International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 10 (2014), pp. 985-990
© International Research Publications House
http://www. irphouse.com

Simulation for Sudoku Solving Logic

Karan Khatter[1] and Shubham Gupta[2]

[1] –Department of Electronics & Communication Engineering-ITM University
Huda Sector 23-A Gurgaon, Delhi (NCR), India
[2,] –Department of Electronics & Communication Engineering-ITM University
Karan Khatter[1]: email – karankhatter93@gmail.com

Abstract

In this paper, we have discussed the history of Sudoku solving along with the new latest software simulation techniques with which we can effectively build up strategies to solve Sudoku. Puzzles constructed from multiple Sudoku grids are common. Although the 9×9 grid with 3×3 regions is by far the most common, many other variations exist.The application and advantages of brain logic Sudoku's are also discussed very briefly in the paper.

1. Introduction

Sudoku is basically a number placement puzzle. The word Sudoku is derived from Su-ji wa dokushin ni kagiru which means "the numbers must be single" or "the digits are limited to one occurrence". The roots of the Sudoku puzzle[1] are originated in Switzerland. Publication of first real real Sudoku was in 1979 by Dell Magazine under the name 'Numbers in Place' and was invented by Howard Garns.
In this paper , the concept of Sudoku solving via matlab simulation is discussed with proper evolved coding and results and discussion. Sudoku was popularized by Japan in 1986 after it was published and given its name by the Japanese number company Nikoli . There were two innovations introduced by Nikoli : the number of givens was restricted to no more than 32, and puzzles became "symmetrical". It became an international hit in the year 2005.
The 9x9 puzzle is the most common puzzle but there are a few variants of the puzzle.

2. Variants of Sudoku Puzzle

In Mini Sudoku, basically the puzzle is played on 6x6 grid with 3x2 regions and uses the numbers 1 through 6 .while in the case of alphabetical Sudoku[2]: It is also known as Wordoku .It contains letters rather than the main word, through which we make a proper and suitable word accordingly. Another Variant is Duidoku, analysis of a two player variant of Sudoku. It is played on 4x4 board.

3. Simulation of Sudoku Solving program

To test and find out a unique solution to Sudoku without using much intelligence, we tried to focus on developing such a coding which could actually within microseconds find a possible solution to 9X9 Sudoku .The code was simulated in matlab and as part of intervention certain outcomes were achieved.
A Sudoku grid is a special type of latin grid with an advantage property of no repeated values in any of the 9 blocks in 3x3 cells. It was earlier proven that the first order formula does not mention that blocks is valid for Sudoku if and only if it is valid for Latin squares.

The number of classic 9×9 Sudoku solution grids is approximately 6.67×10^{21}. This is roughly 1.2×10^{-6} times the number of 9×9 Latin squares .The number of different solutions, when symmetries such as permutation, reflection as well as rotation [3] are taken into account was non-deterministic values , although defined in range.

However, statistical techniques combined with the definition of a new type of generator allow showing that there are approximately :

- 3.10×10^{37} minimal puzzles,
- 2.55×10^{25} non-essentially-equivalent minimal puzzles.

The general problem of solving Sudoku puzzles on $n^2 \times n^2$ boards of $n \times n$ blocks is known to be NP complete.
Coding which we performed is as follows:Sudoku Solver

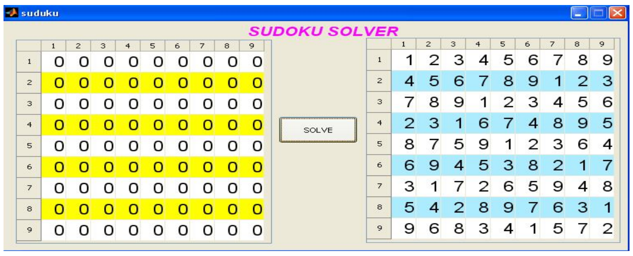
```
function varargout = suduku(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @suduku_OpeningFcn, ...
                   'gui_OutputFcn',  @suduku_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
```

```
 function D = sudoku_solver(D)

onefound = 1;
N = 0;
poss = 1:9;

while onefound
    splits = {};
    onefound = 0;
    for m = 1:9
        rowdata = nonzeros(D(m,:)');
        for n = 1:9
            E = D(m,n);
            if E ~= 0, continue, end

            coldata = nonzeros(D(:,n));
            blk = [ceil(m/3) ceil(n/3)]-1;
            blkdata = nonzeros(D(blk(1)*3+[1:3],blk(2)*3+[1:3]));

            EE = zeros(1,9);
            RCB = [rowdata; coldata; blkdata(:)];
            EE(RCB) = 1;
            Enew = find(~EE);

            if isempty(Enew)
                D = []; return;
            elseif length(Enew) == 1;
                onefound = 1;
                D(m,n) = Enew;
                rowdata = nonzeros(D(m,:)');
            else
                splits{end+1} = [m n Enew];
            end
        end
    end
end
 if isempty(splits)
    return
end

splitlength = cellfun(@length,splits);
splits = splits{find(splitlength == min(splitlength),1)};
m = splits(1); n = splits(2);

for test = 3:length(splits)
```

```
    D(m,n) = splits(test);
    D0 = sudoku_solver(D);
    if ~isempty(D0)
        D = D0;
        return
    end
end
D = [];

function suduku_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = suduku_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function uitable1_CellEditCallback(hObject, eventdata, handles)
if (eventdata.NewData < 0 || eventdata.NewData > 9)
        tableData = get(hObject, 'data');
        tableData(eventdata.Indices(1), eventdata.Indices(2)) = eventdata.PreviousData;
        set(hObject, 'data', tableData);
end

function pbsolve_Callback(hObject, eventdata, handles)
val = get(handles.uitable1,'Data');
val2 = sudoku_solver(val);
if isempty(val2)
    val2 = cell(9);
end

set(handles.uitableAns1,'Data',val2);

function uitable1_CreateFcn(hObject, eventdata, handles)
set(hObject,'Data',zeros(9,9));
```

4. Results and Discussion

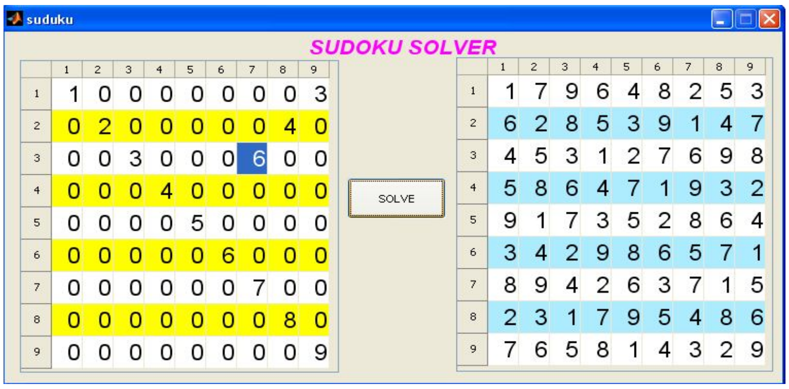
After the coding was implemented on matlab software , further , on simulation we obtained the following results which are defined below in the figures A random 9x9 grid of random no. from 1 to 9 is generated on insertion of 0 at all 81 coordinate places. Another Sudoku Solver emerges as a result to 9x9 grid when we inserted diagonally 1 to 9 in grid and also 1 to 3 row elements 3,4,6 , and as a result random 9x9 solved sudoku emerged .



	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	4	5	6	7	8	9	1	2	3
3	7	8	9	1	2	3	4	5	6
4	2	3	1	6	7	4	8	9	5
5	8	7	5	9	1	2	3	6	4
6	6	9	4	5	3	8	2	1	7
7	3	1	7	2	6	5	9	4	8
8	5	4	2	8	9	7	6	3	1
9	9	6	8	3	4	1	5	7	2

Figure 1: The illustration shows the ideal simulated result with 0’s at all positions.



	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	3
2	0	2	0	0	0	0	0	4	0
3	0	0	3	0	0	0	6	0	0
4	0	0	0	4	0	0	0	0	0
5	0	0	0	0	5	0	0	0	0
6	0	0	0	0	0	6	0	0	0
7	0	0	0	0	0	0	7	0	0
8	0	0	0	0	0	0	0	8	0
9	0	0	0	0	0	0	0	0	9

	1	2	3	4	5	6	7	8	9
1	1	7	9	6	4	8	2	5	3
2	6	2	8	5	3	9	1	4	7
3	4	5	3	1	2	7	6	9	8
4	5	8	6	4	7	1	9	3	2
5	9	1	7	3	5	2	8	6	4
6	3	4	2	9	8	6	5	7	1
7	8	9	4	2	6	3	7	1	5
8	2	3	1	7	9	5	4	8	6
9	7	6	5	8	1	4	3	2	9

Figure 2 : The illustration shows the ideal simulated result with artificial human intelligence

5. Conclusions

Conventional Strategy for Sudoku solving has been discussed in the paper along with a brief history. More on aspect of solving via matlab approach and studying the results and outcomes. The paper also focuses on the aspect of the capacity of providing sensitivity, throughput and flexibility as potential aspects for Sudoku solving, as a further scope certain reforms in the coding are being tried to get across a 10x10 grid Sudoku with 0 to 9 unique digits.

References

[1] Berthier, Denis (2007). The Hidden Logic of Sudoku. pp. 76
[2] Devlin, Keith (January 28–29, 2012). "The Numbers Game ". The Wall Street Journal (Weekend Edition). pp. C5.
[3] Berthier, Denis (December 4, 2009). "Unbiased Statistics of a CSP – A Controlled-Bias Generator". In Elleithy, Khaled. Innovations in Computing Sciences and Software Engineering. pp. 165-170.