ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/303553939

Recursive Backtracking for Solving 9*9 Sudoku Puzzle

Article *in* Bonfring International Journal of Data Mining · January 2016

DOI: 10.9756/BIJDM.8128

CITATIONS
2

READS
2,064

2 authors:

Dhanya Job
Mahatma Gandhi University
3 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Varghese Paul
Rajagiri School of Engineering and Technology
73 PUBLICATIONS **271** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Machine Learning for Data Analytics View project

Security Enhancement of English Text View project

All content following this page was uploaded by Varghese Paul on 14 December 2017.

The user has requested enhancement of the downloaded file.

Recursive Backtracking for Solving 9*9 Sudoku Puzzle

Dhanya Job and Varghese Paul

Abstract--- *Nowadays Sudoku is a very popular game throughout the world and it appears in different medias, including websites, newspapers and books. There are numerous methods or algorithms to find Sudoku solutions and Sudoku generating algorithms. This paper explains possible number of valid grids in a 9*9 sudoku and developed a programming approach for solving a 9*9 sudoku puzzle and the results have been analysed in accordance with various number of clues for 9*9 sudoku.*

Keywords--- *Back Tracking, Enumeration, Pseudocode, Sudoku Puzzle.*

I. INTRODUCTION

SUDOKU Puzzle was invented by American Howard Garns in 1979 .[1] In the year 1984 Maki Kaji of Japan has published in the magazine of his puzzle company 'Nikoli' and gave the name Sudoku to the puzzle game, which means "Single Numbers". The puzzle became popular in Japan and New Zealander Wayne Gould, wrote a computer program that would generate Sudokus. It has become a regular puzzle game in many leading newspapers and magazines around the world and is enjoyed by the people globally. The name Sudo is derived from 'Sujiwadokushionikagiru', which is the Japanese word means the digits must remain single.

II. THEORETICAL BACKGROUND

A. Sudoku Defenition

A Sudoku consists of a 9×9 square grid containing 81 cells.[2]The grid is subdivided into nine 3×3 blocks. Some of the 81 cells are filled in with numbers from {1,2,3,4,5,6,7,8,9 }. These filled-in cells are called givens or clues. The goal of the player is to fill in the whole grid using the nine digits so that each row, column and block contains each number exactly once and this constraint on the rows, columns, and blocks is known as One Rule. The solution of a Sudoku Puzzle requires that every row, column and block contain all the numbers in the set[1,2,….9] and every cell will be occupied by only one number.

Most frequently a 9×9 grid made up of 3×3 sub grids, starting with several numerals given in some of the cells ("givens"). Each row, column, and region or Block must contain only one instance of each numeral. Fig 1is an example for solved Sudoku. A unique solution exists for a Sudoku Puzzle which can be determined by solving for all possible solutions.

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

Fig. 1: Solution to Sample Sudoku Puzzle

III. COMPUTATIONAL PERSPECTIVE

The background gives an introduction to the basic structure of Sudoku.[3]Some commonly used algorithms are backtracking, rule-based and Boltzmann machine.

Backtracking method is probably the most basic Sudoku game solving strategy for computer algorithms. This algorithm is a brute-force method which tries different numbers, and if it fails it backtracks and tries a different number. In this paper Backtracking method is used to solve the puzzle.

Rule Based Method, this method[5] uses several rules that logically proven, that a square either must have a certain number or roles out numbers that are impossible. This method is very similar to how humans solve Sudoku and the rules used are in fact derived from human solving methods.

The Boltzmann machine algorithm models Sudoku by using a constraint solving artificial neural network problems. Puzzles are seen as constraint or limits that describe which node that cannot be connected to each other. These constraints are encoded into weights of an artificial neural network and then they are solved until a valid solution appears, with active nodes indicating chosen the appropriate digits[10]. This algorithm is a stochastic algorithm in contrast to the other two algorithms.

IV. ENUMERATING SUDOKU GRIDS

FelgenhauerandJarvis [6] enumerated total number of valid Sudoku solutions. The mathematical derivation can be shown as follows.

Consider a 9* 9 sudoku with 9 blocks B1,B2,….B9.

The Block representation is shown in Table 2.1.

Table 2.1: Block Representation

B1	B2	B3
B4	B5	B6
B7	B8	B9

Dhanya Job, Department of Computer Science, Santhigiri College, Thodupuzha, India. E-mail:dhanyajob@gmail.com

Varghese Paul, Department of Information Technology, Cochin University, Cochin, Kerala. E-mail:vp.itcusat@gmail.com

DOI: 10.9756/BIJDM.8128

Take the top left Block which is B_1 in the standard form. So we can arrange numbers from 1 to 9 in 9! ways. So N_1 will be N_0/9!.Possibilities for B2 and B3. Consider the top row of Sudoku grid other than the first block. The possible ways for arranging the numbers are Given below.

Table 2.2: Solutions for First Row

(4,5,6)	(7,8,9)
(4,5,7)	(6,8,9)
(4,5,8)	(6,7,9)
(4,5,9)	(6,7,8)
(4,6,7)	(5,8,9)
(4,6,8)	(5,7,9)
(4,6,9)	(5,7,8)
(5,6,7)	(4,8,9)
(5,6,8)	(4,7,9)
(5,6,9)	(4,7,8)

Since the first Block B_1 is in standard form, as per the solution of table 2.1 the top row of Sudoku Blocks B_2 and B_3 can be filled as follows

Table 2.3: Solution for First Row when B1 is in Standard Form

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

Three set of numbers written in 3! Ways. Same number of combination is possible for its reversal that is (7,8,9) and (4,5,6).

In total we have $2*(3!)^6+18*3*(3!)^6$

$=56*(3!)^6$

ie,2612736 possible combinations and Number of possibilities for top three rows of Sudoku grid= 9!* 2612736 = 948109639680.Same will be true for the Blocks B_4 to B_6 and B_7 to B_9.Number of Sudoku grids=$(948109639680)^6 / (9!)^9$ which is equal to $6.657*10^{21}$

V. Solving Sudoku

One of the Popular solution for Sudoku game is based on backtracking and dancing links algorithms. Backtracking, is a common technique in artificial intelligence, is a brute force search technique that explores a constrained set of possibilities until solution is reached[7].

Backtracking is a systematic method to iterate through all the possible arrangements of a search space[8]. It is a general method or technique which must be customized for single application. In the general case, we will model our solution as a vector x = (x1; x2; :::; xn), where each element ai is selected from a finite ordered set Si. Such a vector might represent an arrangement where xi contains the ith element of the permutation. [9]Or the vector might represent a given subset S, where xi is true if and only if the ith element of the set is in S.

PseudoCode for BackTracking

Pseudo code for solving Sudoku using Backtracking is given below.

```
bool Solve(configuration conf)
{
if (no more choices) // BASE CASE
return (conf is goal state);
for (all available choices) {
try one choice c;
 // solve from this point,, if works out, you're done.
if (Solve(conf with choice c made)) return true;
unmake choice c;
 } return false; // tried all choices and no solution is  found
}
```

VI. Implementation

The above defined Pseudo Code have been used to solve a 9*9 sudoku in JAVA.

```
boolSolveSudoku(Grid<int>&grid)
{
int row, col;
if (!FindUnassignedLocn(grid, row, col))
return true; // all locations successfully assigned.
for (intnum = 1; num<= 9; num++) { // options are 1-9
if (NoConflicts(grid, row, col, num)) { // if # looks ok
grid(row, col) = num; // try assign #
if (SolveSudoku(grid)) return true; // recur if succeed stop
grid(row, col) = UNASSIGNED; // undo & try again
}
}
return false; // this triggers backtracking from earlier decisions}
```

VII. Analysis

To get an idea of how back tracking algorithm performs it is suitable to plot solving times. The backtrack algorithm is an efficient algorithm in its performance.The backtrack algorithm was tested on 30 puzzles with different number of clues in the time limit of 20 seconds.

In figure 2 the Number of clues is plotted against the Solving time. From the graph, it can be seen that time for solving seems to decrease at approximately exponential rate as number of clues increases. It can also be observed that the solving times is around time limit of 20s. This probably means that the time for solving would have continued to increase for the other unsolved puzzles.

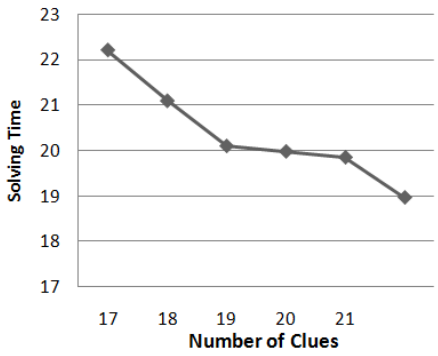

Fig. 6.1

Table 6.1. A plot showing the results of the backtracking approach. Note that the y-axis represents solving times and the x-axis represents number of clues.

Table 6.1: Experimental Results

Number of Puzzles	5	5	5	5	5	5
Number of Clues	17	18	19	20	21	22
Solving Time	22.2	21.1	20.1	19.98	19.85	18.96

Among the 30 puzzles we have taken for doing the experiment, it have been found out the average solving time is 20.365 s.

VIII. Conclusion and Future Work

In this paper we used backtracking algorithm for solving Sudoku puzzles with different number of clues. e have implemented the algorithm using java and also compared the results.The results reveals that the application program developed by us performs well for solving 30 puzzles of size 9*9 with various clues. Future work includes studying neural network and develop an algorithm using Neural network to solve a 9*9 sudoku puzzle. Overall there is space for larger studies with more algorithms for Solving Sudoku Puzzles of other sizes.Also implementation of Sudoku Puzzles in Data security remains as a future work.

Reference

[1] Hung-Hsuan Lin, I-ChenWu, "An Efficient Approach To Solving The Minimum Sudoku Problem", ICGA Journal, Vol. 34, No. 4, Pp. 191–208, 2011.

[2] Neil Robertson, Daniel P. Sanders, Paul Seymour, Robin Thomas, The Four-Colour Theorem, J. Combin. Theory Ser. B 70(1):2–44, DOI: 10.1006/jctb.1997.1750, 1997.

[3] Thomas C. Hales, A proof of the Kepler conjecture, Annals of Mathematics. Second Series 162(3): 1065–1185, doi:10.4007/annals. 2005.162.1065, 2005.

[4] Harald A. Helfgott, Major arcs for Goldbach's theorem, http:// arxiv.org/ abs/1305.2897, 2013.Computing, Vitoria, Espirito Santo, Brazil, Pp: 75–81, 2002.

[5] Hung-Hsuan Lin, I- Chen Wu, "Solving the minimum Sudoku Problem", TAAI, International Conference on Technologies and Applications of Artificial Intelligence Pp. 456-461, 2010

[6] Bertram Felgenhauer, Frazer Jarvis, Mathematics of sudoku I, Mathematical Spectrum, Vol. 39, No. 1, Pp. 15–22, 2006

[7] Faisal Abu-Khzam, "Kernelization Algorithms for d-Hitting Set Problems", in Proceedings of the 10th Workshop on Algorithms and Data Structures (WADS 2007), LNCS, Vol. 4619, Pp. 434–445.

[8] Leonid Khachiyan, EndreBoros, KhaledElbassioni, Vladimir Gurvich, "A New Algorithm for the Hypergraph Transversal Problem, in Computing and Combinatorics", 11th Annual International Conference, COCOON 2005, LNCS, Vol. 3595, Pp. 767–77.

[9] Rolf Niedermeier, Peter Rossmanith, "An efficient fixed-parameter algorithm for 3-Hitting Set", Journal of Discrete Algorithms 1, Pp. 89-102, 2003.

[10] Jean-Paul Delahaye, "The Science behind Sudoku, Scientific American", Vol. 294, No. 6, Pp. 80–87, 2006.

View publication stats