Implementation of a Sudoku Solver Using Backtracking and a Greedy Heuristic

The idea of using better heuristics to improve the performance of a backtracking solver is effective in many domains. In this project we explore whether the performance of your backtracking Sudoku solver might be improved by using a better heuristic for selecting the next cell to assign.

Your programming project

In this project, you will enhance your backtracking Sudoku solver with a greedy heuristic from the CS literature for selecting the next cell. Update your `chooseNextCell` method so that it computes, for each unassigned cell, the number of reasonable values that this cell might take (how many values do not appear in this cell's row or column or box), and then returns a cell with the minimum number of possible values (breaking ties randomly). The intuition behind this heuristic is that we want to assign a value to a cell with minimum remaining values, while there are still reasonable values to give it.

After your own testing convinces you that your implementation is correct, run and time your solver on the required input set. Preserve the runtimes and compare them with your previous times on the same data.

Submission instructions

By the deadline, submit your working solver, evidence of its correctness, and a laboratory report comparing the performance of your backtracking with greedy heuristic solver to your previous solver implementations.

This material is based upon work supported by the National Science Foundation under Grant No. 1140753.