Codebook optimization using Jaya Algorithm for image compression

Suvojit Acharjee, Prof. Sheli Sinha Chaudhuri

Department of ETCE, Jadavpur University

acharjeesuvo@gmail.com, shelism@rediffmail.com

Abstract.

Image compression is required to effectively manage today's communication infrastructure. The Linde–Buzo–Gray (LBG) algorithm is a powerful and reliable lossy image compression technique. Iterative refining is used by LBG to construct a local codebook. Various evolutionary algorithms were used to create a global codebook. However, the performance of various evolutionary algorithms is not consistent. Furthermore, these algorithms include algorithm specific parameters such as acceleration rate in particle swarm optimization (PSO), discovery probability in cuckoo search (CS), and so on. Improper value selection for these tuning parameters might lead to local minima. This article proposes utilizing the Jaya Algorithm (JA) to optimize the vector quantization codebook. JA has no algorithm specific tuning parameters, and its success is determined only by generic evolutionary algorithm parameters like maximum iteration and initial population numbers. The output from the proposed algorithm outperforms the CS, and PSO based vector quantization algorithms as well as the state-of-the-art pattern-based masking LBG algorithm.

Keywords. LBG, Jaya Algorithm, Image Compression, PSO, cuckoo search, Codebook.

1. INTRODUCTION

Compression can be classified into lossy and lossless categories. Lossless compression does not lose any information during compression. However, lossy compression allows the loss of information up to an extent. The removed information is often redundant in nature. The loss of this information has a negligible effect on the signal. The image compression techniques mainly employ the vector quantization algorithms (VQA) for lossy compression. The most popular VQA, Linde–Buzo–Gray (LBG) [1] algorithm works by minimizing the Euclidian distance between the codeword and the image vector. LBG successfully generates locally optimized codebook for image vectors. But it does not guarantee a global optimal code word. Also, the output of the LBG algorithm varies massively based on the initial condition of the algorithm. Therefore, researchers employ several evolutionary algorithms to find a global codebook for VQ.

The ant colony optimization-based (ACO) VQA [2] used wavelet coefficients and bidirectional graphs. Though the output of this process was better than the output of LBG, the speed of the algorithm was very slow. VQ was successfully accomplished using the Particle swarm optimization (PSO) algorithm [3]. The PSO based algorithm was faster than the ACO-based algorithm. However, the convergence of PSO becomes very unstable if any particle in the swarm has a high velocity. PSO was further modified in Quantum-behaved

Particle Swarm Optimization [4] (QPSO) which calculates the local particle using the particle's best fitness as well as the global best fitness. The Firefly Optimization Algorithm (FOA) [5] is based on the social characteristics of the firefly. Fireflies use their bioluminescence property for mating communication. The dark firefly moves toward the brighter firefly during mating. [6] But FOA face difficulty when brighter fireflies are absent from the search space. Chiranjeevi et al. [7] proposed a FOA-based VQA that solves this problem by allowing the firefly to fly at random in the absence of the bright firefly. The Bat Algorithm (BA)-based VQA (BALBG) [8] outperforms the FOA-based VQA (FOALBG) with proper tuning parameters such as frequency, pulse rate, and loudness. However, evolutionary algorithms such as PSO, BA, FOA, and others contain a number of algorithm-specific parameters. The algorithm's performance varies significantly when these operational parameters are not properly tuned. The discovery probability is the single algorithm-specific parameter in the cuckoo search (CS) algorithm [9]. As a result, it is far simpler than BA and FOA. VQ based on CS was simple to implement [10]. Despite the fact that CS reduced the number, the performance of these evolutionary algorithms is still heavily dependent on algorithmic-specific parameters. Incorrect tuning of these parameters can lead to local optima and delay convergence. The pattern-based masking LBG (PBMLBG) algorithm [11] used the local histogram peak to find the repetitive patterns inside the image vectors. These patterns are then used as the initial codebook of the LBG algorithm. But improper selection of the histogram peaks can lead to improper results [11]. Rao et al. [12] proposed the Jaya algorithm (JA), which depends only on general parameters of evolutionary algorithms. Previously, the JA successfully optimized a variety of engineering problems, including the block matching algorithm for motion estimation [13], fuzzy PIDF controller [14], and many others. The absence of algorithmic specific parameters helps the proposed algorithm to outperform the BALBG and PBMLBG.

The manuscript is structured as follows. Section two will introduce different VQ techniques based on evolutionary algorithm. Section 3 will outline the JA as well as the proposed JA based VQA. In section 4, the performance of the proposed algorithms is compared with the performance of state-of-the-art algorithms. Section 5 will bring the article to a conclusion.

2. OVERVIEW: VECTOR QUANTIZATION

The VQ is a lossy process that maps the image vectors into codewords. The image vectors are generated by subdividing an image into non-overlapping blocks. All codewords are stored and indexed inside the codebook. Every image vector will be assigned the index of its nearest codeword based on the Euclidian distance. The size of the codebook controls the quality of the output as well as the speed of the process. The decoder reconstructs the image based on the index and the codebook.

2.1. LBG Algorithm

LBG algorithm [1] is the most popular algorithm to construct the codebook for an image vector. LBG reduce the distortion between the original and recovered image through repetitive iteration. The steps of the algorithm are following.

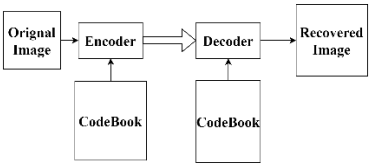

Figure 1. Encoding and Decoding process of vector quantization.

A) Generate initial codebook by randomly selecting N_c number of codewords from the image vectors. Size of each image vector as well as the codewords are N_b.

B) Map the image vectors $\{I_1, I_2, \ldots, I_M\}$ to its nearest codewords $\{C_1, C_2, \ldots, C_{Nc}\}$.

$$flag_{mi} = \begin{matrix} true & if\ |C_i - I_m| < |C_{j\in 1,..,N_c : j\neq i} - I_m| \\ false & otherwise \end{matrix} \quad (1)$$

$flag_{mi}$ describes the position in the $\{M \times N_c\}$ Boolean flag matrix. This matrix is only true when m^{th} image vector belongs to i^{th} codeword.

C) Compute the centroid of the new clusters. These centroids are the new codewords C'_i.

$$C_i' = \frac{\sum_{m=1}^{M} I_m * flag_{mi}}{Number\ of\ vectors\ in\ the\ cluster} \quad (2)$$

D) Compute the distortion (d') with new codewords Algorithm terminates when present distortion is larger or equal to previous iteration (d). Else, the process repeats from step 2.

$$Algorithm\ continue\ when \quad (d - d') > 0 \quad (3)$$

2.2. Evolutionary Algorithms based LBG Algorithm

The codebook produced from LBG is localized one. Evolutionary algorithms were employed to find the globally optimized codebook. The steps of any evolutionary algorithm-based vector quantization algorithms are following.

A) Run the LBG algorithm and assign the output as one of the *n* number of solutions. Initialize rest of the *(n-1)* solutions randomly.

B) Calculate the fitness of every solution using equation 4.

$$F = \frac{N_b}{\sum_{i=1}^{N_c}\sum_{m=1}^{M}\sum_{j=1}^{N_b} ||I_{mj} - C_{ij}|| * flag_{mi}} \quad (4)$$

C) Rank the solutions based on their fitness and generate new solutions for next iteration.

D) Until the termination requirements were met, repeat the process from step B.

3. JAYA ALGORITHM BASED VECTOR QUANTIZATION

JA is one of those evolutionary algorithms which does not contain any algorithm specific operational parameter. It only depends on the general evolutionary algorithm parameters i.e., maximum iteration and initial population size. Therefore, it is free from the possibility of premature convergence due to improper selection of tuning parameters. The steps of JA are described below.

A) Set the population size & maximum iteration. Also, randomly initialize the population.

B) Calculate the fitness of the solutions and identify best (X_{best}) and worst (X_{worst}) solution.

C) Generate new solutions using equation 5. r_1 and r_2 are random numbers between 0 & 1.

$$X_i' = X_i + r_1 * (|X_i - X_{best}|) - r_2 * (|X_i - X_{worst}|) \quad (5)$$

D) Accept the new solution if the fitness of new solution is better than the previous one.

E) Continue from step 3 until the termination requirements are met.

The new solution generation process in Jaya algorithm pushes the old solution closer towards the best solution. At the same time, old solution moves further away from the worst solution. The JA based VQA is referring the codebook as a solution. The fitness of the solutions or the codebooks are calculated based on equation 4. The aim of JA based VQA is to obtain an optimal codebook which will maximize the fitness. The details of JA based VQA algorithm are as follows.

A) Divide the whole image into image vectors $I = \{I_1, I_2, \dots, I_M\}$.

B) Merwe et al. [15] demonstrated that using a local codebook as the initial population of an evolutionary algorithm reduces the convergence time of evolutionary algorithms. Therefore, LBG algorithm was used to generate a codebook which was assigned as one of the initial populations in JA based VQA. Other populations are selected randomly from the image vectors. Initial population consist of *n* codebooks.

C) The algorithm uses Equation 4 to calculate the fitness of the population. The solutions are sorted based on their fitness. The codebook with the highest fitness will be at the top of the list, while the codebook with the lowest fitness will be at the bottom.

D) Equation 6 will generate the new solution for next iteration. This equation is a more specific version of equation 5 which was more elaborated for JA based VQA.

$$C'_{ijk} = C_{ijk} + r1 * (|C_{ij1} - C_{ijk}|) - r2 * (|C_{ijn} - C_{ijk}|) \quad (6)$$

E) The new codebook C'_i will only replace C_k when the fitness of new codebook C'_k is better than the fitness of old codebook C_i.

F) The algorithm checks for the termination conditions. When the iteration is smaller than maximum iteration and error is not converged then repeat the process from step C.

4. RESULT AND DISCUSSION

4.1. Dataset, Initial Condition and Performance metrics

In this experiment, the performance of the proposed algorithm's output is compared to the output of LBG, PSOLBG, QPSOLBG, FOALBG, BALBG, and PBMLBG utilizing bit rate per pixel (bpp), peak signal to noise ratio (PSNR), and calculation time. Four standard test images of 512×512 resolution ("Lenna.jpg", "baboon.png", "peppers.png", and "goldhill.png") are used to compare the performance of the proposed algorithm with the benchmark algorithms. The test images are divided into nonoverlapping 4×4 blocks to construct the image vectors. Therefore, each image creates 16384 image vectors. This experiment sets the initial population to 30 and the maximum iterations to 30.

$$bpp = \frac{\log_2 N_c}{N_b} \quad (7)$$

$$PSNR = 10 \log_{10} \left(\frac{255^2}{\text{mean square error between initial \& reconstructed image}} \right) dB \quad (8)$$

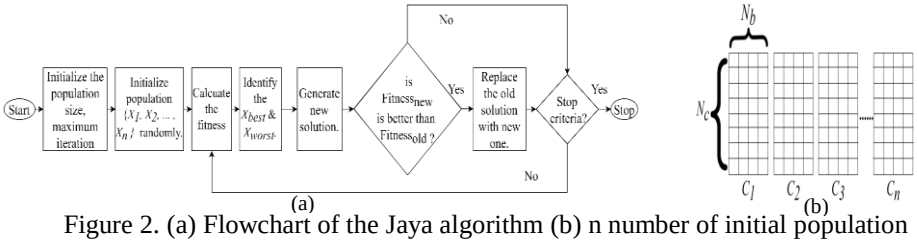

Figure 2. (a) Flowchart of the Jaya algorithm (b) n number of initial population

4.2. Discussion

Proposed algorithm was compared with the benchmark algorithms utilizing the PSNR, bpp and computation time. The PSNR and the computation time presented in this manuscript are the average of the output from five separate execution of the algorithms. The codebook size is {8,16,32,64,128,256,512}. For "Lenna" image, the bpp vs PSNR performance in figure 3a clearly indicates that the proposed algorithm outperforms every other algorithm except PBMLBG. However, for every other test image, the bpp vs PSNR performance of the proposed algorithm outperforms all benchmark algorithms. This demonstrates that the suggested algorithm's reconstructed picture is superior than those produced by other techniques under consideration.

Tables one to four illustrate the execution time required by various algorithms with varying codebook sizes. Conclusions may be derived from this table that the proposed algorithm is quicker than the others for most of the scenario. However, the execution time of the proposed algorithm increases with the bpp. The speed of the proposed algorithm is lagging behind all benchmark algorithms when bpp is 0.5625. The proposed technique is always slower than the PBMLBG because the proposed algorithm requires more iterations to converge than the PBMLBG. When bpp is low, the proposed algorithm minimizes the calculation time of codebook optimization. However, it increases with bpp. Nevertheless, the proposed method consistently delivers the best reconstructed image.

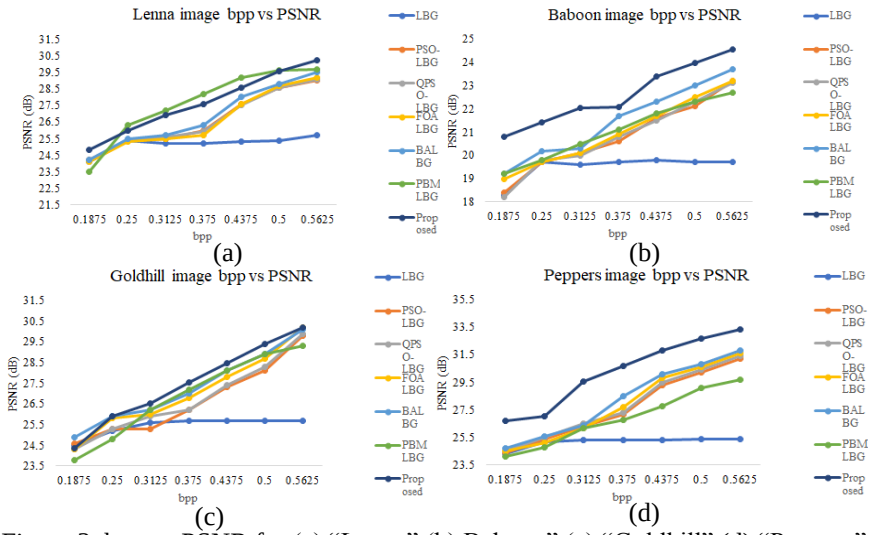

Figure 3. bpp vs PSNR for (a) "Lenna" (b) Baboon" (c) "Goldhill" (d) "Peppers"

Table 1: Average computation time (sec) at bpp =0.375 and block size=64

Image	LBG	PSOLBG	QPSOLBG	FOALBG	BALBG	PBMLBG	Proposed
Lenna	10.1	632.12	632.34	1453.21	678.12	11.2	254.11
Peppers	11.7	612.23	605.21	1301.21	612.23	11.63	147.68
Baboon	11.98	545.21	587.12	1450.21	743.12	10.77	138.82
Goldhill	14.71	598.32	645.32	1201.32	476.43	11.12	151.89

Table 2: Average computation time (sec) at bpp =0.4375 and block size=128

Image	LBG	PSOLBG	QPSOLBG	FOALBG	BALBG	PBMLBG	Proposed
Lenna	13.98	645.32	643.23	1012.21	598.21	11.89	511.63
Peppers	19.21	613.23	687.2	1102.2	643.23	11.71	375.32
Baboon	23.12	476.12	523.51	1051.97	976.21	12.12	303.59
Goldhill	17.98	845.12	876.12	1333.21	512.54	11.23	285.79

Table 3: Average computation time (sec) at bpp =0.5 and block size=256

Image	LBG	PSOLBG	QPSOLBG	FOALBG	BALBG	PBMLBG	Proposed
Lenna	20.84	902.92	887.71	786.77	662.13	12.11	802.15
Peppers	19.17	771.25	767.52	1001.32	587.21	12.12	663.96
Baboon	27.26	621.27	577.1	1021.92	578.21	12.13	620.47
Goldhill	30.72	925.34	554.32	843.21	987.34	12.33	610.04

Table 4: Average computation time (sec) at bpp =0.5625 and block size=512

Image	LBG	PSOLBG	QPSOLBG	FOALBG	BALBG	PBMLBG	Proposed
Lenna	34.21	701.23	723.18	962.38	787.24	12.22	1530.00
Peppers	37.32	962.21	901.27	667.25	512.23	12.12	1310.60
Baboon	21.32	643.23	698.47	751.26	843.12	12.33	1300.30
Goldhill	37.09	576.25	621.12	972.77	934.12	11.33	1240.30

5. CONCLUSION

Proposed algorithm in this article optimizes the codebook of VQ using JA. The proposed algorithm outperforms all benchmark algorithms in terms of PSNR. In this way, proposed algorithm increases the quality of reconstructed image with respect to all benchmark algorithm. Also, JA converges quicker than PSOLBG, QPSOLBG, BALBG, and FOALBG when bpp is low. Future versions of the proposed method may include the PBMLBG outputs as an additional initial solution to further speed up convergence.

6. REFERENCES

[1] Y. Linde, A. Buzo, and R. Gray, “An Algorithm for Vector Quantizer Design,” IEEE trans. commun., vol. 28, no. 1, pp. 84–95, 1980

[2] Rajpoot, N., Hussain, A., Ali, U., Saleem, K., & Qureshi, M., “A novel image coding algorithm using ant colony system vector quantization,” in International workshop on systems, signals and image processing Poznan, Poland, 2004, pp. 13–15.

[3] H.-M. Feng, C.-Y. Chen, and F. Ye, "Evolutionary fuzzy particle swarm optimization vector quantization learning scheme in image compression," Expert Syst. Appl., vol. 32, no. 1, pp. 213–222, 2007.
[4] Y. Wang et al., "A novel quantum swarm evolutionary algorithm and its applications," Neurocomputing, vol. 70, no. 4–6, pp. 633–640, 2007.
[5] Yang, X. S., & He, X. (2013). Firefly algorithm: recent advances and applications. arXiv preprint arXiv:1308.3898.
[6] Horng, M. H. (2012). Vector quantization using the firefly algorithm for image compression. Expert Systems with Applications, 39(1), 1078-1091.
[7] Chiranjeevi, K., Jena, U. R., Murali Krishna, B., & Kumar, J. (2016). Modified firefly algorithm (MFA) based vector quantization for image compression. In Computational Intelligence in Data Mining—Volume 2 (pp. 373-382). Springer, New Delhi.
[8] Karri, C., & Jena, U. (2016). Fast vector quantization using a Bat algorithm for image compression. Engineering Science and Technology, an International Journal, 19(2), 769-781.
[9] Yang, X. S., & Deb, S. (2014). Cuckoo search: recent advances and applications. Neural Computing and applications, 24(1), 169-174.
[10] Chiranjeevi, K., & Jena, U. R. (2018). Image compression based on vector quantization using cuckoo search optimization technique. Ain Shams Engineering Journal, 9(4), 1417-1431.
[11] Bilal, M., Ullah, Z., & Islam, I. U. (2021). Fast codebook generation using pattern-based masking algorithm for image compression. IEEE Access, 9, 98904-98915.
[12] Zitar, R. A., Al-Betar, M. A., Awadallah, M. A., Doush, I. A., & Assaleh, K. (2021). An intensive and comprehensive overview of JAYA algorithm, its versions and applications. Archives of Computational Methods in Engineering, 1-30.
[13] Dash, B., Rup, S., Mohanty, F., & Swamy, M. N. S. (2019). A hybrid block-based motion estimation algorithm using JAYA for video coding techniques. Digital Signal Processing, 88, 160-171.
[14] Debnath, M. K., Sinha, S., & Mallick, R. K. (2017). Application of fuzzy-pidf controller for automatic generation control using jaya algorithm. Int J Pure Appl Math, 114(9), 51-61.
[15] Van der Merwe, D. W., & Engelbrecht, A. P. (2003, December). Data clustering using particle swarm optimization. In The 2003 Congress on Evolutionary Computation, 2003. CEC'03. (Vol. 1, pp. 215-220). IEEE.

Biographies

Suvojit Acharjee is a Ph.D. scholar at Jadavpur University. He has a research interest on soft computing, video coding and signal compression.

Prof. Sheli Sinha Chaudhuri has been serving the ETCE Department of Jadavpur University for the last 23 years. She has a research interest on the field of Machine Learning and Computer Vision.