Optimización de codebook para compresión de imágenes con cuantización vectorial mediante algoritmos genéticos

Nelson Gerardo Peltzer [1], Damian Javier Benassi [2]

1 Ingeniería en Informática. Facultad de Ingeniería y Ciencias Hídricas(FICH). Universidad Nacional del Litoral. Ciudad Universitaria. Santa Fe. Argentina.
nelson.peltzer@gmail.com
2 Ingeniería en Informática. Facultad de Ingeniería y Ciencias Hídricas(FICH). Universidad Nacional del Litoral. Ciudad Universitaria. Santa Fe. Argentina.
damianbe86@gmail.com

Abstract. Los algoritmos genéticos ofrecen un gran abanico de aplicaciones en distintas áreas de la informática. En este trabajo se utilizaron para encontrar un *codebook* óptimo en la compresión de imágenes mediante cuantización vectorial.
Los resultados obtenidos fueron bastante buenos tanto en la tasa de compresión como en la calidad lograda. A pesar de tratarse de una operación lenta, creemos que vale la pena avanzar en esta línea de investigación con fines académicos y tal vez como posible opción práctica en determinadas áreas en las que se necesite buscar el codebook pocas veces en relación a la cantidad de compresiones y descompresiones que se vayan a realizar.

Keywords. Compresión de imágenes, algoritmos genéticos, cuantización vectorial.

1 Introducción

1.1 El método de cuantización vectorial

La cuantización vectorial (VQ) es un método de compresión con pérdida de información, que puede ser aplicado a la compresión de imágenes. Consiste básicamente en discretizar la imagen en pequeños bloques (codewords) de forma que con un número pequeño de éstos se pueda construir una aproximación de la misma. Es decir, siendo n

la cantidad total de bloques de *mxm* pixeles en los que se puede dividir la imagen, encontrar un conjunto de *c* codewords (codebook), *con c < n*, a partir de los cuales se consigue la imagen comprimida.

Para esto, dado un determinado codebook, se divide la imagen en un número determinado de bloques y se elige, para cada uno, cual es el codeword más parecido. Luego solo se guarda un valor que indique el codeword elegido para ese bloque. Este proceso se realiza para todos los bloques que conforman la imagen.

La reconstrucción de la imagen se realiza reemplazando cada valor almacenado con el codeword correspondiente. De esta manera, se logra comprimir la información a almacenar debido a la redundancia existente pero con el costo implícito de perder calidad en la imagen.

La parte de este método que más atención requiere es la de selección de un buen diccionario de bloques (codebook), ya que de esto depende tanto la tasa de compresión como la calidad de la imagen resultante [1].

Es por esto que, a pesar de la gran complejidad computacional que poseen los algoritmos genéticos (GA), los mismos se ofrecen como una solución atractiva para el problema de optimización del codebook en VQ.

Por esta razón, consideramos que el presente trabajo posee un gran valor desde el punto de vista académico ya que explora una solución no convencional para un problema existente.

1.2 Algoritmos genéticos

Los algoritmos genéticos están contenidos dentro de la computación evolutiva. Esta es una rama de la inteligencia artificial inspirada en la teoría de la evolución propuesta por científicos como Charles Darwin y Jean-Baptiste Lamarck.

Una particularidad de los algoritmos genéticos es que realizan la búsqueda en un dominio codificado del problema. Cada individuo que representa una posible solución al problema posee un material genético (genotipo) que lo caracteriza como más o menos apto de acuerdo a una función de aptitud o fitness que mide que tan buena es la solución representada por ese individuo. La idea base del algoritmo es que a través del principio de selección del más apto, estas poblaciones evolucionan colectivamente a lo largo del tiempo generando individuos que cumplan cada vez mejor con las metas indicadas.

Para que la evolución sea posible es importante destacar como condición fundamental la existencia de diversidad en la población. Para garantizar esta diversidad es que se usa la mutación de individuos.

Las metas en los GA se traducen en una función de fitness que pondera la capacidad del individuo para resolver el problema deseado. [2]

2 Descripción del algoritmo genético propuesto

2.1 Representación de individuos

En este problema los individuos a optimizar son los codebooks que son a su vez, como se había mencionado, un conjunto de codewords.

Por esta razón el problema consiste en transformar un conjunto de sub-imágenes en una representación binaria llamada cromosoma.

Las imágenes que vamos a utilizar poseen tres canales de colores con 8 bits de profundidad por canal.

Para la codificación se usó la misma forma de almacenar en memoria que poseen las imágenes de la librería utilizada [8], es decir, primero el canal rojo, luego el verde y por último el azul.

Para visualizar este concepto se presenta el siguiente ejemplo:

Pixel 1: RGB(170,85,42)	Pixel 2: (170,85,127)
Pixel 3: RGB(170,85,85)	Pixel 4: (170,127,85)

Fig. 1. Ejemplo de un codebook de 2x2 pixeles

El gen de la Fig. 1 estará representado por:

Binario(R1.R2.R3.R4.G1.G2.G3.G4.B1.B2.B3.B4)
Binario(170.170.170.170.85.85.85.127.42.127.85.85)
(10101010.10101010.10101010.10101010.01010101.01010101.01010101.
01111111.00101010.01111111.01010101.01010101)

Como un codebook es un conjunto de n codewords distintos, el cromosoma es una concatenación de los genes que representan a cada codeword.

2.2 Inicialización de la población

Para inicializar la población se optó por construir los codewords a partir de fragmentos de la imagen original tomados de forma aleatoria. Esta técnica provoca una solución inicial bastante buena para el problema. No obstante, gracias a la

utilización del algoritmo evolutivo se consigue una mejora importante en el individuo mejor calificado.

Para este trabajo se consideraron codebooks de 16, 32, 64, 128, 256 y 512 codewords de tamaño.

2.3 Función de *fitness*

La elección de una buena función de aptitud resulta fundamental para el buen funcionamiento de un GA.

En este caso, usamos el MSE (del inglés: mean square error) entre la imagen generada a partir un individuo particular y la imagen original. Este valor nos da el grado de error que existe entre la imagen comprimida por el individuo y la original [9].

$$MSE = \sum_{y=1}^{M} \sum_{x=1}^{N} \left[I(x,y) - I'(x,y) \right]^2 \tag{1}$$

Como la función de fitness debe ser mayor cuanto más apto sea el individuo, tomamos el MSE de forma negativa.

$$fitness = -MSE \tag{2}$$

2.4 Mecanismo de selección

En esta etapa es donde se seleccionan los progenitores que van a engendrar los nuevos individuos de la próxima generación. Para esta tarea se utilizó el método de competencia, que consiste en tomar grupos de n individuos al azar, hacerlos competir y seleccionar al ganador de éstos como progenitor. Luego, se arman las parejas y se realiza la cruza.

La cantidad n de individuos que se utilizó fue del 5% del total en cada iteración. [2]. Es decir, se toma el 5% de los individuos al azar y se realiza la competencia. Se toma el ganador y se lo selecciona para la cruza. Luego se toma otro 5% del total (incluidos los que se tomaron anteriormente) y se vuelve a realizar la competencia. Así hasta que se completa la nueva población cruzando a los ganadores.

2.5 Operadores de variación

Operador de cruza y reemplazo. La operación de cruza consiste en seleccionar un punto en el genotipo de los padres e intercambiar toda la cadena de bits entre los mismos a partir de ese punto. De este proceso saldrán dos nuevos individuos que formarán parte de la nueva generación mientras que los padres serán eliminados. Antes de ingresar los individuos a la nueva población se calcula para cada uno, mediante una probabilidad de mutación del 5%, si debe ser mutado o no.

La operación de cruza se realiza con cierto grado de aleatoriedad. Es decir, existe una probabilidad de cruza del 90%. Si se da que los padres no deben cruzarse, entonces los dos pasarán a formar parte de la nueva generación sin cambios, salvo que caigan dentro de la probabilidad de mutación. [3]

Operador de mutación. Para realizar la mutación de individuos se optó por un método que cambie de manera drástica al individuo y obtener de esta forma una búsqueda más global de la solución, ya que de lo contrario, el método se estanca rápidamente en mínimos locales.

La operación de mutación propuesta consiste en tomar un punto al azar en el genotipo del individuo a mutar y a partir de este punto invertir sus valores binarios. Este método genera una gran variación en el individuo. Se optó por ésta operación de mutación tan drástica por el hecho de que la cantidad de bits en cada individuo es muy grande por lo que un cambio de pocos bits no hace una gran diferencia. Por otro lado el hecho de que el algoritmo encuentre una buena solución en la primera época hace que la población se estanque rápidamente sin una operación de mutación que modifique en gran medida los individuos.

2.6 Parámetros de evolución

Para realizar la evolución, se tomó un valor fijo de 60 individuos y se evolucionó por 30 generaciones.

Los GA funcionan mejor mientras la población sea más diversificada. Por lo que teniendo pocos individuos es probable que la población converja a un estado en el que todos los individuos sean parecidos entre sí.

Para nuestro algoritmo se intentó tener la población más grande posible considerando la posibilidad de correrlo en tiempos razonables el algoritmo en computadoras

de prestaciones estándar al día de la fecha y arribando con estos parámetros a soluciones aceptables [3].

3 Variante propuesta

El efecto visual que se obtuvo tanto al reducir el tamaño del codebook como al aumentar el tamaño de los codewords fue similar al producido por la reducción de la profundidad de color en una imagen determinada provocando un efecto de falsos bordes en las zonas homogéneas.

Esto se debe a que la función de aptitud considera más la similitud de la forma geométrica, que el valor de intensidad o brillo de un codeword determinado [7].

Para contrarrestar este efecto propusimos guardar un factor de corrección de iluminación (alpha) al momento de asignar a un bloque de la imagen un determinado codeword. Este valor alpha se calcula como la razón entre la iluminación del bloque de la imagen original y el codeword que va a ocupar ese fragmento de la imagen.

Luego de cuantizar este valor se guarda para aplicarlo a cada bloque individual en el momento de la descompresión.

Se utilizaron 8 bits para la cuantización de este valor, por lo que este factor provoca un aumento del tamaño en bytes igual a la cantidad de bloques en que se dividió la imagen.

4 Resultados y discusión

4.1 Hardware utilizado

Para realizar las pruebas se utilizó el siguiente sistema:
Sistema operativo: Debian 7
Microprocesador: Intel Core 2 Duo e7200 3.2 GHz
Memoria R.A.M.: 2 Gb

4.2 Resultados generales

Para probar el funcionamiento del algoritmo se definieron distintos tamaños de codebooks y codewords. Para medir el desempeño del método se utilizo una medida estándar de compresión de imágenes: el Peak Signal Noise Ratio (PSNR)

$$PSNR = 20 * \log_{10}(\frac{255}{\sqrt{MSE}}) \quad (3)$$

La elección de esta medida se debe a que la mayoría de los trabajos consultados la utilizan y nos resulto útil usarla para comparar nuestros resultados.

Vale la pena mencionar que la medida de calidad PSNR no siempre tiene una relación directa con la percepción humana. Para esto existen otras medidas como HVS (Human Visual System) [9].

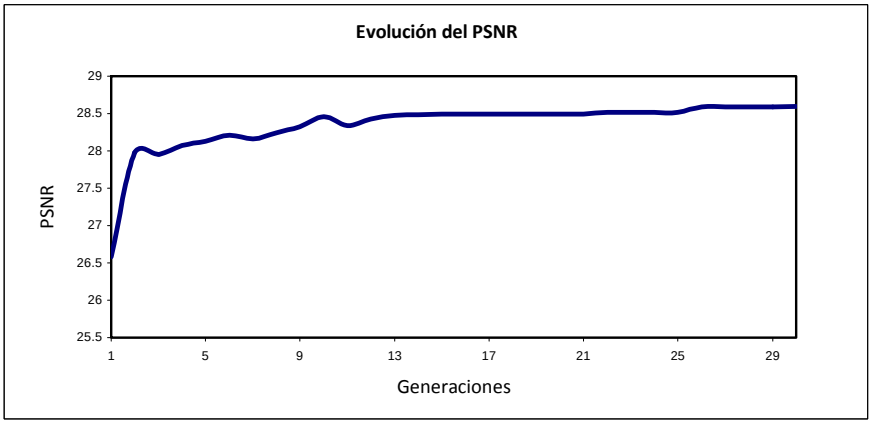


Fig. 2. Evolución del PSNR del mejor individuo utilizando codewords de 2x2 píxeles y un codebook de 256 elementos con la imagen lena.bmp.

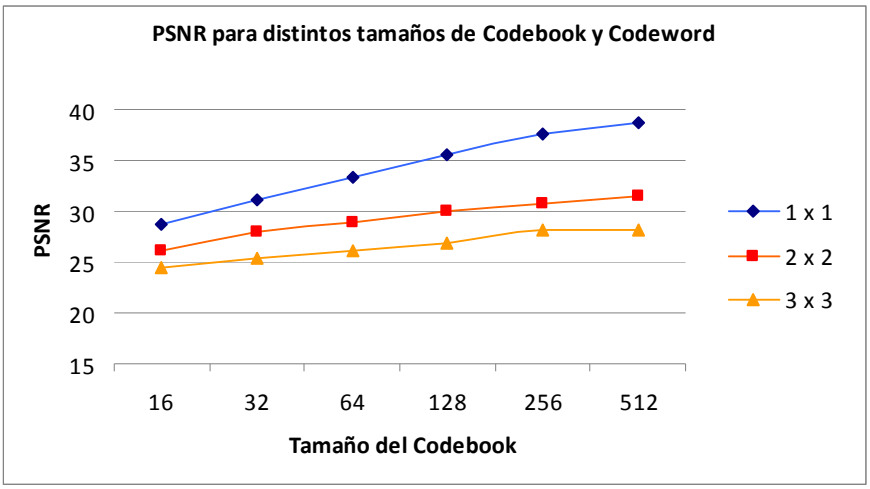


Fig. 3. PSNR para distintos tamaños de codebook y codewords calculados utilizando la imagen lena.bmp

En la Fig. 3 se puede apreciar como el PSNR obtenido mejora casi linealmente al duplicar el tamaño del codebook. O dicho de otra forma, debemos duplicar el tamaño del codebook para obtener una mejora no tan significativa del PSNR. Además vemos en la Fig. 3 que las pendientes de las rectas tienden a disminuir en el tramo que va entre 256 y 512.

Por todo esto concluimos que el tamaño óptimo de codebook es de 256 ya que la mejora que otorgaría un codebook de 512 elementos no es significativa y estaríamos usando un codebook del doble de tamaño con respecto al de 256.

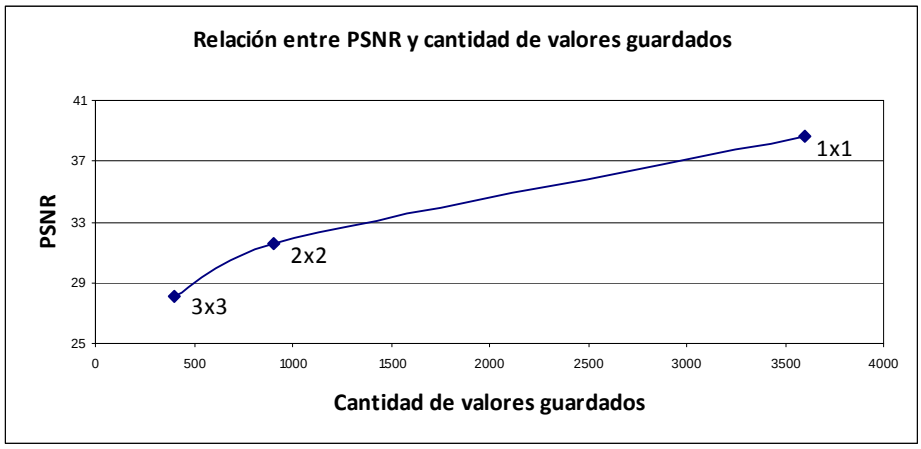


Fig. 4. Relación entre el PSNR y cantidad de valores de asignación de codewords para una imagen de 60x60 pixeles.

En la Fig. 4 la relación entre el PSNR y la cantidad de valores guardados necesarios para comprimir una imagen. Esta última cantidad es una medida directa de la tasa de compresión del método propuesto.

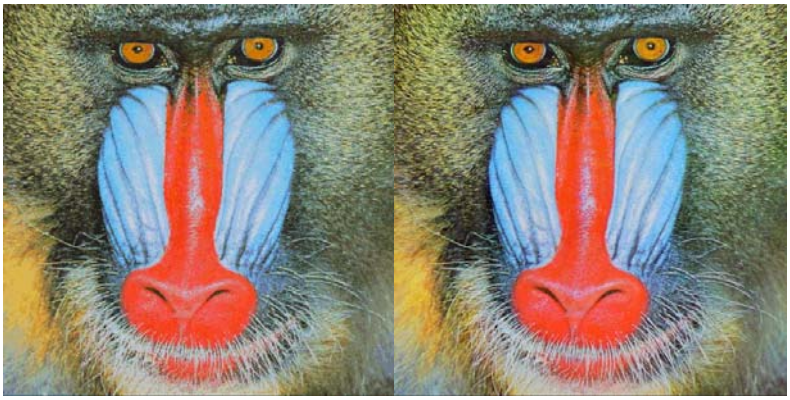
Podemos observar que al pasar de un tamaño de codeword de 3x3 a 2x2 el PSNR crece mas pronunciadamente que al pasar de 2x2 a 1x1; y que además entre 2x2 y 1x1 aumenta notablemente la cantidad de valores guardados necesarios para realizar la compresión y por ende el tamaño de la imagen.

Por lo tanto concluimos que el tamaño de codeword óptimo para nuestro método es de 2x2 pixeles.

(a) (b)

(c) (d)

Fig. 5. Aplicación del tamaño de codeword óptimo (2x2 pixeles) a la imagen lena.bmp para codebooks de a)16, b)21, c)256 y d)512 codewords



a) b)

Fig. 6. a) Imagen baboon.bmp comprimida con un codebook de 256 elementos y codewords de 2x2 pixeles. (PSNR: 29,7865) b) Imagen baboon.bmp sin tratar

a) b)

Fig. 7. a) Imagen peppers.bmp comprimida con un codebook de 256 elementos y codewords de 2x2 pixeles. (PSNR: 29,8179) b) Imagen peppers.bmp sin tratar

a) b)

Fig. 8. Imagen lighthouse.bmp comprimida con un codebook de 256 elementos y codewords de 2x2 pixeles. (PSNR: 30,7) b) Imagen lighthouse.bmp sin tratar

4.3 Comparación con otros métodos

Para comparar el desempeño del método contra otros conocidos se realizó una gráfica PSNR-CR (del inglés: compression ratio).

$$CR = \frac{t(I)}{t(I')} \tag{4}$$

Donde $t(I)$ representa el tamaño de la imagen original y $t(I')$ el tamaño de la imagen comprimida (ambos en bits).

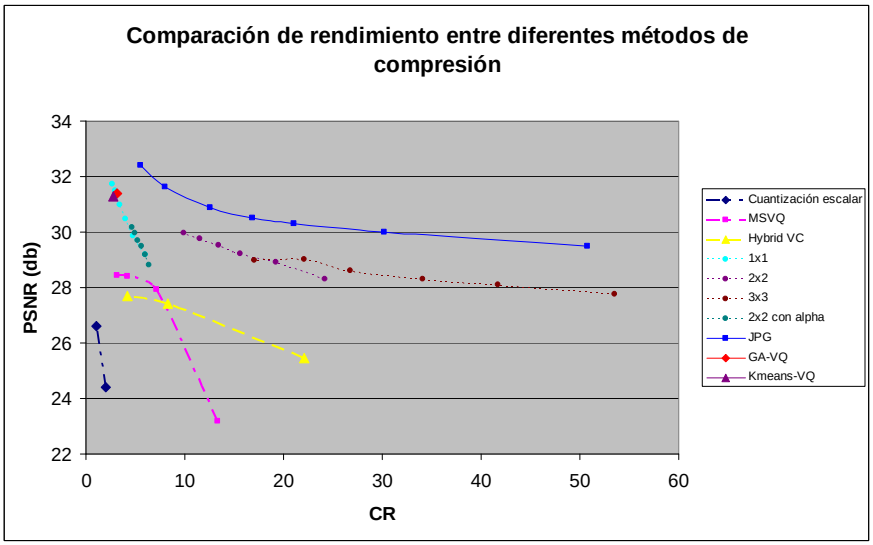


Fig.9. Comparación de distintos métodos de compresión de imágenes aplicados sobre lena.bmp

Para obtener valores de compresión del algoritmo JPG y la cuantización escalar utilizamos el software Gimp 2.8.

Los valores para los métodos Hybrid VQ, MSVQ [1], GA-VQ [4], y KMeans-VQ [11] fueron obtenidos de las publicaciones citadas.

En la Fig. 9 observamos que con el método propuesto podemos cubrir un amplio rango de CR y PSNR si consideramos distintos tamaños de codewords. Esto resulta una ventaja competitiva frente a otros métodos de compresión con cuantización vectorial considerados.

Además podemos observar que el método propuesto presenta relaciones PSNR-CR prácticamente idénticas a los métodos GA-VQ y KMeans-VQ y un desempeño notablemente superior que los métodos MSVQ y Hybrid VC.

Como era de esperar JPG funciono mucho mejor que todos los demás métodos considerados.

4.4 Variante propuesta

Si bien con la variante propuesta se mejoró el PSNR de la imagen resultante, el CR decreció abruptamente en las pruebas. No obstante, considerando que el alpha es un valor que no crece con la cantidad de codewords y crece exponencialmente con la cantidad de bloques, puede considerarse como una mejora interesante cuando se desea procesar imágenes que pueden dividirse en bloques grandes.

Por otro lado, como puede verse en la Fig. 10 el método aplicado a codewords de 2x2 con el agregado del alpha ofrece una mejora con respecto al método con codewords de 1x1 sin variante, ya que obtiene valores mayores o iguales de PSNR con mejor CR y además pudiendo ejecutarse en tiempos menores.

a) b)

Fig. 10. a) Imagen lena.bmp procesada con un codebook de 256 elementos y codewords de 3x3 pixeles. b) Imagen lena.bmp procesada con la variante propuesta, un codebook de 256 elementos y codewords de 3x3

5 Conclusión

El método propuesto mostró un desempeño mayor o igual que los otros algoritmos de compresión de imágenes por cuantización vectorial.

Sin embargo hay que mencionar que el tiempo de cómputo que se necesitó para realizar la compresión de estas imágenes fue mucho más elevado que en otras técnicas que no utilizan GA. Asimismo esta técnica puede ser aprovechada en campos donde se necesite calcular un codebook muy pocas veces o que se utilice el mismo codebook para comprimir una gran cantidad de imágenes.

Puede resultar una técnica interesante en la compresión de foto carnet, donde la cantidad de imágenes a comprimir es muy grande y todas poseen similares características de color y forma. En este caso teniendo solo un codebook grande generado con ésta técnica pueden comprimirse y descomprimirse todas las imágenes.

6 Bibliografía

1. S.Esakkirajan, T. Veerakumar, V. Senthil Murugan and P.Navaneethan.: Image Compression Using Hybrid Vector Quantization. In: International Journal of Information and Communication Engineering 4:1 2008
2. Yimin C.; Yixiao W.; Qibin S.; Longxiang S.: Digital Image Compression Using a Genetic Algorithm. In: Real-Time Imaging, Volume 5, Number 6, December 1999 , pp. 379-383(5)
3. Jenq-Shyang Pan.: Improved Algorithms For VQ Codeword Search, Codebook Design And Codebook Index Assignment.
4. Salah Awad Salman. : Image Compression Using Vector Quantization and Genetic Algorithms. In: Anbar Journal for Engineering Sciences © AJES / 2007.
5. Mohammed A.F. Al-Husainy. : A Tool for Compressing Images Based on Genetic Algorithm. In: Information Technology Journal 6 (3) 457-462,2007. ISSN 1812-5638. © 2007 Asian Network for Scientific Information.
6. G. Mohanbaabu and P. Renuga.: Still image compression using texture and non texture prediction model. In: American Journal of Applied Sciences Volume 9, Issue 4,Pages 519-525.
7. A Vadivel, A K Majumdar, Shamik Sural.: Performance comparison of distance metrics in content-based Image retrieval applications
8. The CImg Library `http://cimg.sourceforge.net`
9. Yusra A. Y. Al-Najjar, Dr. Der Chen Soong.: Comparison of Image Quality Assessment: PSNR, HVS, SSIM, UIQI. In: International Journal of Scientific & Engineering Research, Volume 3, Issue 8, August-2012, ISSN 2229-5518.
10. Gimp. GNU Image Manipulation Program `http://www.gimp.org/`
11. S.Vimala. : Convergence Analysis of Codebook Generation Techniques for Vector Quantization using K-Means Clustering Technique. In: International Journal of Computer Applications (0975 – 8887), Volume 21– No.8, May 2011.