Sudoku Difficulty Notes

1 Depth

The idea of depth focuses on how many ways a player can progress with the puzzle. A state, S, of the sudoku is said to derive a cell, c, if the value of c can be found solely using logical techniques on S. Let the depth of a cell, c, be defined as follows:

- If c has its value in the initial state, depth(c) = 0
- Otherwise, if c can be derived from the initial state depth(c) = 1
- Otherwise, if c can be derived from the state only made up of every depth 0 and 1 cell, then depth(c) = 2
- Otherwise, if c can be derived from the state only made up of every depth 0, 1, ... k-2 and k-1 cell, then depth(c) = k

An important note with depth is that if a cell can be derived from a given state, the cell could also be derived from that state with more cells filled in. This means that depth 1 cells can always be derived in a sudoku solve, no matter how many other cells have been filled in.
Difficulty can be estimated using depth by looking at how many depth 1 cells there are. A puzzle will feel easier to the player if they have more ways to progress at a given state. This is because it would likely take a player longer to find a way to progress in a state with 1 way to progress than it would in a state with 2 ways to progress. Because each depth 1 cell can be derived at any point, there are more ways to progress at these points. So if a puzzle has many depth 1 cells, it would likely be easier for the solver.
However, once these depth 1 cells are all filled in, they are no longer options for progress, but all depth 2 cells would be able to be derived. So for a given amount of depth 0 and 1 cells, if there are more depth 2 cells, the puzzle will likely seem easier.

Maybe multiply amount of cells in each depth by a fixed coefficient and find the total to estimate difficulty?

Calculating the depth of each cell doesn't take long, as it is the same as solving a sudoku in a specific order, which can be done quickly.

2 State Space

Depth of cells can only be used to estimate difficulty based on an estimate of how many ways the player can progress at a given point. However, it is possible to work out exactly how many ways a player can progress.
Consider a state space for a given sudoku puzzle, where:

- Each node is a state of the sudoku with some cells filled in.
- Let A' be identical to A except A' also has cell c filled in. There exists a directed edge from state A to state A' when cell c can be derived from state A.
- The initial state is the initial grid given in the sudoku puzzle.
- The goal state is the completed sudoku grid where every cell has been filled.

Note that for each state, all values filled in will be correct as they must have been derived from a previous state, and only correct values can be derived.
Also note that there will be no cycles in the state space, as deriving a cell can't remove the values of other cells, so each derivation increases the number of filled cells by exactly one.
The number of ways a player can make progress at a given cell is that cell's out-degree, and the number of ways a player can complete a sudoku from the initial state is equal to the number of distinct paths from the initial state to the goal state. This can be used to estimate the difficulty of a sudoku, as if there are more ways to complete a sudoku, it will likely be easier for the player to find one of them.
As the state space graph is directed and acyclic, the number of paths can be found by in O(V + E) time by first topological sorting and then counting the number of paths from each node in reverse order.
The issue with this method is the number of nodes, as there are 81 cells and each can either be filled in or not, there are up to 2^{81} nodes in the state space. This makes counting every path unreasonable, so estimates have to be used.

3 Grouped States

One method of estimating the number of ways to solve a sudoku is by 'grouping' states where the newly filled in cell is the same. Consider states P and Q that can both derived cell c, where $\mathrm{P_{ij}}$ is the cell that is i rows down and j columns right in P, and value(d) is the value of cell d. Let R be the result of grouping P and Q, where

$$R_{ij} = \begin{cases} value(c), & if\ c = R_{ij} \\ value(P_{ij}), & if\ P_{ij}\ isn't\ empty \\ value(Q_{ij}), & if\ Q_{ij}\ isn't\ empty \\ empty, & otherwise \end{cases} \tag{1}$$

Note that if both $\mathrm{P_{ij}}$ and $\mathrm{Q_{ij}}$ both aren't empty, they will both have the same value.
States can be grouped to make a new state if they can derive the cell that corresponds to the new state. This means the old states do not need to be visited again from this new state, as the new state has all the information from every state leading into it.
This can be applied to the state space by keeping track of which cell was newly derived for that state and the depth of this cell. The node for cell c is created by grouping all nodes that can derive cell c and have depth less than the depth of c. This method gives a graph with no more than 81 nodes, and where solving the sudoku can me modelled by finding a topological ordering of the graph.

Counting the number of topological orderings is hard