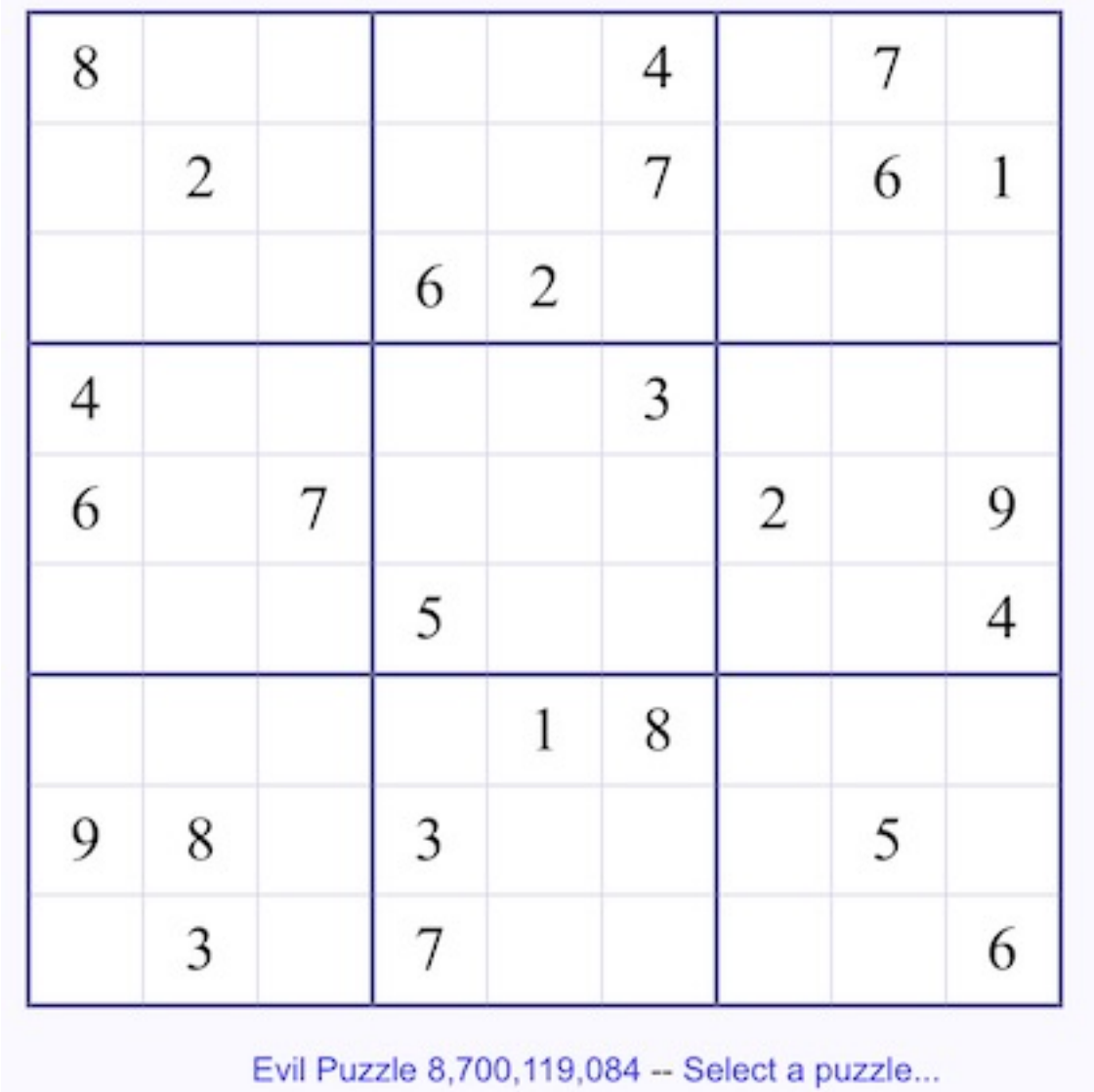
HLSudokuSolver Overview

HLSudokuSolver solves puzzles by the repeated use of 3 different algorithms. They are described in detail below.

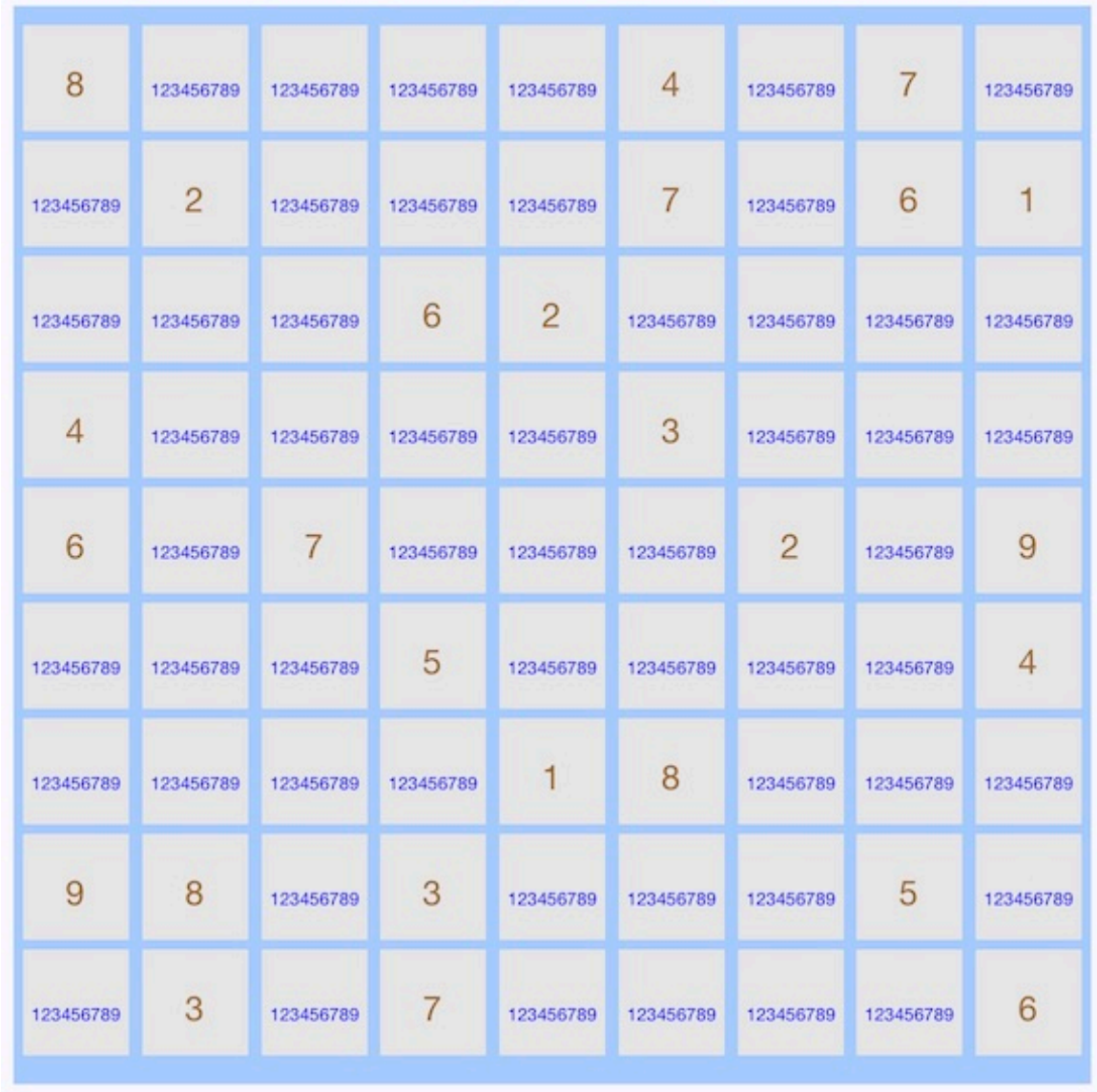
From an initial puzzle, data is imported to the Solver and can then be stepped through the solving process.

The first step is to import the positions and values of the 'Given' data



Scroll down for the rest of this document.

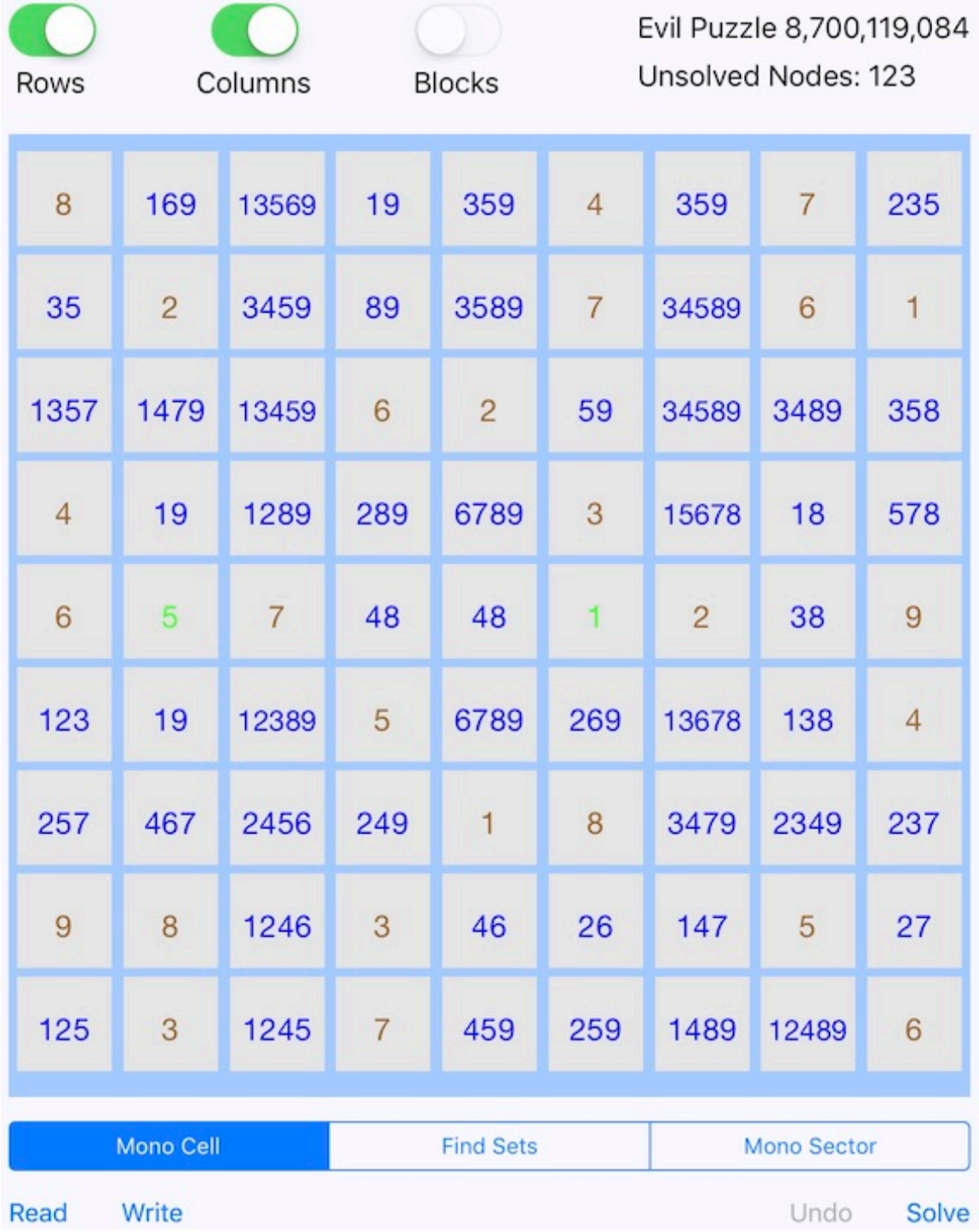
When the 'Go to Solver' button is tapped the Solver panel loads with this initial data:



The cells with a single brown value are the initial 'Given' data and all other cells are filled with the full set of possible solutions. The full set is { 1 2 3 4 5 6 7 8 9 }.

There is a method called PrunePuzzle and it is called at this point and also every time any Solve operation is performed. It's not considered a 'Solve' operation but only reduces the data set to what is known. It does this by going through each row and removing from that row any values that have been solved on that row (cell has only one value). Then it does this for all columns and finally for all blocks.

After pruning the initial puzzle becomes this:

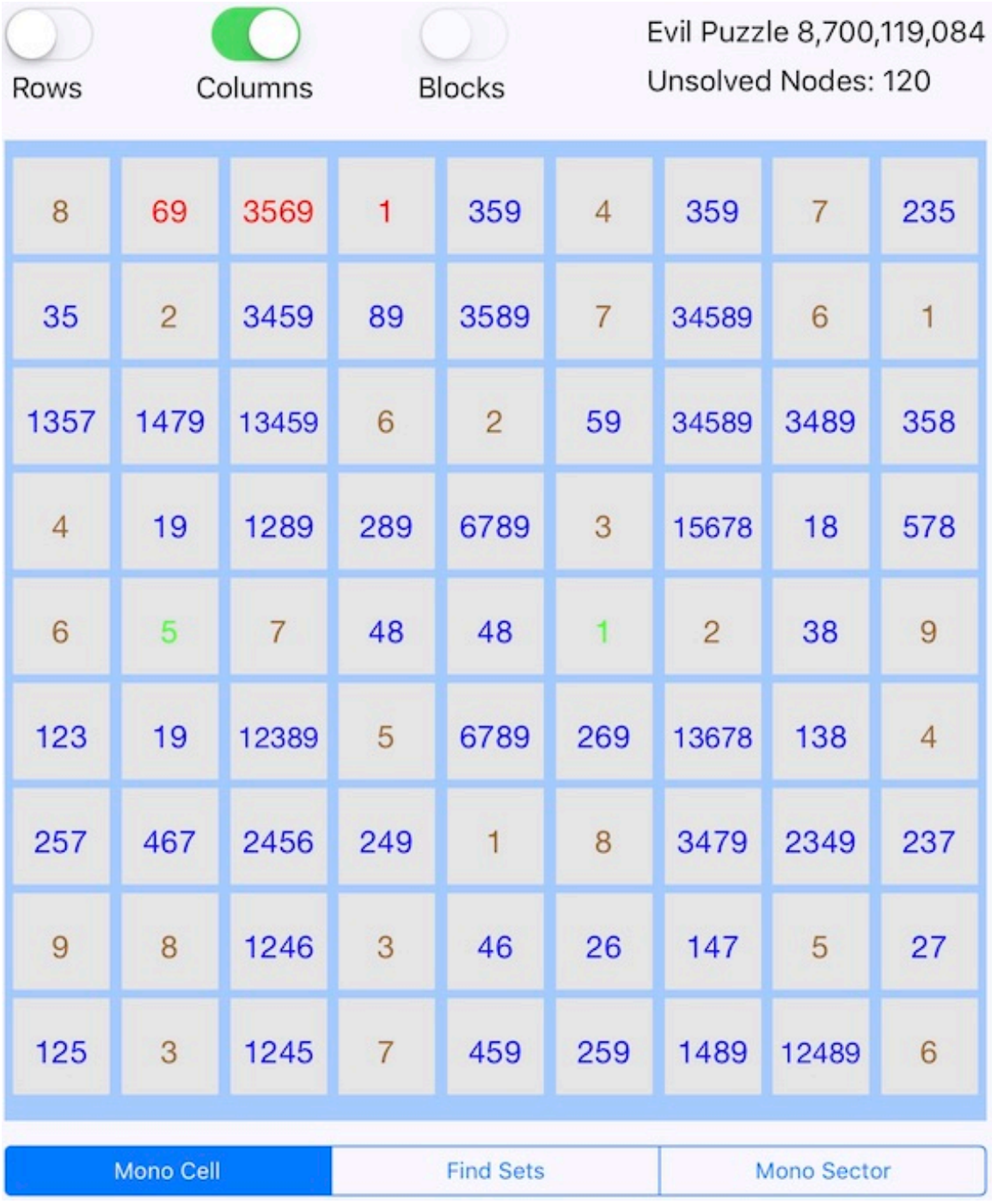


Note the Undo button is dimmed. This is because we are at initial conditions and there is nothing to undo yet.

Notice on row 5 values 1 and 5 have been 'Solved' and are therefore green. Just from initial conditions these 2 values were solved.

Mono Cell Algorithm

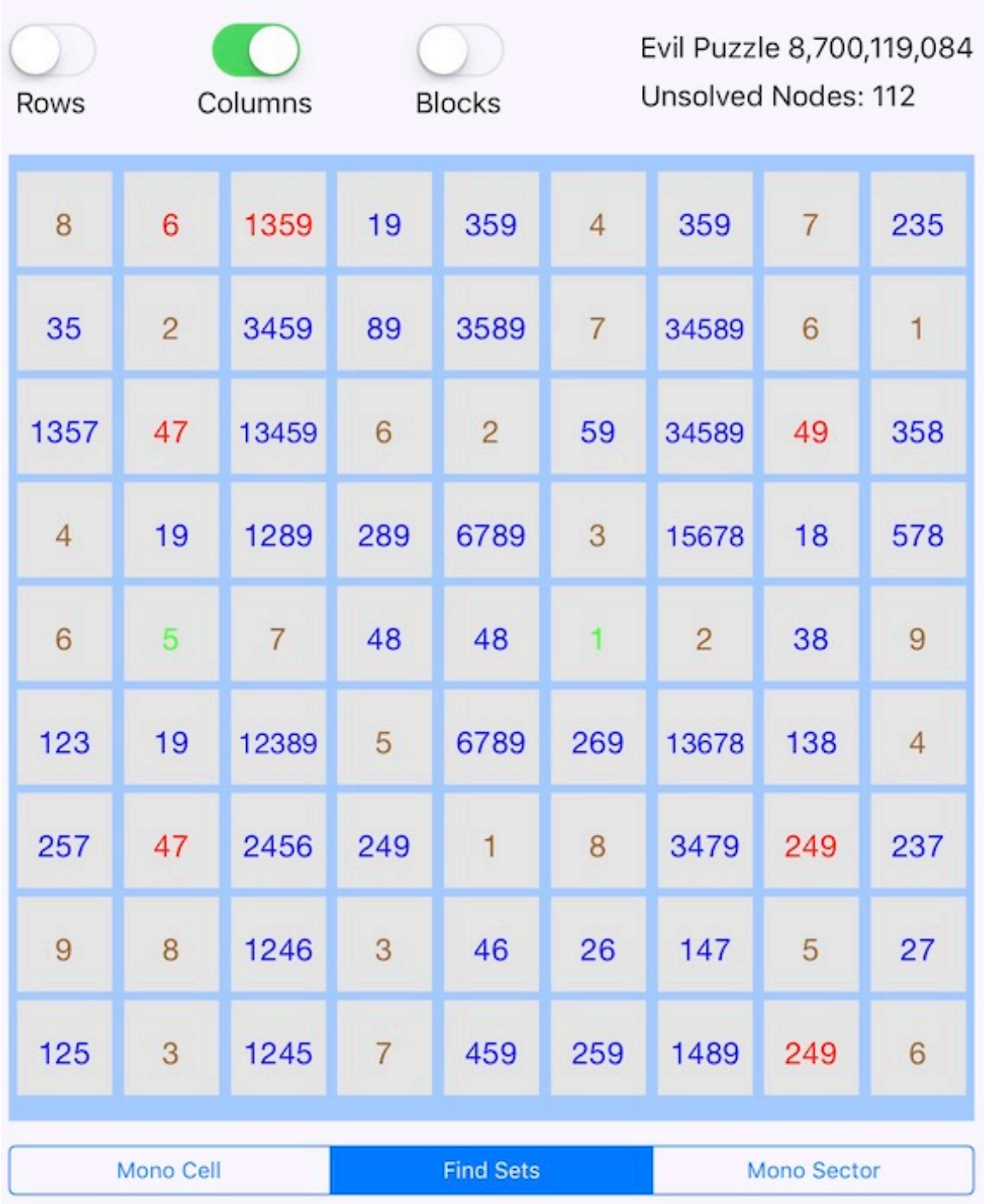
The first solve operation to be described is 'Mono Cell' and it can be performed on rows and columns. It could also be applied to blocks but running the algorithm on rows and columns will find all there is to find using this approach. It checks on each row for an occurrence of a value being found in only one cell. If such a cells are found then those cells are considered Solved at this point.



Here on column 4 the value 1 was solved and then row one columns 2 and 2 were pruned having the 1 removed from their data set.

The color red indicates that the values in that cell have changed during the last Solve operation. Typically you would perform this operation on both rows and columns at the same time and then proceed to the next Solve operation. Often, simply reapplying the same algorithm will continue to make progress. But at some point this may stop and another algorithm will need to be applied.

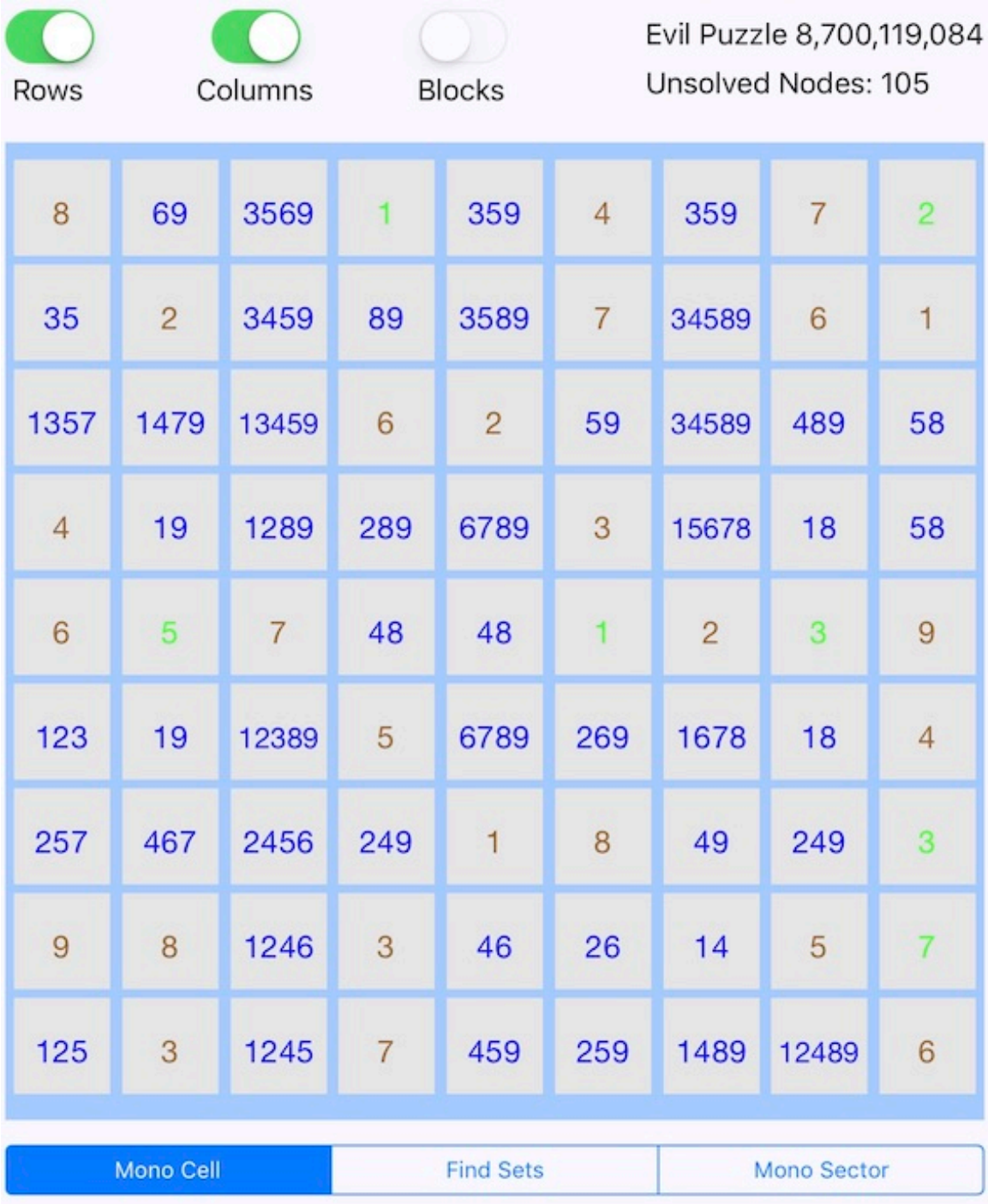
Find Sets Algorithm



The 'Find Set' algorithm is by far the most powerful method. It can be applied to rows, columns, and blocks. Using columns as an example and the initial puzzle data we apply the Find Sets algorithm. On column 2 it finds 2 cells that contain the set of { 1 9 }. Those 2 cells contain the solutions for values 1 and 9. It is either 1 then 9 or 9 then 1. Either way, values 1 and 9 can be removed from all other cells on that column. This solves for the value 6 and also allows for pruning in row 3 and 7. Further pruning occurs on column 3. The set { 1 3 8 } is found in 3 cells in column 8 which allows for pruning in that column.

Mono Sector Algorithm

The 'Mono Sector' algorithm can be applied to rows and columns. To see how this works it is best to first run the Mono Cell on both rows and columns repeatedly until the Unsolved Nodes value doesn't change. Doing this on our initial data set gives us the puzzle below.



Each row has 3 sectors. A sector contains 3 consecutive columns that reside in the same block. For example, rows 1, 2, and 3 have 3 cells in block 1 (columns 1, 2, and 3), 3 cells in block 2 (columns 4, 5, and 6) and 3 cells in block 3 (columns 7, 8, and 9).

After applying Mono Sector to rows we have the result below.

Looking at row 2 column 7 we see that the value 8 has been removed. This was possible because the algorithm noticed that in block 2 the value 8 must be on row 2. It must be on row 2 because in block 2 it's not on row 1 or 3. Therefore, it can't be in the first or last sector of row 2, as it must lie in the middle sector of row 2.

Let's look at the last row where the value 5 was removed from the first sector of row 9. This was possible because for block 8 the value 5 must be in row 9 (middle

sector). Since we know that the value 5 must reside in either cell[9,5] or cell[9,6] we can prune 5 from cell[9,1] and cell[9,3].

Puzzle Solved

There are many different paths to solve most puzzles. Here is our puzzle solved with one application of Find Sets for rows, columns, and blocks using the data set from the previous solve operation (Mono Sector- rows).

Written By: Matthew Homer Last Modified: 03-03-2018

Homer Labs

San Jose, CA