University of Tehran

Faculty of Engineering

School of ECE

AI

report sheet

Niloufar Mortazavi

SID : 220701096

Genetic Algorithm

This code implements a genetic algorithm to approximate a target image by evolving a population of abstract representations built from colored triangles. The algorithm optimizes the arrangement and properties of triangles within each "chromosome" (representing an individual solution) to closely resemble the target image over successive generations.

This implementation combines mutation, crossover, and selection strategies within the genetic algorithm to progressively evolve an abstract representation that closely resembles the target image. The use of elitism, dynamic triangle addition, and adaptive fine-tuning helps enhance convergence efficiency and accuracy.

Functions

Here's an explanation of how the `cross_over` and `mutate` functions work in this genetic algorithm:

`cross_over` Method

The `cross_over` function creates a new chromosome (child) by combining traits from two parent chromosomes (`parent1` and `parent2`). Here's a step-by-step breakdown:

1. Initialize the Child Chromosome: A new chromosome (`child`) is created with the same attributes as the parents, including the maximum width and height and the number of triangles.

2. Combine Parent Triangles: The function iterates over the triangles of both parents. For each pair of corresponding triangles, it randomly selects one triangle from either `parent1` or `parent2` with a probability of 50%.

3. Return the Child: The resulting child chromosome contains a mix of triangles from both parents, enabling a combination of traits from both parent solutions, which helps in exploring new solutions in the search space.

`mutate` Method

The `mutate` function introduces small, random changes to a chromosome, helping the algorithm explore variations and maintain genetic diversity. Here's how it works:

1. Random Mutation Check: With a probability defined by `mutation_rate`, one of the triangles in the chromosome is randomly chosen to be mutated.

2. Color or Position Mutation:

- Color Mutation: There's a 50% chance that the mutation will adjust the color of the chosen triangle. Each color component (R, G, B, alpha) is changed by a random value between -5 and +5, ensuring the new value stays within the valid color range (0 to 255).

- Position Mutation: Alternatively, the position of two points within the triangle may be adjusted. Each selected point's x and y coordinates are shifted by a random value between -5 and +5, ensuring the points stay within image boundaries.

This mutation introduces small changes to color and position, allowing the algorithm to explore slight variations of each solution and improving the algorithm's ability to refine and converge toward an optimal solution.

`fitness` Method

The fitness function in this genetic algorithm uses the mean squared error (MSE) to measure the similarity between a generated image and a target image. MSE is calculated by taking the average of the squared differences between corresponding pixel values in the two images. By normalizing pixel values to a range between 0 and 1 (by dividing by 255), the function calculates how closely the generated image matches the target. A smaller MSE value indicates a closer match, so the function returns the negative MSE, allowing the algorithm to maximize fitness by minimizing the error. This approach effectively guides the genetic algorithm towards generating images that more closely resemble the target.

Here are some examples of how my code works (triangle numbers=50):

Test 1

Population size 50

Population size 100

Population size 150

Increasing the population size in the genetic algorithm enhances the resemblance of the output image to the target. With a larger population, the algorithm explores a broader range of image configurations, maintaining diversity and reducing the risk of converging to suboptimal solutions. This allows for more effective selection, crossover, and mutation, which helps propagate beneficial traits. As a result, larger populations lead to output images that more closely approximate the target image.

Test 2

Population size 50

Population size 100

Population size 150

Comparison of Fitness Improvement Rates by Population Size

1. Population Size 50:

- Initial Fitness: Generation 0 began with a mean fitness of -0.199.

- Generation 50 Fitness: By generation 50, the mean fitness reached -0.068, representing an improvement of (| -0.199 - (-0.068) | = 0.131).

- Generation 100 Fitness: By generation 100, the mean fitness was -0.025, with a total improvement of (| -0.199 - (-0.025) | = 0.174).

- Summary: The fitness improved by 0.131 over the first 50 generations and by 0.174 over 100 generations. This population size showed a moderate improvement rate.

2. Population Size 100:

- Initial Fitness: Generation 0 started with a mean fitness of -0.203.

- Generation 50 Fitness: By generation 50, the mean fitness reached -0.064, indicating an improvement of (| -0.203 - (-0.064) | = 0.139).

- Generation 100 Fitness: By generation 100, the mean fitness was -0.020, resulting in a total improvement of (| -0.203 - (-0.020) | = 0.183).

- Summary: This population size showed a slightly higher improvement rate than the population of 50, with a total improvement of 0.183 after 100 generations.

3. Population Size 150:

- Initial Fitness: Generation 0 started with a mean fitness of -0.202.

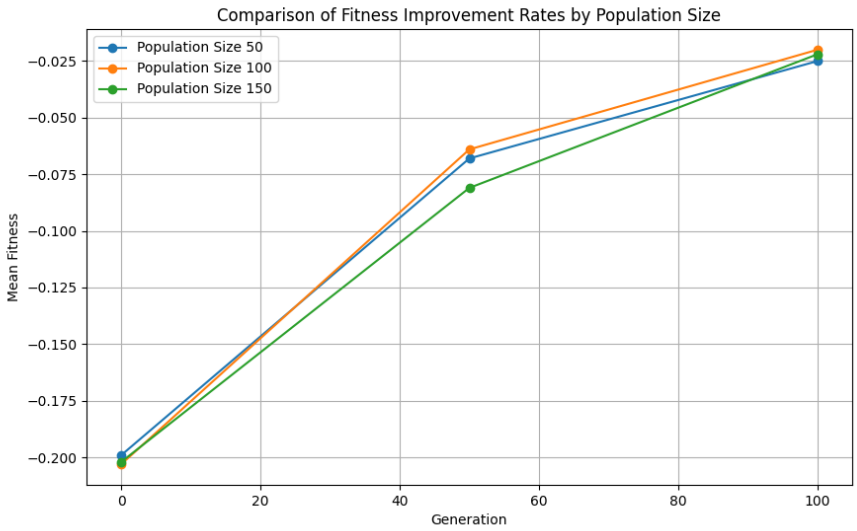
- Generation 50 Fitness: By generation 50, the mean fitness was -0.081, marking an improvement of (| -0.202 - (-0.081) | = 0.121).

- Generation 100 Fitness: By generation 100, the mean fitness reached -0.022, resulting in a total improvement of (| -0.202 - (-0.022) | = 0.180).

- Summary: Although the initial improvement rate was slightly lower in the first 50 generations (0.121), by generation 100, the improvement rate increased to match the total improvement of population size 100 (0.180). This larger population size ultimately led to a more stable convergence.

Conclusion

- Population Size Impact: Increasing the population size enhances the algorithm's ability to explore the solution space, leading to better fitness improvements over time.



*__Increasing the number of triangles__ in the genetic algorithm improves the resemblance to the target image by allowing for finer detail and more complex shapes. Here are the same tests with triangle number 200 and population size 100:

Q1- State Space Calculation for a Chromosome in the Genetic Algorithm

each chromosome represents a set of triangles, where each triangle is defined by three points (each with x and y coordinates) and a color with transparency. The total state space of a chromosome depends on the possible configurations of these attributes.

1. **Point Coordinates**:
    - Each triangle has three points, and each point has x and y coordinates. These coordinates can take any value within the width (W) and height (H) of the target image.
    - For a single point, there are W×H possible positions. Since each triangle contains three points, there are $(W \times H)^3$ possible configurations for the points in one triangle.
2. **Color**:
    - Each triangle has an RGB color with an alpha channel, which defines the color and transparency. Assuming 8-bit values, each channel can take on 256 possible values. This results in 256^4 possible configurations for each color.
3. **State Space for a Single Triangle**:
    - Combining the points and color, the total state space for a single triangle is: $(W \times H)^3 \times 256^4$
4. **Total State Space for the Chromosome**:
    - If each chromosome contains n triangles, the total state space of a chromosome is: $((W \times H)^3 \times (256)^4)^n$

This exponential growth in state space as the number of triangles increases implies that even a small number of triangles results in a very large state space.

Q2- Two ideas that can speed up the Convergence in this problem

1. Implementing Elitism

In genetic algorithms, elitism involves directly passing the best-performing chromosomes from one generation to the next. In each generation, a few chromosomes with the highest fitness scores are preserved without any changes and transferred to the next generation. This approach is effective because it ensures that the best solutions are retained across generations, which helps the algorithm converge more quickly by consistently keeping the highest-quality chromosomes in the population.

2. Using Adaptive Mutation Rates

A fixed mutation rate can slow convergence if it is too low or lead to excessive randomness if it is too high. Adaptive mutation dynamically adjusts the mutation rate based on the algorithm's progress. For example, higher mutation rates can be used in the early stages of evolution to explore a larger solution space, while lower mutation rates can be applied in later generations to fine-tune the best solutions found. This adaptive approach helps balance exploration and exploitation, which can lead to faster convergence.

Q3- There are so many different strategies to choose the next generation in genetic algorithm. Explain 2 of them.

There are multiple strategies for selecting individuals to form the next generation in genetic algorithms. Here, we describe two popular methods: Roulette Wheel Selection and Tournament Selection.

1. Roulette Wheel Selection

Roulette Wheel Selection is a probabilistic method that assigns each individual a selection probability based on fitness. Higher fitness individuals get larger "slices" on a conceptual wheel, increasing their chance of being chosen while allowing lower-fitness individuals a small chance. This approach maintains genetic diversity, helping to prevent premature convergence.

2. Tournament Selection

In Tournament Selection, a fixed number of individuals (typically two or more) are randomly chosen from the population to compete in a "tournament." The individual with the highest fitness in each tournament is selected to be part of the next generation. This process is repeated until the required number of individuals is selected. The advantage of this method is its simplicity and flexibility, as the selection pressure (i.e., the likelihood of choosing higher-fitness individuals) can be adjusted by changing the tournament size. Larger tournaments increase selection pressure, while smaller tournaments maintain more diversity in the population.