CIS579 Artificial Intelligence

Genetic Algorithm

Project Report by Kyrylo Krysko

CONTENTS

1. INTRODUCTION

In this assignment, we are implementing a Genetic Algorithm.

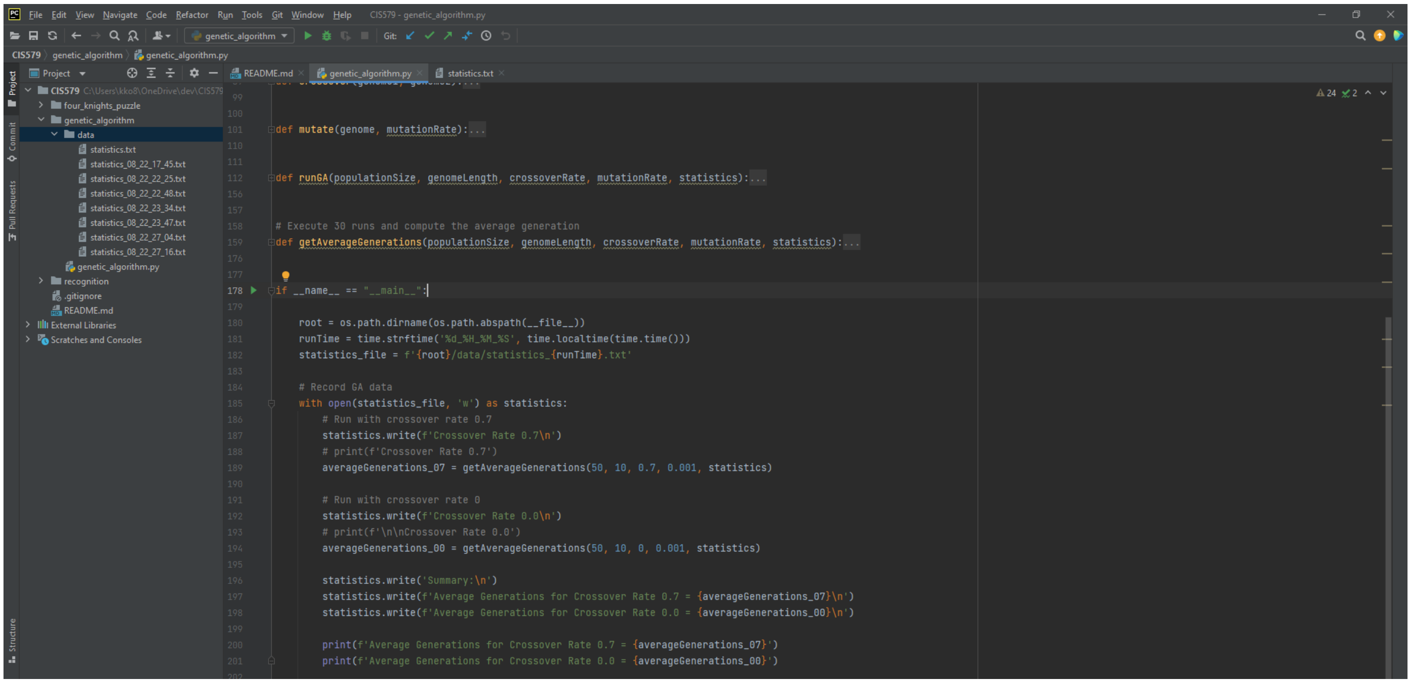
Conditions:

- Roulette-wheel sampling,
- Population size 50,
- Single-point crossover rate 0.7 and 0.0,
- Bitwise mutation rate 0.001.
- Fitness function: f(x) = number of ones in x, where x is a genome of length 10

The goal is to execute 30 runs of the Genetic Algorithm with Crossover Rate 0.7 and 30 runs with Crossover Rate 0.0 and compare results.

2. IMPLEMENTATION

The program implemented with Python3. Each time we execute the program, it runs Genetic Algorithm 30 times with crossover rate 0.7 and 0 and records statistics to a separate file: <program root>/data/statistics_<time stamp>.txt

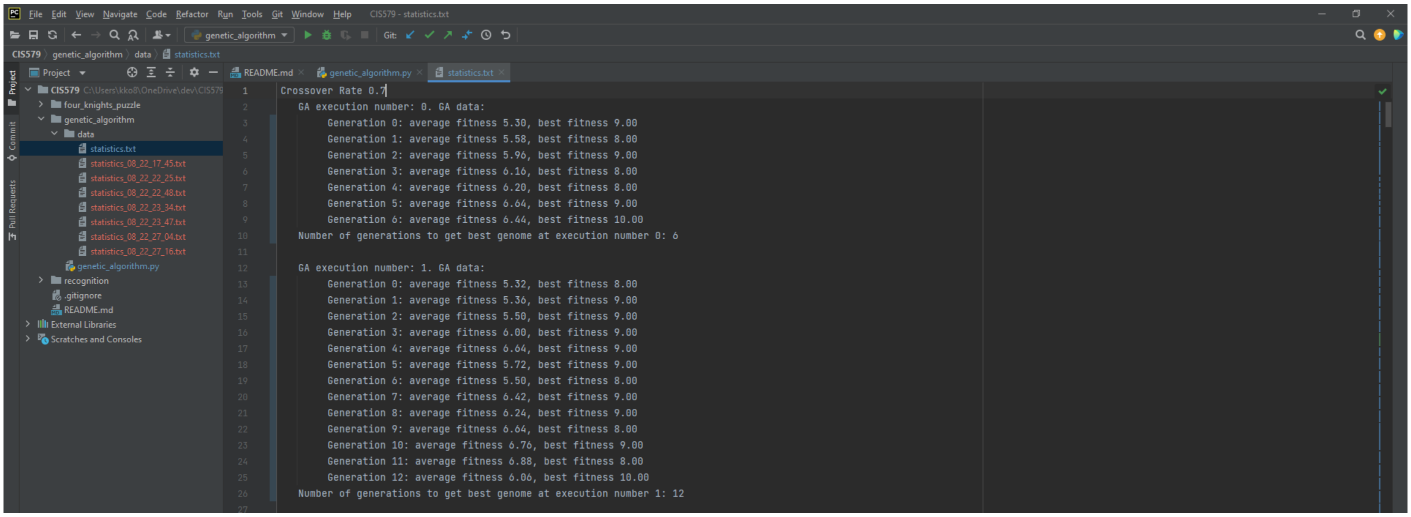

File Edit View Navigate Code Refactor Run Tools Git Window Help
CIS579 - genetic_algorithm.py
genetic_algorithm
Git:
CIS579 〉 genetic_algorithm 〉 genetic_algorithm.py
Project
README.md
genetic_algorithm.py
statistics.txt
CIS579 C:\Users\kko8\OneDrive\dev\CIS579
four_knights_puzzle
genetic_algorithm
data
statistics.txt
statistics_08_22_17_45.txt
statistics_08_22_22_25.txt
statistics_08_22_22_48.txt
statistics_08_22_23_34.txt
statistics_08_22_23_47.txt
statistics_08_22_27_04.txt
statistics_08_22_27_16.txt
genetic_algorithm.py
recognition
.gitignore
README.md
External Libraries
Scratches and Consoles
Commit
Pull Requests
Structure
⚠24 ✓2

```python
def mutate(genome, mutationRate):...


def runGA(populationSize, genomeLength, crossoverRate, mutationRate, statistics):...


# Execute 30 runs and compute the average generation
def getAverageGenerations(populationSize, genomeLength, crossoverRate, mutationRate, statistics):...


if __name__ == "__main__":

    root = os.path.dirname(os.path.abspath(__file__))
    runTime = time.strftime('%d_%H_%M_%S', time.localtime(time.time()))
    statistics_file = f'{root}/data/statistics_{runTime}.txt'

    # Record GA data
    with open(statistics_file, 'w') as statistics:
        # Run with crossover rate 0.7
        statistics.write(f'Crossover Rate 0.7\n')
        # print(f'Crossover Rate 0.7')
        averageGenerations_07 = getAverageGenerations(50, 10, 0.7, 0.001, statistics)

        # Run with crossover rate 0
        statistics.write(f'Crossover Rate 0.0\n')
        # print(f'\n\nCrossover Rate 0.0')
        averageGenerations_00 = getAverageGenerations(50, 10, 0, 0.001, statistics)

        statistics.write('Summary:\n')
        statistics.write(f'Average Generations for Crossover Rate 0.7 = {averageGenerations_07}\n')
        statistics.write(f'Average Generations for Crossover Rate 0.0 = {averageGenerations_00}\n')

        print(f'Average Generations for Crossover Rate 0.7 = {averageGenerations_07}')
        print(f'Average Generations for Crossover Rate 0.0 = {averageGenerations_00}')
```

The program code attached to submission.

3. RESULTS

The program was executed 7 times and results gathered in the table below. Full list of statistics files attached to submission.



```
Crossover Rate 0.7
   GA execution number: 0. GA data:
        Generation 0: average fitness 5.30, best fitness 9.00
        Generation 1: average fitness 5.58, best fitness 8.00
        Generation 2: average fitness 5.96, best fitness 9.00
        Generation 3: average fitness 6.16, best fitness 8.00
        Generation 4: average fitness 6.20, best fitness 8.00
        Generation 5: average fitness 6.64, best fitness 9.00
        Generation 6: average fitness 6.44, best fitness 10.00
   Number of generations to get best genome at execution number 0: 6

   GA execution number: 1. GA data:
        Generation 0: average fitness 5.32, best fitness 8.00
        Generation 1: average fitness 5.36, best fitness 9.00
        Generation 2: average fitness 5.50, best fitness 9.00
        Generation 3: average fitness 6.00, best fitness 9.00
        Generation 4: average fitness 6.64, best fitness 9.00
        Generation 5: average fitness 5.72, best fitness 9.00
        Generation 6: average fitness 5.50, best fitness 8.00
        Generation 7: average fitness 6.42, best fitness 9.00
        Generation 8: average fitness 6.24, best fitness 9.00
        Generation 9: average fitness 6.64, best fitness 8.00
        Generation 10: average fitness 6.76, best fitness 9.00
        Generation 11: average fitness 6.88, best fitness 8.00
        Generation 12: average fitness 6.06, best fitness 10.00
   Number of generations to get best genome at execution number 1: 12
```

Program Execution Attempt	Average Generations for Crossover 0.7	Average Generations for Crossover 0.0
1	16.6	30
2	15.9	29
3	14.2	25.9
4	17.9	28.7
5	17.0	27.2
6	15.7	25.5
7	15.2	27.6

4. CONCLUSION

As we can see from 7 program execution with crossover rate = 0 we get best genome with much more generations.

If crossover rate is 0, this means the genetic algorithm will not use the crossover operation at all. In this case, the genetic algorithm will rely only on mutation (and the selection of existing good solutions) to explore the search space. This may slow down the rate of improvement in the population, as potentially beneficial combinations of genes from different individuals cannot be explored via crossover.