Genetic Algorithm-Based Image Approximation Using Triangle Representation for Efficient Compression

Yashpreet Malhotra
yashmalhotra9323@gmail.com

***Abstract*—This paper presents a novel image approximation algorithm using genetic algorithms (GA) to achieve extreme compression for black-and-white images. The method optimizes the shape and placement of a limited number of triangles to approximate the target image, achieving recognizable results with as few as 139 bytes of data. The genetic algorithm's evolutionary process is leveraged to iteratively improve image quality through mutation, crossover, and selection operators. The approach offers a highly compact representation of images, demonstrating significant size reductions compared to traditional image compression algorithms. The proposed technique balances visual fidelity with extreme data efficiency, making it a promising method for lossy image compression in constrained environments. Test results on sample images demonstrate the method's effectiveness and potential applications.**

***Keywords*—genetic algorithm, image Approximation, Enetic algorithm Image Compression,Black-and-White Images,Polygonal Approximation,Triangle Representation**

I. Introduction

Genetic Algorithms (GAs) (1) are a class of evolutionary computation techniques that provide approximate solutions to optimization problems where deterministic or polynomial-time solutions are either unknown or computationally prohibitive. Inspired by the principles of natural selection and genetics, GAs simulate the process of evolution in search of high-quality solutions by iteratively improving a population of candidate solutions over successive generations.

In a typical GA framework, an initial population—often composed of randomly generated or heuristically chosen solutions—is evolved using biologically inspired operations such as *mutation*, *crossover*, and *selection*. Each member of the population, referred to as a *specimen* or *individual*, encodes a potential solution to the problem, commonly in the form of a chromosome. The quality or effectiveness of each individual is quantified by a predefined *fitness function*, which guides the evolutionary search toward more optimal regions of the solution space. Through mutation, random variations are introduced; crossover recombines components of two parent solutions; and selection ensures that the most promising individuals are propagated into future generations.

This paper introduces a novel approach to lossy image compression leveraging the strengths of GAs in optimizing complex, high-dimensional search spaces. The method targets binary (monochromatic) images and seeks to approximate their visual content using a minimal set of geometric primitives—in this case, filled triangles. Each triangle is defined by parameters such as vertex coordinates, orientation, and fill color (black or white), and collectively, they form a compressed representation of the original image.

The core idea is to encode these triangle parameters as chromosomes and use GA operations to optimize their configuration such that the rendered image closely resembles the original input while consuming minimal storage. The evolutionary process iteratively refines the triangle set, with each generation moving toward a more visually accurate and data-efficient approximation. Various mutation and crossover techniques are explored to perturb triangle configurations, while different selection strategies are evaluated to balance exploitation and exploration during convergence.

A major contribution of this work lies in demonstrating the feasibility of achieving extreme compression ratios without relying on traditional transform-based or entropy-encoding methods. Experimental results show that visually recognizable reconstructions can be achieved using as few as 100–150 bytes, a significant reduction when compared to standard lossy compression formats. While some degradation in image fidelity is inevitable due to the lossy nature of the technique, the preserved structural features and visual cues make the results acceptable for specific use cases where storage or transmission bandwidth is severely constrained.

This GA-based image approximation approach represents a promising direction for lightweight compression systems, particularly in embedded applications, low-power devices, and data-scarce environments. The remainder of this paper details the chromosome representation, the design of genetic operators tailored to image structures, and the empirical evaluation of the algorithm's performance across various image datasets.

II. Related Work

Genetic Algorithms (GAs) have emerged as a powerful heuristic optimization tool particularly suitable for solving complex, nonlinear, and multi-dimensional problems for which no efficient or exact algorithm is available. One of the primary advantages of GAs is their population-based, iterative search mechanism, which allows for gradual solution refinement. Furthermore, they provide the flexibility to terminate the search at any point, returning the best solution obtained so far—an invaluable feature for time-constrained or real-time applications.

In the domain of industrial and operational research, GAs have been extensively explored for scheduling tasks, where the combinatorial nature and vast search spaces render traditional approaches like linear programming inefficient or infeasible. Studies have shown that GAs provide robust and adaptable solutions in manufacturing schedules and workforce management (6). Similarly, transportation logistics and planning benefit from GA-driven models, offering flexible alternatives in scenarios involving multiple constraints and dynamic data (3).

A closely related class of optimization problems is the Vehicle Routing Problem (VRP), a generalization of the classic Traveling Salesman Problem (TSP). Both are known to be NP-hard and have seen successful applications of genetic algorithms for near-optimal route planning. In (4), a practical GA-based framework is proposed to address VRPs, showcasing the ability of GAs to adapt to real-world constraints such as delivery windows, fleet sizes, and route capacities. Likewise, genetic algorithms have demonstrated competency in approximating solutions to the TSP, outperforming certain deterministic methods in scalability and robustness (5).

Another notable field of application is machine learning, particularly in the optimization of synaptic weights in artificial neural networks. For instance, the MarIO project (2) exemplifies the use of genetic algorithms to evolve a neural agent capable of autonomously navigating and playing levels of the Super Mario video game. The success of such frameworks underscores the potential of GAs in evolving control policies and learning behaviors in simulated environments.

In the broader context of artificial intelligence, GAs are employed in genetic programming, where the objective is to automatically generate executable code to control agents or systems. A well-known example is the Robocode framework (7), where virtual robots are programmed using genetically evolved strategies to compete in dynamic combat scenarios. This approach has applications in autonomous systems, robotics, and strategy planning, highlighting the versatility of genetic algorithms in behavioral optimization.

In the financial sector, GAs have gained traction for solving portfolio optimization problems, where investors aim to maximize returns while minimizing risks under multiple constraints. The adaptability of GAs allows for efficient search in large solution spaces typical in financial modeling (8). Furthermore, in algorithmic and automatic trading systems, GAs are utilized to evolve trading strategies that can adapt to market fluctuations, as detailed in (9).

Within the realm of image processing, GAs have proven effective when used alongside neural networks or as standalone optimizers for tasks such as image segmentation (10) and enhancement (12). These applications demonstrate the capacity of GAs to explore the high-dimensional parameter spaces involved in visual data interpretation. While most image compression methods today rely on algorithmic speed and mathematical transformations to achieve either lossless (exact reproduction) or lossy (perceptually similar) compression (11), genetic algorithms offer an alternative that trades off speed for adaptability and extreme compression capabilities. They enable solutions tailored to niche scenarios such as embedded systems or artistic rendering, where traditional methods may fall short.

Collectively, these studies underscore the broad utility of genetic algorithms across a wide range of disciplines. Their ability to navigate complex search spaces, adapt to evolving constraints, and generate high-quality approximations positions them as a compelling approach for problems that defy conventional solutions.

III. GA-Based Image Compression

A. Problem Statement

The objective of this work is to approximate a binary (black-and-white) image using a minimal set of geometric primitives—specifically, triangles—through the application of Genetic Algorithms (GAs). The input image, denoted as IMG, is modeled as a square binary matrix of size $n \times n$, where each pixel a_{ij} belongs to the set $\{0, 1\}$, with 0 representing black and 1 representing white:

$$IMG = (a_{ij}) \in \{0, 1\}^{n \times n} \tag{1}$$

The compressed approximation of the image consists of k triangles, each defined by three vertices located on integer coordinates within the image grid, and a binary color value. Each triangle t_k is represented as a quadruple:

$$t_k = (A, B, C, c), \quad A, B, C \in \{1, \ldots, n\}^2, \ c \in \{0, 1\} \tag{2}$$

The set of pixels covered by triangle t_k is described by the function $I(t_k)$, which computes all interior points using barycentric coordinates derived from vertices A, B, and C:

$$\begin{aligned} &I(t_k) = \{a_{ij}\}, \\ &\exists \alpha, \beta \in [0, 1] : \\ &\quad i = \beta(\alpha A_1 + (1 - \alpha)B_1) + (1 - \beta)C_1 \\ &\quad j = \beta(\alpha A_2 + (1 - \alpha)B_2) + (1 - \beta)C_2 \end{aligned} \tag{3}$$

This formulation enables a geometric abstraction of the image using only a compact set of triangle descriptors.

B. Chromosome Representation

In the context of genetic algorithms, each candidate solution—referred to as a specimen or chromosome—is encoded as a triplet (k, T, C) representing an approximation Q of the input image. This representation is defined as:

- $k \in \mathbb{Z}$ — the number of triangles composing the approximation,
- $T \in \mathbb{Z}^{k \times 3 \times 2}$ — the coordinate triplets (vertices) for each triangle,
- $C \in \{0, 1\}^k$ — the binary color (black or white) assigned to each triangle.

This compact encoding supports efficient mutation and crossover operations during the genetic optimization process.

C. Fitness Function

To evaluate how well an approximation (k, T, C) replicates the input image, a rasterized image R is synthesized by overlaying the k triangles. The color of each pixel (i, j) in R is determined by the topmost triangle that covers it:

$$R_{i,j}(k,T,C) = \begin{cases} C_v, & \exists v : (i,j) \in I(T_v),\ \nexists w > v : (i,j) \in I(T_w), \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Rather than using a conventional fitness function where higher values indicate better solutions, an error metric E is employed. This error function computes the percentage of mismatched pixels between the original image and the generated approximation:

$$E = \frac{\sum_{i,j} |IMG_{i,j} - R_{i,j}|}{n^2} \cdot 100, \quad 1 \le i, j \le n \tag{5}$$

Lower error values indicate higher visual similarity between the original and approximated image.

D. Genetic Operators

Three fundamental genetic operators are utilized in the evolutionary cycle: mutation, crossover, and selection. Their implementations are tailored to operate on triangle-based chromosomes:

- **Mutation Operators:**
  - *M1:* Randomly removes a triangle from the chromosome.
  - *M2:* Inserts a new triangle with randomly generated vertices and color.
  - *M3:* Alters one, two, or all three vertices of an existing triangle to new coordinates.
- **Crossover:** Two parent chromosomes exchange subsets of triangles, randomly combining their genetic material to produce offspring solutions.
- **Selection:** After calculating the error metric E for each specimen, a proportion of the worst-performing individuals is removed from the population and replaced with newly generated specimens.

These operators collectively promote exploration of the search space while preserving useful traits.

E. Program Flow

The complete flow of the compression program proceeds through the following stages:

1) Load the target image from storage.
2) Generate an initial population of specimens, each starting with one randomly initialized triangle.
3) Apply mutation operators probabilistically to introduce variation.
4) Evaluate the error metric E for all individuals in the population.
5) Log generation statistics including current error and triangle count.
6) Perform selection by eliminating high-error individuals.
7) Apply crossover to replenish the population with new specimens.
8) Repeat Steps 3–7 for a predefined number of generations or until convergence.

F. Software Architecture

The implementation of the proposed GA-based image compression algorithm is done in the C programming language. External libraries are avoided (except standard GNU C Library), ensuring a portable and lightweight codebase. As a trade-off, the system supports only the BMP image format with fixed resolution and color depth.

For each generation, the program outputs diagnostic information including generation number, best error rate, and number of triangles in the current best solution. Additionally, the best approximation image is saved to disk at every generation, enabling visual tracking of convergence.

The binary representation of each approximated image consumes minimal space. The breakdown of storage requirements is as follows:

- 8 bits for the number of triangles,
- $3 \times 2 \times 9 \times k$ bits to encode triangle coordinates (3 vertices, 2D each, 9 bits per coordinate),
- k bits for triangle colors.

Hence, the total compressed size in bytes is:

$$\text{Total Size} = 1 + \frac{55}{8}k \tag{6}$$

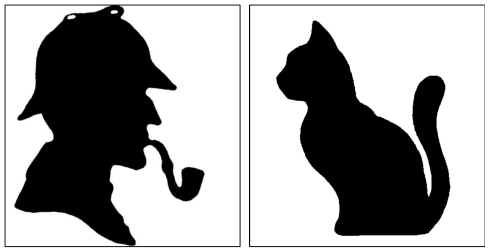
This formulation enables highly compact representations—suitable for extreme compression scenarios—at the cost of some image fidelity.

IV. Test Results

To thoroughly evaluate the efficiency and performance of the proposed GA-based image compression method, two moderately complex binary silhouette images were selected as test cases. These images, displayed in Fig. 1, include: (a) a silhouette of Sherlock Holmes and (b) a silhouette of a cat. Both images are sized at 512×512 pixels, offering a balanced challenge in terms of feature complexity and compression viability.

The genetic algorithm was executed on each of these images using a fixed population size of 2048 individuals. Mutation was introduced in every generation through three operators with differing intensities: the triangle deletion operator (*M1*) was applied with a probability of 10%, the insertion operator (*M2*) with 50%, and the vertex mutation operator (*M3*) with 80%. These mutation strategies introduce both structural and spatial diversity, essential for exploring the solution space effectively.

To accelerate convergence, the algorithm adopted an elitist replacement strategy: the 1800 least-fit individuals in each generation were replaced by new specimens created via crossover



(a) (b)

Fig. 1: Selected binary input images used for testing. (a) Sherlock Holmes silhouette, (b) Cat silhouette.

between randomly chosen, more successful individuals. While this elitist strategy ensures rapid improvement in fitness, it may potentially reduce genetic diversity and lead to local optima.

To study the algorithm's approximation behavior under various complexity constraints, a maximum polygon (triangle) count k_{max} was varied across the following set of values: 5, 10, 15, 20, 30, 40, 50, 60, 70, and 80. For each configuration, the algorithm was executed for 1000 generations, ensuring sufficient time for convergence while observing the evolution of approximation quality.

Figs. 2 and 3 show intermediate approximation results for both images using $k_{max} = 20$ triangles, visualized after 25, 50, 100, 200, 300, and 500 generations. The reconstructions demonstrate that significant visual accuracy is attained early in the evolutionary process, especially between 100 and 200 generations. Beyond this point, improvements become more incremental, emphasizing the principle of diminishing returns.

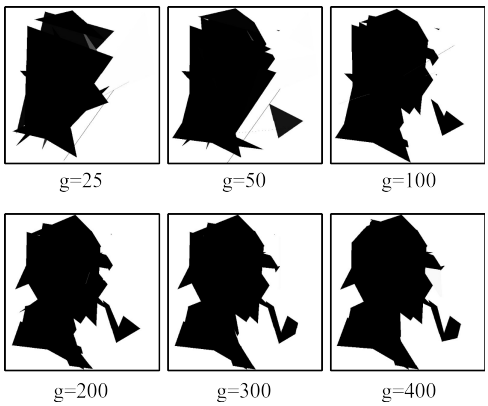


Fig. 2: Progressive evolution of the Sherlock image approximation using $k_{max} = 20$ triangles at various generation counts.

Figs. 4 and 5 provide a quantitative visualization of the best fitness score per generation for multiple values of k_{max}, showcasing the convergence trends across the two images. As expected, configurations with fewer triangles exhibit limited expressiveness and higher error rates. When the number of polygons is increased beyond approximately 30, additional improvements in approximation quality taper off. However, a few finer details continue to emerge with up to 60 triangles.

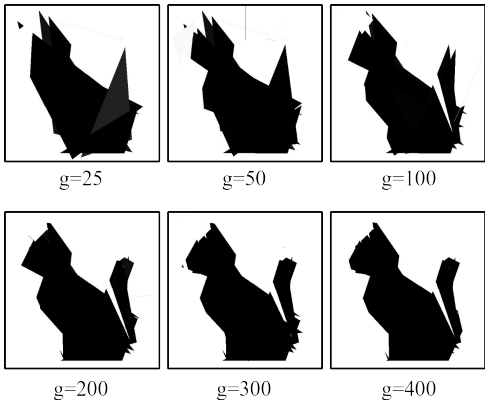


Fig. 3: Evolution of the cat image reconstruction with $k_{max} = 20$ triangles over different generation milestones.

Importantly, a recognizable and compressed visual approximation using only 20 triangles can be encoded in just 139 bytes, illustrating the remarkable compression potential of the proposed GA-based method.

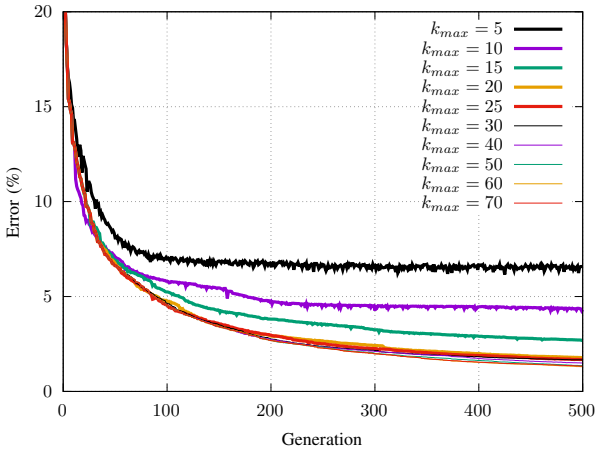


Fig. 4: Evolution of the error metric during the compression of the Sherlock image. Each curve corresponds to a different polygon count k_{max}.

Tab. I presents the final error rates and the resulting file sizes for both test images after 1000 generations across multiple k_{max} values. These empirical results confirm that even with a limited number of triangles (e.g., $k_{max} = 15$–20), the algorithm delivers a compelling trade-off between approximation fidelity and storage efficiency.

V. Comparison to Traditional Compression Algorithms

In order to assess the competitiveness of the proposed approximation-based compression scheme, its storage efficiency was benchmarked against both domain-specific and general-purpose compression methods. Specifically, the same test images were processed using two widely-adopted image compression standards—PNG (lossless) and JPEG (lossy)—as well as two general data compressors: RAR and BZIP2.

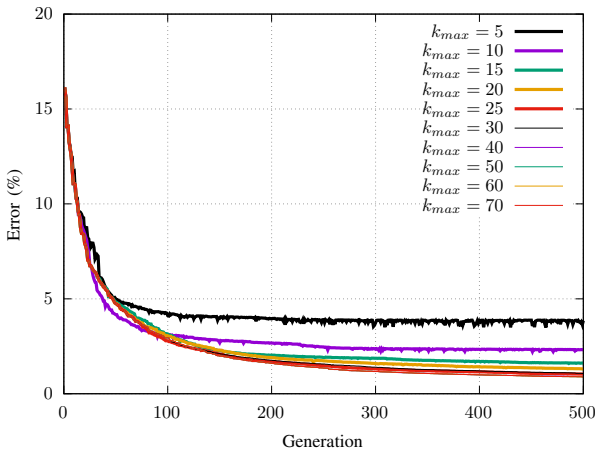


Fig. 5: Error convergence for the cat image across generations for various triangle counts.

TABLE I: Approximation error and encoded size after 1000 generations for varying k_{max} values.

Image	k_{max}	Final Error (%)	Size (bytes)
[r=10]c,1.5cmSherlock	5	6.338	36
	10	4.218	70
	15	2.401	105
	20	1.439	139
	25	1.220	173
	30	1.301	207
	40	1.123	276
	50	1.006	345
	60	0.940	414
	70	0.898	483
,gray5			
[r=10]c,1.5cmCat	5	3.775	36
	10	2.260	70
	15	1.320	105
	20	1.199	139
	25	0.837	173
	30	0.807	207
	40	0.532	276
	50	0.612	345
	60	0.569	414
	70	0.562	483

For the image formats, PNG compression was performed at its highest level, and JPEG was configured for the lowest quality setting to minimize file size. Metadata such as EXIF tags and color profiles were stripped to ensure a fair comparison. For the general-purpose tools, RAR was executed with `-m5 -md4096` options, and BZIP2 used the `-9` switch—both indicating maximum compression effort.

To compare with the proposed method, approximations were generated with $k_{max} = 20$ and $k_{max} = 50$. The resulting compressed sizes are presented in Tab. II. As seen, the GA-based technique significantly outperforms traditional methods in terms of file size, achieving reductions by over an order of magnitude compared to JPEG or PNG. While the proposed approach is lossy and best suited for binary or highly abstracted images, its compression ratios make it highly attractive in storage-constrained scenarios such as embedded systems or low-bandwidth applications.

TABLE II: Comparison of compressed file sizes using standard algorithms and the proposed method.

Method	[r=2]c,1.5cmCompression Compressed Size (bytes) Sherlock	Cat
JPEG	2443	2521
PNG	2688	2247
RAR	2059	1553
BZIP2	1859	1690
GA ($k_{max} = 20$)	139	139
GA ($k_{max} = 50$)	345	345

VI. Summary

A genetic algorithm based image approximation method was proposed. The algorithm can approximate black and white images with a limited (definable) number of black and white triangles. The GA is implemented with 3 mutation and a crossover operator. Selection operator uses an elitist approach. Formal definition was given to the fitness function computation.

The proposed algorithm was implemented as a C language program and tested with different input images and with several maximal number of triangles.

The algorithm can generate a recognizable quality approximation of images with moderate complexity in less than 150 bytes with computation time under 10 miutes.

VII. Conclusion and Future Work

A. Conclusion

This study presented a genetic algorithm-based method for compressing black-and-white images using triangle approximations. By evolving polygonal representations of images under fixed constraints on the number of triangles, the proposed technique achieves significant reductions in file size while preserving visual recognizability.

Experimental results using binary silhouettes demonstrated the effectiveness of the method across various polygon counts, showing a rapid convergence in early generations and diminishing error improvements as the number of triangles increased. Notably, the method was able to compress an image into as little as 139 bytes with $k_{max} = 20$ triangles, vastly outperforming traditional image compression algorithms such as JPEG, PNG, and even general-purpose data compressors like RAR and BZIP2 in terms of compression ratio.

The encoding method leverages elitist evolutionary strategies and multiple mutation operators to explore a vast search space efficiently. Despite the method's lossy nature, the resulting approximations are suitable for scenarios where extreme compression is prioritized over pixel-perfect accuracy, such as in embedded systems, icon generation, or progressive rendering.

B. Future Work

Although the current approach demonstrates high compression efficiency for binary images, there are several promising directions for extending and enhancing this framework:

- **Color Image Support:** Extending the algorithm to support grayscale and color images by encoding color attributes per triangle and optimizing both geometry and chromaticity.
- **Adaptive Polygon Strategies:** Incorporating adaptive mesh refinement where the number of triangles varies dynamically based on image regions' complexity, rather than a global fixed k_{max}.
- **Parallel and GPU Acceleration:** Implementing parallel processing techniques using GPU acceleration to significantly reduce the runtime of evolution and fitness evaluations.
- **Integration with Existing Codecs:** Exploring hybrid systems that integrate this approximation technique with standard codecs like JPEG 2000 or WebP for layered compression.
- **Perceptual Error Metrics:** Replacing pixel-wise fitness evaluation with perceptual quality metrics such as SSIM or MS-SSIM to better align compression output with human visual perception.
- **Security and Steganography:** Investigating potential applications in secure image sharing and data hiding through controlled polygon transformations.

The results of this work open up a low-data-cost direction for image encoding that merges the strengths of optimization techniques with minimalistic geometric representations. With further refinements, this approach could find widespread application in low-resource environments, mobile platforms, and real-time rendering systems.

References

[1] Sivanandam, S. N. and Deepa, S. N., "Introduction to Genetic Algorithms", Springer-Verlag Berlin Heidelberg, 2008, ISBN 978-3-540-73189-4

[2] Baldominos A., Saez Y., Recio G., Calle J. "Learning Levels of Mario AI Using Genetic Algorithms", In: Advances in Artificial Intelligence. CAEPIA 2015. Lecture Notes in Computer Science, vol 9422. Springer, Cham. doi: 10.1007/978-3-319-24598-0_24

[3] C. Lin, J. Yu, J. Liu and C. Lee, "Genetic Algorithm for Shortest Driving Time in Intelligent Transportation Systems", 2008 International Conference on Multimedia and Ubiquitous Engineering, 2008, pp. 402-406, doi: 10.1109/MUE.2008.16.

[4] Zhu, K. Q., "A new genetic algorithm for VRPTW", Proceedings of the international conference on artificial intelligence. 2000.

[5] Ahmed, Z. H. "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator", International Journal of Biometrics & Bioinformatics (IJBB) 3.6 (2010): 96.

[6] Lim, C., and Eoksu S., "Production planning in manufacturing/remanufacturing environment using genetic algorithm", Proceedings of the 7th annual conference on Genetic and evolutionary computation. 2005.

[7] Shichel Y., Ziserman E., Sipper M., "Using Genetic Programming to Evolve Robocode Players" In: Genetic Programming. EuroGP 2005. Lecture Notes in Computer Science, vol 3447. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-31989-4_13

[8] S. Slimane and M. Benbouziane, "Portfolio Selection Using Genetic Algorithm" In: Journal of Applied Finance & Banking , Vol. 2, No. 4 (2012): pp. 143-154.

[9] R.J. Kuo, C.H. Chen, Y.C. Hwang, "An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network", Fuzzy Sets and Systems, Volume 118, Issue 1, 2001, Pages 21-45, ISSN 0165-0114, 10.1016/S0165-0114(98)00399-6.

[10] Pare, S., Bhandari, A. K., Kumar, A., Singh, G. K., Khare, S. "Satellite image segmentation based on different objective functions using genetic algorithm: a comparative study." In 2015 IEEE international conference on digital signal processing (DSP) (pp. 730-734). IEEE.

[11] Jayaraman, S., S. Esakkirajan, and T. Veerakumar. "Digital image processing", Mc Graw Hill India, 2009, ISBN 978-0070144798

[12] Deborah, H., Arymurthy, A. M., "Image enhancement and image restoration for old document image using genetic algorithm", In 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies (pp. 108-112). IEEE.