Solving Sudoku with MATLAB

Raluca Marinescu Andrea Garcia Ivan Castro
Eduard Paul Enoiu
Mälardalen University, Västerås, Sweden
{rmu09001, aga09001, ico09002, eeu09001}@student.mdh.se

March 13, 2011

Abstract

Contents

1 Introduction **3**

2 Related Works **3**

3 Implementation **4**

3.1 GUI Design 4

3.2 An Algorithm for Solving Sudoku Puzzles 4

4 Experimental Results **5**

5 Conclusions **5**

1 Introduction

Games and Puzzles have been a platform for the application of mathematics, artificial intelligence and other fields techniques. The past years have brought a very popular puzzle called Sudoku. The typical Sudoku puzzle grid is a 9-by-9 cells into nine 3-by-3 squares. The rules of the game are: Every row, column, and square (of 3-by-3) must be filed with each of the numbers 1 till 9 and that number cannot appear more than once in any of the row, column, or square. For more general information on Sudoku the reader can acces this website[1].

Sudoku has led other researchers to some advances in algorithm design and implementation. This work was largely motivated by the interesting mathematical concepts behind it.

This paper will present a MATLAB implementation of a Sudoku Solver by using GUI environment. The remainder of this paper will cover some research work concering the solving of Sudoku puzzles, the algorithm and GUI implementation, continuing with some experimental results before concluding the paper.

2 Related Works

Sudoku puzzles can be viewed as a interesting problem for different fields of mathematics, computer science, artificial intelligence, physics, and others. There are many researchers from these fields who have proposed algorithms to solve puzzles in many different ways.

In [5] the authors proposes a search based solution in order to solve Sudoku puzzles by using some heuristic in a modified steepest hill ascent. Some researchers, as found here [6], are suggesting the design of a genetic algorithm by representing the puzzle as a block of chromosomes, more precise as an array of 81 integers. Any crossover appears between the 3x3 grids and any mutations occur only inside the 3x3 grids. From the experiments done in [6] the algorithm based on genetic algorithms performs well though is does not solve all cases. Geem in [3] proposes a model for a Sudoku solver based on harmony search that mimics the characteristics of a musician. As mentioned in [4] the performance of the algorithms is not that good and it solves puzzles in less than 40 seconds and in less than 300 iterations. Santos-Garcia and

[1]http://goo.gl/IIf0r

Palomino suggests here [8] a method for solving Sudoku puzzles using simple logic with rewriting rules to mimic human intelligence. This method works but, as the author admits, it is not performing well when compared with other techniques. Others like [9] are suggesting neural networks by modelling an energy driven quantum neuron (Q'tron) neural network to solve the Sudoku puzzles.

In [1], Barlett and Langville proposes a solution based on binary integer linear programming (BILP). To formulate in a simple way the method we can say that it uses binary variables to pick a digit for a cell in Sudoku puzzle. The model in [1] is solved in MATLAB and it works very fast (16 sec. per second).

In [7], Russell Norvig is suggesting a solution based on backtracking. The algorithm is a combination of constraint propagation and direct search by assigning a value to a square on the grid and then propagating these constraints to other squares.

We have decided to solve the puzzles using backtracking as in [7] because of the simplicity and performance mentioned by Norvig in his paper. We will present in the next section our own version of the algorithm using recursive backtracking and constraint propagation implemented in MATLAB.

3 Implementation

3.1 GUI Design

3.2 An Algorithm for Solving Sudoku Puzzles

The algorithm was implemented in MATLAB using direct search and constraint propagation. This constraint propagation was implemented in such a manner that:

- When a value is assigned to a square that same value cannot be used as a possible assignment in all related squares
- If a square has only one single value for possible assignment, that value is immediately assigned.

Our MATLAB program has only 5 steps:

- Find all the possible values for all empty cells

- Assign a value to a square on the grid
- Propagate constraints to other square
- If all the cells have more than one possible value we fill in a tentative value for one cell

This algorithm, though it is simplistic, it should by the nature of the algorithm class perform in a fairly way. This algorithm performs in up left to right and up to down fashion.

To see how our program works, we will use a simpler 4-by-4 grid with 2-by-2 blocks. As mentioned in [2] these kinds of puzzles are called Shidoku (”Shi” means ”four” in Japanese). Figure 1(a) shows a Shidoku puzzle. Figure 1(b) through 1(f) shows how we get to a solution by using our algorithm. In Figure 1(b), the possible assignment, are represented as smaller digits. For example in Figure 1(b) line 1, row 1, we have two possible values to choose from. If a cell contains only one candidate is filled immediately.

In the next section we are presenting some testing results under different puzzles and we will describe how our MATLAB behaves to various situations.

4 Experimental Results

5 Conclusions

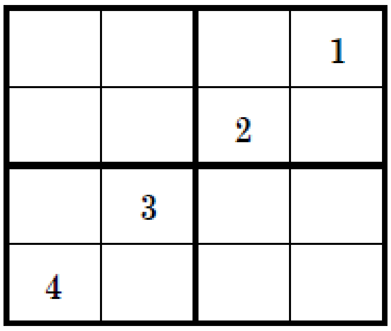
References

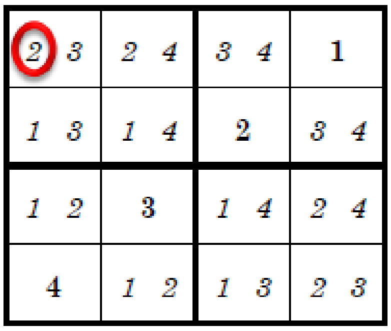
[1] A.C. Bartlett and A.N. Langville. An integer programming model for the Sudoku problem. *Preprint, available at http://www. cofc. edu/˜ langvillea/Sudoku/sudoku2. pdf*, 2006.

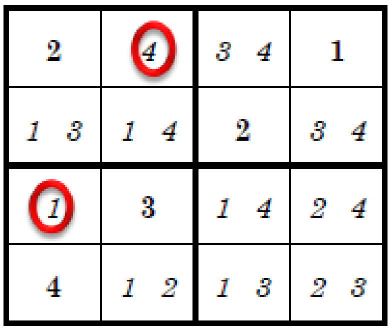
[2] JF Crook. A pencil-and-paper algorithm for solving Sudoku puzzles. *Notices of the AMS*, 56(4):460–468, 2009.

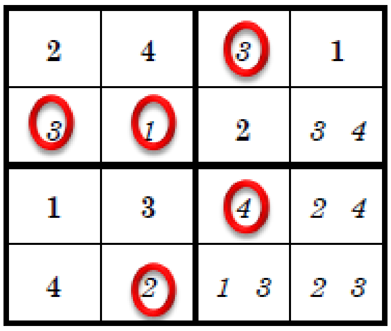
[3] Z. Geem. Harmony search applications in industry. *Soft Computing Applications in Industry*, pages 117–134, 2008.

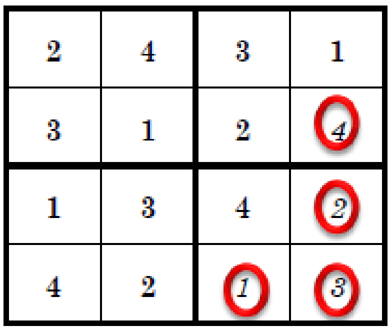
[4] R.C. Green II. Survey of the Applications of Artificial Intelligence Techniques to the Sudoku Puzzle. 2009.


(a) A Shidoku puzzle


(b) The possible candidates for all squares


(c) Inserting value 2 and constraint propagation


(d) Constraint propagation


(e) Insert the remaining values to complete the puzzle

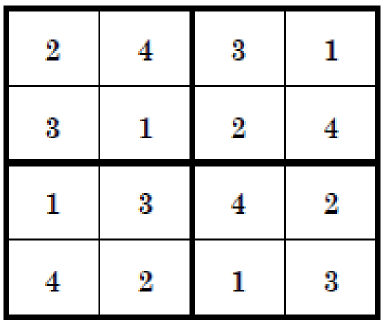

(f) The puzzle completed

Figure 1: Algorithm Exemplification by using a simple example

[5] SK Jones, PA Roach, and S. Perkins. Construction of heuristics for a

search-based approach to solving Sudoku. *Research and Development in Intelligent Systems XXIV*, pages 37–49, 2008.

[6] T. Mantere and J. Koljonen. Solving, rating and generating Sudoku puzzles with GA. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1382–1389. Ieee.

[7] P. Norvig. Solving every sudoku puzzle. *Preprint.*

[8] G. Santos-Garcia and M. Palomino. Solving Sudoku puzzles with rewriting rules. *Electronic Notes in Theoretical Computer Science*, 176(4):79–93, 2007.

[9] T.W. Yue and Z.C. Lee. Sudoku Solver by Q'tron Neural Networks. *Intelligent Computing*, pages 943–952, 2006.