PROJECT REPORT

INFO6205 Programming Structures and Algorithms

DECEMBER 7, 2019

SHUBHANKAR DANDEKAR | KARAN RACCA | URVASHI JAIN
dandekar.s@husky.neu.edu | racca.k@husky.neu.edu | jain.ur@husky.neu.edu

Objective:

Find an initial pattern that grows the most in the game of life.

Terminology:

***Cell*:** Each individual on the pattern is a cell. A cell can either be alive or dead. An alive cell is represented by white color, while a dead cell is represented by black color.

***Pattern*:** Pattern is a two-dimensional grid of cells.

Process:

Summary: We used genetic algorithms to start with randomly generated patterns to reach a possible initial pattern that grows the most when the rules of Conway's Game of Life is applied to their cells.

Coefficients:

GRID_SIZE: Board size of the Game of Life

MAX_GENERATIONS: Maximum number of Generations the genetic algorithms should run

POPULATION_SIZE: Size of the population in each generation

MUTATION_COEFFICIENT: Percentage by which to mutate a pattern for the next generation

MUTATION_FACTOR: Probability for a pattern to be mutated

INITIAL_FACTOR: Factor of the board *GRID_SIZE (Board Size)* which would be used as the initial pattern

MAX_ALIVE: Stop the game of life when percent of living cells go beyond.

MIN_ALIVE: Stop the game of life when percent of living cells go below

TOP_FITTEST_COEFFICIENT: Percent of population taken to the next generation

LEAST_FITTEST_COEFFICIENT: Percent of population to be filled with new random initial patterns

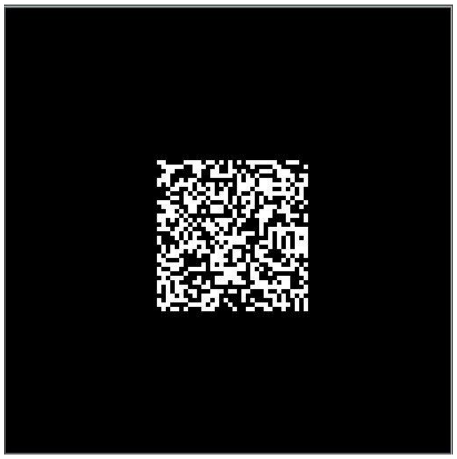

Figure 1: Example of a Pattern with GRID_SIZE=100; INITIAL_FACTOR=3 i.e. a 33 X 33 grid

Steps:

1. Generate **POPULATION_SIZE** random initial patterns of size ***GRID_SIZE/INITIAL_PATTERN X GRID_SIZE/INITIAL_PATTERN*** on board of size ***GRID_SIZE X GRID_SIZE.*** This acts as the 1st generation in the genetic algorithm
2. Each of the initial patterns goes through the game of life. The game of life terminates when the percent of living cells is between ***MIN_ALIVE*** and ***MAX_ALIVE*** or when a repeated pattern is detected. Game of Life acts as a fitness function which assigns a fitness score to each of the initial patterns. The fitness score is the average number of living cells. Higher the fitness score, more does the initial pattern grow
3. The initial pattern with highest fitness score is set as the best possible initial pattern that grows the most
4. Now when we have fitness scores of the entire population, top ***TOP_FITTEST_COEFFICIENT*** percent of the total population with the highest fitness scores are selected as candidates for the next generation
5. More ***LEAST_FITTEST_COEFFICIENT*** percent are populated with new random initial patterns. While the remaining are selected randomly from the current population using the concept of the **Wheel of Fortune.** *The Wheel of Fortune is a concept where an item is selected randomly from a collection of items, where the probability of selecting an item is high if its fitness score is high.*
6. Each of the candidate, except the ***TOP_FITTEST_COEFFICIENT*** percent of the new population, has a probability of ***MUTATION_FACTOR*** to be mutated. If a pattern does mutate, it mutates by reversing the state of ***MUTATION_COEFFICIENT*** percent of total cells in the initial pattern; i.e. a living cell dies, and a dying cell comes to life
7. This contributes to the new population for the next generation.
8. Step 2 through Step 6 are repeated for ***MAX_GENERATIONS*** times
9. In the end of the program, we get one of the best initial patterns that grows the most in the game of life

Running the program

1. Run *Runner.java* with an argument **false,** after the setting the coefficients as per your test in *Library.java*
2. Wait for the algorithm to continue for ***MAX_GENERATIONS.*** You can stop the algorithm mid-way which gives you the best initial pattern until that generation being run
3. In the end, “image.bmp” is saved, which is the solution; i.e. one of the best initial patterns to grow the most in the game of life
4. You can run the game of life with this solution by running the *Runner.java* with an argument **true**
5. This is a visual demonstration of the initial pattern growing with the rules of game of life

<u>Our example:</u>

Coefficients for our example:

GRID_SIZE: 100

MAX_GENERATIONS: 100

POPULATION_SIZE: 100

MUTATION_COEFFICIENT: 0.2

MUTATION_FACTOR: 40

INITIAL_FACTOR: 15

MAX_ALIVE: 95

MIN_ALIVE: 1

TOP_FITTEST_COEFFICIENT: 10

LEAST_FITTEST_COEFFICIENT: 10

Observations:

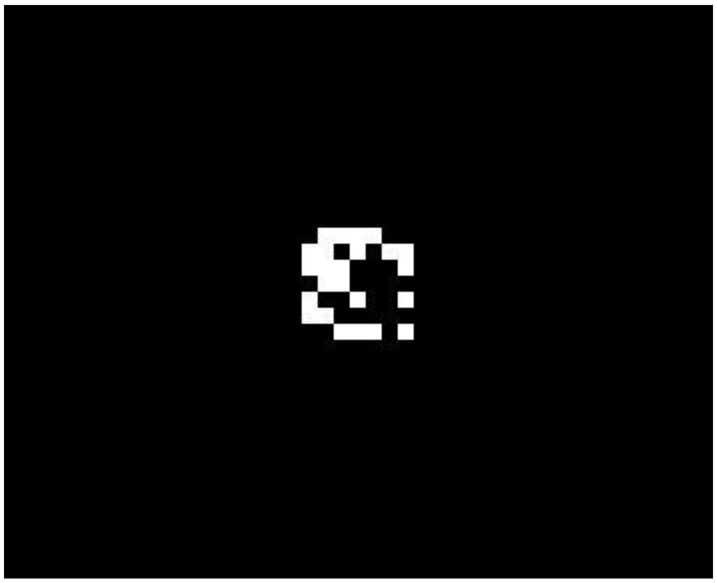

Figure 2: The best possible initial pattern

The Genetic Algorithm generated 100 random 7 X 7 (***100/15***) initial patterns. Each pattern went through the game of life to return a fitness score. The game of life is played on a board of size 100 X 100. After 100 generations, it came up with a possible solution, i.e. an initial pattern that will grow the most.

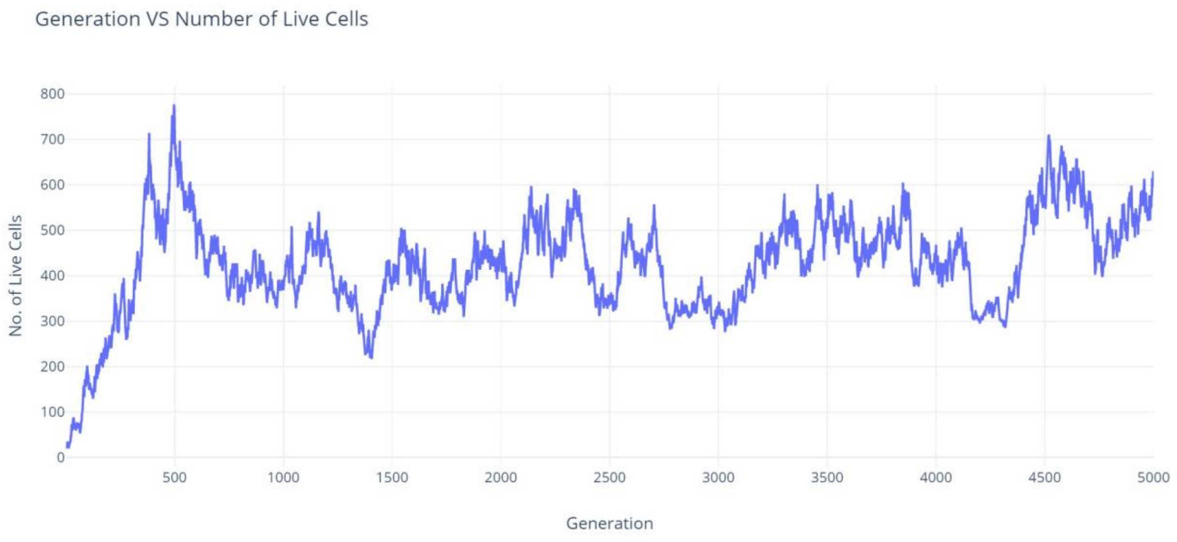

Figure 3: Plot of growth of the initial pattern shown in Figure 1

As it can be observed, the best initial pattern which has about 20 living cells, grows for about 5000 generations. It grows as high as 800 living cells. Also, on an average about 500 cells are alive throughout the game. Hence, it proves that the genetic algorithm comes up with an initial pattern that grows the most.