MATH:7450
Topological Data Analysis Project: Population Diversity of Genetic Algorithm for Solving Sudoku Puzzles

Gregory Tanner,
Applied Math and Computational Sciences,
University of Iowa

Brice Merwine,
Applied Math and Computational Sciences,
University of Iowa

February 22, 2018

1 Abstract

2 Introduction

In this project, we use the Topological Data Analysis (TDA) Mapper algorithm to analyze the generational results from genetic algorithms applied to solving Sudoku puzzles. Specifically, we aim to answer questions about the diversity of candidate solutions over time.

3 Background

3.1 Problem Description

Sudoku is a popular pencil and paper puzzle that takes place on a 9-by-9 grid divided into nine 3-by-3 blocks. The puzzle begins with certain cells (givens) of the grid filled in with the numbers 1-9. The goal of the puzzle is to fill in the remaining grid such that:

1. Every row contains the numbers 1-9 exactly once.
2. Every column contains the numbers 1-9 exactly once.
3. Every 3-by-3 block contains the numbers 1-9 exactly once.

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

(a) Unsolved Sudoku Puzzle [1]

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

(b) Solved Sudoku Puzzle [2]

Genetic algorithms are a biology-inspired stochastic optimization strategy that aims to mimic the processes of evolution and natural selection. In search of an optimal solution, the algorithm produces a population of solutions that evolve over successive generations. The main evolutionary operators that are applied are recombination (also called crossover) and mutation. In recombination, information from two (or more) parents is randomly combined to create an new offspring solution. In mutation, a small portion of the information of a solution is randomly changed. At each generation, a certain number of current population and offspring is chosen to survive. This process repeats until a termination condition is reached. The general scheme of a genetic algorithm follows[3]:

Algorithm 1 Genetic Algorithm Pseudocode

```
INITIALIZE population with random candidate solutions;
EVALUATE each candidate;
repeat
    SELECT parents;
    CROSSOVER pairs of parents to create offspring;
    MUTATE the resulting offspring;
    EVALUATE new candidates;
    SELECT individuals for the next generation;
until TERMINATION CONDITION is met
```

There are several papers detailing the application of genetic algorithms to Sudoku [4, 5, 6, 8]. For this project, we follow the construction of Segure et al. [6]:

Encoding The candidate solutions are represented as 81-element arrays containing the numbers 1-9. The candidate solutions are initialized such that each 3-by-3 block contains the numbers 1-9 exactly once. This leaves only the row and column constraints to check.

Evaluation Each candidate solution is evaluated by taking the sum of the number of row and column repetitions. An additional penalty of 100 times the number of row and column conflicts is imposed. The goal is to minimize this fitness value, and the exact solution has a fitness value of 0.

Parent Selection Two parents are selected by a pair of binary tournaments. In each tournament, two random candidates are selected from the population, and the candidate with the better fitness score is selected to be a parent.

Recombination The recombination selects entire 3-by-3 blocks from the parents to create the offspring. This maintains that each block contains the numbers 1-9 exactly once.

Mutation The mutation operator swaps two non-fixed elements within a block.

Survivor Selection We evaluate several survivor selection mechanism and investigate their impact on diversity.

Termination Condition The genetic algorithm was allowed to run for a fixed number of generations or until the exact solution (fitness value = 0) is found.

Segura et al. noted the importance of maintaining diversity in the solution population when trying to solve Sudoku puzzzles with genetic algorithms [6]. They analyze the performance of several different survivor selection mechanisms in maintaining diversity.

3.2 Topological Data Analysis Mapper Algorithm

The TDA Mapper algorithm was developed by Singh, Mémoli and Carlsson [7]. The purpose of the TDA mapper algorithm is to allow a method to visualize higher dimensional data in a manageable fashion. Inspired by the Reeb graph, the mapper utilizes topological properties to extract information from the shape of the data. The TDA mapper algorithm consists of these basic steps:

- Obtain and preprocess the data into a manageable format.
- Use a filter function to map the data onto one (or possibly two) dimensions.
- Cover the image of the data by overlapping intervals (patches for two dimensions). The preimage of these intervals (patches) create overlapping bins in the data.
- Cluster each overlapping bin separately.
- Create a graph from the results. Each vertex corresponds to a cluster, and edges are placed between intersecting clusters.

The TDA mapper algorithm allows us to visualize our data as a lower dimensional graph, to which we can use topology to examine and draw conclusions.

4 Results

5 Discussion

6 Conclusion

7 Acknowledgement

The authors would like to acknowledge the TDA Mapper course taught by Dr. Isabel Darcy Spring 2018. Also, the authors would like thank Cole Stiegler for the countless conversations we had about implementing genetic algorithms.

8 Author Contribution

Gregory Tanner developed the Matlab codes for solving Sudoku puzzles with genetic algorithm. He contributed to all of the sections of this project report.

Brice Merwine contributed to the background section.

9 Funding Sources and Conflicts of Interest

Gregory Tanner is supported by a fellowship from the University of Iowa Graduate College. He is conflicted about the amount of time that he has spent on the genetic algorithms for Sudoku puzzle problem.

References

[1] `https://upload.wikimedia.org/wikipedia/commons/e/e0/Sudoku_Puzzle_by_L2G-20050714_standardized_layout.svg`.

[2] `https://upload.wikimedia.org/wikipedia/commons/1/12/Sudoku_Puzzle_by_L2G-20050714_solution_standardized_layout.svg`.

[3] A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[4] Timo Mantere and Janne Koljonen. Solving and rating sudoku puzzles with genetic algorithms. In *New Developments in Artificial Intelligence and the Semantic Web, Proceedings of the 12th Finnish Artificial Intelligence Conference STeP*, pages 86–92, 2006.

[5] Yuji Sato and Hazuki Inoue. Solving sudoku with genetic operations that preserve building blocks. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 23–29. IEEE, 2010.

[6] Carlos Segura, S Ivvan Valdez Peña, Salvador Botello Rionda, and Arturo Hernández Aguirre. The importance of diversity in the application of evolutionary algorithms to the sudoku problem. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 919–926. IEEE, 2016.

[7] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *SPBG*, pages 91–100, 2007.

[8] John M Weiss. Genetic algorithms and sudoku. In *Midwest Instruction and Computing Symposium (MICS 2009)*, pages 1–9, 2009.