SUDOKU SOLVING USING GENETIC ALGORITHM

BY TEAM 506,

BHASHMI FATNANI

KARUN KESAVADAS

KUMARI DEEPSHIKHA

Contents

1. INTRODUCTION

1.1. PROBLEM STATEMENT

This project aims to solve a Sudoku puzzle which is a NP hard problem using Genetic Algorithm.

1.2. NP HARD PROBLEMS

NP hardness is the property of class of problems that are “at least as hard as the hardest problem in non-deterministic polynomial time” a common example is an important decision problem in complexity theory and cryptography, the subset sum problem.

NP-Hard problems are problems which has no known algorithm to efficiently solve them in polynomial time and thus time to find the solution grows exponentially depending on the problem size.

Many existing decision, optimization and search problems like Travelling salesman problem, Hamiltonian path, graph coloring is categorized as NP-Hard problems.

1.3. SUDOKU

The objective this puzzle is to fill a 9x9 grid with digits in such a way that each column, each row and each of the nine 3x3 grids that make the one 9x9 grid contains all digits from 1 to 9. Initially the puzzle has some values already filled in the cells.

Rules:

1. Each row, column, and nonet can contain each number (typically 1 to 9) exactly once.
2. The sum of all numbers in any nonet, row, or column must match the small number printed in its corner. For traditional Sudoku puzzles featuring the numbers 1 to 9, this sum is equal to 45.

2. GENETIC ALGORITHM

Genetic algorithm is inspired by the theory of natural evolution proposed by Charles Darwin which states “descent with modification” by natural selection. This algorithm reflects the process of natural selection where the best fit individual is selected for reproduction to produce offspring of the next generation.

The process of natural selection starts with the selection of fittest individuals from a population. They produce different set of individuals which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their child will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals is considered.

2.1 PHASES OF GENETIC ALGORITHM

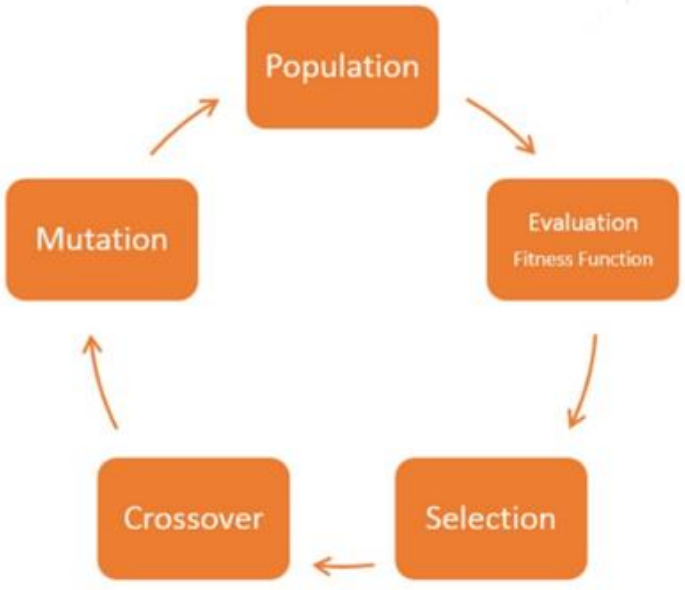


Figure 1: Phases of Genetic Algorithm

INITIAL POPULATION

Initial population is the first step in functioning of Genetic Algorithm. Population contains a set of chromosomes is the subset of solutions in the current generation. First generation is generally created randomly.

There are two methods to initialize population:

a. Random Initialization: Populating the generation by random solutions.
b. Heuristic Initialization: Populate the generation with solutions having predefined rules

FITNESS FUNCTION

Fitness function is also known as evaluation function, it evaluates how close a given solution is to optimum solution of desired problem thus determining how fit a solution is.

In genetic algorithm, each solution is string of binary numbers known as chromosome. We need to test solutions and find the best solution out of all and so the solution needs to be

attached a score which will indicate the closeness of the best solution with respect to the desired solution. This score is generated by applying the fitness function.

SELECTION

In Genetic Algorithm a chromosome(a string of binary numbers representing a solution) is selected to be parents for the next step which is crossover. Parents are selected according to the fitness assigned so, a solution/chromosome having higher fitness value has a higher probability of getting selected as a parent for crossover rather than the solution/chromosome having a low fitness value. The fit parents are the “Genotypes”.

CROSSOVER

Crossover is to create a next generation from the best individual(the parent) selected through the selection step. For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously.

The resulting child from the crossover is a “*Phenotype”*.

MUTATION

After Selection and Crossover now, you have a new set of population with better genes in individuals as compared to the first set of population or generation.

Mutation is a small random tweak in the chromosome to get a new solution. It is done to create diversity in the genetic solution to ensure that the all individuals are not the same.

3. GENETIC ALGORITHM IMPLEMENTATION

The Sudoku puzzle to be solved is read from a text file. It is represented as a collection of rows, columns and blocks.

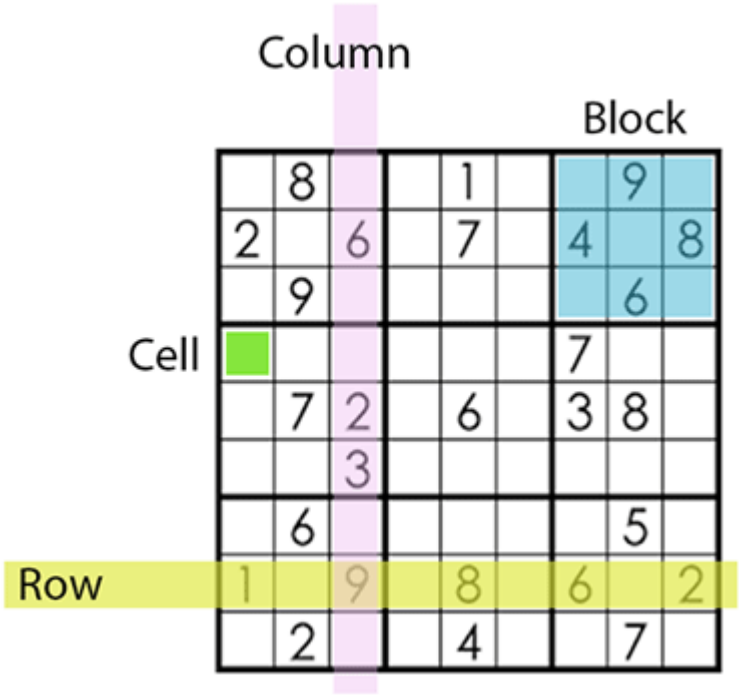


Figure 2 Sudoku Grid Annotations

3.1 INITIAL POPULATION

The empty cells in the sudoku grid are detected and a list of valid values for each cell is determined. The valid values will not conflict with the given values in its row, column or grid.

An initial population of individuals is created by filling random valid values into the empty cells. The number of conflicts for an individual is determined by counting the number of duplicate values in each row, column and block.

3.2 FITNESS FUNCTION

The fitness of an individual is determined using the formula **(216 – conflicts)/216**.

3.3 SELECTION

If we do not have an individual with 0 conflicts, we select parents for crossover using Tournament Selection. In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent.

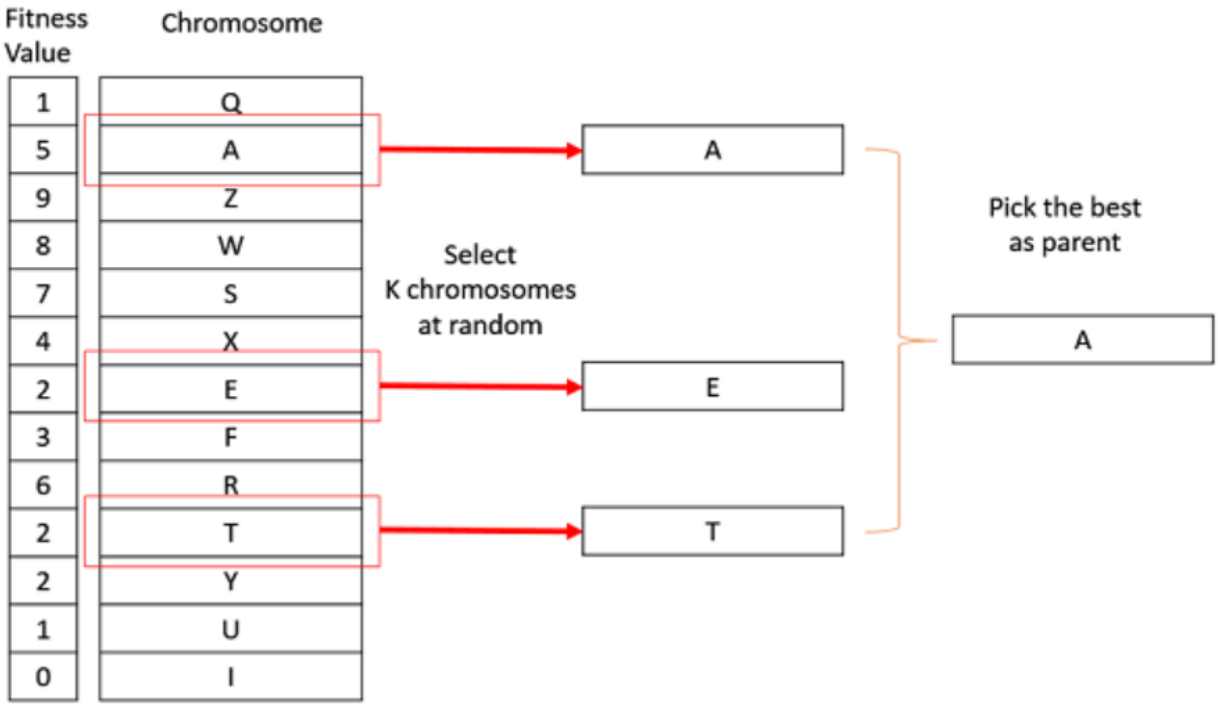


Figure 3 Tournament Selection Process

3.4 CROSSOVER

Uniform crossover is performed between the genotypes of the selected parents i.e. each empty cell of the resulting individual will have values from a random parent. The resulting individuals formed are known as the phenotypes.

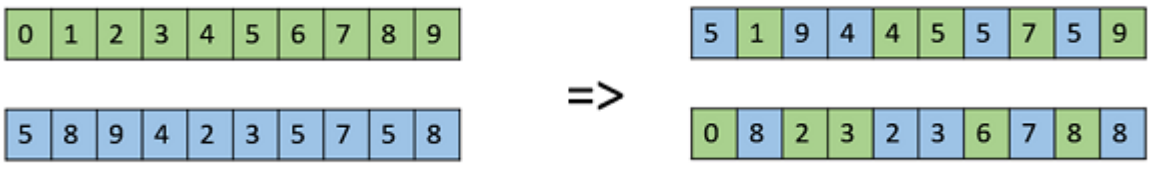


Figure 4 Process of Uniform Crossover

3.5 MUTATION

Mutation is carried out by randomly swapping the values of two cells in a row.

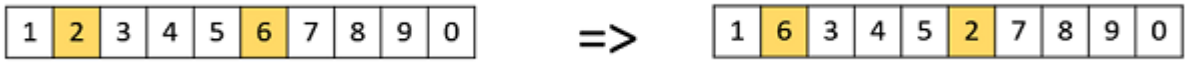


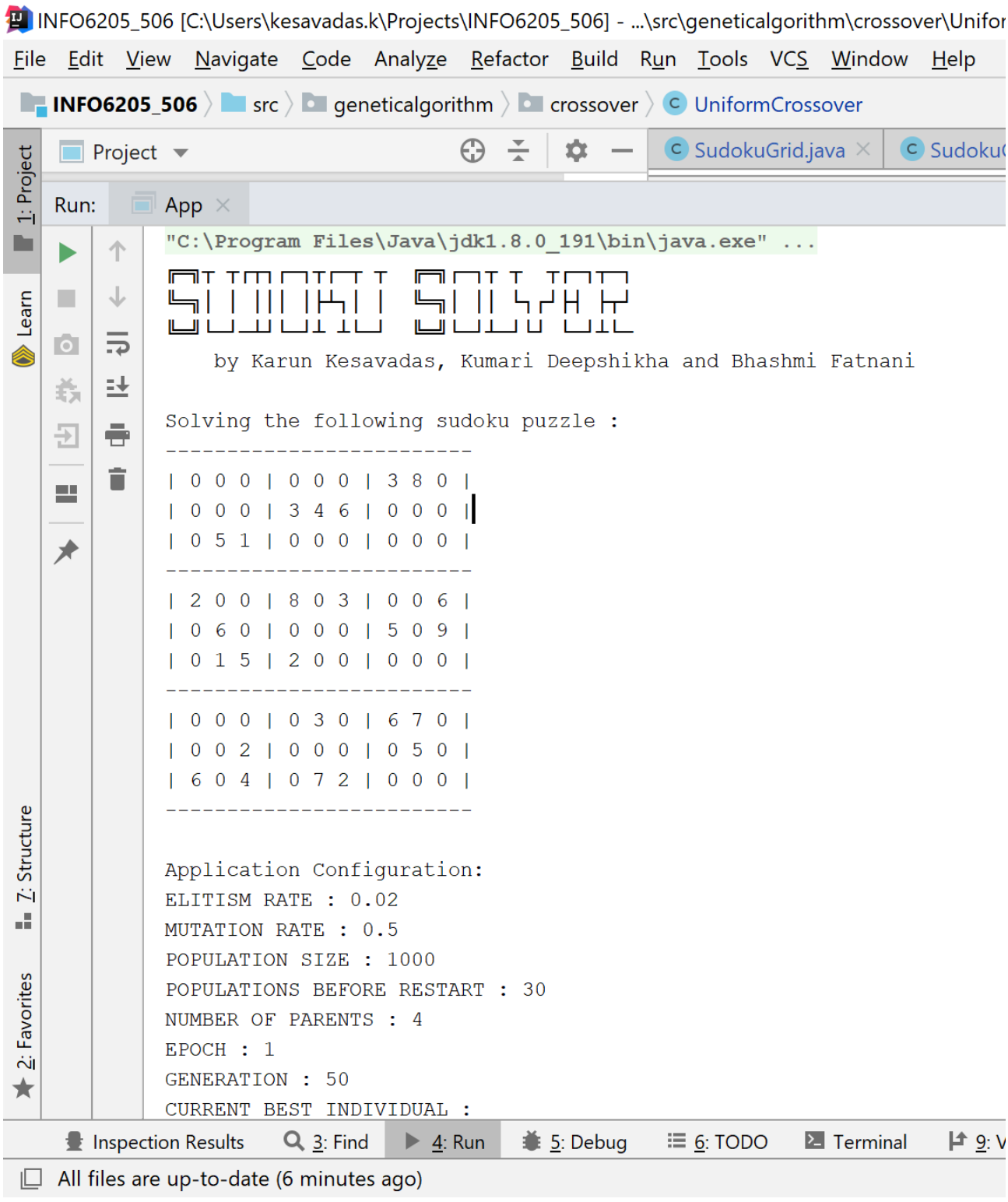
Figure 5 Swap Mutation Process

3.6 TERMINATION CONDITION

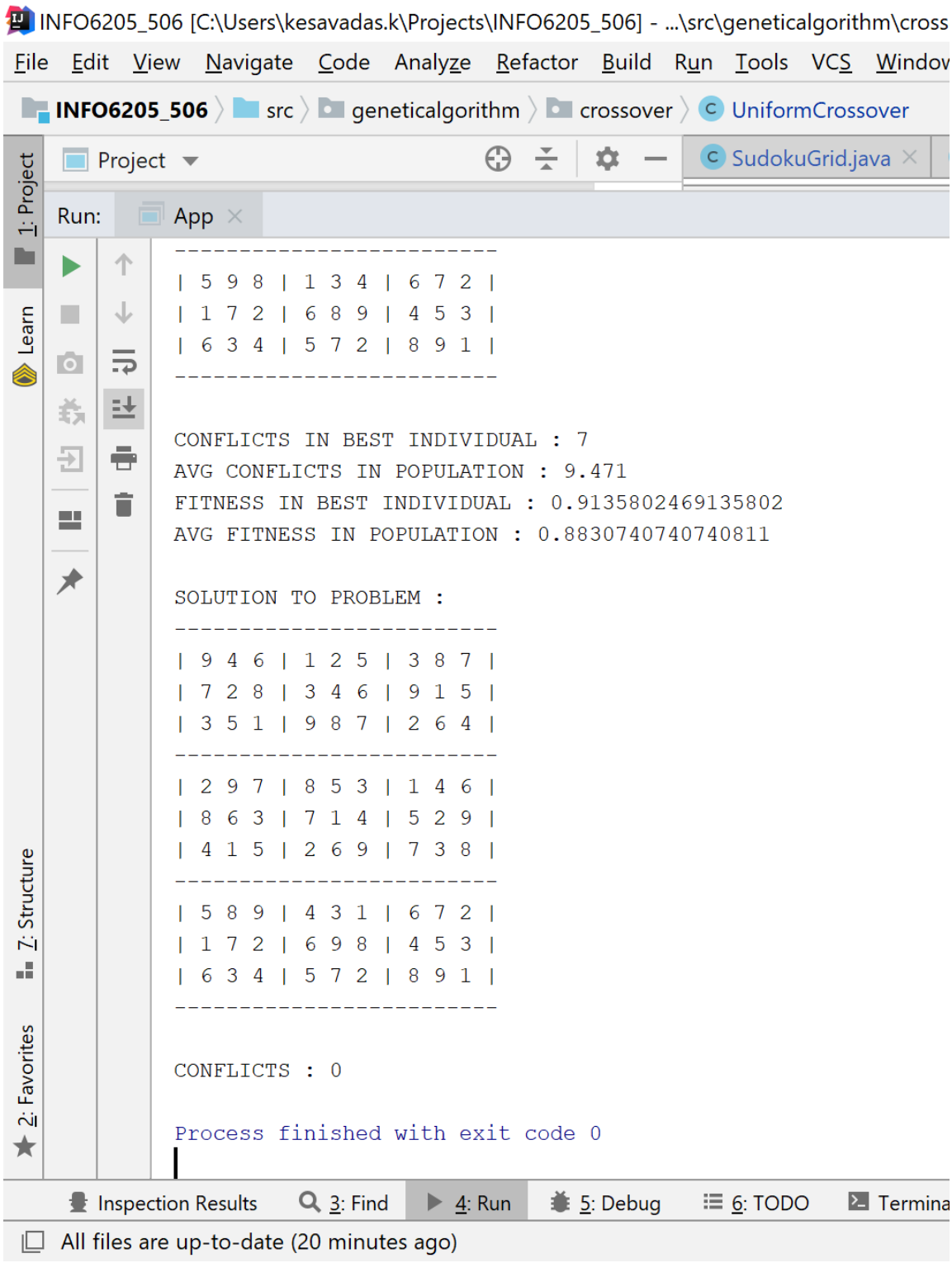
When an individual is created with fitness value of 1 i.e. having 0 conflicts, it is considered as the solution to the given puzzle. If the fitness of the current best individual is not better than the fitness of the best individual 20 generations(this value can be modified in the configuration file) ago, the population is reset with a population containing the best individuals from previous generations.

4. CONCLUSION

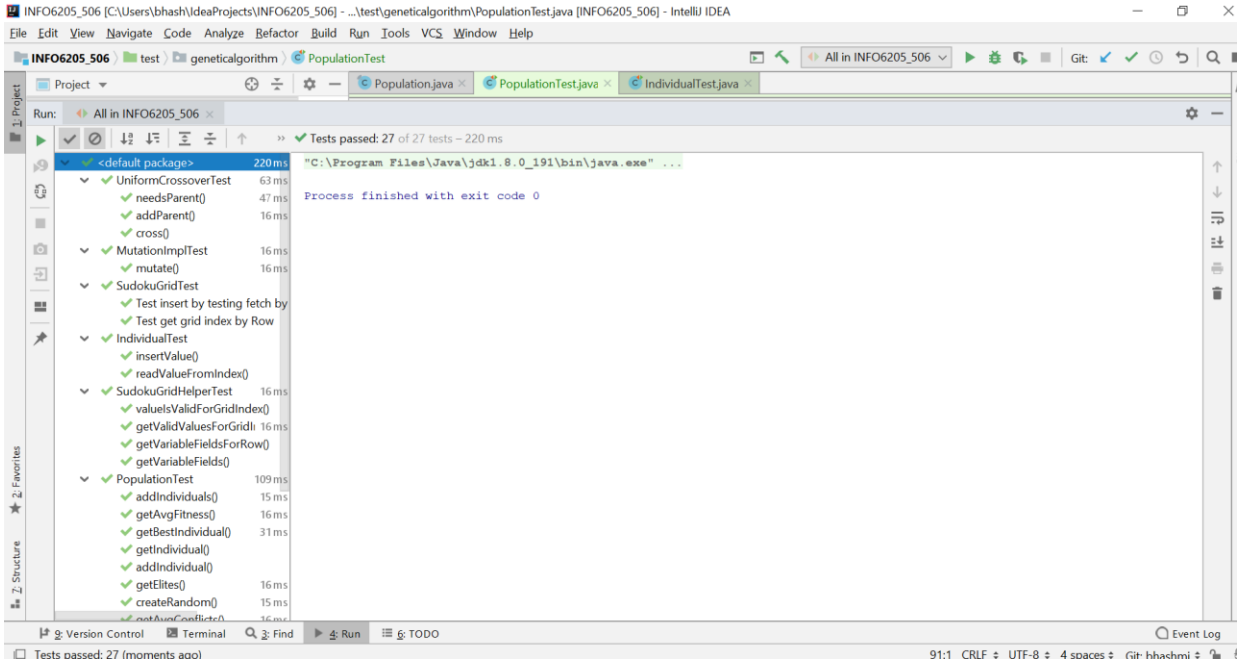
We successfully solved the sudoku puzzle given below using genetic algorithm :

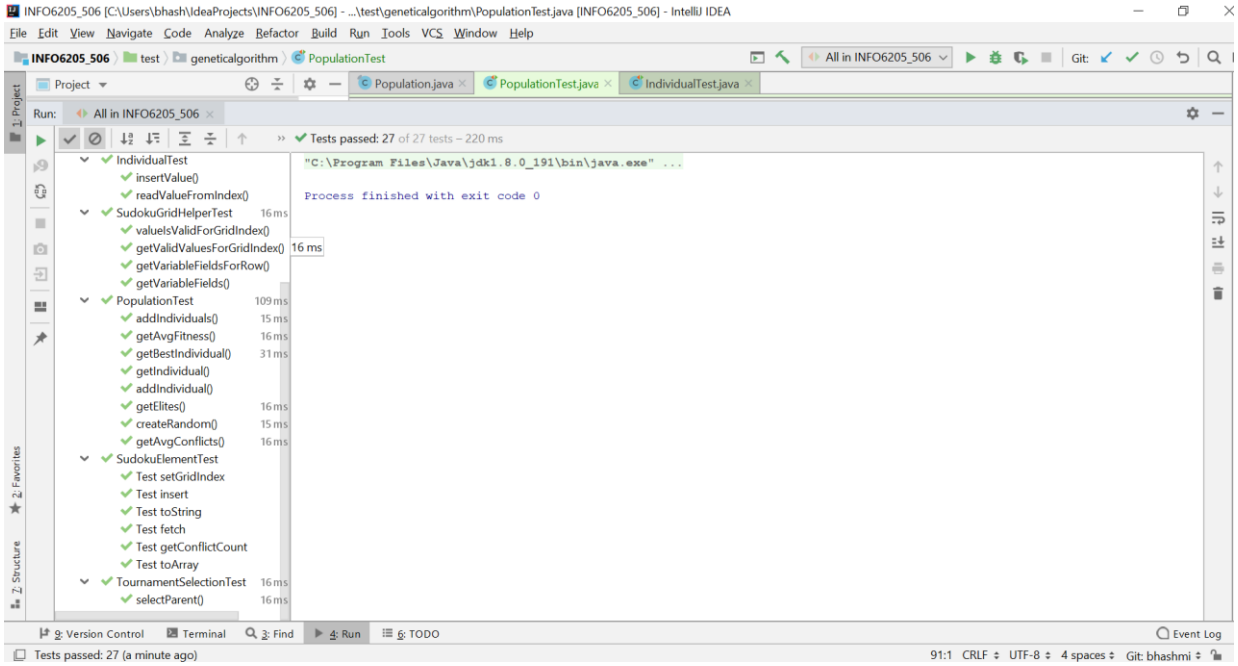


We arrived at the solution after 32 restarts and 52 generations :



5. TEST CASES


INFO6205_506 [C:\Users\bhash\IdeaProjects\INFO6205_506] - ...\test\geneticalgorithm\PopulationTest.java [INFO6205_506] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
INFO6205_506 test geneticalgorithm PopulationTest
All in INFO6205_506
Git:
Project
Population.java
PopulationTest.java
IndividualTest.java
Run: All in INFO6205_506
Tests passed: 27 of 27 tests – 220 ms
<default package> 220 ms
UniformCrossoverTest 63 ms
needsParent() 47 ms
addParent() 16 ms
cross()
MutationImplTest 16 ms
mutate() 16 ms
SudokuGridTest
Test insert by testing fetch by
Test get grid index by Row
IndividualTest
insertValue()
readValueFromIndex()
SudokuGridHelperTest 16 ms
valueIsValidForGridIndex()
getValidValuesForGridI 16 ms
getVariableFieldsForRow()
getVariableFields()
PopulationTest 109 ms
addIndividuals() 15 ms
getAvgFitness() 16 ms
getBestIndividual() 31 ms
getIndividual()
addIndividual()
getElites() 16 ms
createRandom() 15 ms
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
Process finished with exit code 0
9: Version Control
Terminal
3: Find
4: Run
6: TODO
Event Log
Tests passed: 27 (moments ago)
91:1 CRLF UTF-8 4 spaces Git: bhashmi


INFO6205_506 [C:\Users\bhash\IdeaProjects\INFO6205_506] - ...\test\geneticalgorithm\PopulationTest.java [INFO6205_506] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
INFO6205_506 test geneticalgorithm PopulationTest
All in INFO6205_506
Git:
Project
Population.java
PopulationTest.java
IndividualTest.java
Run: All in INFO6205_506
Tests passed: 27 of 27 tests – 220 ms
IndividualTest
insertValue()
readValueFromIndex()
SudokuGridHelperTest 16 ms
valueIsValidForGridIndex()
getValidValuesForGridIndex() 16 ms
getVariableFieldsForRow()
getVariableFields()
PopulationTest 109 ms
addIndividuals() 15 ms
getAvgFitness() 16 ms
getBestIndividual() 31 ms
getIndividual()
addIndividual()
getElites() 16 ms
createRandom() 15 ms
getAvgConflicts() 16 ms
SudokuElementTest
Test setGridIndex
Test insert
Test toString
Test fetch
Test getConflictCount
Test toArray
TournamentSelectionTest 16 ms
selectParent() 16 ms
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
Process finished with exit code 0
9: Version Control
Terminal
3: Find
4: Run
6: TODO
Event Log
Tests passed: 27 (a minute ago)
91:1 CRLF UTF-8 4 spaces Git: bhashmi