Search

Smit Patel

June 23, 2020

1 Assume you are given an Asterisk instance with a randomly assigned value from 1 to 9 in every empty cell. How might you define a search problem that would start at this initial state and gets to a goal state? Briefly detail a possible state space, start state, goal test, and successor function. (5 marks)

1.1 Assumptions

- There are **no** empty cells present in the start state.
- There possibly exists duplicates in a given row, column or an house.

1.2 Search Algorithm - Backtracking Search

Since we do not have an empty cells present at the start state, we will pick the first cell at (0, 0) and see if it is unique in the row, column and the in the house(s) it is part of. Say it is not unique and there exists duplicates, if such situation occurs find all values not used in the row, column, and house (also the current value) and find the common value. If no common value exists keep existing value and move. If common value does exist change it to that value. What this will do is both remove the duplicate and up hold the constraints of asterisk sudoku.

1.3 Search Specifications

- Start State: Initialized each cell with random integer between 1 to 9.
- State Space: All possible configurations of asterisk sudoku board possible (valid and invalid configurations). 9^{81} possible combinations.
- Goal Test: Check if the configuration of asterisk sudoku board is valid, by checking if each row, column and houses have all different values.
- Successor Function: Picks the cell that is violating the asterisk sudoku property.

2 When might such a search algorithm be preferable to plain backtracking with MRV? When might plain back tracking be preferable? (5 marks)

2.1 Plain Backtracking with MRV

This type of algorithm is preferable over plain backtracking when we have more duplicates as minimum remaining value will be useful in detecting the next viable option for that field.

2.2 Plain Backtracking

Since plain backtracking is more useful when we have empty cells it would be preferable to use plain backtracking when we have least duplicates possible in the state. Plain backtracking can also be bad since it can go down a path looking for a solution where its not possible and waste time.