Simulación y Análisis de Algoritmos de Compresión Empleados en un Sistema de Comunicaciones Digitales.

José Miguel Hurtado Madrid
Universidad Tecnológica de Puebla
miguelhm13@hotmail.com

Juan Pedro Cervantes De La Rosa
Universidad Tecnológica de Puebla
pedrocerv@yahoo.com.mx

Resumen

El presente trabajo presenta los resultados de las simulaciones realizadas a tres algoritmos de compresión que serán implementados en un sistema de comunicaciones digitales, el cual, transmite millones de datos por segundo. La compresión de datos consiste en tomar una trama de símbolos y transformarlos en códigos o claves, para representar la señal en un menor espacio. En las comunicaciones digitales y en los sistemas computacionales se emplea para la reducción del volumen de datos. El espacio que ocupa la información codificada sin compresión es el cociente entre la frecuencia de muestreo y la resolución. Cuantos más bits se empleen mayor será el tamaño del archivo. Se realizaron las simulaciones de los algoritmos en un software especializados en cómputo de matrices, en este caso se empleó MATLAB, tomando datos provenientes de escaneos del sistema de detección de defectos.

Palabras clave: Compresión, Comunicación Digital, RLE, Huffman y Shannon Fano.

Introducción

La compresión de datos consiste en tomar una trama de símbolos y transformarlos en códigos o claves, para representar la señal en un menor espacio. En las comunicaciones digitales y en los sistemas computacionales se emplea para la reducción del volumen de datos. El espacio que ocupa la información codificada sin compresión es el cociente entre la frecuencia de muestreo y la resolución. Cuantos más bits se empleen mayor será el tamaño del archivo.

La resolución viene impuesta por el sistema digital con que se trabaja y no se puede alterar el número de bits arbitrariamente; por ello, se utiliza la compresión, para transmitir la misma cantidad de información que ocuparía una gran resolución en un número inferior de bits, la compresión es un caso particular de la codificación, cuya característica principal es que el código resultante tiene menor tamaño que el original.
La compresión de datos implica la reducción en el número de caracteres transmitidos así mismo la probabilidad de error en la transmisión se reduce, aumentando las prestaciones del sistema [1].

METODOLOGÍA EN LA COMPRESIÓN DIGITAL DE SEÑALES
Existen 2 tipos sobre los cuales se clasifican los algoritmos de compresión de datos, son compresión con pérdida y compresión sin pérdida.

Un algoritmo de compresión con pérdida se basa en eliminar datos de la señal para reducir el tamaño, con lo que se suele reducir la calidad [2]. En la compresión con pérdida, la tasa de bits (bit rate) puede presentar una longitud constante o variable [3].

Una vez realizada la compresión, no se puede obtener la señal original, aunque sí una aproximación cuya semejanza con la original está en función del tipo de compresión. Estos algoritmos se aplican principalmente en la compresión de imágenes, videos y sonidos [4].

Los algoritmos de compresión de datos sin pérdida se caracterizan, por recuperar la señal original partiendo de la señal procesada, con lo que no se tiene una perdida en la información en la señal, la compresión en este tipo de

algoritmos es de menor medida que el tipo de compresión con perdida, esto debido a los métodos empleados en cada tipo de compresión [5].

A. *Algoritmo de Compresión Huffman*

El algoritmo de compresión sin pérdida Huffman se caracteriza por el empleo de un diccionario que se construye a partir de la naturaleza de los datos. Es un algoritmo de longitud variable, ya que el tamaño de la palabra para cada valor está en función de las probabilidades del mismo, para la descompresión de este tipo de algoritmo es necesario que el sistema de recepción de datos contenga el mismo diccionario que el sistema transmisión, ya que sin este no es posible recuperar la información recibida [6].

B. *Algoritmo de Compresión RLE*

La compresión Run Length Encoding (RLE) es una forma de compresión de datos en la que secuencias de datos con el mismo valor consecutivas son almacenadas como un único valor más su recuento. Esto es más útil en datos que contienen muchas de estas secuencias; por ejemplo, gráficos sencillos con áreas de color plano, como iconos y logotipos.

El primer byte contiene un número que representa el número de veces que el carácter se repite en la trama, el segundo byte contiene al propio carácter. En otros casos se codifican en un solo byte: 1 bit (0 o 1) y 7 bits para especificar el número de caracteres consecutivos [7].

C. *Algoritmo de Compresión Shannon-Fano*

Codificación Shannon-Fano, posee un código prefijo basado en un conjunto de símbolos y sus probabilidades, garantiza que todas las longitudes de palabras de código están a un bit de su ideal teórico – log P(x). Fue propuesta por Claude Elwood Shannon, en su artículo “Una Teoría Matemática de la Comunicación”, de 1948. El método fue atribuido a Robert Fano, quien posteriormente lo publicó como un informe técnico [8].

Desarrollo

Se realizaron las simulaciones correspondientes a los algoritmos de compresión RLE, Huffman y Shannon-Fano considerando la información del sistema de detección de defectos, como los datos que se desean comprimir:

1) *Compresión Huffman*

Se basa en crear un diagrama de árbol, en el cual cada rama representa un símbolo y la probabilidad de ocurrencia, representa la codificación de los mensajes de la fuente [9].

El código para cada mensaje se construye siguiendo el camino desde el punto de inicio del árbol hasta la rama que representa el mensaje. Además, si el decodificador implementa el mismo árbol usado para comprimir, la decodificación no será más que leer bits e interpretar el camino para la codificación, como en la Fig. 1 [10].

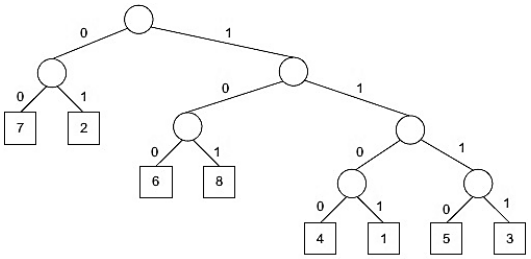

Fig. 1 Diagrama de árbol para la codificación Huffman

La codificación es inversamente proporcional a la probabilidad de aparición del mensaje. El símbolo con mayor probabilidad, se le asignará una codificación más corta asignándole menos símbolos del alfabeto de salida.

Por cada iteración se toman los dos nodos con menor probabilidad del árbol y se crea un nodo para ambos que contendrá la probabilidad sumada de los símbolos. Los nodos con menor probabilidad se localizan en la parte inferior del diagrama.

Generalmente los descompresores de este tipo no tienen posibilidad de conocer previamente las probabilidades de los mensajes, pues sólo recibe los códigos asignados a los mensajes; en consecuencia el árbol ya procesado ha de ser transmitido al descompresor, junto con los datos [11].

B) *Compresión RLE*

El código Run-Length Encoding es utilizado cuando la información tiende a ser poco cambiante, y con mayor tendencia hacia un solo símbolo. Si una señal tiene una alta probabilidad de ocurrencia de uno de sus símbolos, entonces se procederá a agrupar éste, colocando el valor de ese símbolo seguido del número de veces que se repite.

A medida que la señal tienda a ser más estable la codificación será más eficiente, ya que los grupos son más numerosos.

Para agruparlos se debe primero definir cuál de los símbolos se agrupará (en el caso binario el “0” o el “1”), así podemos seleccionar el más probable para esta agrupación.

Luego se debe seleccionar el valor de m, el cual representa la cantidad de bits utilizados para expresar el número de veces que se repite el símbolo seleccionado. Para valores pequeños de m, el código se hace ineficiente cuando los tramos repetidos son grandes, mientras que si este número es muy grande, se perderán muchos bits transmitidos por el “overhead” introducido [12].

El código Run-Length Encoding consta de escribir primero el valor del símbolo repetido, y luego con m bits la cantidad de bits repetidos menos 1 (esto es para aprovechar los m bits completos). Los otros símbolos se copian en el código de manera normal y directa. En la Fig. 2 se puede observar un código Run-Length por agrupación del símbolo “0” y con m=3, de modo que se pueden agrupar hasta 8 bits [13].

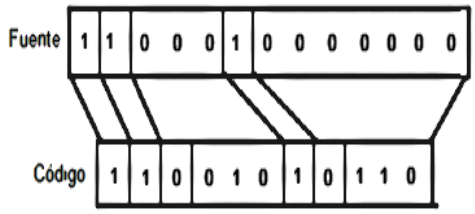

Fig. 2 Código Run-Length Encoding

Para este código la Tasa de Compresión (TC) se define como:

$$TC = \frac{NCod}{NOri} x100\% \qquad (1)$$

En la ecuación (1), NCod es el número total de bits en la palabra codificada, mientras que NOri es el número total de bits en la palabra original.

Para tener una idea de las razones compresión producidas por RLE, consideramos una cadena de N caracteres, que necesita ser comprimida. Se supone que la cadena contiene M repeticiones, con una longitud media de L elementos cada una. Cada una de las M repeticiones, se sustituye por 3 caracteres (el cambio de código, el contador y el dato), por lo que el tamaño de la cadena comprimida $(N - M)(L + M)\ 3 = (N - M)\ (L - 3),$ y el factor de compresión es:

$$\frac{N}{(N - M)\ (L - 3)} \qquad (2)$$

Una variante de la codificación Run-Length Encoding para textos es la codificación de diagramas. Este método es adecuado cuando el número de elementos distintos del bloque a comprimir se reduce sólo, a un número limitado de caracteres, por ejemplo solamente las letras, los dígitos y los signos de puntuación [14].

C) ***Compresión Shannon-Fano***

En la codificación Shannon-Fano, los símbolos se ordenan del más al menos probable, y se dividen en dos subconjuntos cuyas probabilidades totales son tan próximas a ser iguales como sea posible [15]. A continuación todos los símbolos tendrán el primer dígito de sus códigos asignados; los del primer subconjunto recibirán el "0" y los del segundo el "1". Mientras exista algún subconjunto con más de un término, se repetirá el mismo proceso para determinar los sucesivos dígitos de sus códigos, cuando uno de los subconjuntos ha sido reducido a un símbolo, esto significa que el código del símbolo es completo y que no formará el prefijo del código de ningún otro símbolo.

1. Para una secuencia de símbolos, se calcula la correspondiente lista de frecuencias de aparición de los símbolos.
2. Se ordena la lista de símbolos según su frecuencia en orden decreciente.
3. Se divide la lista en dos partes, de forma que la suma total de frecuencias de la mitad superior sea lo más cercana posible a la suma total de la parte inferior
4. A la mitad superior de la lista se le asigna el dígito binario 0, y a la mitad inferior se le asigna el dígito binario 1. Esto significa que los códigos de los símbolos en la primera mitad empezarán todos con 0 y los códigos en la segunda mitad empezarán todos con1.

5. Cada una de las mitades, se subdivide en grupos y se agregan bits (dígitos binarios) a los códigos hasta que cada grupo conste de un único símbolo.
6. Se pueden representar los símbolos a modo de árbol binario.
7. Se calcula la entropía como:

$$X = \frac{Largodelaserie}{frecuencia} \qquad (3)$$

$$Entropía = Log\,2\,(X) \qquad (4)$$

8. Una vez calculada la entropía se calcula la entropía en el mensaje (cantidad de bits necesarios para representar el símbolo en el mensaje) Entropía * frecuencia del símbolo.
9. Finalmente el cálculo de los bits de código a transmitir está dado por la representación binaria (0,1) del símbolo y los bits de mensajes es la multiplicación de los bits de códigos * la frecuencia del símbolo [16].

Resultados

Se realizaron las simulaciones de los algoritmos en un software especializados en cómputo de matrices, en este caso se empleó MATLAB, tomando datos provenientes de escaneos del sistema de detección de defectos.

El objetivos de estas simulaciones es determinar el algoritmo que presente el mejor comportamiento con la información dada, para esto se contemplan diferentes factores a considerar para cada algoritmo.

En la Fig. 3 se observa el número de datos sin procesar contra el número de datos comprimidos, en la cual se resalta el factor de compresión obtenido por este algoritmo, aplicando el criterio de tasa de compresión se obtiene que el porcentaje de compresión obtenida para este algoritmo es del 25.8 %.

En la tabla 1 se presenta una comparación entre el uso del canal en la transmisión de datos, sin comprimir y los datos una vez aplicado el algoritmo de compresión RLE, presenta de manera numérica la eficiencia de este algoritmo de compresión.

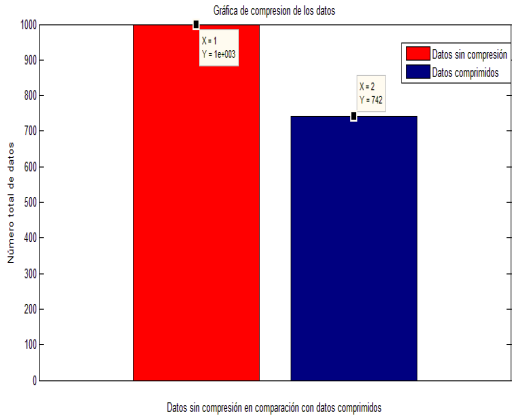

Fig. 3 Resultado de una compresión de datos RLE

Tabla 1. Comparación del uso de la línea de transmisión de una compresión RLE

Número de símbolos sin comprimir	3000
Número de símbolos comprimidos	2226
Número de datos	1000
Ratio de Compresión de datos	0.7420
Porcentaje de Compresión de datos (%)	25.8000

A continuación se presenta parte de la trama de la simulación del algoritmo RLE:

Datos codificados
1404138913881389238813891404138813912389138 81
4011388238923882389138813892388138923881400 14

Un extracto del diccionario utilizado para la realización de los algoritmos de compresión Huffman y Shannon Fano se muestra en la Tabla 2.

Factor de compresión obtenido por este algoritmo, aplicando el criterio de tasa de compresión se obtiene que el porcentaje de compresión obtenido para este algoritmo es del 30.6 % que es muy similar al resultado de la

compresión Shannon Fano que corresponde a un 30.3% de compresión como se observa en la figura 5, así como en las Tablas 2 y 3 se muestra un análisis numérico de los algoritmos de compresión correspondientes a una compresión Huffman y una compresión Shannon Fano.

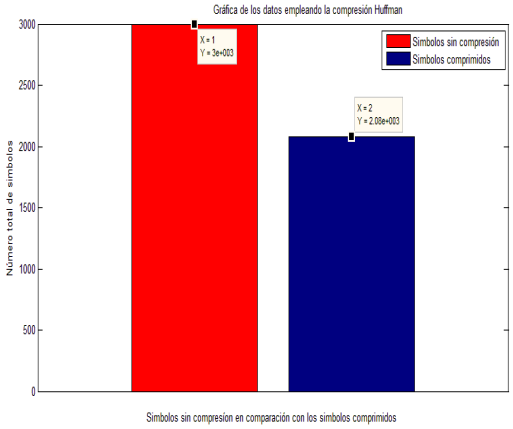

Fig. 4 Resultado de una compresión de datos Huffman

Tabla 2. Comparación del uso de la línea de transmisión de una compresión Huffman

Número de símbolos sin comprimir	3000
Número de símbolos comprimidos	2082
Número de datos	1000
Ratio de Compresión de datos	0.6940
Porcentaje de Compresión de datos (%)	30.6000

A continuación se presenta parte de la trama de la simulación del algoritmo Huffman:

Datos codificados
1101001000101100111001010011101001010100010 1
0010001000111111010101100010001000000000010100

Tabla 2. Diccionarios Huffman y Shannon Fano

Dato	Shannon-Fano	Dato	Huffman
384	111110	384	111011
385	11111100	385	1110101
388	0	388	0
389	10	389	10
391	11110	391	11100
392	1111111100	392	11101000
400	1110	400	1111
401	11111101	401	11101001
404	110	404	110
405	1111111101	405	1110101110

El resultado de la simulación de la compresión Huffman se muestran en la Fig. 4 donde se puede observar el factor de compresión obtenido por este algoritmo, aplicando el criterio de tasa de compresión se obtiene que el porcentaje de compresión obtenido para este algoritmo es del 30.6 % que es muy similar al resultado de la compresión Shannon Fano que corresponde a un 30.3% de compresión como se observa en la figura 5, así como en las Tablas 2 y 3 se muestra un análisis numérico de los algoritmos de compresión correspondientes a una compresión Huffman y una compresión Shannon Fano .

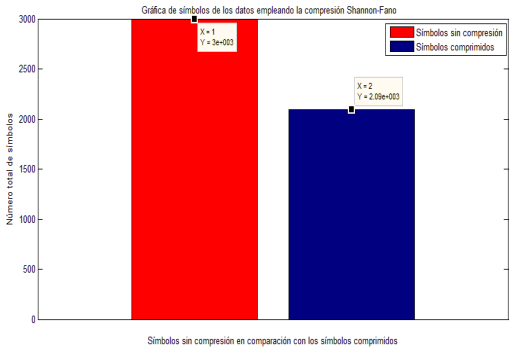

Fig. 5 Resultado de una compresión de datos Shannon Fano

Tabla 3. Comparación del uso de la línea de transmisión de una compresión Shannon Fano

Número de símbolos sin comprimir	3000
Número de símbolos comprimidos	2091
Número de datos	1000
Ratio de Compresión de datos	0.6970
Porcentaje de Compresión de datos (%)	30.3000

Se presenta una parte de la trama de la simulación del algoritmo Shannon Fano:

Datos codificados
1101001000101100111101010011111101010001
0100100010001110110101011000100010000000000

Conclusión

En este trabajo se desarrolló un sistema de compresión de datos en una plataforma software, el cual, emplea algoritmos de compresión sin perdida. Estos se caracterizan por la utilización de diccionarios basados en datos estadísticos de la información que se desea comprimir. Debido a la naturaleza de los datos fue necesario el desarrollo de estos diccionarios, por lo que uno de los principales resultados de este trabajo se centra en el análisis de los algoritmos empleando dichos algoritmos.

El empleo de la plataforma MATLAB permitió la codificación, así como pruebas de compresión y su posterior análisis, empleando cantidades de datos muy extensas, del orden de millones, con esto se asegura que los resultados obtenidos representen el coeficiente de compresión lo más preciso posible.

Se consideró la relación de compresión como primer parámetro de comparación. Dentro de este apartado, el algoritmo Huffman presentó el mejor comportamiento con un porcentaje de compresión del 30.6%.

Como segundo parámetro de comparación se tomó la velocidad de compresión que requiere cada algoritmo, en esta comparación, el algoritmo RLE presentó el mejor comportamiento con una velocidad de ejecución mayor a los algoritmos restantes, esto debido a que este algoritmo no requiere de la búsqueda en el diccionario por cada valor a comprimir.

Como tercer parámetro a considerar se encuentra la entropía, el cual nos proporciona información sobre el uso del canal. De este apartado el algoritmo RLE fue el que presentó el mejor desempeño, con un valor de 4 que muy cercano al valor ideal para esta información cuyo valor ideal es 4.0576, lo que implica que el algoritmo tiene un uso del canal óptimo para este tipo de información. De los parámetros que se consideraron para este trabajo, el algoritmo con mejor desempeño es el algoritmo RLE.

Bibliografía

- Denecker Koen and Van Overloop Jeroen, An *Experimental Comparison of several Lossless Image Coders for Medical Images*, 1068-0314/97, 1997 IEEE.

- Kenneth C. Adkiiis, Mary Jo Shalkhausert, Steven B. Bibyk, *Digital Compression Algorithmsf For Hdtv Transmission,* Dept. of Electrical Engineering The Ohio State University Columbus, Ohio 43210, CH2868 8/90/0000-1022$1.00 0 1990 IEEE.

- Gary Breed, *Bit Error Rate: Fundamental Concepts and Measurement Issues,* High Frequency Electronics, Summit Technical Media, LLC, 2003.

- José E. Briceño M., *Transmisión de Datos*, Publicaciones de la Facultad de Ingeniería Escuela de Ingeniería Eléctrica, Taller de Publicaciones de la Facultad de Ingeniería, ULA, 2005.

- *Compresión de fuente*, notas de clase, Departamento de Ingeniería Telemática ETSET de Barcelona Universidad Politécnica de Cataluña.

- Luis Gabriel Rueda, Manuel Osear Ortega, *Un Modelo de Codificación Dinámica del Método de Huffman, para la Compresión de Datos en Línea*, 1er. Congreso Argentino de Ciencias de la Computación, Domicilio: Ignacio de la Rosa y Meglioli (5400) San Juan.

- David Salomón, Giovanni Motta, David Bryant, *Compresión de datos, La referencia completa*, Springer-Verlag London Limited, Cuarta Edición, Primera en español, ISBN-10: 1-84628-602-6, 2007.

- MacKay D., Information *Theory, Inference, and Learning Algorithms*. Cambridge University Press. Version 7.0 (third printing) August 25, 2004.
- Viraktamath S.V., Attimarad G. V., Bhate Gaurav, *Impact of Selection of Source Coding Technique on the Efficiency* , International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011).

- Anderson Jennings T., *The Mathematics of Cryptography & Data Compression,* Mathematics, Computer Science, & Engineering, Carroll College 2012.

- Amir Said William A. Pearlman, *Digital Signal Compression*. Cambridge University Press, United States of America, 2011.

- C. H. Messom, S. Demidenko2, K. Subramaniam and G. Sen Gupta, *Size/Position Identification in Real-Time Image Processing using Run Length Encoding,* EEE Instrumentation and Measurement, Technology Conference Anchorage, AK, USA, 21-23 May 2002.

- Kussay Nugamesh Mutter, Zubir Mat Jafri, Azlan Abdul Aziz, *Automatic Fingerprint Identification Using Gray Hopfield Neural Network Improved by Run-Length Encoding,* Fifth International Conference on Computer Graphics, Imaging and Visualization, 978-0-7695-3359-9/08 © 2008 IEEE.

- Jie Liang Chengjie Tu and Trac D. Tran. *Adaptive Runlength Coding*. The Johns Hopkins University Department of Electrical and Computer Engineering, Baltimore, MD 21218.

- Xiaoyu Ruan and Rajendra Katti, *Using Improved Shannon-Fano Codes for Data Encryption,* Department of Electrical and Computer Engineering, Seattle, USA, July 9 14, 2006.

- N. Abramson, *Teoría de la Información y Codificación,* Paraninfo, Quinta edición, Madrid, 1981.