**Sudoku Solver Project**

**Abbreviations and Definitions:** In this essay we have tried to use the same terminology, which is commonly used in other journals and research papers. In the following paragraph, there is a brief description of some the abbreviations and definitions that are used in the text.

* **Sudoku:** is a logic-based, combinatorial number placement puzzle [4]. The word “Sudoku” is short for ***Su-ji wa dokushin ni kagiru*** (in Japanese), which means ***“the numbers must be single”***.
* **Box (Region, Block):** A region is a 3x3 box like the one. There are 9 regions in a traditional Sudoku puzzle.
* **Cell (Square):** is used to define the minimum unit of the Sudoku board.
* **Candidates:** the number of possible values that can be placed into an empty square.
* **Clues:** the given numbers in the grid at the beginning.
* **Grid (board):** The Sudoku board consists of a form of matrix or windows.

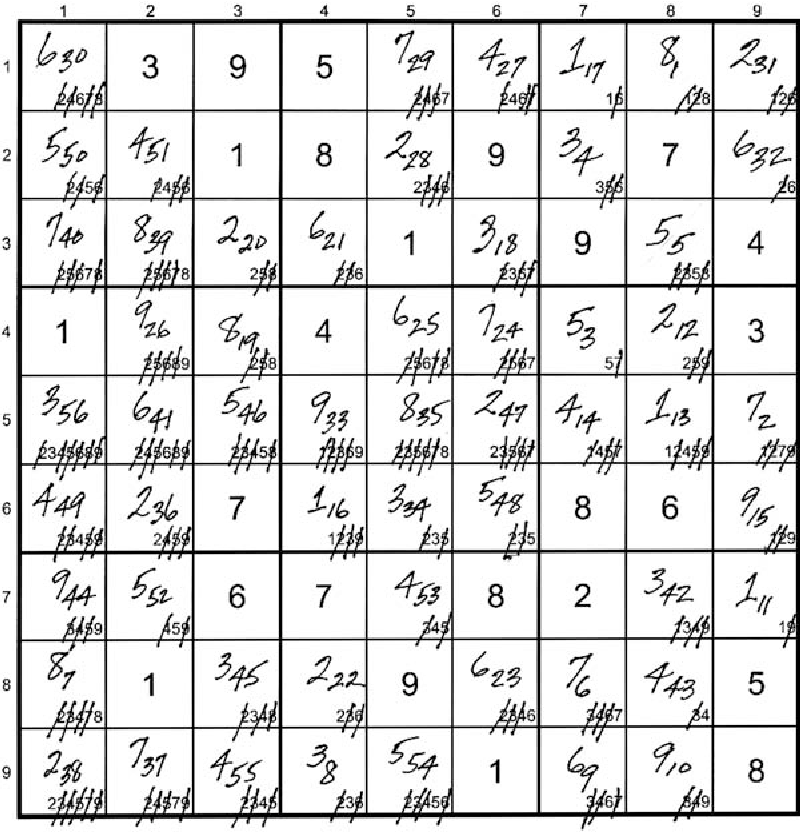
**Short About Sudoku**

Sudoku is a logic-based puzzle that is played by numbers from 1 to 9. The Puzzle first appeared in newspapers in November 1892 in France and then Howard Garns an American architect presented it in its modern form [1,5]. There are already many journals, papers and essays that researched about Sudoku Solvers and most of them present different type of algorithms. Sudoku’s popularity is based on several reasons. First of all, it is fun and fascinating, and very easy to learn because of its simple rules. There are currently many different type of Sudoku puzzles, classic Sudoku that contains a 9X9 grid with given clues in various places, mini Sudoku that consists of a grid with 4X4 or 6X6 sizes. The other type of Sudoku is Mega Sudoku that contains a grid with 12X12 or 16X16 sizes [2]. In this text, the focus is mostly on the classic Sudoku, i.e. 9X9 grid. Furthermore, Sudoku has become so popular, compared to other games, all over the world because its rules are easy to understand and it can improve our brain and also it is fun. The structure of the puzzle is very simple, especially the classic puzzle. This essay is mainly focused on classic puzzle of a 9X9 grid. There already exist a number of digits in the board that make the puzzle solvable. It means that some numbers are already placed in the Sudoku board before starting playing. The board consists of 81 cells, which is divided into nine 3X3 sub boards and each 3X3 sub board is called “box” or “region”. The main concept of the game is to place numbers from 1 to 9 on a 9X9 board so that every row, column and box contains any numbers but once. This means that no number is repeated more than once. Generally, the puzzle has a unique solution. There are certain techniques to solve the puzzle by hand and these rules can be implemented into a computer program.

**Examined Algorithms**

**🡪Pencil-and-paper algorithm**

In this work, we implement a solution based on some strategies used by humans when solving the puzzle, therefore, it is called pencil-and-paper algorithm. The paper-and-pencil algorithm contains human strategies. These strategies have been examined below in more details. These techniques are almost easy to understand by human players, but it might be hard to search in the puzzle, since there are several things to look for at the same time. As there are puzzles with different types of difficulty, the easy and medium puzzles can be solved using some simple techniques such as unique missing method, naked singles. However, to solve difficult problems we may examine other techniques as well (locked candidates, naked and hidden pairs, triplets etc.)



**🡪Brute force algorithm**

Kovacs describe some of the brute force methods used for solving Sudoku puzzles [9]. The simplest method randomly produces a solution to the puzzle called “unconstrained grid”, after that the program checks whether it is a valid solution. If not, the process is repeated until a solution is found. This algorithm can be applied simply and will find a valid solution for any problems because it will go through all possibility solutions. However, this method can be time consuming but according to Kovacs the algorithm can be optimized.

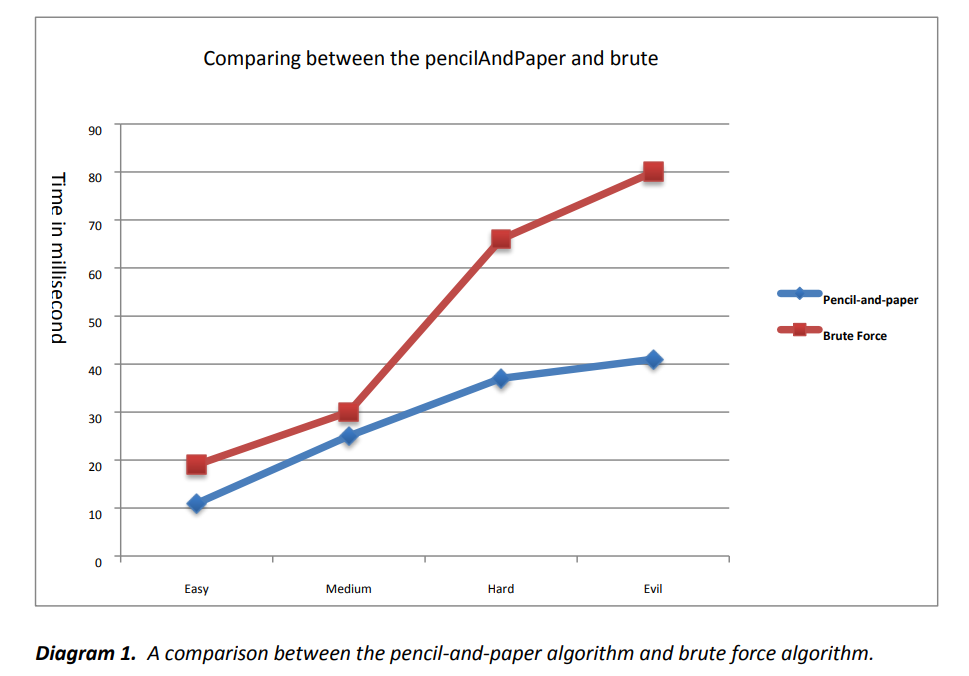
Generally, the brute force algorithm goes through the empty squares, filling in numbers from the existing choices, or removing failed choices if a “dead-end” is reached. For example, Brute force solve a puzzle by inserting the digit “1” in the first square. If the digit is allowed to be there by checking row, column and box then the program go to the next square, and put the digit “1” in that square. The program discovers that the “1” is not allowed, then the digit increments by one i.e. it has become 2. When a square is noticed where none of the digits (1 to 9) is permitted, then the program backtracks and comes back to the prior square. The value in that square increases by 1. The process is repeated until the correct digits fill all 81 squares.

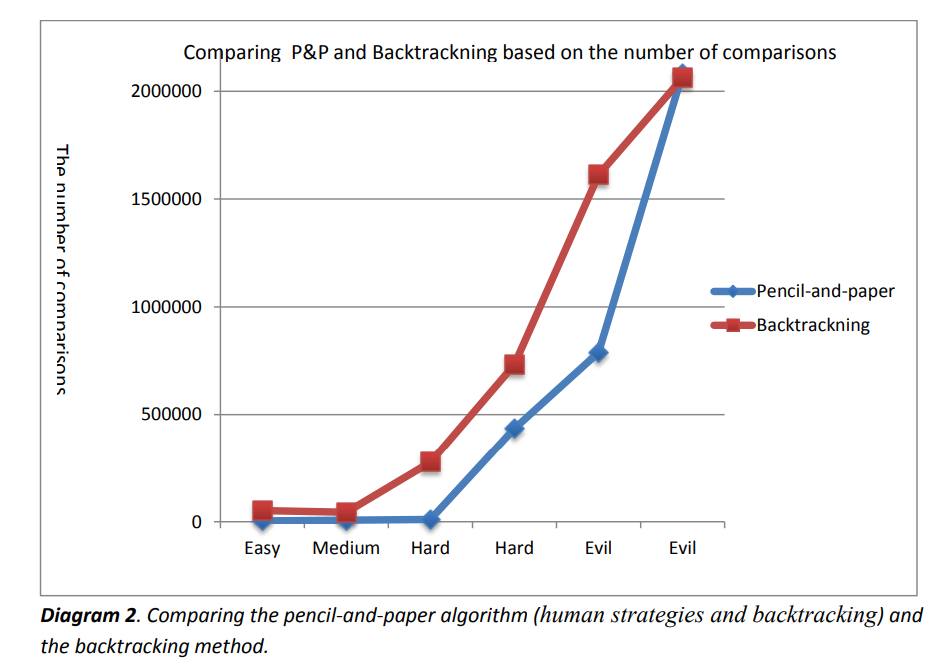
**🡪Backtracking (guessing method)**

In order to solve puzzles with even more difficult levels such as hard and evil the backtracking method has been used to complete the algorithm. A human player solves the puzzle by using simple techniques. If the puzzle is not solvable by using the techniques the player, then tries to fill the rest of the empty squares by guessing.

The backtracking method, which is similar to the human strategy (guessing), is used as a help method to the pencil-and-paper algorithm. In other words, if the puzzle cannot be filled when using the unique missing method and the naked single method, the backtracking method will take the puzzle and fill the rest of empty squares. Generally, the backtracking method find empty square and assign the lowest valid number in the square once the content of other squares in the same row, column and box are considered. However, if none of the numbers from 1 to 9 are valid in a certain square, the algorithm backtracks to the previous square, which was filled recently.

**Comparing**

In this section we present the result of the testing and also examine the differences between the pencil-and-paper algorithm and the brute force algorithm.



**Algorithms**

🡪**Pencil-And-Paper Algorithm: -**

*Pencil-and-paper (puzzle [][]){*

*puzzle [][]*

*Solve the puzzle{*

*For 1 to 10*

*Naked Single Method (puzzle[][])*

*check all rows (puzzle[][]) //check if there is only one candidate that*

*is missing, place that missing digit in the empty square*

*check all columns (puzzle[][])*

*check all boxes (puzzle[][])*

*}*

*If the puzzle is solved: print*

*Otherwise: recursiveBacktracking (puzzle[][])*

*Print the solved puzzle*

*}*

**🡪The Backtracking algorithm: -**

*recursiveBacktrackning(Puzzle[][]){*

*Puzzle [][] //global*

*solvePuzzle(row,col){*

*if (no more choices): the puzzle is solved!*

*If (puzzle[row][col]= notEmpty):*

*move to the next square.*

*for 1 to 9: if(checkRow(row,col,digit) & checkCol(row,col,digit) & checkBox(row,col,digit){*

*puzzle[row][col]= digit;*

*move to the next square*

*}*

*if not valid number is found go the previous square that was recently filled*

*}*

*}*