**Solution to Sudoku(a Constraint Satisfaction Problem) using different algorithms**

**Submitted by**

**Sreepradhana Thirumalaiswami-110349668**

**Sadhana Kumaravel-110338732**

1)Backtracking:

It’s the breadth first search implementation. Each variable(cell in the Sudoku board) is assigned a value satisfying constraint with previously assigned cell values. In case, there’s no solution left for any cell at any step, the recursion stack backtracks to the previous assignment stage and tries new assignment for the variable. In case of absence of any solution to a problem, backtracking ends in root with no solution obtained. Since solving Sudoku is NP-Complete, the solution is obtained at some stage of traversal in the tree of value assignments.

2)Minimum Remaining Values Heuristic:

The most constrained variable has minimum remaining values. Therefore the most constrained position in the board is taken and assigned values first followed by assignment of values to positions constrained lesser. This is an improvement over backtracking and increasing the speed of the algorithm to a great extent.

3)Forward checking:

This method keeps track of values for unassigned board positions and reports failure in case a cell in the neighborhood runs out of values to be assigned.

4)Constraint Propagation:

This is a step ahead of Forward checking in detecting failures (no numbers available satisfying constraints for a particular board position) early.In case a possible value for a cell is removed on checking for constraints, the values for all cells in the neighborhood which could be possibly affected are computed. In case any cell has no possible values left, failure is reported.

5)Min-conflicts:

Among the conflicting positions in the board, one is chosen random and the algorithm works to reduce conflict at the particular cell. In case the conflicts corresponding to all cells turn to be zero, the algorithm stops there and returns value, else the iterations goes upto a maximum number of steps set and reports the board state at the given step.

Implementation:

1. Simple Backtracking- Time complexity is very high
2. Backtracking+MRV – increases speed of backtracking by a large number
3. Backtracking+MRV+Forward Checking- Detects failure by propagating information from assigned to unassigned variables
4. Backtracking+MRV+Constraint Propagation- detects failure faster than Forward checking by propagating values to all possible cells which can affected
5. Min-Conflicts Heuristic –This search method avoids searching less promising regions of state space.

Conclusions:

1. Backtracking is not preferred for CSP since it’s a brute force implementation consuming maximum time and space than other methods always.
2. Backtracking+MRV- Preferred over backtracking to increase speed without any propagation of assigned values to unassigned ones(simple computation).
3. Backtracking+MRV+Forward Checking- competitive in terms of detecting conflicts right before failure with minimal checking (only immediate neighbors). Used in cases when failures (conflicts in assignments) occur midway. This is not suitable in cases when failures occur at the last assignment stage.
4. Backtracking+MRV+Constraint Propagation- Very competitive in terms of detecting failures and pruning the entire branch. But the propagation of constraints takes time since change in value for a variable requires recomputation for all associated variables. This is used when a CSP reaches failure when assigning value at the last stages.
5. Min-conflicts: This method converges based on the order of conflicting positions chosen. This hill-climbing approach can be used when one doesn’t want to check for solutions assigning less promising values.