AI Algorithms for Sudoku

- Priya Sahu (B20AI031)
- Tanmay Agrawal (M22MA011)
- Poonam Suthar (B20ES007)
- Nikita Sharma (M20DH015)

Abstract

Sudoku is a puzzle played on an n × n grid N , where n is the square of a positive integer m. The most common size is n=9. The grid N is partitioned into n subgrids of size m × m. The player must place exactly one number from the set N={1, …, n} in each row and each column of N , as well as in each subgrid. A grid is provided with some numbers already in place, called givens. The origins of Sudoku can be traced back to the 19th century when number games first appeared in newspapers. The modern version of the puzzle was likely created in the 1970s and became a world success in the following decade. Nowadays, many daily newspapers offer Sudoku alongside crosswords and other puzzles. Figure 1 depicts a 9 × 9 Sudoku. Figure 1(a) shows the initial grid with 22 givens, and Figure 1(b) provides the solution.

a

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

b

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

Figure 1 An instance of a Sudoku: (a) A 9×9 Sudoku with 22 givens; (b) The completed Sudoku.

Problem statement

We compare many algorithms namely **Backtracking**, Backtracking with heuristic ,Backtracking with heuristic (Forward checking and Arc consistency), **Hill Climbing** .Our aim is to compare the relative performance of these algorithms. We will assess the efficiency of each algorithm with respect to the input data, that is, to the number of givens, and with respect to the size of instance, in order to evaluate which kind of algorithm works best.

Literature survey

This section reviews already published papers, which are found to be relevant to build the proposed application.

[1] "An FPGA based Sudoku Solver based on Simulated Annealing Methods reviewed on The Sudoku simulated annealing solver -SSAS is a probabilistic Sudoku solver.

[2] "Solving Sudoku Puzzles using Improved Artificial Bee Colony Algorithm" provides Sudoku puzzles belong to a set of hard problems called NP-Complete problems. A Sudoku puzzle is a logic-based combinatorial puzzle with rules that are relatively simple. Various algorithms have been applied to solve this combinatorial problem.

[3] "A Deep Learning approach to solve sudoku puzzle" explains the methodology involves several stages such as image preprocessing and image extraction using OpenCV, OCRing and finally feeding the numerical data extracted to the neural network (tensorflow) model to get the desired output. Camera-based Sudoku recognition with deep belief network a method to detect and recognize a Sudoku puzzle on images taken from a mobile camera. The lines of the grid are detected with a Hough transform.

[4]Solving Sudoku with Boolean Algebra Sudoku solving technique named Boolean Sudoku Solver (BSS) using only simple Boolean algebras. Use of Boolean algebra increases the execution speed of the Sudoku solver.

[5]"A hybrid backtracking and Pencil and paper sudoku solver" moves through empty cells of a Sudoku puzzle tests each candidate or compatible digit of the cell, if there are no violations candidate is placed in the cell.

Novelty

Since, arc consistency , has around 10 percent success rate.

Hill climbing algorithm has around 5 percent success rate.

Simple backtracking algorithm has good success rate but high execution time

The following improvements were made to increase success rate and reduce execution time.

Algo 1

A constraint was used which simplified our sudoku table and make it easier to solve. The constraint of naked twins is that no boxes in units with naked twins in them have digits which are in the naked

twins boxes. By removing those digits from all boxes in a unit except the boxes which make the naked twins, we are making sudoku easier to solve.

Algo 2

It is a combination of Backtrack, minimum remaining values and arc-consistency. Backtrack helps us to backtrack when we detect failure and as we know that MRV chooses the variable with the fewest legal values and tries to fill those values , then we go for arc consistency to keep track of remaining legal values for unassigned variables. Arc consistency detects failure earlier than forward checking and can be run after each assignment. This algorithm is complete because of backtracking and by using MRV and arc-consistency, we reduced the time for solving a sudoku.

Methodology

We used 4 different datasets consisting of following number of puzzles
Easy difficulty puzzles – 100
Medium difficulty puzzles – 100
Hard difficulty puzzles – 100
Random puzzles – 1000
Random puzzles dataset was used for plotting No. of givens V/s average time and Time interval V/s No. of instances.

Algo1

First we created a variable (unitlist) consisting of all the constraints i.e.

1. The same number can't appear in the same row twice.
2. The same number can't appear in the same column twice.
3. Each subgrid must contain a single number from 1 to 9.

Then , we created a set consisting of all the peers for a particular box for which constraints will be applicable.
Then, following functions were created .

- Algo
  It goes through columns, rows and squares which we consider as peers and checking if twin exist in each of these
  If we detect twin in any of the peers then delete those numbers.
- Grid values
  It takes the input strings and convert it into a dictionary. If the value for a particular box is non-given then '123456789' is assigned .
- Display
  It the solved sudoku in the dictionary form and displays it into a 2D grid.
- Eliminate

It goes through all the boxes, and whenever there is a box with a value, eliminate this value from the values of all its peers.

- Only choice
  It goes through all the boxes and when there is a box with a single value , it assigns it.
- Reduce puzzle
  It reduces the number of possible values for a box using the following functions Naked Twins, Eliminate and only choice.

When the number of all possible values for the boxes is one , then sudoku is solved.

Algo 2

The Sudoku Grid is represented as a dictionary in python. The keys of the dictionary will be the variable names, each of which corresponds directly to a location on the board. In other words, we use the variable names Al through A9 for the top row (left to right), down to I1 through I9 for the bottom row. For example, in the example board above, we would have sudoku["B1"] = 9, and sudoku["E9"] = 8. The number 0 is used to represent the tiles which are not filled.

We first defined the constraints for the sudoku problems in the class csp.
Suppose at any instance, my state of sudoku is T.
If T is a complete assignment i.e, sudoku is fully-filled as per constraints (isComplete function does this) , then return it as it is the goal state.
But if T is not complete then do the following :

STEP 1⇒ Select all the unassigned variables which can be assigned using MRV.
STEP 2⇒ Then we fill remaining legal values in those variables.
STEP 3⇒ For all the assignments, check if the assignment is consistent or not i.e, satisfy the Constraints. (IsConsistent function does that).

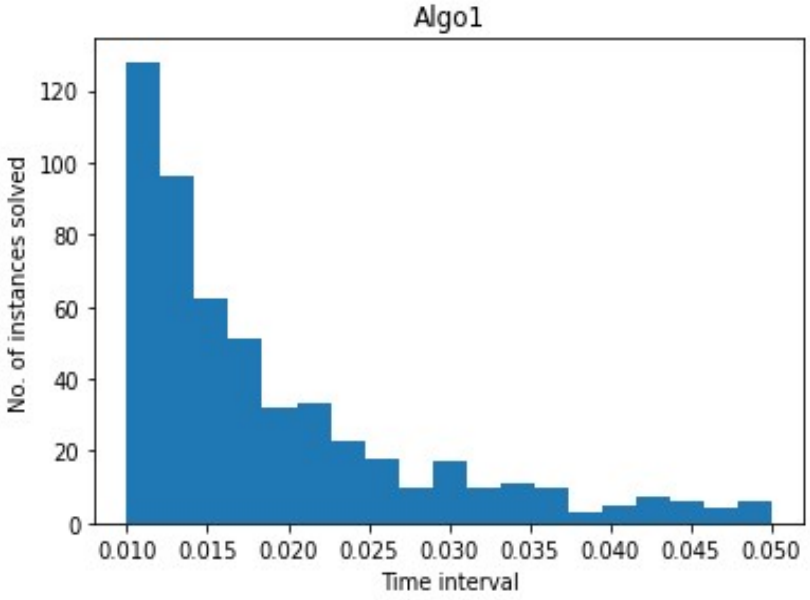
- If not consistent, go back.
- If yes, then by arc-consistency we keep the track of remaining legal values for other variables. (Inference function does that).
  Go to step1.

Algo 3

Step -1. Based on the Sudoku rule: 1. Check for each row and see if there is a unique possible number across a row. If so, update the poss_val and sol. 2. Check for each column and see if there is a unique possible number across a row. If so, update the poss_val and sol. 3. Check for each box and see if there is a unique possible number across a row. If so,update the poss_val and sol.
Step-2 .Check for unique possible values: For each cell, if there is only one possible number, update sol and remove from poss_val.
Step-3 : Run all basic rules until there is no update on sol.
Step-4: In this step create a function that applies this algorithm and eliminates impossible numbers. And if there is any update, re-run all basic rules.

Step-5 : Apply this algorithm: 1. On each row 2. On each column 3. On each box 4. Apply this algorithm until there is no update on sol. 5. By considering the possible numbers within a box, check if there is a possible number only in one row/column. Step-6: If so,then in the same row/column outside the box will be eliminated from all markup.
In main function:
First ,include all possible numbers for each cell. If the cell is already filled, remove all possible numbers for this cell Run basic rules and this algorithm . Check if the final result is solution.

Results



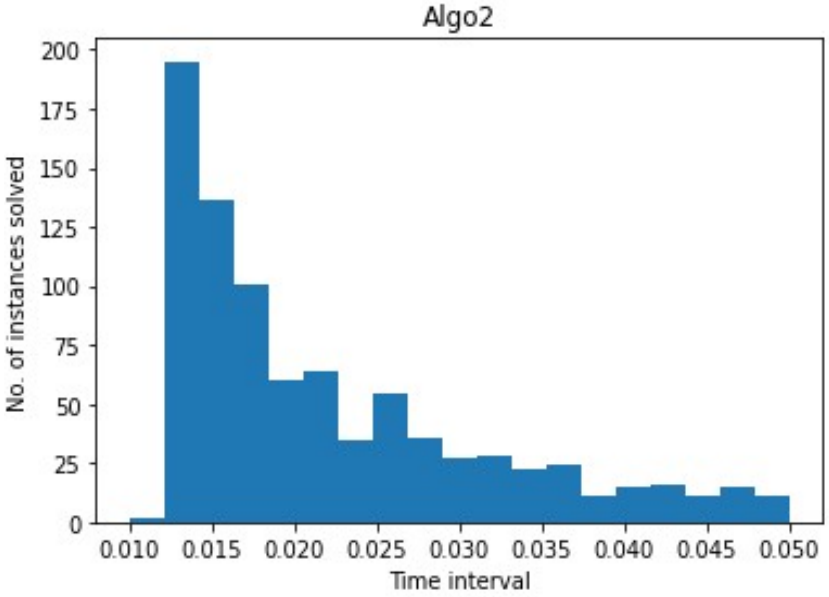
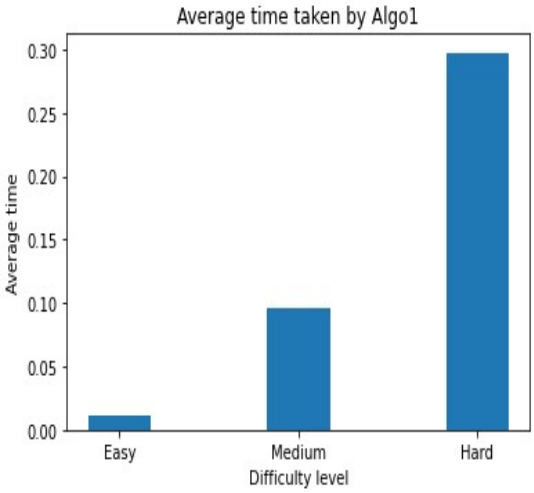
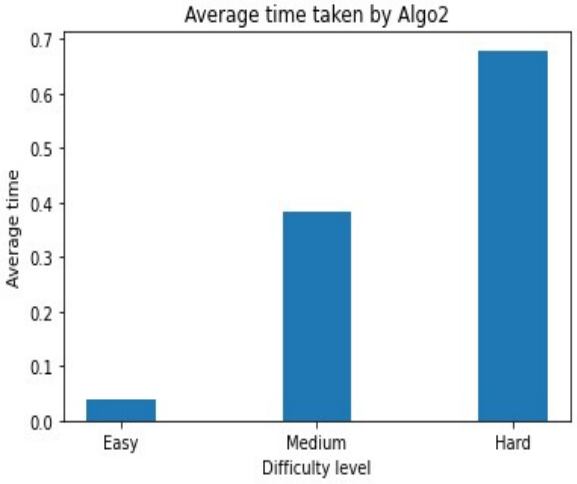
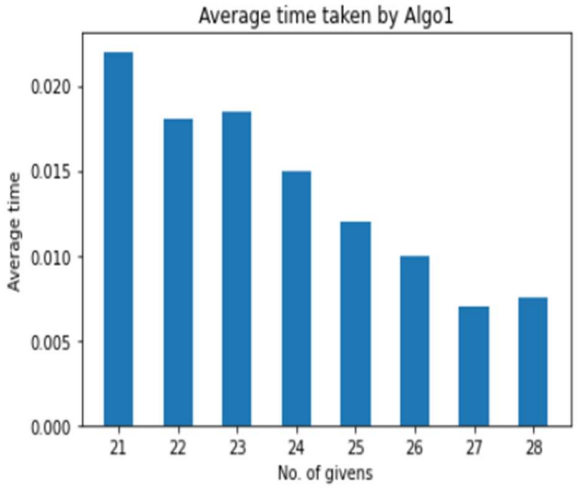


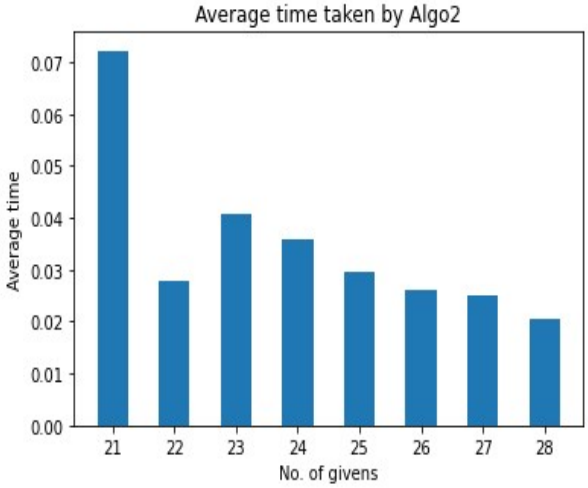
Table 1 : Running time to solve an empty sudoku

Algo 1	Algo 2	Algo 3
46.66 ms	*18.83 ms*	*Failure*









Conclusions

We have compared here three algorithms for solving a sudoku problem

- In general , Algorithm 1 was the fastest , it took around 10ms to solve most of the problems
- For sudoku with different difficulty ratings Algo 1 was faster than Algo 2 in each of the categories i.e. easy, medium, hard
- Algo 2 took around 15ms to solve most of the problems.
- However in solving an empty grid Algorithm 2 took the lowest time.
  And Algorithm 3 was unable to solve it.
- In both the algorithms as the number of givens are increasing the running time decreases.
  Limitations
  Algo2 uses MRV for assignment and it means that it does not consider every possible assignment, So, in cases where assignments through MRV will not take us to the solution, this algorithm will fail. That is, assignments by MRV are not suitable, it will cause problems in future and even backtracking will not help

References

https://warwick.ac.uk/fac/sci/moac/people/students/peter_cock/python/sudoku/

[1] Leandro C Coelho & Gilbert Laporte (2014) A comparison of several enumerative algorithms for Sudoku, Journal of the Operational Research Society, 65:10, 1602-1610, DOI: 10.1057/jors.2013.114

[2] Rajneesh Tiwari, Aritra Sen, Arindam Banerjee, "Generalization through augmentation in deep neural networks for handwritten character recognition", International Journal of Scientific & Engineering Research Volume 11, Issue 9, September 2020

[3] Kshitij Gupta, Sagar Khatri, M. Khan, "A Novel Automated Solver for Sudoku Images", IEEE 5th International Conference for Convergence in Technology (I2CT), March 2019.

[4] A. Van Horn, "Extraction of sudoku puzzles using the Hough transform", University of Kansas, Department of Electrical Engineering and Computer Science, Tech. Rep., 2012.

[5] "A Hybrid Backtracking and Pencil and Paper Sudoku Solver" International Journal of Computer Applications (0975 – 8887) Volume 181 – No. 47, April 2019

Roles of each individual

Tanmay Agrawal
(M22MA011)
Did literature review of different sudoku algorithms and implemented algo 1
Priya Sahu
(B20AI031)
Implemented algo 2 and did plotting and comparison of different algorithms
Poonam Suthar
(B20ES007)
Did literature survey of different sudoku algorithms and implemented algo 3
Nikita Sharma
(M20DH015)
Searched and provided input grid(datasets) for all 3 Algo’s