SOLVING SUDOKU USING GENETIC ALGORITHM

BY TEAM 209,

- CYRIL SEBASTIAN

- SHURAVI BARMAN

- SANSKRUTI KARKHANIS

Problem Statement

To solve a Combinatorial number placement puzzle -Sudoku, which is a NP-Hard problem using Genetic Algorithm.

NP-Hard Problem Explanation

“NP-hardness, in computational complexity theory, is the defining property of a class of problems that are, informally, "at least as hard as the hardest problems in NP". A simple example of an NP-hard problem is the subset sum problem”

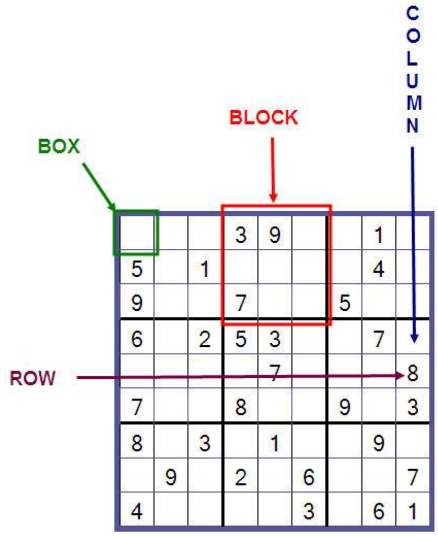
NP hard problems are referred to as the problems whose solutions are very difficult to find but they have a verifiable solution. NP is a set of all the problems which can be verified in a reasonable amount of time.

NP hard problems can be decision problems, search problems and optimization problems.

The best know algorithms for this problem take exponential time.

Rules for Playing a 9x9 Sudoku-

A 9x9 sudoku grid is divided into 3x3 sub grids each of which is called as a block. Some of the numbers in the grid are already filled.

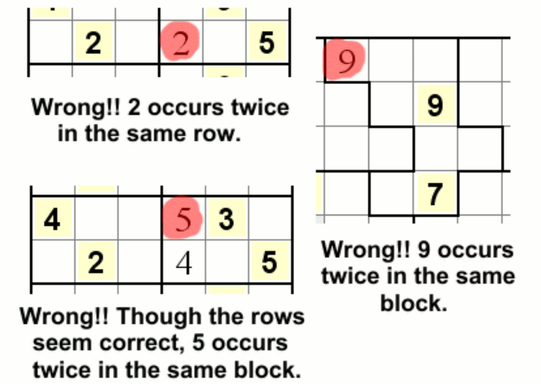


			3	9			1	
5		1					4	
9			7			5		
6		2	5	3			7	
				7				8
7			8			9		3
8		3		1			9	
	9		2		6			7
4					3		6	1

Picture 1. sudoku grid and notations.

The game is to fill the empty boxes in the puzzle with numbers between 1-9.

The numbers can appear only once in each row, column and block.


2
2
5
Wrong!! 2 occurs twice in the same row.
9
9
7
Wrong!! 9 occurs twice in the same block.
4
5
3
2
4
5
Wrong!! Though the rows seem correct, 5 occurs twice in the same block.

Picture 2. sudoku rules

Overview on Genetic Algorithm:

Genetic Algorithm as the name suggests is based on Darwin's Theory of Natural Evolution – "Survival of the Fittest". Genetic Algorithm implements this theory for finding solutions for different optimizations and searching techniques.

GA is an iterative process where, each individual is selected on basis of a fitness function. Each iterative step is referred to as a generation.

The following steps are to be followed while solving a problem using Genetic Algorithm: -

1) **Initialization of Population:**
   A population of multiple chromosomes is initialized. Each chromosome in the population has a different fitness value. This population selected should be diversified in-order to avoid convergence.
   There are two initialization techniques-
   Random Initialization:
   wherein, the population is initialized with totally with random solutions.
   Heuristic Initialization:
   wherein, the population is initialized on set of predefined rules.

2) **Computing Fitness Function:**

Fitness function determines the parameters for selecting the best candidate from set of input chromosomes.

Fitness Function is computed iteratively, in GA to find the best chromosome solution.

It is used for guiding stimulations toward optimum solutions.

3) **Selection of parents**

The fit candidates from a generation are chosen as parents and are passed on to next generation for reproducing the new chromosomes. These fit parents are genotypes.

Different selection methods like Rank Selection, Tournament selection, Roulette wheel etcetera can be used for selecting the genotypes.

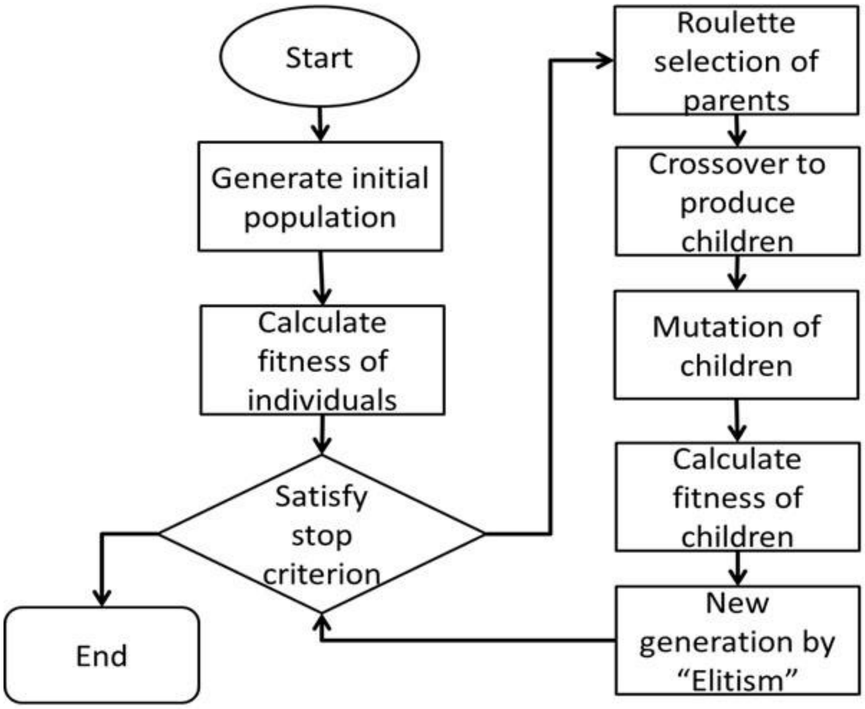
4) **Crossover**

In this step new chromosomes are reproduced by mating the fit chromosomes from the previous generation. The resulting child is a phenotype.

The main purpose of crossover is to produce offspring candidates that help in finding better solutions.

5) **Mutation**

Mutation is producing divergence in the population. It is used to avoid local minima situations.


Start
Generate initial population
Calculate fitness of individuals
Satisfy stop criterion
End
Roulette selection of parents
Crossover to produce children
Mutation of children
Calculate fitness of children
New generation by "Elitism"

Flow-chart for genetic algorithm

Implementation of GA in project:

The sudoku puzzle to be solved is read from a text file. The puzzle is then pre -solved to fill in the obvious numbers in the puzzle.

The Process of Elimination-Solving.

In the elimination-solving process, each empty cell is assumed to hold a set of numbers ranging from 1 to 9.

The number which appear in each cells' row, column and block are removed from its assumed set.

This process is repeated for all the cells in the block. There will be some blocks which will be left with only one number. This number is the solution for that particular cell.

The puzzle with the new number is again pre-solved. The elimination-solving step is continued till no other empty cells can be filled with obvious solutions.

The assumption set of empty cells in each row/column/block is checked. If there appears to be a number that appears in only one solution set. The cell containing the number in its assumption set is filled.

The above steps are repeated recursively until we are left with a partially solved grid, ready to be solved by genetic algorithm.

In case of easy puzzles, the elimination-solving might even solve the entire puzzle.

Genetic algorithm is then used to obtain the fittest solution for the problem.

Following are the steps that are followed-

1. **<u>Initializing the Population</u>**

   Population of individuals is created by filling out the empty boxes in the pre-solved puzzle. The values filled in the empty boxes are randomly selected from the assumption set for each cell. No value in the assumption set can generate a conflict with its column, row or block when the grid is partially solved.

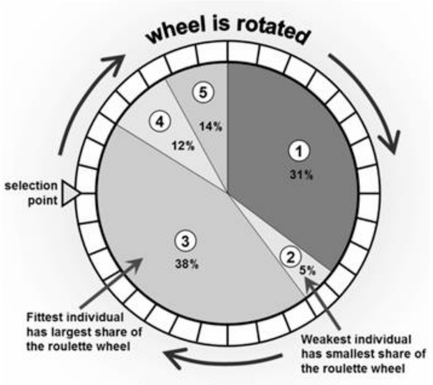
2. **<u>Fitness function</u>**-

   The fitness function chosen for solving the sudoku puzzle is – > **(216-gridconflicts)/216.**
   The maximum number of conflict a number can generate is 8. (It can be placed in 8 incorrect positions and only 1 correct position). There are 9 rows, columns and blocks. Thus, maximum number of conflicts a grid can have is 9*8*3 = 216.

3. **<u>Selection</u>**

   Roulette wheel selection is used to select 2 fit individuals for passing on to next generation. This step ensures that there are fewer conflicts in the future. The fitter parents selected for passing on to next generation are the genotypes.

The criteria for a genotype to be selected is the probability of its individual fitness to the total fitness of the population. Roulette wheel selection avoids local minima.

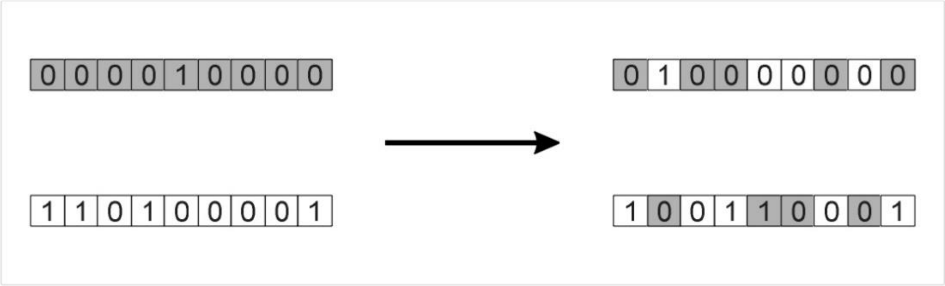

wheel is rotated
selection point
5
14%
4
12%
1
31%
3
38%
2
5%
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

Newcastle University, CS Resources

Picture 3-Roulette wheel

4. <u>Crossover</u>

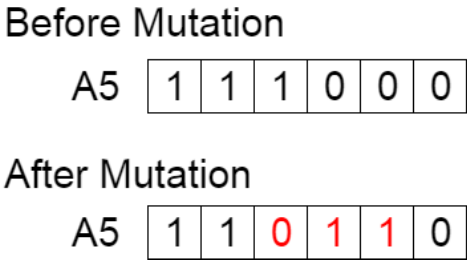
Uniform crossover is performed between the genotypes to form new individuals. The resulting individuals formed are the phenotypes.
The phenotype is then formed by randomly selecting the parent genotype.
Thus, the probability of each parent chromosome being selected is equal. The phenotype formed has the characteristics of its parents as well as its own traits.



Picture-4 Uniform crossover.

5. Mutation

Mutation is performed on the reproduced phenotype to main the diversity. The mutation probability chosen is greater than 0.1



Picture 4- Mutation

Program overview-

Each cell in the sudoku grid has three representations, row representation, column representation and block representation.

Each cell contributes in calculating the total grid conflict.

Solution is obtained when the grid conflict reaches zero value.

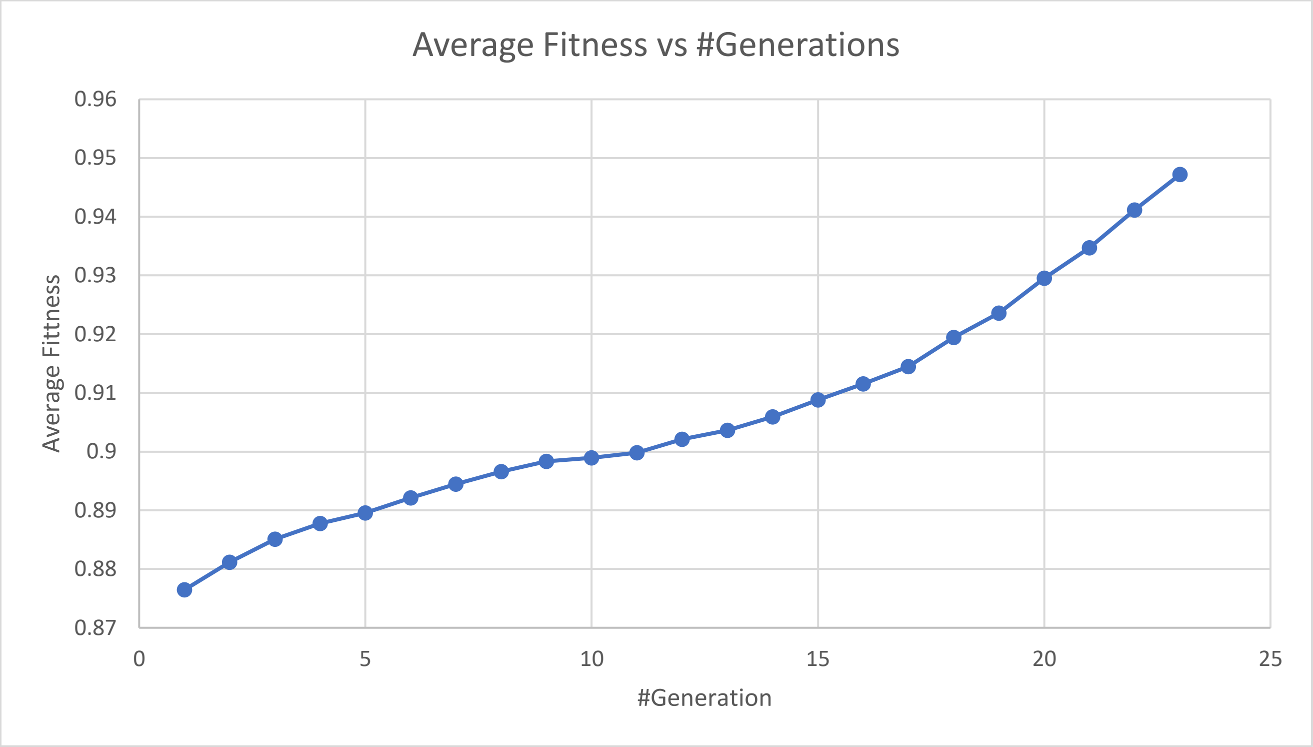
If the grid conflict does not arrive to a zero value after every 20 generations, The process of calculating the conflicts is restarted .

Observation Table:

Sr.No	Generation passed	Average Fitness
1	1	0.876458333
2	2	0.881171296
3	3	0.885074074
4	4	0.887787037
5	5	0.889550926
6	6	0.892125
7	7	0.894490741
8	8	0.896587963
9	9	0.898351852
10	10	0.898944444
11	11	0.899828704
12	12	0.902097222
13	13	0.903643519
14	14	0.905935185
15	15	0.908810185
16	16	0.911541667
17	17	0.914509259
18	18	0.919430556
19	19	0.923606481
20	20	0.929560185
21	21	0.934708333
22	22	0.941175926
23	23	0.947185185

Observation Table for the average fitness vs #generations when solution is obtained

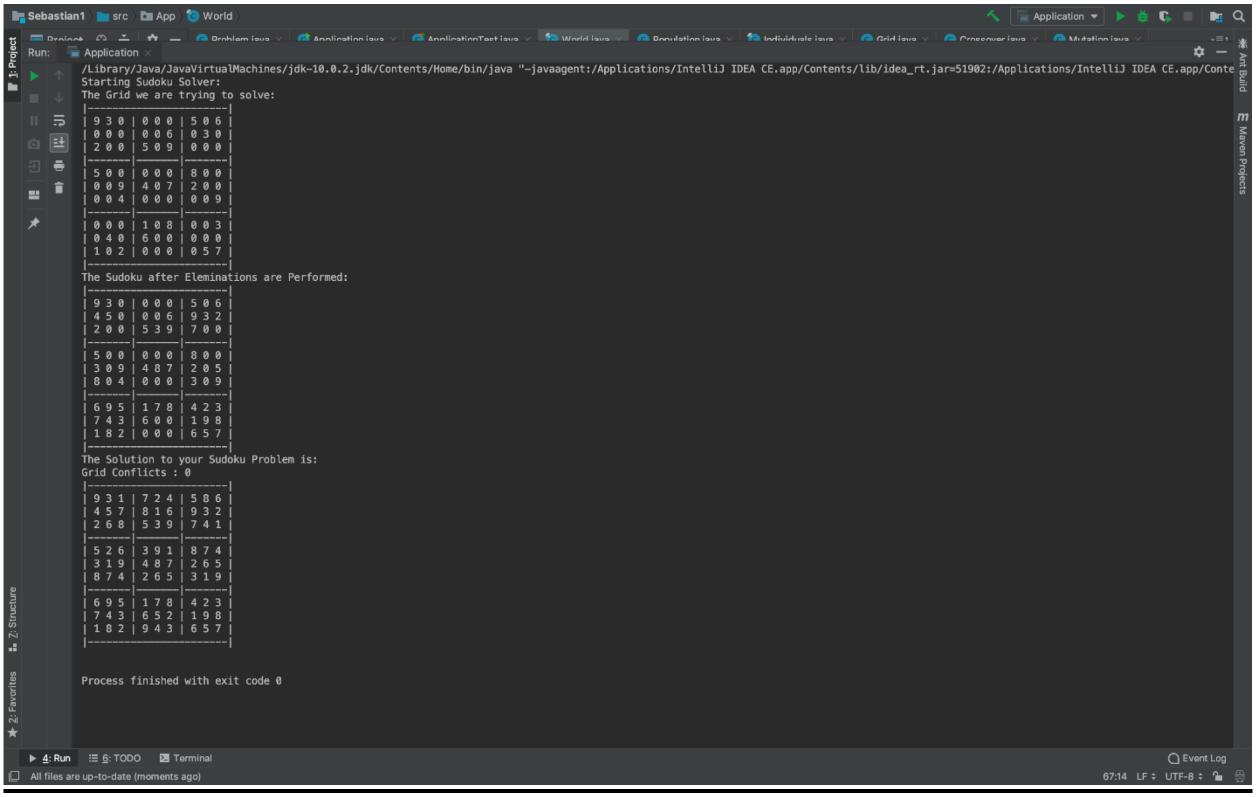
Graphical Representation: when Solution is obtained



Conclusion:

Difficult sudoku puzzles are examples of NP- hard problems. Each cell in the grid has a very large search space. Finding a solution for the puzzle can become very difficult. Genetic algorithm being an iterative and optimization technique based on Natural selection solves the puzzles by using an appropriate fitness function, Selection technique , Crossover and Mutation technique.

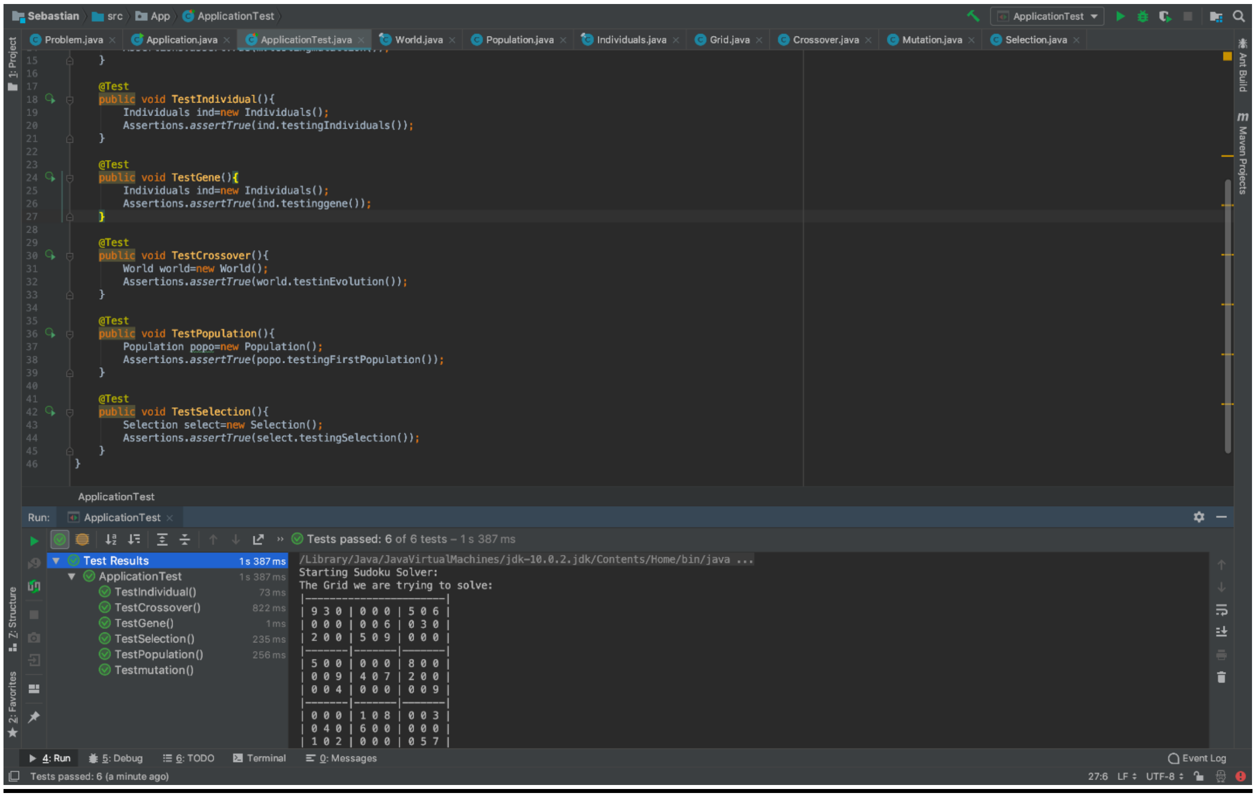
Program Run Snapshot:



```
/Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=51902:/Applications/IntelliJ IDEA CE.app/Conte
Starting Sudoku Solver:
The Grid we are trying to solve:
|-----------------------|
| 9 3 0 | 0 0 0 | 5 0 6 |
| 0 0 0 | 0 0 6 | 0 3 0 |
| 2 0 0 | 5 0 9 | 0 0 0 |
|-------|-------|-------|
| 5 0 0 | 0 0 0 | 8 0 0 |
| 0 0 9 | 4 0 7 | 2 0 0 |
| 0 0 4 | 0 0 0 | 0 0 9 |
|-------|-------|-------|
| 0 0 0 | 1 0 8 | 0 0 3 |
| 0 4 0 | 6 0 0 | 0 0 0 |
| 1 0 2 | 0 0 0 | 0 5 7 |
|-----------------------|
The Sudoku after Eleminations are Performed:
|-----------------------|
| 9 3 0 | 0 0 0 | 5 0 6 |
| 4 5 0 | 0 0 6 | 9 3 2 |
| 2 0 0 | 5 3 9 | 7 0 0 |
|-------|-------|-------|
| 5 0 0 | 0 0 0 | 8 0 0 |
| 3 0 9 | 4 8 7 | 2 0 5 |
| 8 0 4 | 0 0 0 | 3 0 9 |
|-------|-------|-------|
| 6 9 5 | 1 7 8 | 4 2 3 |
| 7 4 3 | 6 0 0 | 1 9 8 |
| 1 8 2 | 0 0 0 | 6 5 7 |
|-----------------------|
The Solution to your Sudoku Problem is:
Grid Conflicts : 0
|-----------------------|
| 9 3 1 | 7 2 4 | 5 8 6 |
| 4 5 7 | 8 1 6 | 9 3 2 |
| 2 6 8 | 5 3 9 | 7 4 1 |
|-------|-------|-------|
| 5 2 6 | 3 9 1 | 8 7 4 |
| 3 1 9 | 4 8 7 | 2 6 5 |
| 8 7 4 | 2 6 5 | 3 1 9 |
|-------|-------|-------|
| 6 9 5 | 1 7 8 | 4 2 3 |
| 7 4 3 | 6 5 2 | 1 9 8 |
| 1 8 2 | 9 4 3 | 6 5 7 |
|-----------------------|


Process finished with exit code 0
```

Unit Test Cases Screenshot:



```
    }

    @Test
    public void TestIndividual(){
        Individuals ind=new Individuals();
        Assertions.assertTrue(ind.testingIndividuals());
    }

    @Test
    public void TestGene(){
        Individuals ind=new Individuals();
        Assertions.assertTrue(ind.testinggene());
    }

    @Test
    public void TestCrossover(){
        World world=new World();
        Assertions.assertTrue(world.testinEvolution());
    }

    @Test
    public void TestPopulation(){
        Population popo=new Population();
        Assertions.assertTrue(popo.testingFirstPopulation());
    }

    @Test
    public void TestSelection(){
        Selection select=new Selection();
        Assertions.assertTrue(select.testingSelection());
    }
}
```

ApplicationTest

Run: ApplicationTest

Tests passed: 6 of 6 tests – 1 s 387 ms

```
▼ Test Results                1 s 387 ms
  ▼ ApplicationTest           1 s 387 ms
      TestIndividual()             73 ms
      TestCrossover()             822 ms
      TestGene()                    1 ms
      TestSelection()             235 ms
      TestPopulation()            256 ms
      Testmutation()
```

```
/Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home/bin/java ...
Starting Sudoku Solver:
The Grid we are trying to solve:
|-----------------------|
| 9 3 0 | 0 0 0 | 5 0 6 |
| 0 0 0 | 0 0 6 | 0 3 0 |
| 2 0 0 | 5 0 9 | 0 0 0 |
|-------|-------|-------|
| 5 0 0 | 0 0 0 | 8 0 0 |
| 0 0 9 | 4 0 7 | 2 0 0 |
| 0 0 4 | 0 0 0 | 0 0 9 |
|-------|-------|-------|
| 0 0 0 | 1 0 8 | 0 0 3 |
| 0 4 0 | 6 0 0 | 0 0 0 |
| 1 0 2 | 0 0 0 | 0 5 7 |
```

Tests passed: 6 (a minute ago)

References used-

1. https://en.wikipedia.org/wiki/Genetic_algorithm.

2. https://ijcsits.org/papers/vol1no12011/2vol1no1.pdf.

3. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=538609&tag=1.

4. https://www.sciencedirect.com/science/article/pii/S0377042705000774.