ARTIFICIAL INTLLIGENCE PROJECT REPORT

SUBMITTED BY:

DOA AHMED CS-067
HIBA MAHBOOB CS-068
MUHAMMAD AHMED KHAN CS-128

Submitted to :
Miss Hameeza

TABLE OF CONTENTS

ABSTRACT

Sudoku is a combinatorial puzzle where the goal is to fill a 9x9 grid with digits from 1 to 9, ensuring that each number appears only once in each row, column, and 3x3 subgrid. Solving these puzzles can be computationally intensive due to the large solution space, especially for more complex configurations. Genetic Algorithms (GAs), inspired by natural evolution, provide an effective method for solving such optimization problems. Using selection, crossover, and mutation, GAs evolve potential solutions by iteratively improving a population of candidate Sudoku grids. A fitness function evaluates each grid based on how well it satisfies Sudoku's constraints, guiding the algorithm toward a valid solution with fewer conflicts over generations. This approach offers a novel and efficient way to solve Sudoku puzzles and can be adapted to different puzzle complexities.

GENETIC ALGORITHM

Genetic Algorithm (GA) is a metaheuristic optimization technique inspired by the process of natural selection and evolution. It works by creating a population of potential solutions to a problem and evolving them over generations through mechanisms like selection, crossover (recombination), and mutation. Each solution is evaluated using a fitness function to measure its quality. The best solutions are selected for reproduction, while less fit solutions are discarded. Over successive generations, the population converges towards an optimal or near-optimal solution. GAs are widely used for solving complex optimization problems where traditional methods may struggle.

Genetic algorithm’s example used is a sudoko game.In a genetic algorithm for solving Sudoku, a population of grid configurations is generated, with each grid evaluated based on how many Sudoku constraints it satisfies. Over generations, the algorithm uses crossover and mutation to create new grids, gradually improving the solutions until a valid Sudoku grid is found. The fitness function guides the process by penalizing grid configurations with rule violations.

GENETIC ALGORITHM SUDOKO SOLVER

The project consists of the following main parts:

1. Defining genes and chromosomes
2. Making the first generation
3. Fitness function
4. Crossover and mutation
5. Implementing the genetic algorithm

Defining Genes and Chromosomes

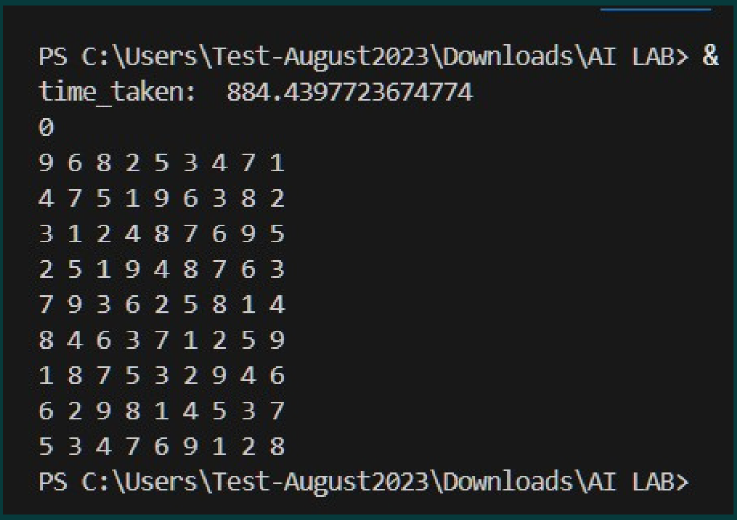
A gene represents a row of the Sudoku puzzle and is a permutation of the set {1,2,3,4,5,6,7,8,9}. A chromosome consists of 9 genes, each gene representing a row of the actual Sudoku puzzle. The `make_gene` function creates a gene, while the `make_chromosome` function creates a chromosome.

Making First Generation

The `make_population` function creates a population of a specified size. Each member of the population is a Sudoku puzzle represented as a chromosome. **Fitness Function** The fitness function calculates how "fit" a chromosome (puzzle) is based on the following criteria: - For each column: subtract (number of times a number is seen) - 1 from the fitness for that number - For each 3x3 square: subtract (number of times a number is seen) - 1 from the fitness for that number. The higher the fitness, the closer the puzzle is to being solved. **Crossover and Mutation** The crossover function takes two chromosomes as input and makes two new chromosomes by combining them. This crossover function decides the parent of each gene separately, so the result is independent of the location of the genes.The mutation function applies a random change to a chromosome with a specified probability. In this case, the mutationfunction randomly changes a gene in a chromosome.

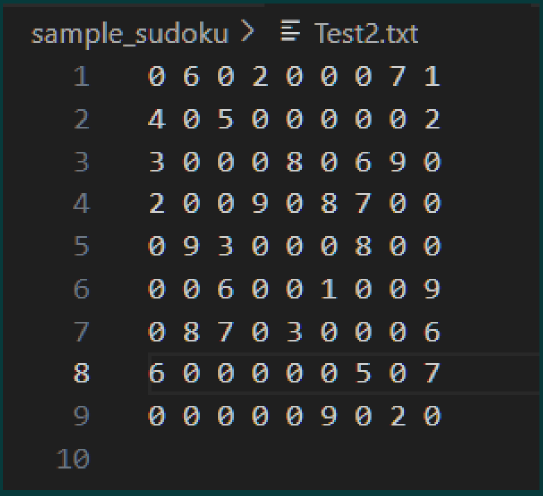
Implementing The Genetic Algorithm The genetic algorithm consists of the following steps: 1. Read the puzzle from the input file. 2. Create the initial population. 3. Calculate the fitness of each chromosome. 4. Select the mating pool from the current population.

TEST RUN



```
PS C:\Users\Test-August2023\Downloads\AI LAB> &
time_taken:  884.4397723674774
0
9 6 8 2 5 3 4 7 1
4 7 5 1 9 6 3 8 2
3 1 2 4 8 7 6 9 5
2 5 1 9 4 8 7 6 3
7 9 3 6 2 5 8 1 4
8 4 6 3 7 1 2 5 9
1 8 7 5 3 2 9 4 6
6 2 9 8 1 4 5 3 7
5 3 4 7 6 9 1 2 8
PS C:\Users\Test-August2023\Downloads\AI LAB>
```

INITIAL GRID



sample_sudoku > Test2.txt

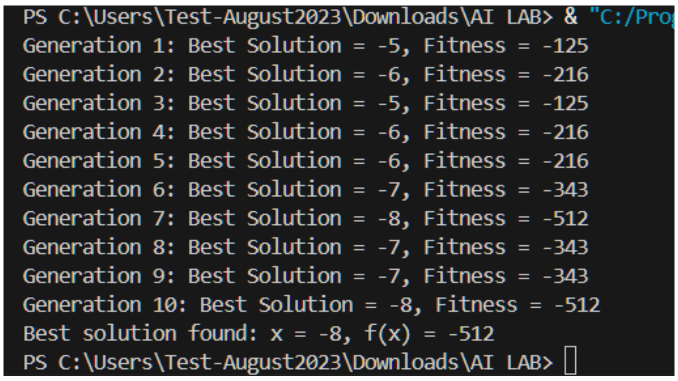
```
0 6 0 2 0 0 0 7 1
4 0 5 0 0 0 0 0 2
3 0 0 0 8 0 6 9 0
2 0 0 9 0 8 7 0 0
0 9 3 0 0 0 8 0 0
0 0 6 0 0 1 0 0 9
0 8 7 0 3 0 0 0 6
6 0 0 0 0 0 5 0 7
0 0 0 0 0 9 0 2 0
```

SIMPLE EQUATION IMPLEMENTATION

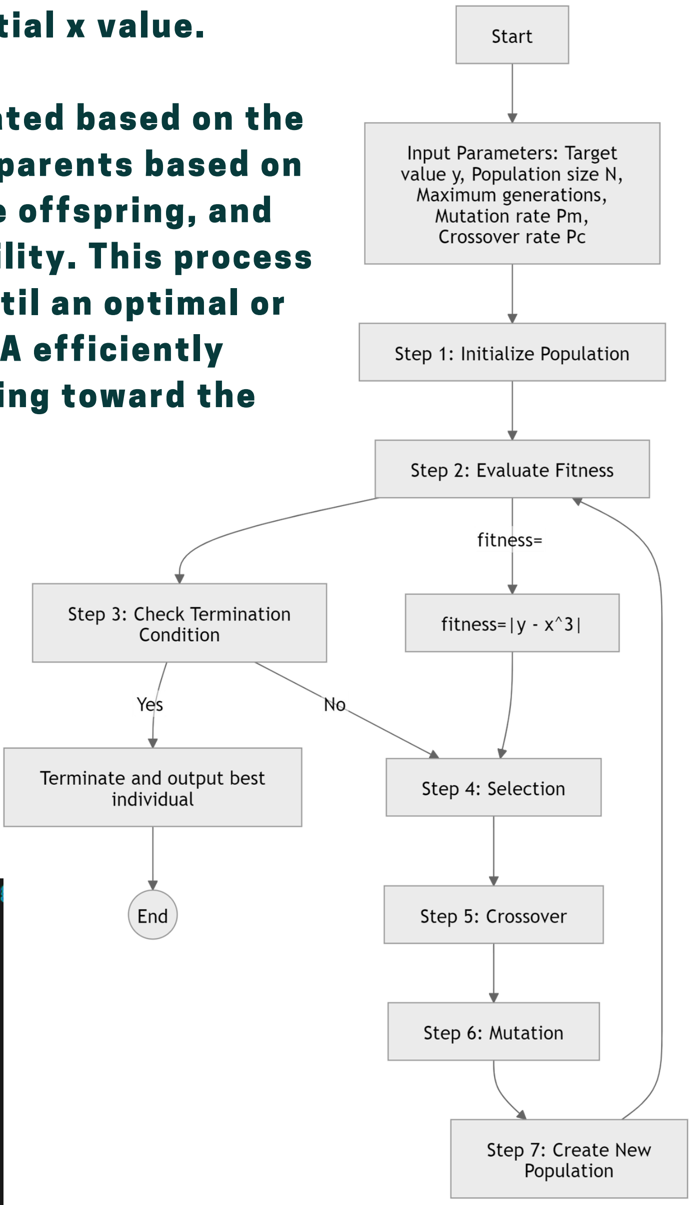
this implementation is a genetic algorithm (GA) to maximize the functionf(x) = x^3 for integer values of x. It starts by generating a population of random solutions, each representing a potential x value.

The fitness of each solution is evaluated based on the cube of x. The algorithm selects two parents based on fitness, performs crossover to create offspring, and applies mutation to introduce variability. This process repeats over multiple generations until an optimal or near-optimal solution is found. The GA efficiently explores the solution space, converging toward the best value for x.

Below is a test run for the code:



```
PS C:\Users\Test-August2023\Downloads\AI LAB> & "C:/Prog
Generation 1: Best Solution = -5, Fitness = -125
Generation 2: Best Solution = -6, Fitness = -216
Generation 3: Best Solution = -5, Fitness = -125
Generation 4: Best Solution = -6, Fitness = -216
Generation 5: Best Solution = -6, Fitness = -216
Generation 6: Best Solution = -7, Fitness = -343
Generation 7: Best Solution = -8, Fitness = -512
Generation 8: Best Solution = -7, Fitness = -343
Generation 9: Best Solution = -7, Fitness = -343
Generation 10: Best Solution = -8, Fitness = -512
Best solution found: x = -8, f(x) = -512
PS C:\Users\Test-August2023\Downloads\AI LAB>
```


Start
Input Parameters: Target value y, Population size N, Maximum generations, Mutation rate Pm, Crossover rate Pc
Step 1: Initialize Population
Step 2: Evaluate Fitness
fitness=
fitness=|y - x^3|
Step 3: Check Termination Condition
Yes
No
Terminate and output best individual
Step 4: Selection
End
Step 5: Crossover
Step 6: Mutation
Step 7: Create New Population

CONCLUSION

In conclusion, using a Genetic Algorithm (GA) to solve Sudoku provides an innovative and effective approach for tackling this combinatorial puzzle. By simulating natural selection, GAs iteratively improve potential grid configurations through selection, crossover, and mutation, gradually converging on a valid solution. This approach not only demonstrates the adaptability of GAs to solve complex optimization problems but also highlights their potential for puzzles and problems with large solution spaces. While not always the fastest method, GAs offer a unique, probabilistic way to explore solutions, making them particularly useful for complex or highly constrained Sudoku puzzles.