Artificial Sudoku Player

Why a “Learnable RL model”?

Imagine a young kid who wants to solve a sudoku and who only knows the basic rules of Sudoku:

- 9 different numbers in each of the 9 lines,
- 9 different numbers in each of the 9 columns,
- 9 different numbers in each of the 9 blocks.

How can he learn on his own without having a parent looking over his shoulder and telling him "that's good" when a number is right and worse telling him "that's wrong" when a number is misplaced, or even worse, who solves the sudoku instead of the child?

At the beginning, the kid will apply the basic rule to the letter and will search for singles by proceeding randomly by row, by column and by block, without searching by cell.

Progressively, he will proceed by block and more particularly will search at the intersections of blocks and rows and columns, but still not by cell.

Searching by cell is not systematic for a beginner because it is not explicitly specified in the basic rule.

At this stage of the resolution, after having found some or better all of the singles, he will search for the single “candidates” proceeding by block and their intersections with the rows and columns. The candidates located in the intersections are grouped by 2 or 3.

Instinctively the child will tend to guess the most likely candidate among the 2 or 3 candidates, but this guessing process will not take him very far in the resolution!

He will also proceed by “Trial and Error”, but that will not go very far either.

He can memorize the location and numbers of the candidates but to make the task easier he will probably start marking the candidates.

The purpose of pencil-marks is not only to indicate the potential singles but also to indicate that there cannot be other candidates neither in the same row as the marks, nor in the same column, nor in the common block.

And so on …

At this stage, the kid has already applied two solving techniques: the singles spotting rule and the row/column-block interaction rule.

Just as an individual is able of learning to solve a sudoku puzzle by himself, it must be possible to design an AI model other than a supervised learning model.

Functional Bases

The AI player must mimic the solving process of a sudoku puzzle that an experienced human player apply on paper, i.e. without any other visual aid but the grid and the pencil-marks, and certainly not with the aid of any computer solver. His only tools are a pencil and an eraser.

At first sight a sudoku puzzle is made of cells, filled-in or not, grouped in rows, columns and blocks, but for solving that puzzle a human player unconsciously and logically proceeds by sector of cells and more precisely by intersection of sectors.

A sector of a cell includes the cell itself and the 20 other cells that are either in the same row or in the same column or in the same block.

Unconsciously the thought process of a human player takes place in the candidates' state space. That space contains for an empty grid 81 invisible clones per each of the 9 different candidate digits. For a human player that space is never comprehensively visible hence the pencil-marks.

There are basicaly only three types of "common sense" action that a player takes step-by-step in that solving process:

1. Fill in an empty cell with the correct digit.
2. Pencil-mark (mentally and/or physically) one or more "potential" entries and/or one or more participating candidates that are key for solving the puzzle.
3. Erase (mentally and/or physically) one or more pencil-marks.

Environment

If the 9 different blocks were not highlighted on a grid, visually a sudoku puzzle would be a simple set of 81 cells arranged in a grid of 9 rows by 9 columns.

The non-linear row/column-block relationship would not be taken into account when representing the candidates' state space with a 9x9x9 binary array of coordinates (candidate, row, column).

The candidates' state space could be represented by a 9x9x9x9 binary array of coordinates (candidate, row, column, block.) Even so, that's a non-linear state space.

A binary number alone do not determine the proper candidate's state. Actually a candidate's state is a function of all its clones and the other candidates that are present in its cell's sector.

The candidates' state is best represented by a 4-component vector :

- [0] - total of the candidates in the candidate's cell,
- [1] - total of the candidate's clones in its cell's row,
- [2] - total of the candidate's clones in its cell's column,
- [3] - total of the candidate's clones in its cell's block.

The state space could then be represented by a 9x81x4 integer array of dimensions (candidate,cell, state vector).

Notice that is a functional state space.

Graphs

The pencil-marks that a human player pens make a pattern that is typical of a particular solving technique.

- A single pencil-mark corresponds to a definitive entry of a digit in a cell.
- Two or 3 pencil-marks correspond to a pattern typical of the row/column-block interaction solving technique.
- Four pencil-marks correspond to either a naked pair or a hidden pair solving technique.

Obviously for more advanced solving techniques there may be more than 4-pencilmark patterns, e.g. a triplet (3 candidates in 3 different cells) requires 7, 8 or 9 pencil-marks.

Those patterns and pencil-marks constitute graphs and nodes.

The graphs are bi-directed and circular, and starting from the 4-node graphs most of these are alternating graphs.

Circular means that if the correct order of a 4-node graph is 1-2-3-4-1, there may be no crossing or shortcut such that the order would be 1-2-4-3-1 or 1-3-2-4-1.

Adjacent nodes in the graph are linked by their respective location in the grid and their respective state.

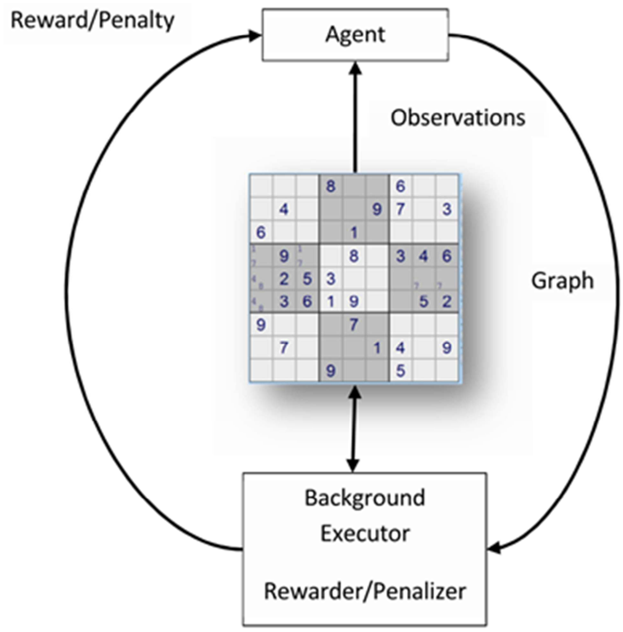
Alternating means that by traversing the graph, the links between successive nodes alternate from strong link/weak link to weak link/strong link and so on.

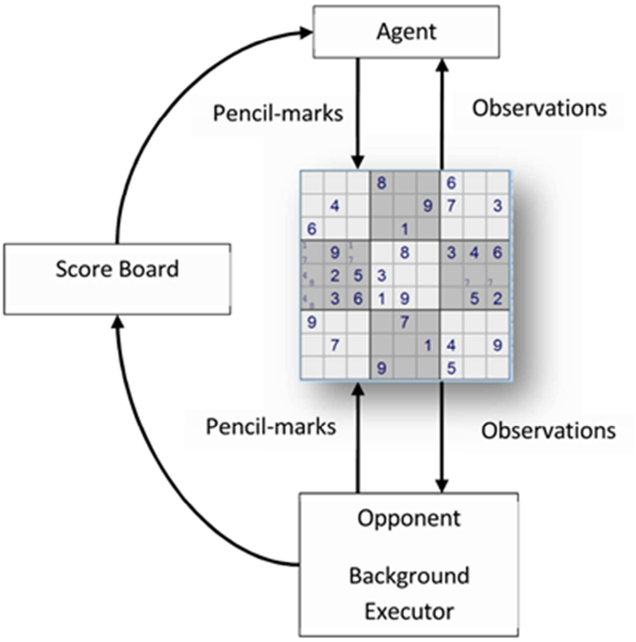
In Sudoku, they have defined two different links: the Strong Link and the Weak Link.

Very simply, a strong link is when two candidates are mutually exclusive, while in a weak one they are not.

The underlying goal of this competition is to develop an 'Experienced Graph Builder' which corresponds to a policy network in the RL domain.

So far, I have imagined two conceptual learning models: a classic RL loop and a 2-player game.





Rules

Less pre-processing is preferred to avoid ending up with a programmed model instead of a learnable one.

The codes of the algorithms of the solving techniques will not be implemented.

Hint

The link between adjacent nodes in a graph is determined by their relative locations in the grid and one or more equalities of the components of their state vectors.