,9,12,3,11,5,4,7,2,1,8,6;
3,4,6,5,12,8,10,2,7,11,1,9;
8,8,1,9,7,4,6,5,3,12,10,2;
7,7,10,2,3,11,1,9,3,6,5,8;
12,1,2,11,8,3,9,6,5,4,7,10;
4,4,9,7,10,12,5,1,6,8,2,11;
6,6,8,10,4,7,2,11,12,3,1,1;
```

Board 1:

```
12,3,4;
1,9,-,-,11,8,-,-,-,10,-,-;
5,2,10,-,-,12,1,-,-,8,-,-;
-,8,3,4,-,7,-,9,-,1,6,5;
-,-,9,-,4,-,-,5,12,6,3,-;
11,12,8,-,3,1,-,-,-,9,7,-;
3,5,4,-,9,11,-,-,-,-,10,-;
9,-,2,12,-,-,-,-,-,-,-,7;
-,3,-,8,-,5,-,7,10,-,-,2;
-,-,-,-,1,6,-,10,3,-,9,-;
10,1,-,2,-,-,11,8,9,-,-,-;
-,-,11,-,-,9,12,6,-,-,2,-;
-,-,5,-,10,-,-,-,4,-,-,12;
```

Board 2:
12,3,4;
-,2,12,-,-,-,1,4,-,8,-,-;
-,9,-,-,-,2,-,-,-,4,1,10;
10,1,-,-,5,-,6,9,-,-,-,12;
-,-,-,-,1,7,-,-,-,-,-,-;
9,-,6,-,2,-,-,12,5,7,-,-;
4,-,3,-,-,-,-,-,11,-,2,-;
-,5,-,12,-,-,-,-,-,1,-,4;
-,-,1,2,11,-,-,5,-,10,-,6;
-,-,-,-,-,-,3,7,-,-,-,-;
12,-,-,-,10,3,-,8,-,-,6,11;
1,10,5,-,-,-,9,-,-,-,12,-;
-,-,7,-,12,5,-,-,-,9,10,-;

Board 3:
12,3,4;
-,3,-,-,11,-,-,-,10,-,-,12;
-,11,-,-,-,-,-,-,-,-,7,-;
7,-,-,2,-,-,-,9,-,-,6,-;
-,4,-,5,-,-,-,-,-,6,-,-;
-,-,7,-,4,10,9,12,-,5,-,-;
-,-,11,-,-,-,-,-,4,-,10,-;
-,8,-,6,-,-,-,-,-,11,-,-;
-,-,3,-,10,6,12,4,-,7,-,-;
-,-,5,-,-,-,-,-,12,-,9,-;
-,5,-,-,9,-,-,-,8,-,-,11;
-,9,-,-,-,-,-,-,-,-,2,-;
3,-,-,8,-,-,-,7,-,-,5,-;