

*Faculty of Engineering and Technology*

*Computer Science Department*

*COMP338 - Artificial Intelligence*

**Project 1-Genetic Algorithm**

**Prepared by:**

Nour Manasrah, 1211163

Raymaa Elian, 1200798

**Instructor:** Dr. Mohammad Helal

**Table of Contents**

[**1. Introduction** 4](#_Toc186492647)

[**2. Background** 4](#_Toc186492648)

[**2.1 What is a Genetic Algorithm?** 4](#_Toc186492649)

[**2.2 Main Components of a Genetic Algorithm** 4](#_Toc186492650)

[**3. Problem Formulation** 5](#_Toc186492651)

[**3.1 Password Representation** 5](#_Toc186492652)

[**3.2 Chromosome, Gene, and Fitness Function** 5](#_Toc186492653)

[**4. Implementation Details** 5](#_Toc186492654)

[**4.1 Initialization** 5](#_Toc186492655)

[**4.2 Fitness Evaluation** 6](#_Toc186492656)

[**4.3 Selection** 6](#_Toc186492657)

[**4.4 Crossover** 6](#_Toc186492658)

[**4.5 Mutation** 6](#_Toc186492659)

[**4.6 Termination Criteria** 6](#_Toc186492660)

[**5. Parameter Tuning Analysis** 7](#_Toc186492661)

[**6. Results and Discussion** 7](#_Toc186492662)

[**6.1** **Performance Metrics** 7](#_Toc186492663)

[**6.2 Convergence Rate** 7](#_Toc186492664)

[**7. Snapshots of Program Execution** 8](#_Toc186492665)

[**8. Conclusion** 9](#_Toc186492666)

[**9. References** 9](#_Toc186492667)

**Table Of Figures**

[Figure 1: genetic algorithm 4](#_Toc186490794)

[Figure 2: genetic algorithm operations 5](#_Toc186490795)

[Figure 3: genetic algorithm components 5](#_Toc186490796)

[Figure 4: first result 8](#_Toc186490797)

[Figure 5: second result 8](#_Toc186490798)

# **1. Introduction**

This project implements a password-breaking algorithm using a Genetic Algorithm. It seeks to arrive at a randomly generated 32-bit password through an iterative improvement of guesses in conformance with the principles of natural selection and evolution. This project demonstrates the strength of GAs in solving difficult optimization problems within the domain of Artificial Intelligence.

# **2. Background**

## **2.1 What is a Genetic Algorithm?**

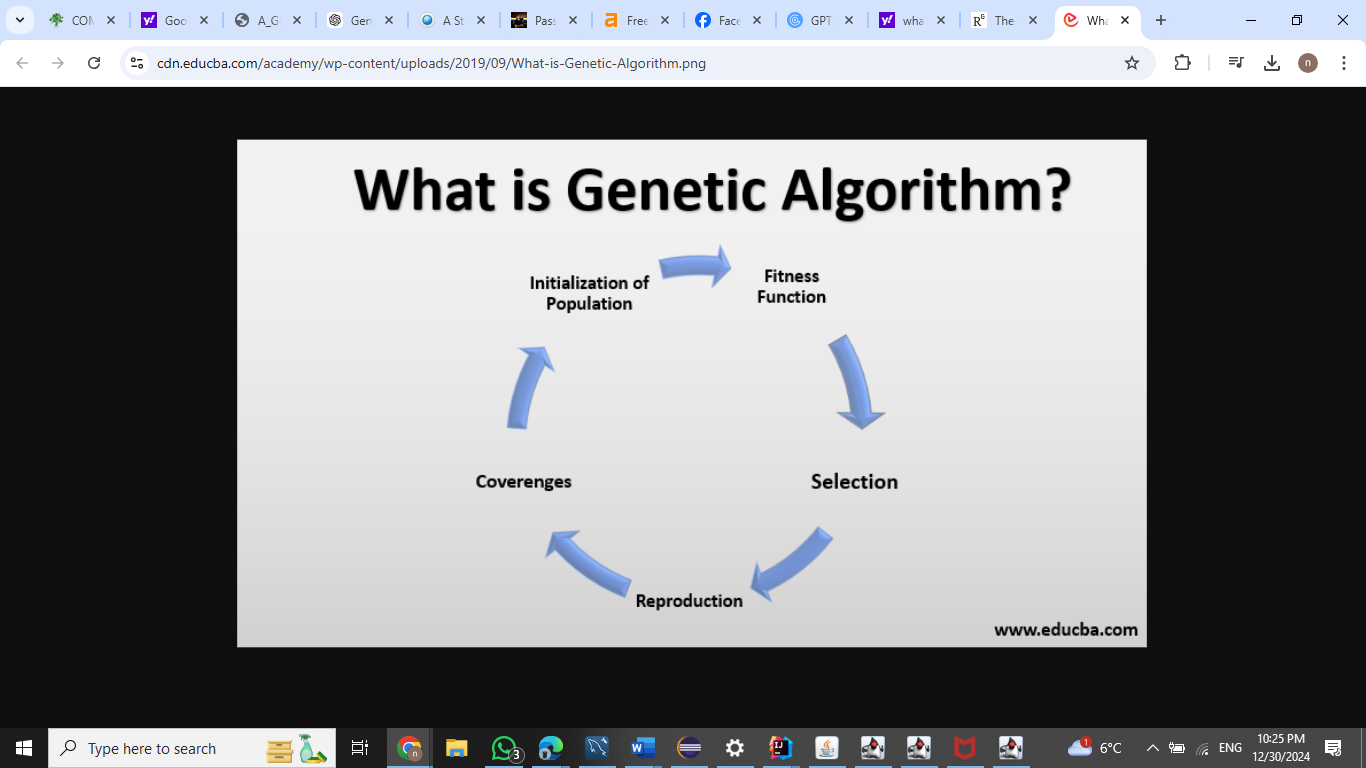
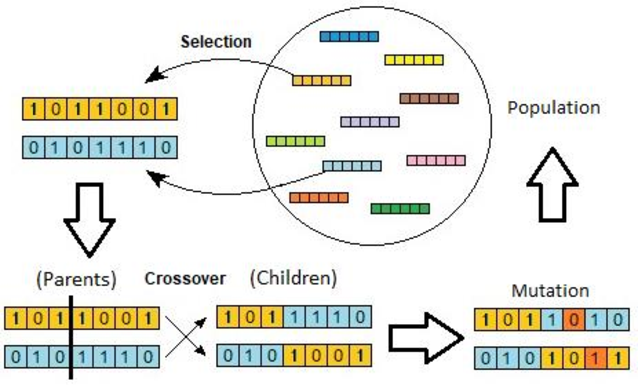
A Genetic Algorithm is a search heuristic that draws inspiration from Charles Darwin's theory of natural selection. It utilizes biological concepts such as reproduction, mutation, and survival of the fittest in evolving solutions to computational problems over successive generations.

Figure 1: genetic algorithm

## **2.2 Main Components of a Genetic Algorithm**

* Population: The set of individuals (solutions).
* Chromosome: The individual in question, represented by a 32-bit sequence.
* Gene: An individual element (bit) of the chromosome.
* Fitness Function: Measures the quality of a solution.
* Selection: Process of choosing the best individuals for reproduction.
* Crossover: Combining genes of two parents to produce offspring.
* Mutation: Randomly altering genes to introduce diversity.



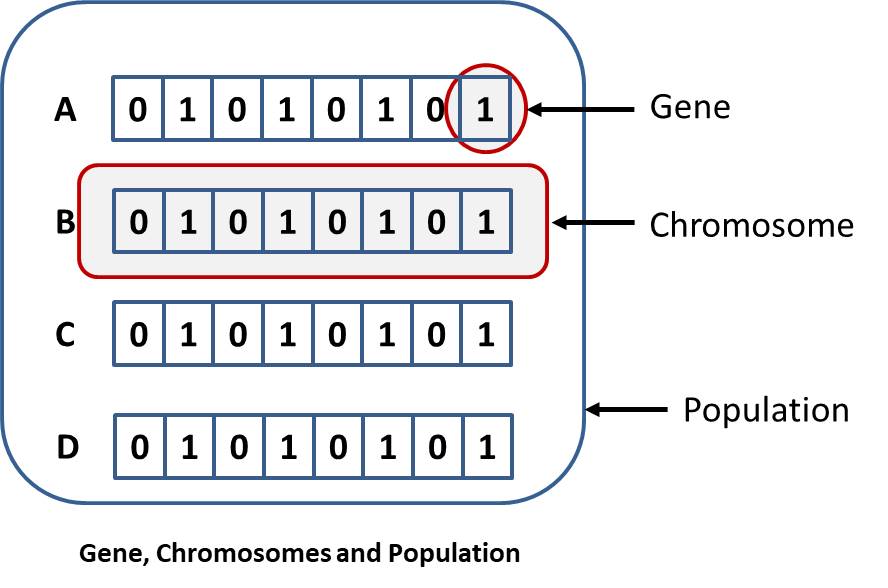


Figure 2: genetic algorithm operations

Figure 3: genetic algorithm components

# **3. Problem Formulation**

## **3.1 Password Representation**

The password is represented as a 32-bit binary sequence, e.g., 10101011010101110101010101101010. Each potential guess is treated as a chromosome in the population.

## **3.2 Chromosome, Gene, and Fitness Function**

* Chromosome: A 32-bit binary sequence (guess).
* Gene: A single bit in the chromosome.
* Fitness function: Counts the number of matching bits between a guess and the target password.

# **4. Implementation Details**

## **4.1 Initialization**

Create an initial population of random 32-bit binary sequences

Population size = 101.

## **4.2 Fitness Evaluation**

For each individual of the population, evaluate the fitness as a number of bits matching the target password.

## **4.3 Selection**

Select the individuals with higher fitness for reproduction using roulette wheel selection or tournament selection.

## **4.4 Crossover**

Perform single-point crossover between a pair of selected parents in order to generate offspring. Example:

Parent 1: 10101010 | 11111111

Parent 2: 00000000 | 11001100

Offspring: 10101010 | 11001100

## **4.5 Mutation**

Introduce random changes on small percentages of genes in the offspring. Example: Flip a bit from 0 to 1.

## **4.6 Termination Criteria**

Stop the algorithm at:

* A guess matches the target password.
* A maximum number of generations is reached.

# **5. Parameter Tuning Analysis**

In our analysis of parameter tuning, we found that changing population sizes, crossover rates, and mutation rates significantly affected convergence. Larger populations enhanced diversity and helped avoid local optima, but they also raised computational costs. Smaller populations, on the other hand, faced the risk of stagnation. Increasing crossover rates boosted exploration but could sometimes disturb effective solutions. Mutation rates over 5% frequently led to divergence by making promising solutions too random. The convergence plots showing generations against fitness illustrated the balance needed between exploration, exploitation, and computational efficiency, highlighting the importance of well-adjusted parameters.

# **6. Results and Discussion**

## **Performance Metrics**

**-Time to convergence:**

Based on the console output, the algorithm converged to a solution in **7 generations**.

**-Fitness achieved:**

The best fitness achieved is **30**, and the algorithm identified the solution as 10011100000001011000111001010001.

## **6.2 Convergence Rate**

The convergence rate measures how quickly the algorithm improves the fitness score over generations.

**Formula:**

Convergence Rate= (Final Fitness−Initial Fitness)/Generations to Converge

**Calculation:**

* Initial Fitness: 22(from Generation 0)
* Final Fitness: 30 (from Generation 6/7)
* Generations to Converge: 7

Convergence Rate=30−22/7=8/7≈1.14 fitness units per generation

# **7. Snapshots of Program Execution**

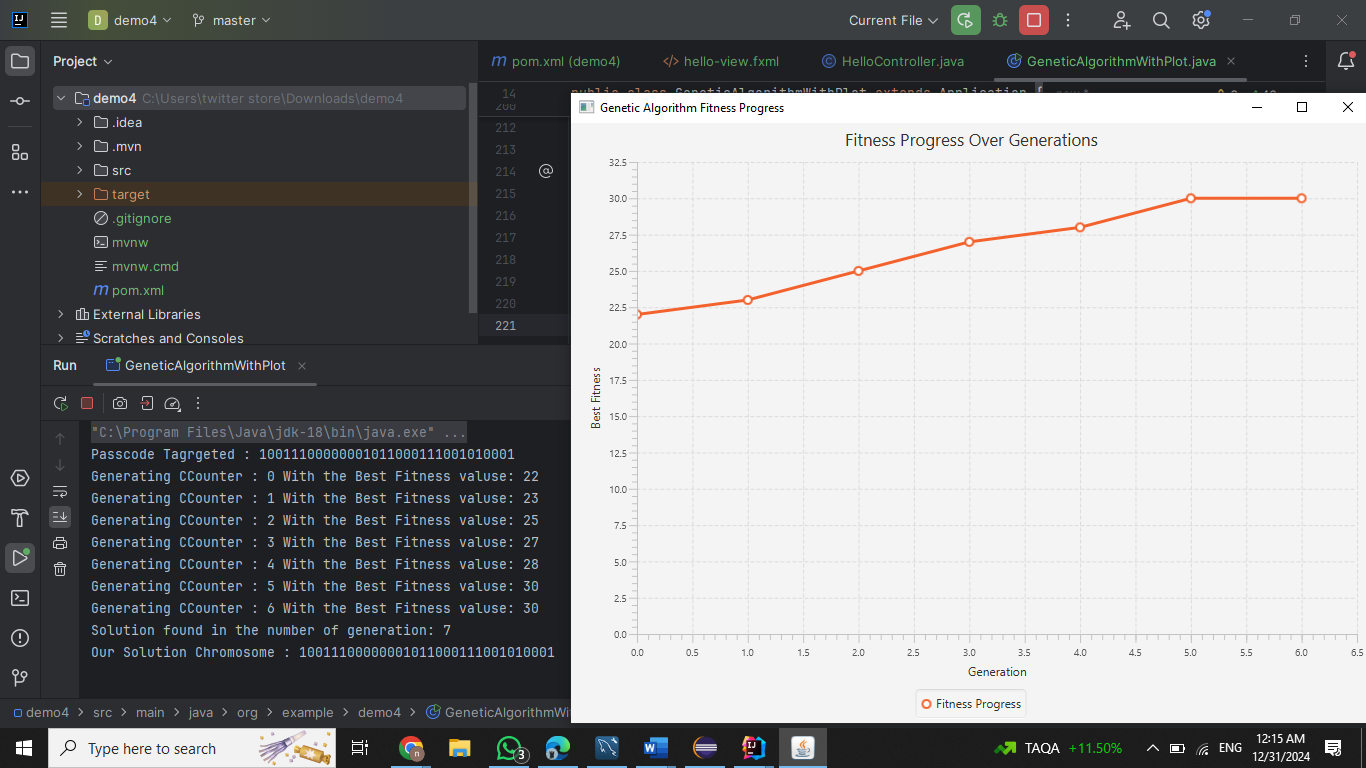


Figure 4: first result

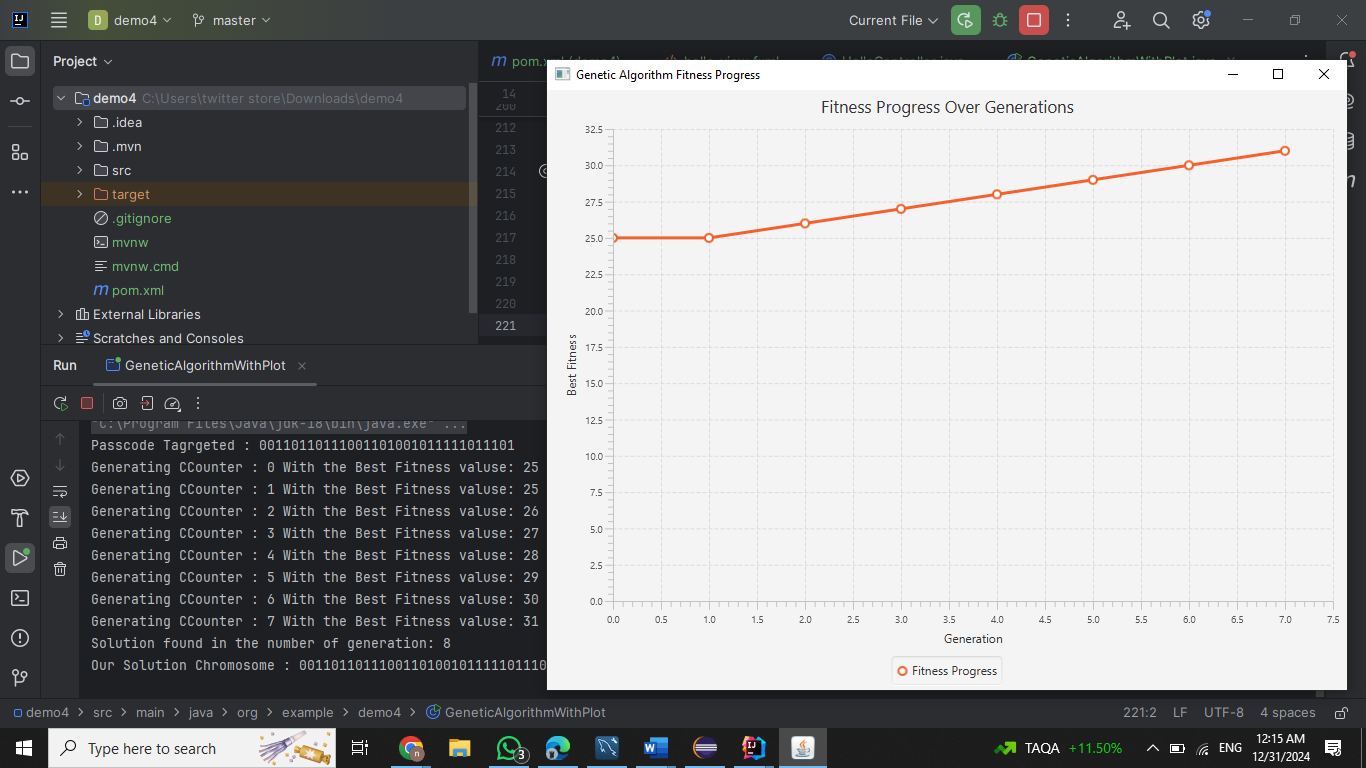


Figure 5: second result

# **8. Conclusion**

Finally, the Genetic Algorithm was able to find the 32-bit password, which proved it worthy for optimization problems. Also, the project highlighted parameter tuning and the balance between exploration and exploitation.

# **9. References**

[1]. <https://www.geeksforgeeks.org/genetic-algorithm/>

[2]. <https://towardsdatascience.com/optimization-using-genetic-algorithms-11e0b443aa39>

[3]. <https://research.ijcaonline.org/volume118/number20/pxc3903559.pdf>

[4]. <https://www.researchgate.net/figure/stages-of-genetic-algorithm_fig2_360918320>

[5]. <https://www.educba.com/what-is-genetic-algorithm/>

[6]. <https://www.kindsonthegenius.com/basics-of-genetic-algorithm-ga-explained-in-simple-terms/>