AI Project Report- 2

E Saileswara Reddy
CS21B1078

Problem statement : Solving Sudoku using AI

Algorithm used : Simulated Annealing

Explanation:

Simulated annealing is a type of local search which allows moves to inferior neighbors with a probability that is regulated over time.

How AI will help to solve this problem:

We are assigning random values for temperature and cooling rate and no of iterations. Then we try multiple and find optimal ones to reach the goal.

States:

- A state is represented in form of matrix which is 9*9.

  This 9*9 contains 9 ,3*3 sub matrix.
- Maximum number of iterations as 100,000.
- Initial Temperature
- Cooling rate

Initial State:

A 9 by 9 matrix with some zeroes or any user defined initial state.

Assumption:

We assume that 0 on board represents empty cell and 1 to 9 represents a filled cell.

Constraints:

We should fill numbers from 1 to 9 by following below rules:

- In an entire row no two numbers should be same
- In an entire column no two numbers should be same
- In a 3*3 sub matrix no number should be same.

Procedure:

1. We declare a function called solve_sudoku in which we pass the board(matrix) as parameter.
2. Next, we are checking where the 0 values are present in board.
3. We store this in empty_cells variable.
4. Next, we randomly fill the board with some random numbers row wise. Such that in a row no common number is present.
5. After Randomly applying the numbers for the board we will calculate the cost of the board made above.

Cost Calculation:

   a. Initially we are assigning cost as 0.
   b. We calculate the cost on number of repeated numbers in each row, column and 3*3 sub matrix.
   c. For row cost calculation iterate along the row and check if any element is repeated in the row.
   d. If repeated then increase the cost with 1.
   e. Similar to this for column iterate along the column and check.
   f. And in case of 3*3 sub matrix Store the 3*3 matrix in a new variable and calculate the cost for it.

6. After calculating the Cost of randomly initialized board we check if the cost is equal to 0.
7. If it is equal to zero then the board is the required answer.
8. Else we check if iteration is less than maximum iteration.
9. If it is less then we again we will modify the above randomly initialized table. We modify by swapping two randomly selected empty cells in a row which are not the original. (Here empty cells means before randomly initializing the value we calculated the empty cell position those values)
10. If it is more than maximum iterations then we terminate with printing sudoku not solved.
11. Next calculate the cost for newly obtained board.
12. Next calculate the difference between this cost and previous cost.
13. If difference is less than 0 means new cost is less than previous cost.
14. In this case update the table to this and update the cost.
15. Else then calculate the probability that program will accept it i.e, e^(-difference/current temperature).
16. If probability is greater than some probability which is randomly generated then update the state and cost to this.
17. Increase the iteration and decrease the temperature according to the cooling rate as temperature= current Temp* cooling rate.
18. In this way iterate through the number of iterations.
19. If the maximum number of iterations are reached and the temperature is zero then return the existing board and say sudoku not solved.
20. Next return the board to main function. Then print the board.

Output:

Output will be solved sudoku or sudoku is not solved (If the maximum number of iterations is reached).