Misr International University

Faculty of Computer Science

CSC – Advanced Ai

Constant Area Coding (CAC) Optimization

Submitted by:

Ahmed Hossam

Mahmoud Osama

Ahmed Ehab

Submitted to:

Dr. Alaa Hamdy

Table of Contents

Explanation

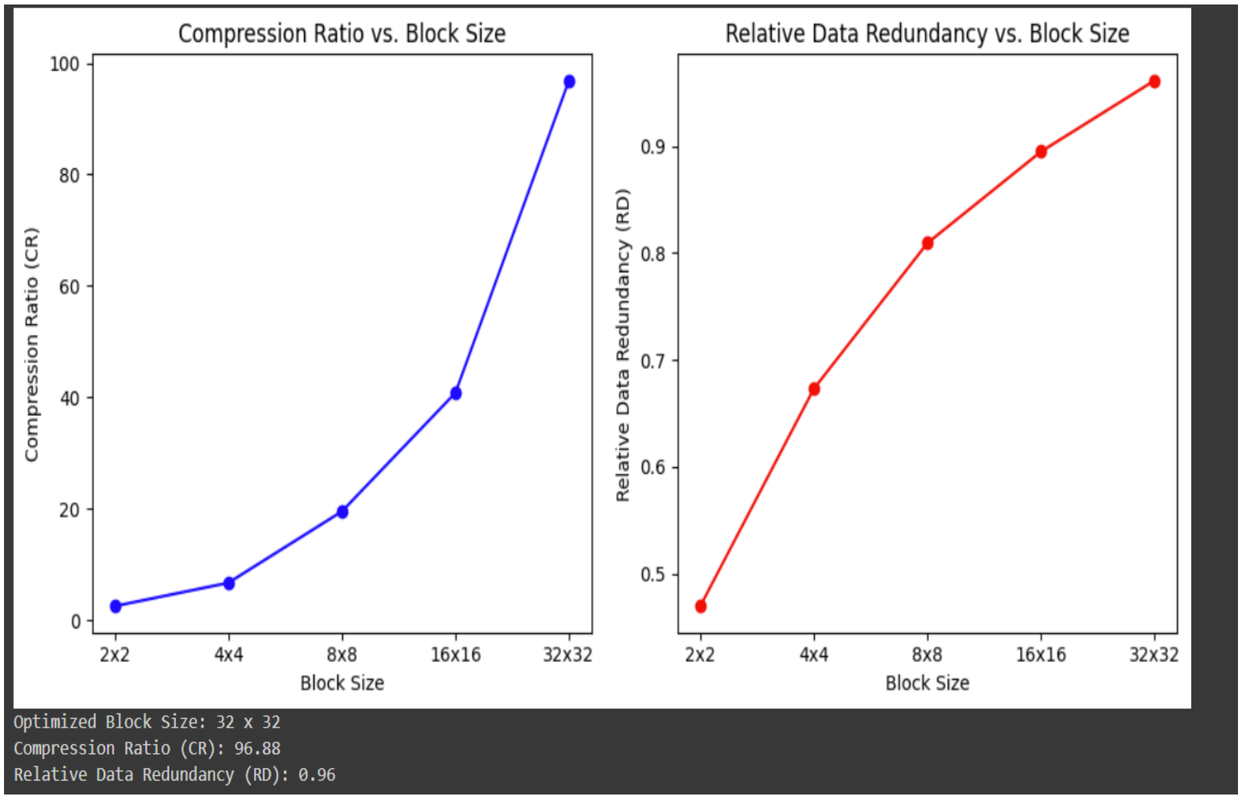
The main code implements a genetic algorithm (GA) to find the best block size for compressing an image. The goal is to maximize the compression ratio (CR) while minimizing the relative data redundancy (RD). The code starts by defining the parameters, including the image itself, the population size, the number of generations, and the mutation rate.

The GA loop begins by initializing a population of random block sizes within the defined range. Each individual in the population represents a block size using the dimensions (p, q). The fitness of each individual is evaluated using a fitness function that calculates the CR and RD. The fitness scores are then used for parent selection using roulette wheel selection. Crossover is performed by swapping dimensions between selected parents, and mutation is applied to the offspring with a probability determined by the mutation rate.

The offspring are added to the population, which is sorted based on fitness, and only the top individuals are retained. This process continues for the specified number of generations. After the GA loop, the best block size and its corresponding fitness values (CR and RD) are extracted from the population.

Methodology

- The implemented Genetic Algorithm follows a standard evolutionary process. It starts with a randomly initialized population of block sizes, evaluates their fitness using the defined fitness function, selects parents based on their fitness, performs crossover and mutation to generate offspring, replaces a portion of the population with the offspring, and repeats this process for a specified number of generations. The individuals with higher fitness scores have a greater chance of being selected and passing their genetic information to the next generation, simulating natural selection.
- The methodology involves preprocessing the binary image, performing image segmentation, calculating the compression ratio, and determining the relative data redundancy. By following these steps, the optimum block size for binary image compression can be identified, leading to efficient image storage and transmission.



```
Optimized Block Size: 32 x 32
Compression Ratio (CR): 96.88
Relative Data Redundancy (RD): 0.96
```

Test Cases and Scenarios:

- Different Image: Use a different image with varying dimensions and patterns. Modify the 'image' variable accordingly and observe how the algorithm adapts to find the best block size for compression.

- Varying Parameters: Change the population size, number of generations, and mutation rate to see how it affects the convergence and the quality of the solution. Increase the population size and number of generations for a more exhaustive search, but note that it may also increase the computation time.

- Different Fitness Function: Modify the fitness function to consider other factors or add constraints, such as the trade-off between CR and RD, or incorporating additional image quality metrics like peak signal-to-noise ratio (PSNR).

- Visualize Results: Implement a visualization component to display the best block size and the compressed image for each generation, allowing you to observe the evolution of the algorithm visually.

Limitations

- Speed and Efficiency: The code does not prioritize efficiency and may not be suitable for large images or extensive search spaces. It's recommended to optimize the code, such as using more efficient data structures, parallelizing computations, or implementing heuristics to reduce the search space.
- Generalizability: The code focuses on optimizing the block size for image compression but may not be directly applicable to other types of data compression problems. It is

tailored specifically for the given scenario and may require modifications to be used in different contexts.

- Scalability: As the image size and search space increase, the computation time and memory requirements of the algorithm also increase. Consider applying techniques like scaling down the image or exploring domain-specific optimizations to handle larger images effectively.