Project Report
CSC361

Professor Hasan Mathkour

Abdulkarim Alolayet 444101988
Almothana Alzahrani 444100146
Nawaf Alkhalifa 444102189

Project Summary

An AI agent that uses two distinct AI techniques (backtracking and genetic algorithm) to solve a Sudoku game.

1 Project Description

1.1 Motivation

Sudoku is a very challenging game, and a lot of people fail while playing it. Even when they manage to solve it, they take a lot of time. The AI agent lets the computer solve a problem that's hard for humans to solve. We used two different AI techniques because we wanted to compare and contrast their effectiveness.

1.2 Background

1.2.1 What is Sudoku?

Sudoku is a logic-based number puzzle that involves filling a grid with digits according to specific rules. The most common Sudoku variant features a 9×9 grid divided into nine 3×3 subgrids, known as regions.

1.2.2 Rules of Sudoku

- **Grid Structure:** The standard Sudoku grid consists of 81 cells arranged in a 9×9 format.
- **Numbers:** The digits from 1 to 9 must be placed in the grid.
- **Row Rule:** Each row must contain all digits from 1 to 9 without repetition.
- **Column Rule:** Each column must contain all digits from 1 to 9 without repetition.
- **Box Rule:** Each of the nine 3×3 subgrids must contain all digits from 1 to 9 without repetition.

1.2.3 Gameplay

Starting Point: A Sudoku puzzle is typically provided with some cells already filled with numbers (the "clues"), while others are empty.
Objective: The goal is to fill in the empty cells while adhering to the rules.

1.2.4 What is a Genetic Algorithm?

A genetic algorithm (GA) is a search heuristic inspired by the process of natural selection and genetics. It is used to find solutions to optimization and local search problems.

Key Concepts

- **Population:** A genetic algorithm has a set of potential solutions, called a population.
- **Fitness Function:** Each gene is evaluated using a fitness function, which quantifies how good the gene is.
- **Selection:** Genes are selected based on their fitness scores. Higher fitness genes are more likely to be chosen for reproduction.
- **Crossover:** Selected genes (parents) combine their genetic information to produce one or more offspring.
- **Mutation:** Random changes are introduced to some offspring to maintain genetic variability within the population. This helps prevent early convergence on local optima.
- **Generation:** The process of selection, crossover, and mutation is repeated over multiple generations. The population evolves, ideally leading to better solutions over time.

1.2.5 What is Backtracking?

Backtracking is a problem-solving method used in CSP (constraint satisfaction problems). It helps find solutions to problems by trying different options and going back if a choice doesn't work. It can be optimized by adding inferences such as forward checking and AC3.

How It Works

1. **Try an option:** You begin by choosing one possible option for a problem.
2. **Check if it works:** After making a choice, you check if it leads to a solution. If it does, great! If not, you need to go back.
3. **Go back (Backtrack):** If the option you chose doesn't work, you undo that choice and try a different one.
4. **Repeat:** You keep trying different options and backtracking until you find a solution or exhaust all possibilities.

2 Proposed Approach

2.1 Backtracking

In this approach, we try to fill each cell one by one. For each cell, we try all possible numbers. Before placing a number, we check if it causes any conflicts (like if the number is already in the same row, column, or box).

If there's no conflict, we place the number and move to the next cell. We keep doing this until the whole board is filled. If we get stuck and can't place any number in a cell, we go back (backtrack) to the previous cell, remove the number we placed, and try a different one. We repeat this process until we find a solution or run out of options.

2.2 Genetic Algorithm

Gene/State Each gene is represented as a 2D array containing an integer from 1–9 representing the value for each cell. Each gene follows the box rule.

Fitness Function Maximum fitness is 5 million; each pair of cells that doesn't follow the rule gets 1 mark off. And if one of the pair of cells existed in the original array, there would be an extra penalty of 10 marks.

Crossover A random one-point crossover that crosses between whole blocks and never partial blocks (so the children follow the box rule as well).

Mutation It's a swap mutation where 2 cells swap their value. Both cells have to be in the same block (so the state after mutation adheres to the box rule).

Restart After 5000 generations, we will restart our population over again to produce some randomness.

3 Results

3.1 Backtracking

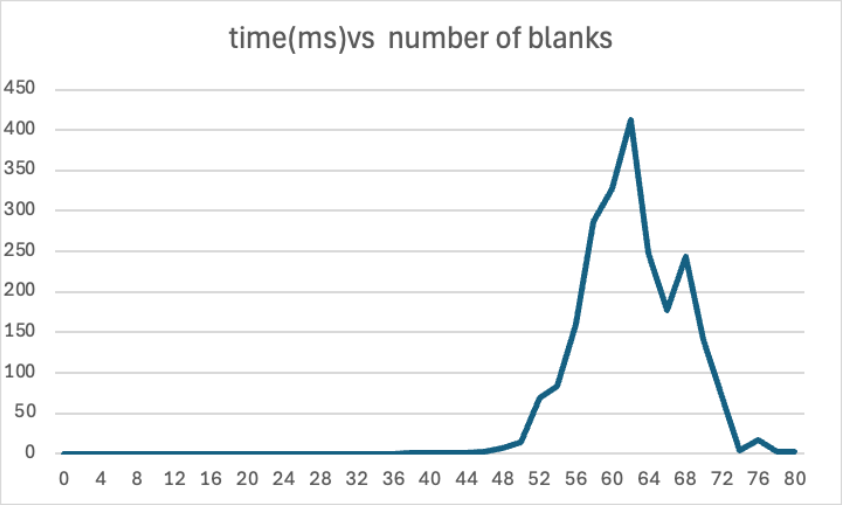

Figure 1: Average time (ms) vs. number of blanks (backtracking)

This graph shows the average time (ms) the backtracking algorithm takes for every number of blanks in the Sudoku grid. We can observe that the peak time taken is not where we would expect it—at the 80 space mark, but rather in the 58–62 range.

3.2 Genetic Algorithm

A graph has been plotted showing how the genetic algorithm performs, with each colour indicating different numbers of empty cells.

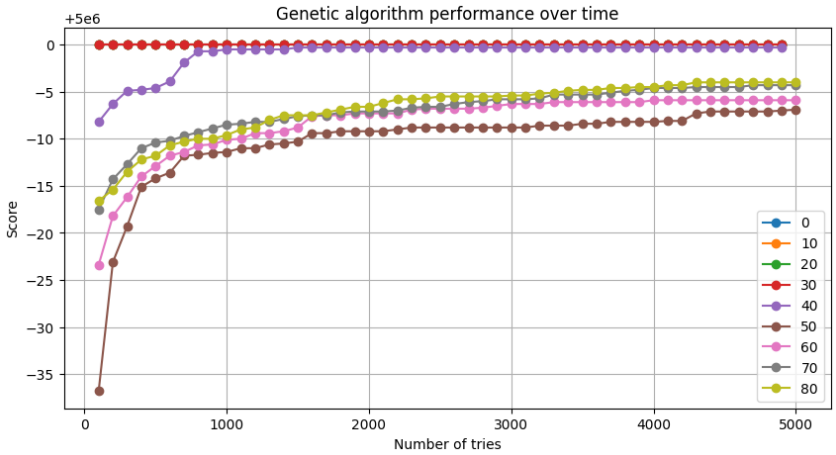

Figure 2: Number of tries vs score (genetic algorithm)

In the previous graph, we can see that with a low number of blanks, like 0, 10, 20, and 30, the Sudoku will get solved almost instantly. However, for a higher number of blanks, the algorithm starts with a high growth that decreases over time.

We also noticed that the worst two graphs were the ones with 50 and 60 blanks, which aligns with the backtracking graph, showing that the most time-consuming puzzles were not the ones with the most blanks.

We can conclude that the genetic algorithms are very good at optimizing solutions, but not very optimal for finding a definitive answer. As we can see, the bottom graphs haven't converged yet. They will do so with a higher number of tries, where the maximum (highest) fitness is 5 million, as we mentioned earlier.

4 Conclusion

In conclusion, Sudoku is better solved using a CSP solution such as backtracking. We implemented the genetic and backtracking algorithms. And we tried to implement reinforcement learning but didn't succeed. Both the genetic and reinforcement learning algorithms failed to keep up with backtracking's efficiency and speed. After doing some digging, we found a research paper that solved Sudoku using a CNN (convolutional neural network) with 99.7% accuracy.

References

- Akin-David, C., & Mantey, R. (2018). Solving Sudoku with Neural Networks [CS230 course project report]. Stanford University.
- Mantere, T., & Koljonen, J. (2007). Solving, rating and generating Sudoku puzzles with GA. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)* (pp. 1382–1388). IEEE.