The Sudoku Project

Ishika De and Yashvi Donga

October, 2021

1 Objective

The goal of this project is to investigate a variety of algorithms (backtracking, brute force, stochastic search and Crook's algorithm) that are capable of solving sudoku puzzles, of ranging difficulties, in order to learn more about sudoku solving techniques.
We also wanted to use the OpenCV library to read a sudoku from an image and solve it.

2 Toolchain

- Languages used: Python, C++, Java.
- Libraries used in Python: Numpy, OpenCV and Keras.

3 Challenges

In the course of this projects we faced a few challenges that helped us learn more about working on a group project and helped us understand our mistakes and improve upon them.
The challenges we faced were as follows:

- **Setting unrealistic deadlines**
  While scheduling tasks, we would tend to set overambitious deadlines, that would mess up our schedule and eventually hinder our progress.
- **Explaining each other's ideas/concepts**
  This project challenged our communication skills. It took us a while to share our ideas and thoughts efficiently.
- **Failure to implement algorithms**
  We were unable to implement few algorithms like simulated annealing and Crook's algorithm for higher difficulty puzzles.
- **Dealing with code errors**
  It was difficult for us to move on with failures in code. We would get stuck and would end up spending more than required time on a particular issue.

4 Learnings

Through this project we wanted to explore different concepts of programming. In addition to this, we also wanted to learn about neural networks and image processing.
By the end of this project we have learnt to:

- **Collaborate using GitLab**
  For this project we chose to use git as our version control tool. It made collaboration easier as we worked on this project remotely.

- **Write the same algorithm in different languages**
  We explored the concepts of backtracking and brute force algorithm and implemented these in different languages. We learnt to translate code from one language to another and learnt the pros and cons of each of the languages used.

- **Explain our code, thought processes and ideas to each other**
  Initially, communicating our ideas during the ideation of the project was challenging, but, through this project we learnt to explain our codes and ideas to each other efficiently.

- **Apply the concept of cost function and thermodynamics in simulated annealing**
  Simulated annealing algorithm was an eye opener as it required us to understand concepts of physics - thermodynamics. The algorithm has an analogy with thermodynamics, specifically with the way that metals cool and anneal. Simulated annealing calculates optimum of a function instead of the energy of a material.

- **Process an image to extract digits of a sudoku**
  Image processing was an concept that interested us. We employed the openCV library in python to read a sudoku from an image and subsequently solve it.

- **Implement neural networks to predict digits of a sudoku from an image**
  To identify the digits from an image we used the MNIST dataset to train the neural network. We learnt how images are fed to a system to capture patterns, in this case, to identify numbers from 1 - 9.

- **Importance of changeability of code**
  We were able to generate 3 different levels of sudoku just by changing the user input. We generated sudoku puzzles of 3 different difficulties - easy, medium and hard in the three different languages.

5 Results and Conclusion

We tested 100 sudoku puzzles in each language for each of the algorithms. The results for the latter are given in the image below.

As give in the table below, backtracking algorithm was visibly faster than the brute force algorithm in all the languages. While brute-force tries every single possibility, back-tracking follow some path and goes back to previous intersection if it's a dead-end. This is the reason why backtracking was was much more time-efficient.

Language	Difficulty	Time taken by an algorithm (milliseconds)	
		Backtracking	Brute force
C++	Easy	0.02	1.11
	Medium	0.08	21.43
	Hard	0.24	48.89
Java	Easy	0.03	18.27
	Medium	0.26	65.07
	Hard	0.40	83.35
Python	Easy	30.96	41.83
	Medium	66.86	253.84
	Hard	175.50	6,520.23

Figure 1: Average time taken to solve a sudoku (tested 100 puzzles).

Considering the languages, C++ took the least amount of time to solve the sudoku and python clearly was the slowest to solve the puzzles, especially puzzles of higher difficulty.
Below is a snapshot of the solved sudoku that was read from an image.

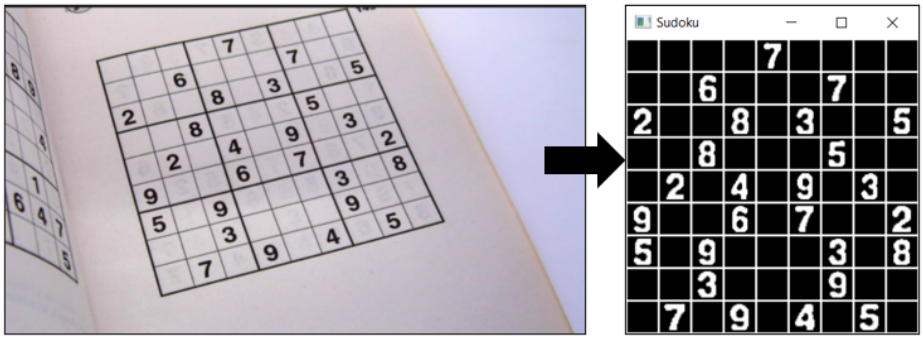

Figure 2: Image processing of a sudoku puzzle image.

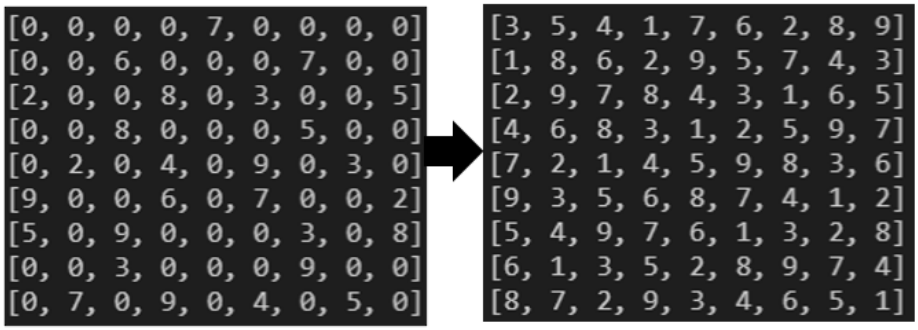

Figure 3: Solved the sudoku from the image

6 Code Repository

Our code repository link: *https://gitlab.com/yashvidonga29/sudoku-project-we*

7 References

- http://people.uncw.edu/tagliarinig/Courses/380/F2015%20papers%20and%20presentations/F2015%20papers%20and%20presentations/Sudoku/Sudoku.pdf
- https://towardsdatascience.com/solving-sudoku-with-ai-d6008993c7de
- http://zhangroup.aporc.org/images/files/Paper_3485.pdf
- https://www.sudokuwiki.org/Brute_Force_vs_Logical_Strategies
- https://stacks.stanford.edu/file/druid:my512gb2187/Agarwal_Kamat_Kurian_Smart_Sudoku_Solver.pdf
- https://blogs.mathworks.com/deep-learning/2018/11/15/sudoku-solver-image_processing-and-deep-learning
- https://www.researchgate.net/publication/264572573_Sudoku_Puzzle_Complexity
- https://www.youtube.com/watch?v=FyyVbuLZav8
- https://www.adrian.idv.hk/2019-01-30-simanneal/
- https://link.springer.com/article/10.1007/s10732-007-9012-8
- https://docs.opencv.org/4.5.2/d0/de3/tutorial_py_intro.html
- https://www.youtube.com/watch?v=aircAruvnKk