**Sudoku Solver with Genetic Algorithm**

**How to Run**

1. git clone [git@github.com:cdivyanambiar/Genetic-Algorithm-Based-Sudoku-Solver.g](mailto:git@github.com:cdivyanambiar/Genetic-Algorithm-Based-Sudoku-Solver.g)
2. cd cd Genetic-Algorithm-Based-Sudoku-Solver
3. For 2x2 use this: **python GA\_Sudoku\_Solver.py "4" "abcd" "Sudoku\_4.json**" [This is the use case given]

For 3x3 use this: **python GA\_Sudoku\_Solver.py "9" "123456789" "Sudoku\_9.json"**

First Parameter is Proper Square of a number

Second the word you need to pass

Third is a file with the random index of Sudoku question generator

I have copied samples for dimension 4 (Sudoku\_4.json) and 9(Sudoku\_9.json)

Note To Run this program **Python 3.x and Numpy** required**.**

**Pseudocode**

Nd = Dimension which is a proper square

Input = the string of length Nd.

**Initialized Genetic variables as follows**:

Nc = 1000 # Number of candidates (i.e. population size).

Ne = int (0.05 \* Nc) # Number of elites that is the individuals in the generation with the best fitness values.

Ng = 10000 # Number of generations.

Nm = 0 # Number of mutations.

* Read Nd
* Read Input
* Read Json file [We can hardcode the grid if required. For creating random grids we can use Jason]
* Validate input :

1. Check Nd is proper square
2. Length of input is equal to Nd
3. Input contain proper Digit/Characters (No white spaces)
4. Input contain duplicate character or 0
5. Whether the Json file exists.

* Load DB (here Json file) for random generation of question
* Generate random grid (Puzzle.New\_Game() method)
* Call Solver which is based on Genetic Algorithm.
* Solver has below algorithm:

1. Randomly initialize the Population:

Here we will initialize the vector of int (Passing index of the given word as word to as numbers are faster than Strings). Here we will find the permutation of 1 to Nd and randomly store those in chromosomes. This is done by the Population.seed() function

1. Determine the Fitness

The fitness function will find which solution is good. We will update the fitness of candidate and sort them. The fitness of a candidate solution is determined by how close it is to being the actual solution to the puzzle. The actual solution (i.e. the 'fittest') is defined as a Nd x Nd grid of numbers in the range [1, Nd] where each row, column and sqrtVal x sqrtVal block contains the numbers [1, Nd] without any duplicates if there are any duplicates then the fitness will be lower.

1. Until convergence repeat:
2. Parent (Tournament) Selection :

Here we will select n random population and perform a tournament among them. Select best from this N population in a stochastic way.

1. Crossover (Generate new population)

Creating 2 child’s from the selected parents from step 3. Here completely new gene get created from the parents by mutating

1. Perform Mutation on child

Here we will insert random string in population to maintain diversity.

1. Calculate fitness child

**Reference**

[**https://www.tutorialspoint.com/genetic\_algorithms/genetic\_algorithms\_fundamentals.htm**](https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_fundamentals.htm)

[**https://sudoku.com/**](https://sudoku.com/)

[**https://en.wikipedia.org/wiki/Genetic\_algorithm**](https://en.wikipedia.org/wiki/Genetic_algorithm)

[**https://www.geeksforgeeks.org/genetic-algorithms/**](https://www.geeksforgeeks.org/genetic-algorithms/)