Algoritmo de Compresión Probabilístico basado en la Teoría de la Información

Carlos Rincón, Denis Rodríguez, Alfredo Acurero, David Bracho, Juan Jakymec
crincon@luz.edu.ve, denisrodriguex@gmail.com, aacurero@luz.edu.ve, drbracho@luz.edu.ve, jjakymec@luz.edu.ve
Unidad de Redes en Ingeniería Telemática, Facultad Experimental de Ciencias, Universidad del Zulia

RESUMEN

La presente investigación tuvo como finalidad diseñar un nuevo algoritmo de compresión sin pérdida basado en los conceptos y fundamentos de la teoría de la información propuesta por Shannon en 1948. Después de analizar los principios de la teoría de la información y los algoritmos de compresión sin pérdida más usados, se logró diseñar un nuevo algoritmo de compresión utilizando para la generación de los códigos, las posiciones de los símbolos en el mensaje. Como resultado del estudio fue posible comprobar la generación de códigos óptimos por parte del algoritmo diseñado, proponiendo los futuros trabajos que permitan describir el rendimiento del algoritmo.

Palabras Clave: Algoritmo de compresión, compresión sin pérdida, teoría de la información

ABSTRACT

The purpose of this research was to design a new lossless compression algorithm based on Shannon in 1948 concepts and fundamentals of information theory. After reviewing these principles and the most commonly used lossless compression algorithms, a new compression algorithm was designed using symbol's position to generate the compressed codes. As a result of the study, it was possible to prove the generation of optimal codes by the new algorithm, putting forth future works to describe the algorithm´s performance.

Key words: Compression algorithm, lossless compression, information theory.

MOTIVACIÓN

Desde que los seres humanos tienen la capacidad de analizar lo que ocurre en su entorno, se ha generado información. Muchas ciencias definen la información, sin embargo no fue sino hasta mediados del siglo pasado, como consecuencia de la aparición de la computación, que se establecieron los mecanismos para cuantificar la información.

A pesar que los sistemas de cómputo tienen como tarea fundamental generar información, la mayoría de los estudiantes y egresados en el área de computación definen la información como un conjunto de datos que estructurados tienen un significado y sólo saben que los computadores almacenan o transmiten la información utilizando la unidad mínima llamada bit.

En 1948, Claude Shannon (1) se encargó de definir un método matemático que permitiera cuantificar la información generada por una fuente de datos. Shannon logró determinar lo que hoy llamamos teoría de la información utilizando un concepto relativamente sencillo: considerando que todo evento tiene una probabilidad de ocurrencia, la cantidad de información producto de este evento puede cuantificarse mediante una función representada como la inversa de la probabilidad de ocurrencia de dicho evento. De este concepto se concluye que un evento con menor probabilidad de ocurrencia generará mayor información que un evento con mayor probabilidad de ocurrencia.

La necesidad del ser humano de procesar mayor cantidad de datos, ha generado un problema debido a que se ha incrementado la cantidad de información generada. La información debe ser almacenada y transmitida, por lo que desde el punto de vista computacional, se ha incrementado en el tiempo la capacidad de almacenamiento y de transmisión de los sistemas informáticos. Sin embargo, la cantidad de información generada excede la capacidad de los sistemas de cómputo, lo que se traduce en la necesidad de establecer mecanismo que permitan aprovechar de manera efectiva los recursos computacionales.

Para resolver el problema planteado, la ciencia de la computación utilizó las mismas técnicas utilizadas en el pasado para resolver problemas similares. Mediante la codificación eficiente de los símbolos, se obtiene la representación de la información original utilizando una menor cantidad de unidades de información.

La compresión de datos es el proceso que permite encontrar la codificación óptima para representar los datos generados por una fuente, con la finalidad de minimizar la cantidad de información producida por la misma. Según Sayood (3), los algoritmos de compresión se clasifican en algoritmos con pérdida (cuando la información que resulta del proceso de compresión no es exactamente igual a la información original) y algoritmos sin pérdida (cuando la información que resulta del proceso de compresión es exactamente igual a la original). El uso de técnicas de compresión con pérdida o sin pérdida dependerá de las características de la información a comprimir.

A principios de los 50, David Huffman (2) alumno de Robert Fano (compañero de Shannon) en respuesta a una asignación en el MIT, planteó una técnica que permite generar códigos óptimos para realizar el proceso de

compresión. Está técnica de compresión sin pérdida se conoce en la actualidad como el algoritmo de compresión de Huffman, el cual mantiene su vigencia a pesar de la introducción de nuevos algoritmos para generar códigos óptimos.

El algoritmo de compresión de Huffman, fue diseñado en 1952 por David Huffman fundamentado en la teoría de la información propuesta en 1942 por su profesor en el Instituto Tecnológico de Massachussets Claude Shannon. A pesar del tiempo que ha pasado desde su diseño y el desarrollo de nuevos algoritmos de compresión de datos, Abraham Bookstein (4) en su investigación titulada "Is Huffman Coding Dead?, concluyó que el algoritmo de Huffman puede utilizarse actualmente para resolver diferentes problemas informáticos que requieran de la compresión de datos.

Una de las aplicaciones en las que actualmente se utiliza el algoritmo de compresión de Huffman es la académica, como lo demuestran los trabajos de Mohamed Hamada (5) y James Keeler (6), donde se implementa el algoritmo de Huffman para demostrar los conceptos implícitos en la teoría de la información.

El propósito de la presente investigación consistió en desarrollar un algoritmo de compresión basado en la teoría de la información, con la finalidad de brindar al mundo de la ciencia de la computación de una herramienta más para implementar soluciones que permitan minimizar la cantidad de bits con la que se debe representar una información.

DISEÑO DEL ALGORITMO

El diseño del algoritmo de compresión probabilístico propuesto se realizó utilizando como fundamento para la generación de los códigos, la posición de los símbolos como método para la creación de los nuevos códigos, evitando así la necesidad de escribir el mensaje nuevamente con diferentes códigos.

Como cualquier algoritmo probabilístico, el algoritmo propuesto se fundamenta en la generación de códigos óptimos utilizando como parámetro la frecuencia de aparición de los símbolos en el mensaje. El concepto de información propuesto por Shannon permite asignar a los símbolos con mayor frecuencia, códigos de menor tamaño, permitiendo así minimizar la cantidad de bits para representar el mensaje, logrando algún grado de compresión.

Proceso de Compresión

El algoritmo de compresión propuesto utiliza la posición de los símbolos del mensaje en vez de generar un código de sustitución. Para tal fin se dispone de una cadena de bits la cual se encargará de mantener la posición de un determinado símbolo. Cada símbolo en el mensaje dispone de una cadena binaria, y es ésta información la que en último término se almacena.

El proceso de compresión consiste en (ver figura 1): 1) recorrer todo el mensaje realizando el cálculo de la frecuencia de los símbolos presentes en el mismo, 2) determinar cuál símbolo del mensaje tiene mayor frecuencia, 3) generar una cola de símbolos ordenándolos según su frecuencia de mayor a menor e inicializar la cadena de bits para el mensaje comprimido, 4) extraer de la cola el símbolo con mayor frecuencia, 5) anexar a la cadena de bits del mensaje comprimido, una cadena de n bits (donde n es el tamaño del mensaje), la cual tendrá 1 en las posiciones del mensaje donde aparezca el símbolo extraído de la cola y 0 en donde éste no aparezca, 6) eliminar el símbolo extraído de la cola del mensaje original, 7) verificar si la cola de símbolos está vacía, en el caso de que sea cierto se finaliza el proceso y en el caso de ser falso ir al paso 4.

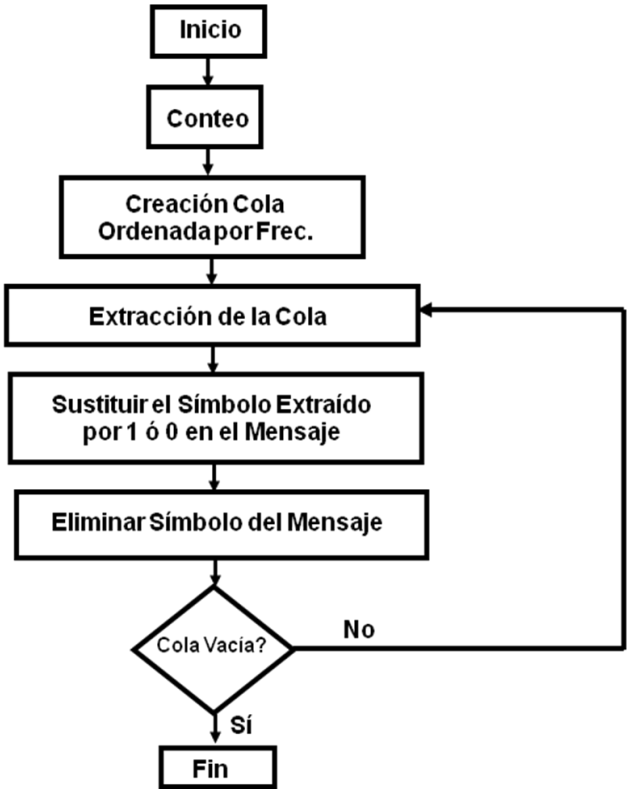

Figura 1. Proceso de Compresión
Fuente Propia

Proceso de Descompresión

Para retornar al mensaje original sólo es necesario conocer el tamaño en bits utilizado para representar cada símbolo presente en el mensaje y la cadena de bits con el mensaje comprimido. Como la secuencia binaria se generó en el orden de mayor a menor con respecto a las frecuencias, el proceso se realiza de forma inversa

empezando por el último símbolo (el de menor frecuencia), sabiendo la cantidad de bits que ocupa, éstos se irán eliminando de la cadena binaria a medida que el proceso avance. Al mismo tiempo una cadena de caracteres se irá formando de acuerdo con el contenido de los bits correspondientes a cada símbolo (ver figura 2).

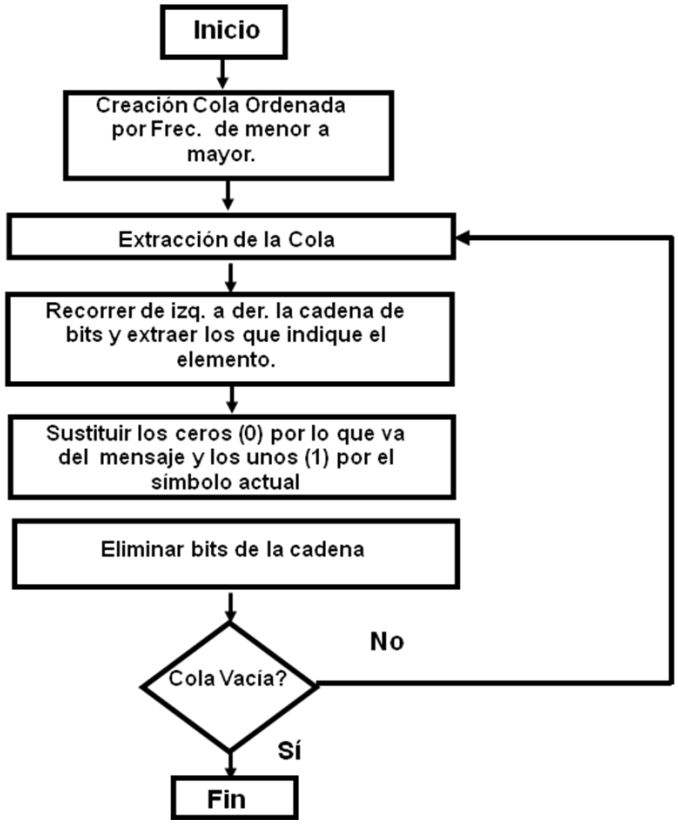

Figura 2. Proceso de Descompresión
Fuente Propia

ANÁLISIS MATEMÁTICO DEL RENDIMIENTO DEL ALGORITMO

Para analizar el rendimiento del algoritmo, al igual que en la mayoría de los análisis matemáticos realizados a los algoritmos de compresión probabilísticos sin pérdida, existen 2 casos a considerar:

El mejor caso: cuando un mensaje está compuesto por n símbolos iguales, el número de bits que utiliza el algoritmo para generar el mensaje comprimido (sin considerar la información de control), será igual a n bits, dado a que el algoritmo representará el mensaje con n 1.

El peor caso: cuando un mensaje está compuesto por n símbolos tomados de un alfabeto conformado por m elementos y en dónde los símbolos aparecen de manera equiprobable en el mensaje, el rendimiento del algoritmo puede representarse mediante la siguiente serie matemática:

$$n+\left(n-\frac{n}{m}\right)+\left(n-\frac{2*n}{m}\right)+....+\left(n-\frac{(m-1)*n}{m}\right)$$

Simplificado, tenemos:

$$\sum_{i=1}^{m} n-\sum_{i=o}^{m-1}\left(\frac{in}{m}\right)=n*m-\sum_{i=0}^{m-1}(i)*\frac{n}{m}=$$

$$n*m-\left(\frac{m*(m-1)}{2}\right)*\frac{n}{m}=n*m-\left(\frac{n*(m-1)}{2}\right)=$$

$$n*\left(m-\frac{m-1}{2}\right)\ \text{bits}$$

El rendimiento (en número de bits para representar el mensaje comprimido) del algoritmo planteado en este análisis matemático, no considera el tamaño en bits de la cabecera que debe anexarse al mensaje comprimido para poder ser descomprimido.

UN EJEMPLO DE LA EJECUCIÓN DEL ALGORITMO

El proceso de compresión planteado se ilustrará en el siguiente ejemplo:

Tamaño del Símbolo: 8 bits.

Mensaje: ABCABCACBAAABCCCCBDF

1) Conteo: A(6), B(5), C(7), D(1), F(1)

2) Ordenar.
Frecuencias: C(7), A(6), B(5), D(1), F(1)
3) Extraer de la Cola de Frecuencias.
Elemento (C).

4) Recorrer para el Elemento (C).

5) Cambiar por bits en la Lista Paralela.

A	B	C	A	B	C	A	C	B	A	A	A	B	C	C	C	C	B	D	F
0	0	1	0	0	1	0	1	0	0	0	0	0	1	1	1	1	0	0	0

Lista Paralela de bits = 00100101000001111000.

6) Eliminar el símbolo del Mensaje.
ABABABAAABBDF

7) Ir al paso 3.

3) Extraer de la Cola de Frecuencias.
Elemento (A).

4) Recorrer para el Elemento (A).

5) Cambiar por bits en la Lista Paralela.

A	B	A	B	A	B	A	A	A	B	B	D	F
1	0	1	0	1	0	1	1	1	0	0	0	0

Lista Paralela de bits = 1010101110000

6) Eliminar el símbolo del Mensaje.
BBBBBDF

7) Ir al paso 3.

3) Extraer de la Cola de Frecuencias.
Elemento (B).

4) Recorrer para el Elemento (B).

5) Cambiar por bits en la Lista Paralela.

B	B	B	B	B	D	F
1	1	1	1	1	0	0

Lista Paralela de bits= 1111100

6) Eliminar el símbolo del Mensaje.
DF

7) Ir al paso 3.

3) Extraer de la Cola de Frecuencias.
Elemento (D).

4) Recorrer para el Elemento (D).

5) Cambiar por bits en la Lista Paralela.

D	F
1	0

Lista Paralela de bits= 10

6) Eliminar el símbolo del Mensaje.
F

7) Ir al paso 3.

3) Extraer de la Cola de Frecuencias.
Elemento (F).

4) Recorrer para el Elemento (F).

5) Cambiar por bits en la Lista Paralela.

F
1

Lista Paralela de bits = 1

6) Eliminar el símbolo del Mensaje.
--

8) Unir las listas de bits generadas.

00100101000001111000	1010101110000	1111100	10	1
C	A	B	D	F

Quedando como resultado final la siguiente cadena de bits la cual no es más que la concatenación de todas las cadenas binarias que se generaron por cada símbolo y dispuestas en el orden en se crearon.

C= 20; A=13; B=7; D=2; F=1;

0010010100000111100010101011100001111100101 = 43 bits.

Mientras que el mensaje original:
ABCABCACBAAABCCCCBDF = 20*8 = 160 bits.

Es importante recalcar que a la cantidad de bits utilizados para representar el mensaje comprimido hay que sumarle la información de control necesaria para descomprimir el mensaje.

En el siguiente ejemplo se muestra como trabaja el proceso de restauración del mensaje.

Ejemplo:
Teniendo: C= 20; A=13; B=7; D=2; F=1;
0010010100000111100010101011100001111100101.

1) Se ordena en orden inverso al que obtuvimos.
F = 1;
D = 2;
B = 7;
A = 13;
C = 20;

2) Se extrae al primero de la cola.
Se extrae F.

3) Se obtiene el bloque desde el final de la Cadena Binaria del tamaño que indica la cabecera para F.
(1) Último bloque de tamaño que F indica.
Mensaje = F.

4) Se elimina el bloque de F de la Cadena Binaria.

00100101000001111000101010111000001111110010

5) Ir al paso 2.

2) Se extrae al primero de la cola.
Se extrae D.

3) Se obtiene el bloque desde el final de la Cadena Binaria del tamaño que indica la cabecera para D.
(10) Último bloque de tamaño que D indica.
Mensaje = F.
1 indica que se coloca D
0 Indica que se coloca un elemento de lo que estaba antes en el mensaje.
Mensaje = DF.

4) Se elimina el bloque de D de la Cadena Binaria.
001001010000011110001010101110000111110 0

5) Ir al paso 2.

2) Se extrae al primero de la cola.
Se extrae B.

3) Se obtiene el bloque desde el final de la Cadena Binaria del tamaño que indica la cabecera para B.
(1111100) Último bloque de tamaño que B indica.
Mensaje = DF.
1 indica que se coloca B
0 Indica que se coloca un elemento de lo que estaba antes en el mensaje.
Mensaje = BBBBBDF.

4) Se elimina el bloque de B de la Cadena Binaria.
00100101000001111000101010111 0000

5) Ir al paso 2.

2) Se extrae al primero de la cola.
Se extrae A.

3) Se obtiene el bloque desde el final de la Cadena Binaria del tamaño que indica la cabecera para A.
(1010101110000) Último bloque de tamaño que A indica.
Mensaje = BBBBBDF.
1 indica que se coloca A
0 Indica que se coloca un elemento de lo que estaba antes en el mensaje.
Mensaje = ABABABAAABBDF.

4) Se elimina el bloque de A de la Cadena Binaria.
00100101000001111000

5) Ir al paso 2.

2) Se extrae al primero de la cola.
Se extrae C.

3) Se obtiene el bloque desde el final de la Cadena Binaria del tamaño que indica la cabecera para C.
(00100101000001111000) Último bloque de tamaño que C indica.
Mensaje = ABABABAAABBDF.
1 indica que se coloca C
0 Indica que se coloca un elemento de lo que estaba antes en el mensaje.
Mensaje = ABCABCACBAAABCCCCBDF

4) Se elimina el bloque de C de la Cadena Binaria.
--

Como se puede apreciar, se logra la restauración de la cadena original del mensaje, empezando por el último elemento del alfabeto creado hasta llegar al primero.

CONCLUSIONES Y TRABAJOS FUTUROS

Como resultado de la investigación, se logró diseñar un nuevo algoritmo de compresión probabilístico sin pérdida. La innovación del algoritmo propuesto se fundamenta en la utilización de las posiciones de los símbolos en el mensaje para generar los códigos de estos símbolos, dado a que normalmente los algoritmos de compresión probabilísticos sin pérdida generan sus códigos a partir de la cantidad de información que los símbolos aportan al mensaje. La utilización del tamaño del mensaje y la posición de los símbolos en el mensaje como parámetros fundamentales para la generación de los códigos, garantiza que dichos códigos sean óptimos.

Para analizar el rendimiento del algoritmo planteado, se propone realizar un análisis inicial sobre el efecto de los parámetros fundamentales de la teoría de la información (tamaño del mensaje, entropía, entre otros) sobre el algoritmo, para luego realizar una comparación de rendimiento entre el algoritmo producto de la investigación y los algoritmos probabilísticos sin pérdida más utilizados (Huffman, Shannon-Fano, entre otros).

REFERENCIAS

(1) Shannon, C. E. 1948. A mathematical theory of communication. Bell Systems Technical Journal, 27:379-423, 623-656.

(2) Huffman, D. 1952. A method for the construction of minimum redundancy codes. In Proc. IRE. 40, 9, 1098—1101

(3) Bookstein, A., Klein, S. T., and Raita, T. 1993. Is Huffman coding dead? (extended abstract). In Proceedings of the 16th Annual international ACM SIGIR Conference on Research and Development in information Retrieval (Pittsburgh, Pennsylvania, United States, June 27 - July 01, 1993). R. Korfhage, E. Rasmussen, and P. Willett, Eds. SIGIR '93. ACM, New York, NY, 80-87. DOI= http://doi.acm.org/10.1145/160688.160697

(4) Sayood, K. 1996. Introduction to Data Compression. Morgan Kaufmann Publishers, INC. San Francisco, California, USA.

(5) Hamada, M. 2007. Web-based tools for active learning in information theory. In Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (Covington, Kentucky, USA, March 07 - 11, 2007). SIGCSE '07. ACM, New York, NY, 60-64. DOI= http://doi.acm.org/10.1145/1227310.1227332

(6) Keeler, J. 2004. Graphical implementation of Huffman and arithmetic coders. J. Comput. Small Coll. 19, 5 (May. 2004), 289-290.