OPTIMIZACIÓN DEL USO DE ANCHO DE BANDA EN LOS ENLACES DE TRANSMISIÓN DE DATOS POR MEDIO DE ALGORITMOS DE COMPRESIÓN.

OPTIMIZATION OF THE USE OF BANDWIDTH IN DATA TRANSMISSION LINKS THROUGH COMPRESSION ALGORITHMS.

Luis Benavides Castillo, Mgs.
Magíster en Auditoría en Tecnologías de la Información (Ecuador).
Docente de la Facultad de Ingeniería en Sistemas, Telecomunicaciones y Electrónica de la Universidad Espíritu Santo, Ecuador.
lebenavides@uees.edu.ec

Mónica Flores Marín, MsC.
Máster en Business Administration (Ecuador).
Docente de la Facultad de Ingeniería en Sistemas, Telecomunicaciones y Electrónica de la Universidad Espíritu Santo, Ecuador.
mflores@uees.edu.ec

Gabriel José Ortega Jouvin, Ing.
Ingeniero en Telecomunicaciones (Ecuador).
Jefe de producto de Networking y Telefonía en la empresa La Casa del Cable, Guayaquil, Ecuador.
gjortega@uees.edu.ec

Iván Stefano Cherrez Chica, Sr.
Estudiante de la Universidad Espíritu Santo, Ecuador.
icherrez@uees.edu.ec

ARTÍCULO DE INVESTIGACIÓN
Recibido: 6 de septiembre de 2018.
Aceptado: 24 de octubre de 2018.

RESUMEN

Este trabajo se lleva a cabo mediante un diseño experimental, enfoque cuantitativo. Se presenta un estudio comparativo en términos de tasa de compresión y consumo de

ancho de banda entre los algoritmos RLE, LZW y Huffman Adaptativo para seis archivos de texto, 15 de audio y 20 imágenes. Para realizar esto, se utiliza Matlab como herramienta de simulación de los algoritmos. Los principales resultados de este trabajo muestran que RLE incrementa en promedio el 50,56% del tamaño de los archivos en el 85,37% de los casos. Mientras que, LZW y Huffman Adaptativo logra reducir el tamaño en un 65,85% y 97,56% de los casos respectivamente. Asimismo, Huffman Adaptativo tiene el mejor valor en ahorro de ancho de banda con un promedio de 25,99%, seguido por LZW con un 13,59%. En base a estos resultados, se concluye que el algoritmo Huffman Adaptativo es el más idóneo para lograr un mayor ahorro de ancho de banda frente a los otros dos algoritmos probados.

Palabras clave: Algoritmos de compresión, ancho de banda, RLE, LZW, Huffman Adaptativo.

ABSTRACT

This paper represents an experimental study, with a quantitative approach of three compression algorithms. A comparison in terms of compression ratio and bandwidth consumption is driven between the algorithms RLE, LZW, and Adaptive Huffman. The tests are made in a simulated scenario using Matlab with six text files, 15 audio files and 20 images. The main results show that RLE increments the size of the files by 50,56% average in 85,37% of the cases. Meanwhile, LZW and Adaptive Huffman reduced the size of the files in 65,85% and 97,56% of the tests respectively. Adaptive Huffman got the best bandwidth saving average with 25,99%, followed by LZW that hot a 13,59% of average savings. As conclusion, Adaptive Huffman is the ideal algorithm beyond the three tested to get a better bandwidth consumption save.

Keywords: Compression algorithms, bandwidth, RLE, LZW, Adaptive Huffman.

INTRODUCCIÓN

La información no comprimida requiere considerables cantidades tanto de capacidad de almacenamiento como de ancho de banda en las transmisiones. A pesar de los rápidos avances tecnológicos registrados en áreas como disponibilidad de almacenamiento en masa, velocidad de

procesamiento, mejoras en los sistemas de comunicación digitales, entre otros; el ancho de banda continua siendo un recurso limitado (Slaby, Hercik, & Machacek, 2013). Ante todo lo mencionado, se hace inminente el uso de algún mecanismo capaz de representar la información en un tamaño menor al original sin disminuir su calidad en lo absoluto o de forma considerable, a esto se le llama compresión (Hussen, Mahmud, & Mohammed, 2011).

Adicionalmente, Bandyopadhyay, Paul & Raychoudhury (2011) hacen hincapié en que una imagen ocupa más espacio en un disco de almacenamiento o ancho de banda en un sistema de transmisión de lo que ocupa el texto. Ante la variación de archivos que pueden enviarse se debe encontrar la forma de transmitir todos usando la menor cantidad de recursos tales como ancho de banda. Por eso es importante conocer los algoritmos de compresión que logren tener las mejores tasas de compresión para los diferentes tipos de archivos. De la misma forma, Sangvikar & Joshi (2011) recalcan que la enorme cantidad de información que es almacenada, procesada y transmitida día a día sobrepasaría los límites físicos actuales en caso de no ser comprimida.

Como indican Alamin & Mustafa (2015) la razón más importante para comprimir información es que el ancho de banda es un recurso limitado, lo cual influye en la cantidad de bits que se pueden transmitir por un medio en un tiempo determinado. Este límite de ancho de banda, en conjunto con el crecimiento constante de información a transmitirse hacen que la compresión sea vital para las comunicaciones.

De la misma forma, Slaby, Hercik & Machacek (2013) resaltan la importancia que tienen los algoritmos de compresión dado que los recursos son limitados en la capacidad de transmisión de datos. La tarea principal de la compresión que se ve reflejada en los algoritmos es de reducir el ancho de banda utilizado. A esto, Al-Hashemi & Al-Dmour (2011) añaden que la compresión de imágenes es uno de los pilares fundamentales en aplicaciones de almacenamiento en bases de datos, archivado de imágenes, televisión y video conferencias. Por este motivo, resulta de gran importancia realizar un análisis de diferentes algoritmos de compresión y así determinar su eficiencia ante diferentes tipos de secuencias de bits. En este trabajo se propone un análisis desde el punto de vista del ancho de banda.

Este trabajo está enfocado en la comparación de los algoritmos de compresión LZW, RLE y Huffman Adaptativo con la intención de conocer cuál de ellos es más eficiente, tomando como referencia la

tasa de compresión, para este propósito, se trabaja con imágenes en escala de grises y a color; archivos de audio de un solo canal y archivos de texto aleatorio. De esta forma se podrán relacionar estos resultados con el consumo de ancho de banda en una posible transmisión.

1. REVISIÓN TEÓRICA

1.1 Algoritmo.

Un algoritmo es una secuencia ordenada de pasos, exenta de ambigüedad que conduce a la resolución de un problema determinado en un número finito de pasos (Pere-Pau, Jordi, Angela, & Xavier, 2006). Adicionalmente, Neapolitan & Naimipour (2010) indican que un programa informático está compuesto por módulos individuales que buscan cumplir con tareas específicas a las que denominan problemas, donde cada uno de estos problemas tiene tanto entradas como salidas. Consecuentemente, describen a un algoritmo como el proceso paso a paso que resuelve un problema. Las maneras más comunes que se usan para expresar algoritmos son mediante lenguaje común, que está limitado a algoritmos sencillos, o a través de pseudocódigo.

1.2 Ancho de banda.

El ancho de banda es una característica natural de todos los medios y se define como el rango de frecuencias de las señales que se pueden transmitir por el medio físico (Gil, Pomares, & Candelas, 2010).

Sklar (2001) expresa brevemente el ancho de banda dentro del marco de las frecuencias de comunicaciones como la diferencia existente entre el valor más alto y el valor más bajo de frecuencia que usa una señal para la transmisión en un medio. Esta medida se la obtiene en hertz.

En lo que respecta al mínmo ancho de banda que puede utilizar una transmisión en un sistema con una tasa de transmisión de R símbolos por segundo sin que exista interferencia entre símbolos, el teorema de Nyquist lo expresa como R/2 Hertz en un escenario ideal. Por el otro lado, e un escenario más realista esta cifra de 2 símbolos por Hertz se puede ver reducido a un valor entre 1.4 y 1.8 símbolos por Hertz (Sklar, 2001).

1.3 Transmisión de datos.

Fariña (2005) planteó una breve descripción de estos procesos a partir de su configuración en cuatro etapas bien diferenciadas: compresión, transmisión, recepción y descompresión. Las dos primeras son desempeñadas por el emisor mientras que los dos últimas son cumplidas por el receptor. Un ejemplo sencillo lo representa la descarga de un archivo comprimido desde un servidor web en el que previamente la información se procesa y comprime para su transmisión. En el otro lado de la

red de comunicación, una vez la transmisión ha finalizado, el receptor descomprime dicha información para su uso posterior. Un paso más allá del presente ejemplo se pueden encontrar los escenarios de transmisión en tiempo real en el que las cuatro etapas anteriores se llevan a cabo concurrente y constantemente, de tal forma que los procesos de compresión y transmisión, en el origen, se ejecutan simultáneamente a la recepción y descompresión en destino. En estos casos, las técnicas de compresión adaptativas ofrecen una solución más competitiva dado que no necesitan obtener información de forma previa del mensaje y, por tanto, pueden modelarlo y codificarlo como un flujo continuo de datos (Martínez, 2010).

1.4 Compresión de datos.

Muchos programas necesitan usar archivos que tienen un tamaño mayor del que pueden tratar, por lo que es necesario comprimirlos y descomprimirlos. La compresión de datos consiste en reducir el volumen de los datos para que ocupen menos espacios y la descompresión es el proceso contrario, es decir consiste en aumentar el volumen de los datos.

A la hora de comprimir los datos se utiliza una serie de algoritmos que pueden implicar una pérdida de información o no. Cuando no se produce pérdida de información se trata tan solo de compactar la información de forma que se puede recuperar la original. Pero cuando se utilizan algoritmos que provocan la pérdida de datos el usuario decide cuánta información está dispuesto a perder para que el archivo pese menos (Mondelo & Iglesias, 2014).

La compresión de datos es la codificación de un cuerpo de datos D en un cuerpo de datos más pequeño D'. Para comprimir los datos, los métodos de compresión examinan los datos, buscan redundancia en ellos, e intentan removerla. Una parte central en la compresión es la redundancia en los datos. Solo los datos con redundancia pueden comprimirse aplicando un método o algoritmo de compresión que elimine o remueva de alguna forma dicha redundancia. La redundancia depende del tipo de datos (texto, imágenes, sonido, etc), por tanto, no existe un método de compresión universal que pueda ser óptimo para todos los tipos de datos (Salomon, 2002).

El desempeño de los métodos de compresión se mide en base a dos criterios: la razón de compresión y el factor de compresión, siendo el segundo el inverso del primero. Las relaciones para determinar estas medidas están dadas por las igualdades de las ecuaciones 1 y 2. Entre mayor redundancia exista en los datos, mejor razón o tasa de compresión será obtenido (Mondelo & Iglesias, 2014).

$$Razon\ de\ compresión = \frac{N°\ Bytes\ archivo\ comprimido}{N°\ Bytes\ archivo\ original} \quad (1)$$

$$Factor\ de\ compresión = \frac{N°\ Bytes\ archivo\ original}{N°\ Bytes\ archivo\ comprimido} \quad (2)$$

La compresión es la reducción de la cantidad de espacio que se necesita para almacenar información, así como también la cantidad de tiempo que se requiere para realizar la transmisión de información a través de un canal. Por otro lado, el proceso de la detección y corrección de errores es lo opuesto a la compresión, puesto que trata de añadir redundancia a la información para que esta pueda ser revisada luego.

La compresión de datos permite que la información se transmita a una velocidad superior a la velocidad a la que lo haría si no existiese esa compresión. Normalmente, los datos y, en particular, el texto y los gráficos, contienen secuencias repetidas de información idéntica. La compresión de datos funciona al sustituir muchos caracteres de información repetida por unos pocos caracteres y transmitir sólo una copia de las secuencias de datos repetidas (Abramson, 1963).

1.5 Métodos de compresión sin pérdida de información.

También conocidos como *lossless* en inglés se caracterizan porque la información recuperada será idéntica a la original. Además, la tasa de compresión que proporcionan está limitada por la cantidad de información de la señal original. Entre estas técnicas destacan las que emplean métodos estadísticos, basadas en la Teoría de Shannon, que permiten la compresión sin pérdidas. Por ejemplo: codificación de Huffman, Lempel-Ziv, entre otras (Dudek, Borys, & Grzywna, 2007).

1.6 Algoritmos de compresión.

Algunos métodos de compresión así como RLE y Huffman estático se basan en el análisis inicial de las secuencias de datos para tomar las secuencias de datos repetidas y armar un diccionario de equivalencias, asignando códigos más breves, en una segunda etapa se transforma los datos iniciales con las equivalencias, en el algoritmo LZW el diccionario y las cadenas son transformadas en el mismo momento, además el diccionario no es almacenado pues puede ser generado a partir de los datos codificados, obteniendo un archivo de menor tamaño (Blelloch, 2013).

1.7 Lempel Ziv Welch – LZW.

El algoritmo LZW fue llamado así por los tres científicos que colaboraron en su desarrollo Abraham Lempel, Jakob Ziv y Terry Welch y es una variación de LZ78. Es un algoritmo sin pérdida orientado a diccionario. Como se mencionó previamente, los algoritmos que se basan en diccionarios escanean la información en búsqueda de secuencias de datos que ocurran más de una vez (Islam & Arafat, 2014).

La principal ventaja que posee es que puede ser adaptivo. Eso significa que el algoritmo no asume nada sobre las propiedades de la entrada antes de recibirla, por ende, construye el diccionario para la compresión a partir de la entrada de datos como tal. Esta propiedad es muy importante en la compresión para sistemas de comunicación y transmisión de datos. Este método es todo lo contrario a algoritmos de compresión que usan conocimiento predictivo de las propiedades de la entrada como lo hace el algoritmo de Huffman (Yang, Guo, Yong, Guo, & Wang, 2015).

LZW tiene la alta velocidad, buena tasa de compresión y baja ocupación de recursos entre sus ventajas. Adicionalmente, LWZ comprime el contenido carácter por carácter para así combinarlos y formar una cadena. A cada una de estas cadenas que se forman se les asigna un código especial y se las añade al diccionario. Por lo tanto, de ahí en adelante cuando una cadena se repite, únicamente se la refiere con el código previamente añadido al diccionario. Quizás como una desventaja del algoritmo es que requiere una construcción secuencial del diccionario por lo tanto es necesario una comparación constante de los datos de entrada con el contenido del diccionario durante la compresión (Yang, Guo, Yong, Guo, & Wang, 2015).

El proceso de compresión se lo puede resumir en tres etapas que se las revisa a continuación. 1) El diccionario se inicializa, aquí todos los caracteres simples y la tabla de cadenas de codificación se añaden al diccionario; 2) Se lee cada cadena y se la compara con todas las entradas del diccionario; 3) El código de cada cadena que se encontró en el diccionario es escrito en el flujo de datos codificado, en el caso de que no se encuentre ninguna cadena cuyo resultado de la comparación sea exitoso se la agrega al diccionario y se le asigna el código necesario (Islam & Arafat, 2014).

1.8 Algoritmo RLE.

Run Length Encoding tanto Al-Hashemi, Al-Dmour, Fraij & Musa (2011) como Liew, Zain & Liew (2010) lo definen como un algoritmo de compresión de datos sin pérdidas donde los datos originales pueden ser reconstruidos exactamente después de realizada la compresión. El principal objetivo de este algoritmo es la reducción de secuencias de dígitos repetidos. (Slaby, Hercik, & Machacek, 2013) Adicionalmente, como indica Abdmouleh, Masmoudi & Bouhlel (2012) su principio básico es eliminar la redundancia de la información.

Es una de las técnicas más simples de compresión de datos. (Seleznejv & Shykula, 2012) Funciona reemplazando secuencias de valores iguales de datos por dos espacios donde uno representa el conteo del carácter repetido y el otro es el carácter. (Bandyopadhyay, Paul, & Ratchoudhury, 2011) En un ejemplo propuesto por Lie, Zain & Liew (2010) se tiene la cadena de datos binarios 11111100000111111 se lo codificaría como (6,1), (5, 0) y (6, 1), al hacer una conversión del resultado que está en decimal a binario nuevamente este quedaría de la siguiente forma (110, 1),

(101, 0) y (110, 1) obteniédose así un total de 12 bits en comparación con los 17 bits que representaban originalmente a la cadena.

Martinez (2011) divide los datos comprimidos por RLE en dos categorías; 1) Símbolos repetidos: Consiste en los símbolos que pueden comprimirse reemplazándolos de la forma indicada en el párrafo anterior; y 2) Símbolos no repetidos: consiste en los símbolos que no pueden comprimirse porque no están repetidos. Adicionalmente, haciendo referencia a esto Martínez (2011) añade que RLE puede usarse para comprimir cualquier tipo de archivos. Por otro lado, Al-laham & El Emary (2007) señalan que RLE no puede ser muy usado para archivos que no contengan secuencias de valores iguales como pueden llegar a ser los archivos de texto.

Tomando como ejemplo el caso de las imágenes, RLE tiene buenos resultados en la compresión de imágenes monocromáticas que normalmente consiste en ciertos pixeles negros inmersos en un mar de pixeles blancos o viceversa (Al-laham & El Emary, 2007). Continuando con el ejemplo anterior, en casos en los que se tenga mayor variedad en la gama de pixeles como una fotografía, como indica Alamin, Ibrahim & Mustafa (2015), la eficiencia del algoritmo puede decrecer tanto hasta el punto de hacer que el tamaño del archivo aumente en lugar de reducirse. El punto máximo al cual RLE puede expandir un archivo es hasta el doble de su tamaño siendo este el caso en el que cada byte sea diferente del anterior y no se puedan obtener corridas de datos similares. Así como con las imágenes, este comportamiento se repite para cualquier tipo de archivo donde no se encuentren muchas secuencias de caracteres iguales.

1.9 Codificación Huffman.

Fue desarrollada por David A. Huffman en 1952 y corresponde a un código de longitud variable, en el que la longitud de cada código depende de la frecuencia relativa de aparición de cada símbolo en un texto: cuanto más frecuente sea un símbolo, su código asociado será más corto. Además, un código Huffman es un código libre de prefijos; es decir, ningún código forma la primera parte de otro código, esto permite que los mensajes codificados sean no ambiguos (Correa, 2008).

La codificación Huffman es un método muy valorado para la compresión de datos. Sirve como base para varios programas populares que se ejecutan en diversas plataformas. Algunos de ellos, utilizan sólo el método de Huffman, mientras que, en otros forma parte de un proceso de compresión de varios pasos. El método de Huffman es en cierto modo similar al método de Shannon– Fano. En general, produce mejores códigos; al igual que el método de Shannon–Fano, obtiene el mejor código cuando las probabilidades de los símbolos son potencias negativas de dos. La principal diferencia entre los dos métodos es, que Shannon–Fano construye sus códigos de arriba abajo, es decir desde los bits ubicados más a la izquierda hacia los que se encuentran más a la derecha.

Mientras que, Huffman lo hace mediante un árbol de código desde lo más profundo del mismo, hasta arriba, es decir de derecha a izquierda (Sharma, 2010).

1.10 Huffman Adaptativo.

Como fue descrito previamente, el modelo estático de Huffman requiere dos pasadas donde la primera se enfoca básicamente en la recolección de las probabilidades estadísticas del archivo. Mientras que, en la segunda pasada se realiza la compresión de la información de la fuente. Si se desea codificar un símbolo (k+1) usando las estadísticas de los primeros k símbolos, se tendría que recalcular el código usando todo el proceso de Huffman cada vez que un símbolo se transmite. Sin embargo, esto conllevaría un uso muy elevado de procesamiento lo que haría que el proceso no se vuelva práctico a la hora de pensar en costos (Jagadeesh & Ankitha, 2013).

Los sistemas de compresión basados en modelado adaptativo o dinámico procesan el mensaje de entrada en una única pasada en la que el modelo se construye de forma progresiva y el archivo se codifica de acuerdo al estado de dicho modelo en cada paso del proceso. Esto supone la actualización incremental del conocimiento relativo a la distribución de la información del archivo, facilitando la adaptación del modelo a las propiedades particulares de dicho archivo, cumple las propiedades del árbol de Huffman, el código se modifica a medida que se procesan los símbolos basándose en los previamente procesados (Herrera, 2003).

Adicionalmente, Jagadeesh & Ankitha (2013) indican que también existen dos diferencias principales entre los modelos estático y dinámico de Huffman. Primero, que el estático no se recomienda su uso para transferencias en tiempo real, cosa que si se puede hacer con el adaptativo. En segundo lugar, al momento de la transmisión, en el modelo dinámico no se necesita que se envíe información adicional con las probabilidades de cada uno de los elementos, por el contrario, los deduce en la descompresión, así como lo hace en la compresión. Esta última característica hace que además se ahorre cierto ancho de banda en la transmisión.

Para poner en funcionamiento al modelo adaptativo se añadieron dos parámetros a los nodos del árbol, en referencia al modelo estático, que son el peso de cada hoja que representa las veces que se repite el símbolo correspondiente a aquella hoja y un número para cada nodo el cual es único. Si se cuenta con un alfabeto de tamaño n, los 2n-1 nodos tanto internos como externos pueden numerarse como $y_1, \ldots, y_{2n-1}$. Para que el árbol puede mantener su condición de árbol Huffman tiene que cumplir con ciertas propiedades que son las siguientes: a) siendo que x_j es el peso de un nodo y_j, donde $x_1 \leq x_2 \leq \ldots \leq x_{2n-1}$; b) los nodos y_{2j-1} y y_{2j} son descendientes del mismo padre, es decir hermanos, entonces el número nodo del padre debe ser mayor que y_{2j-1} y y_{2j}. Adicionalmente, para que el árbol mantenga continuamente las propiedades mencionadas anteriormente, se debe realizar

un proceso de actualización constante mediante el cual se reacomodan los nodos del árbol en referencia al ingreso de nuevos datos.

2. MATERIALES Y MÉTODOS

Este trabajo propone una investigación cuyo diseño es experimental donde existirán 41 archivos a comprimir, para de esta forma obtener la tasa de compresión y el ancho de banda ocupado antes y después de procesarlos con los algoritmos elegidos. Además, se cuenta con un enfoque cuantitativo y alcance descriptivo.

Adicionalmente, este trabajo se divide en tres etapas. La primera etapa implica el proceso de selección de los archivos y sus tipos. La segunda etapa conlleva el proceso de simulación de la compresión de los archivos. Finalmente, en la tercera etapa se realiza el proceso de modulación de los archivos que fueron comprimidos para así representarlos en el dominio de la frecuencia.

El ambiente de estudio en el cual se realizarán las pruebas es un ambiente simulado, se ejecuta un programa de desarrollo matemático denominado Matlab en su versión R2015b.

Los algoritmos de compresión, es decir, los objetos de estudio se eligieron por conveniencia y en base a sus características. Estos son RLE, LZW y Huffman Adaptativo. De la misma forma, debido a que son los tipos de archivos más comunes en transmisiones de información se optó por trabajar con imágenes, archivos de texto y de audio.

2.1 Ambiente de pruebas.

Con el objetivo de obtener los resultados experimentales que permitan analizar la compresión y su ahorro en ancho de banda, el estudio se realizó en un ambiente simulado donde los algoritmos RLE, LZW y Huffman adaptivo se programaron usando Matlab R2015b en una computadora MacBook Pro con sistema operativo macOS Sierra 10.12 con 4GB de memoria RAM y procesador Intel Core i7.

Se eligieron tres tipos de archivos para este trabajo los cuales son imágenes, texto y audio. La distribución de los archivos se hizo de la siguiente manera: 15 imágenes a color, cinco imágenes blanco y negro; seis archivos de texto y finalmente 15 de audio, sumando un total de 41 archivos. Los formatos utilizados fueron .jpg y .png para imágenes, .txt para texto y .wav para audio. Para el caso de las imágenes es indiferente la extensión que se use, esto no tiene incidencia en el desarrollo de las pruebas.

Los algoritmos se programaron en Matlab usando los fundamentos teóricos de los mismos. Para certificar el correcto funcionamiento de estos algoritmos simulados se realizaron pruebas con las

que se hizo la afinación de los programas mediante el uso de ejercicios previamente realizados y propuestos por diferentes autores. Una vez que la respuesta propuesta por los ejercicios elaborados coincidía con la respuesta obtenida en las simulaciones desarrolladas, se establecía que cumplían con todos los parámetros para considerarse una simulación correcta de los algoritmos de compresión elegidos y por ende estaban listos para usarse en las pruebas del estudio.

En lo que respecta a la cantidad de pruebas por archivo, no se siguió ningún modelo estadístico para determinar un intervalo de confianza de cantidad de pruebas de compresión ni transmisión a un archivo usando las mismas condiciones de compresión puesto que estas pruebas cuando se repetían se obtenía siempre el mismo resultado en el mismo marco de prueba. Esto se debe a que estas pruebas se realizan en un ambiente totalmente simulado y sin tomar en cuenta interferencias ni variaciones no controladas.

2.2 Fuentes de datos.

Se usaron imágenes tanto en escala de grises como a color las cuales eran cuadradas de un tamaño de 100x100 pixeles en lo vertical y horizontal respectivamente. Para su procesamiento en Matlab, las imágenes se leyeron y guardaron en archivos pixel por pixel donde cada uno de ellos era representado por un valor de 0 a 255, es decir 8 bits por pixel equivalente a un byte. Ciertas imágenes blanco y negro se representaban con un arreglo bidimensional de 100x100 objetos. Por otro lado, el resto de las imágenes blanco y negro y la totalidad de las imágenes a color se representan con un arreglo tridimensional formado por tres arreglos bidimensionales de 100x100 objetos, así como se puede apreciar en la figura 8. El total de bytes que se puede obtener en una de las imágenes blanco y negro representadas arreglos bidimensionales es de 10 KB, mientras que, para los arreglos tridimensionales se puede tener un tamaño máximo de 30 KB.

Los archivos de texto estaban generados de forma aleatoria, estando formados palabras del idioma español, pero sin un orden lógico, los cuales se procesaron carácter por carácter. Puesto que estos caracteres eran letras o símbolos se los tuvo que transformar a una variable de tipo uint8 que se pueda utilizar la su compresión. Para realizar lo mencionado, se convirtió cada una de los caracteres a su equivalente en ASCII con lo que se logró representar cada uno de los caracteres por un byte. Al igual que como se hizo con las imágenes, se colocó la cadena en un arreglo horizontal dependiendo de la longitud del texto.

Finalmente, para el audio se usaron archivos de un solo canal y con la ayuda de Matlab se realizaron muestras del audio que se representan en valores de 8 bits de 0 a 255. Estos archivos fueron elegidos a conveniencia, intentando obtener archivos de diferentes modalidades y tamaños. De manera similar a como se hizo tanto en imágenes como en texto, se coloca todos los valores de los bytes en un arreglo horizontal.

En cada uno de los archivos se calculó la cantidad de objetos diferentes encontrados, a este valor se lo denomina tamaño del diccionario. Para las imágenes RGB se puede obtener un máximo de 10 mil valores de datos diferentes, mientras que, para imágenes en blanco y negro, archivos de texto y audio la cantidad máxima de bytes diferentes es de 256. El tamaño del diccionario es importante porque brinda la pauta para conocer de cierto modo la densidad de caracteres diferentes por archivo. Adicionalmente, también se obtuvo el tamaño de cada uno de los objetos a comprimir para poder tener una medida de comparación a futuro.

2.3 Formato de entrada.

El formato de entrada para procesar los archivos fue propuesto como un arreglo unidimensional de una fila y múltiples columnas de formato uint8 que representa enteros de 8 bits. Para esto se concatenó todas las filas de los arreglos bidimensionales y tridimensionales para que las imágenes se amolden al formato de entrada de los algoritmos. Los .txt se obtienen por defecto como un arreglo unidimensional, por lo tanto, podía procesarse directamente una vez convertido a uint8. Por último, los archivos de audio se representaban como un arreglo unidimensional de una columna por varias filas, consecuentemente, sólo se invirtió la disposición del arreglo para que pueda procesarse.

Debido a las condiciones de simulación, se decidió establecer un tamaño máximo de archivos de 60 KB. Esto se debe a que para archivos de mayor tamaño el procesamiento de la simulación toma mucho tiempo y por lo tanto se ralentizaba todo el proceso. Este límite no influye en el estudio por cuanto el tamaño no es un factor relevante para este trabajo.

2.4 Formato de salida.

Los formatos de salida una vez realizada la compresión se obtuvieron de formas diferentes para cada uno de los tres algoritmos de compresión. Para RLE se obtuvo un arreglo unidimensional de un tipo de dato uint8. Por otro lado, debido a que LZW trabaja con un diccionario, los valores del índice del diccionario eran superiores a 256. Por esta razón se optó por usar variables de 16 bits de tipo uint16 que eran capaces de almacenar esos valores, obteniéndose así una representación de 2 bytes por cada uno de estos caracteres. Finalmente, la salida de Huffman Adaptivo se obtenía en una variable de tipo char que representaba a la cadena de bits resultante de la compresión.

2.5 Modulación de datos.

La modulación se realizó con un esquema 4-QAM, donde cada símbolo representa dos bits. No se tomaron en cuenta valores de redundancia, roll off y distancia entre símbolos. Estos parámetros que se omitieron no tienen repercusión en el estudio debido que se aplicaría el mismo valor para todos los casos y, por ende, se convertirían en constantes que son independientes de la compresión aplicada. De la misma forma, la modulación que se elija no influye en los resultados porque esta se

aplica de forma uniforme para todos los archivos, por consiguiente, si se cambia la modulación se lo hará para todos los casos y no de forma individual. Por lo tanto, esta modulación se la eligió por conveniencia. Adicionalmente, debido a que los parámetros tomados en cuenta son únicamente los más relevantes para el trabajo, el ancho de banda se lo obtiene mediante la relación más básica que es un Hertz de ancho de banda es igual a un símbolo por segundo de velocidad de transferencia.

2.6 Análisis de datos.

La tasa de compresión fue calculada a manera de porcentaje relacionando el tamaño del archivo antes y después de la compresión, esto se aprecia de mejor manera en la ecuación 3. El valor obtenido de aquel cálculo se lo divide en tres categorías que son: 1) Si es igual a 100, significa que el tamaño del archivo se mantuvo a pesar de la compresión; 2) Si es mayor a 100 se relaciona con que la compresión incrementó el tamaño del archivo; finalmente 3) Si la tasa es menor a 100 se lo toma en cuenta como una reducción en el tamaño del archivo en relación con el original. Este valor de tasa de compresión se lo utiliza para evaluar la eficiencia de los algoritmos al analizar los archivos.

$$TC = \frac{TD \times 100}{TA} \qquad (3)$$

Donde:

TC = Tasa de compresión.

TD = Tamaño después de la compresión.

TA = Tamaño antes de la compresión.

Se obtuvo un valor denominado tamaño de diccionario que mide la cantidad de caracteres que conforman el archivo evitando repeticiones. Para esto se realizó un programa que evaluaba todos los caracteres de los archivos para así poder obtener el resultado deseado. Además, se calculó un valor denominado densidad de diccionario que relaciona el tamaño del diccionario con la cantidad total de caracteres que tiene el archivo como lo muestra la ecuación 4.

$$\delta = \frac{CT}{D} \qquad (4)$$

Donde:

δ = Densidad de diccionario.

CT = Número total de caracteres del archivo.

D = Diccionario.

3. ANÁLISIS DE RESULTADOS

La tabla 1 muestra una representación general de la eficiencia de compresión para los algoritmos RLE, LZW y Huffman Adaptativo. Aquí se relaciona la tasa de compresión, dado por la ecuación 5 del capítulo anterior, de los algoritmos ante los diferentes tipos de archivos.

Tabla 1. *Tasa de compresión promedio.*

	Tasa de compresión (en porcentaje)			
Algoritmo	**Imágenes**	**Texto**	**Audio**	**Promedio General**
RLE	119,78	195,61	174,04	163,14
LZW	73,61	68,38	110,04	84,01
Huffman Adaptativo	74,66	55,54	80,54	70,25

Fuente: Elaboración propia

La tabla 1 muestra la tasa de compresión promedio de los algoritmos antes los distintos tipos de archivos. Adicionalmente, se encuentra una columna de promedio general que representa la media aritmética de la tasa de compresión combinando todos los tipos de compresión.

Tal y como se definió en el capítulo anterior, los valores mayores al 100% señalan un aumento en el tamaño del archivo, esto significa que mientras menor sea el valor de la tasa de compresión será mejor. RLE es el que peor desempeño mostró, todos sus valores sobrepasan el 100%; para imágenes se obtuvo un valor de 119,78%, en texto 195,61% y con audio un 174,04%. Además, su promedio general de compresión fue de 163,14%.

Los valores obtenidos para LZW señalas que tanto para imágenes como para texto se obtuvieron resultados menores al 100%; sin embargo, para el caso de audio se superó este porcentaje. Los valores están distribuidos de la siguiente forma; para imágenes fue de 73,61%, en texto 68,38% y en audio 110,04% con un promedio general de 84,01%.

Finalmente, con Huffman Adaptativo de forma general se aprecia que tanto los valores para imágenes, texto y audio fueron menores al 100%. Para las imágenes se obtuvo un 74,66%, la compresión del texto reflejó un 55,54% y el audio se representó con un 80,54%. Por último, el promedio general de este algoritmo fue del 70,25%.

Una comparación de los tres algoritmos agrupados por tipo de archivo se puede apreciar en la figura 1. Donde RLE está de color azul, LZW se representa por anaranjado y Huffman Adaptativo se simboliza por un gris. En esta figura se puede destacar que con los archivos de imágenes tanto LZW como Huffman Adaptativo muestran un rendimiento similar; mientras que, RLE se eleva hacia más del 100%. Algo similar ocurre para el texto donde Huffman Adaptativo denota por poco una mejor eficiencia que LZW; por otro lado, RLE se aleja notablemente de los otros dos algoritmos mencionados con un valor cercano al 200%. Finalmente, los archivos de audio forzaron que la tasa de compresión se eleve tanto en LZW, donde ascendió a más de 100%, como en Huffman Adaptativo que de todas formas se mantuvo en un valor menor al 100%. RLE al igual que en los otros tipos de archivos se encontró por sobre el 100%.

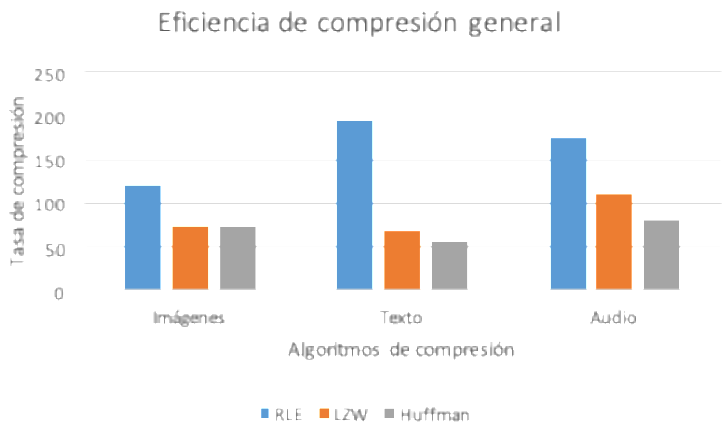

Figura 1. Comparación general de la eficiencia de compresión.
Fuente: Elaboración propia.

Las figuras 2, 3 y 4 muestran gráficas comparativas de los algoritmos RLE, LZW y Huffman Adaptativo para la compresión de imágenes, texto y audio respectivamente. Estas gráficas relacionan la densidad del diccionario, que fue explicada en el capítulo anterior, con la tasa de compresión. En el eje horizontal se encuentran los valores de densidad de diccionario de todas las imágenes, archivos de texto y audio ordenados de menor a mayor. El eje vertical representa los valores en índice porcentual de la tasa de compresión. Asimismo, la línea horizontal de color verde, que se encuentra en 100%, representa la división entre compresión y aumento de información.

En la figura 2, la línea de color anaranjado simboliza la compresión en RLE donde se observa que la mayoría de sus pruebas se encontraron por encima del límite que indica compresión. Además,

se percibe una tendencia a aumentar su valor de tasa de compresión conforme aumenta la densidad de diccionario, a excepción de ciertos valores que aparentemente responden a algún otro factor no especificado.

En la misma figura, la compresión LZW está representada por la línea de color azul. A diferencia de RLE, la mayoría de los resultados están por debajo del 100%. Además, de la misma forma que en el caso anterior, se muestra una tendencia a empeorar la compresión cuando la densidad de diccionario se incrementa.

Finalmente, la línea gris identifica a la compresión Huffman Adaptativo, que, en relación con los otros algoritmos de la gráfica es el de mejor desempeño. En este caso, a diferencia de los anteriores, ninguno de los valores pasa el 100% significando esto que comprime la información en todos los casos. Por el contrario de RLE y LZW no muestra una tendencia tan marcada a desmejorar su nivel de compresión conforme aumenta la densidad de diccionario.

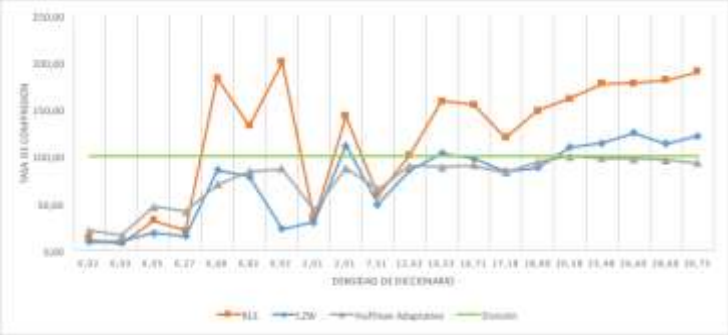

Figura 2. Comparación de RLE, LZW y Huffman Adaptativo en imágenes.
Fuente: Elaboración propia.

En la figura 3, referente a la compresión de texto, los resultados de RLE se enmarcan bajo el color anaranjado mostrando en todos los casos tener valores cercanos al 200%. Esto quiere decir que casi se duplica el tamaño del archivo en cada prueba realizada. Por otra parte, no se nota relación alguna con la densidad de diccionario.

La línea azul de la misma figura representa la compresión por LZW donde todos los puntos estás ubicados debajo de la línea verde de división. Por consiguiente, existe compresión en todos los casos. Adicionalmente, se aprecia una ligera tendencia empeorar la tasa de compresión conforme se incrementa la densidad de diccionario.

Por último, para complementar la gráfica, la línea gris simboliza la técnica Huffman Adaptativo. Para este caso se aprecia el mejor rendimiento de compresión; además, siendo muy constante a pesar de las variaciones de densidad de diccionario.

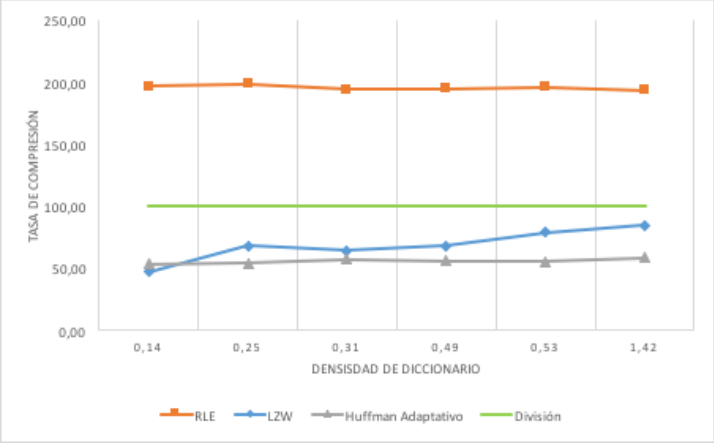

Figura 3. Comparación de RLE, LZW y Huffman Adaptativo en texto.
Fuente: Elaboración propia.

En el caso de la figura 4, referente a la compresión de archivos de audio. Existe mucha variabilidad para RLE representado por la línea anaranjada. Todos los valores de las pruebas de audio para RLE resultaron ser mayores al 100% por lo que no existió compresión. Además, no se precia ninguna relación ni tendencia en relación a la densidad de diccionario.

El comportamiento de LZW se visualiza en la línea azul. En este caso, la mayoría de los puntos se encuentran por debajo de la división del 100%. No se aprecia ninguna relación directa con la densidad de diccionario, pese a esto se observa un incremento considerable en los valores más altos de la densidad.

La línea gris representa la compresión por Huffman Adaptativo. Para esta técnica, todos sus valores se encuentran por debajo del 100%, es decir, existe compresión para todos los casos. De igual manera que en los casos de RLE y LZW, no se aprecia relación con la variación de la densidad de diccionario.

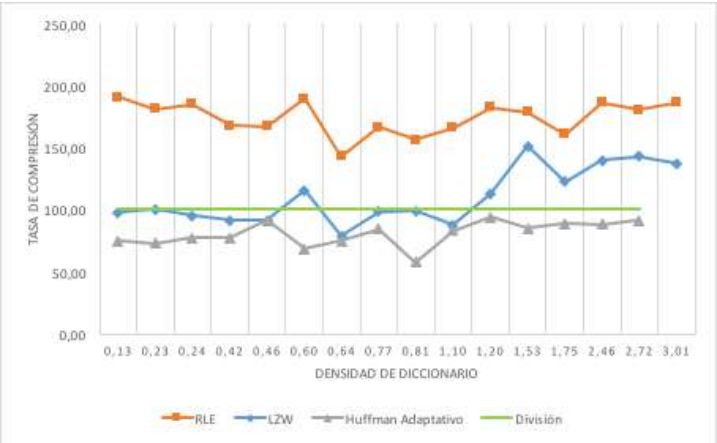

Figura 4. Comparación de RLE, LZW y Huffman Adaptativo en audio.
Fuente: Elaboración propia.

3.1 Ancho de banda.

En lo que respecta al uso de ancho de banda, las figuras 5, 6 y 7 lo representan mediante gráficos de comparación entre el ancho de banda utilizado antes y después de la compresión. Cada una de las figuras representa a los tres algoritmos, siendo estos RLE, LZW y Huffman Adaptativo respectivamente. Para este caso se consideran en conjunto las pruebas realizadas para imágenes, texto y audio llegando así a un total de 41 pruebas. En el eje horizontal se encuentra el número de prueba, el mismo que se relaciona con el ancho de banda que está ubicado en el eje vertical.

En todos los casos, la línea de color azul simboliza al ancho de banda antes de la compresión. Por otro lado, la línea anaranjada representa la utilización del ancho de banda después de la compresión.

La figura 5 muestra la diferencia que existe entre el ancho de banda antes de la compresión y después. El gráfico muestra que, en la mayoría de los casos, la medición realizada después de la compresión ocupa más ancho de banda que el archivo original. Solamente en el 14,63% de las pruebas las mediciones hechas post compresión resultaron ocupar menor ancho de banda del original. En algunos casos se muestran diferencias muy grandes entre los dos campos.

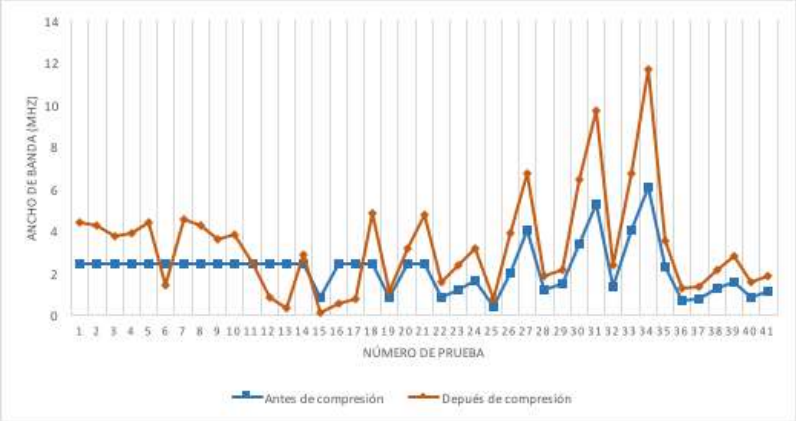

Figura 5. Comparación de uso de ancho de banda antes y después de la compresión RLE.
Fuente: Elaboración propia.

El caso de LZW se notan ciertas diferencias en comparación al referido en la figura anterior. En la figura 6 se aprecia mayor igualdad entre los resultados. A pesar de esto, únicamente el 65,85% de las mediciones realizadas muestran un mejor uso del ancho de banda.

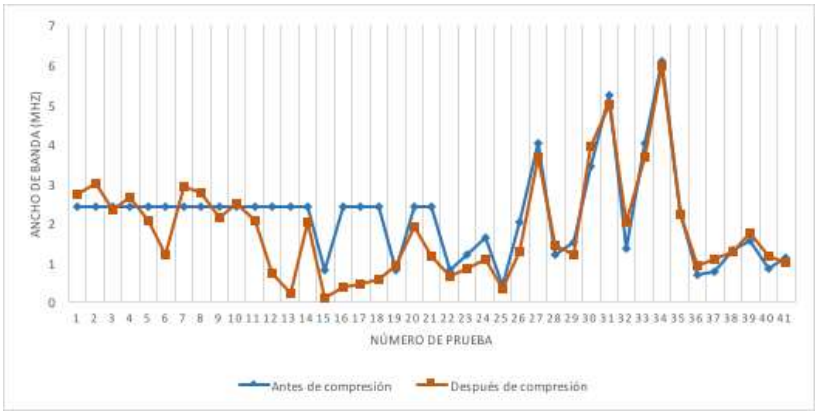

Figura 6. Comparación de uso de ancho de banda antes y después de la compresión LZW.
Fuente: Elaboración propia

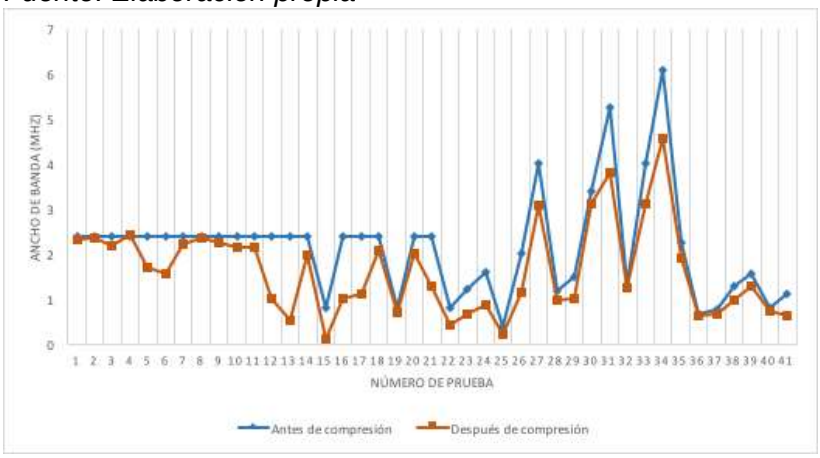

Figura 7. Comparación de uso de ancho de banda antes y después de la compresión en Huffman Adaptativo.
Fuente: Elaboración propia.

Finalmente, para Huffman Adaptativo, esquematizado en la figura 7 se obtuvieron resultados positivos en el 97,56% de los casos. Siendo así, que solo una de las mediciones mostró un pequeño incremento en el ancho de banda utilizado. Aquel incremento es tan reducido que casi no se lo puede apreciar en la figura 7. En relación al resto de los casos, se observan reducciones importantes de ancho de banda para los archivos comprimidos.

4. DISCUSIÓN Y CONCLUSIONES

La compresión de datos, según la literatura, ha demostrado ser fundamental para optimizar y aprovechar recursos dentro de una transmisión, tales como ancho de banda y potencia. Siendo así que la elección un algoritmo en cuanto a características tales como la eficiencia, es un aspecto fundamental. Este parámetro fue tomado en cuenta para la realización de un estudio comparativo entre RLE, LZW y Huffman Adaptativo en condiciones de simulación.

Para llevar a cabo esta tarea, se realizó un análisis teórico de los algoritmos que se utilizaron en este trabajo. Una característica compartida por todos es que son algoritmos sin pérdida, es decir que, al recobrarse la información, esta debe ser exactamente igual a la original. Por otro lado, los algoritmos difieren en fundamento de operación, siendo de esta manera que, RLE es el más sencillo de los tres al estar basado en secuencias de caracteres repetidos. Posteriormente, la revisión teórica de LZW mostró que este tiene un grado más alto de complejidad que RLE dado que trabaja con base en un diccionario, es decir, bloques de caracteres. Finalmente, Huffman Adaptativo está relacionado con árboles de probabilidad que cambian constantemente, lo que hace que este algoritmo sea el de mayor complejidad teórica y a su vez adaptabilidad ante varios ambientes puesto que trabaja asociando cada carácter de manera individual con su probabilidad de aparición.

La tasa de compresión se utilizó como valor cuantificador de los resultados obtenidos en las simulaciones, demostrando que Huffman Adaptativo es el algoritmo que presentó mejores resultados en este aspecto con un valor promedio de 70,25% al reducir el tamaño del 97,56% de los archivos. RLE registró el peor desempeño con 163,14% de tasa de compresión, puesto que en el 85,37% de los casos presentó un aumento en el tamaño de los archivos. Por otro lado, LZW se ubica en una posición intermedia en relación a los algoritmos mencionados anteriormente con 84,01% de promedio en tasa de compresión al disminuir el tamaño en el 65,85% de las pruebas.

Por otra parte, en la sección de resultados y haciendo referencia a las figuras 5, 6 y 7, se demuestra de forma clara la variación de uso de ancho de banda entre las mediciones pre y post compresión. De esta forma se llevó a cabo la relación mencionada entre la compresión y el consumo de ancho de banda. Adicionalmente, se evidencia que en dos de los tres algoritmos de compresión –LZW y Huffman Adaptativo– existe un ahorro de ancho de banda promedio del 13,59% y 25,99% respectivamente. Este no fue el caso de RLE en el cual hubo aumento del ancho de banda promedio de 50,56%.

El algoritmo que demostró ser más eficiente ante todas las circunstancias, además de ser el más estable en sus pruebas fue Huffman Adaptativo. No obstante, LZW fue ligeramente mejor para la compresión de imágenes, pero tuvo variaciones muy grandes al momento de procesar archivos de texto y audio. Por último, RLE cumple su función de compresión solamente ante condiciones ideales, es decir, con archivos que contengan abundantes secuencias de caracteres repetidos. Dado que se usaron archivos aleatorios, RLE obtuvo los resultados más altos en todos los casos de compresión llegando a ser el menos eficiente de los tres.

La principal limitación de este trabajo recae en que las pruebas se realizaron únicamente simulando condiciones ideales de transmisión. Por lo tanto, los resultados que se obtuvieron no toman en cuenta factores externos que puedan alterar el desarrollo de las pruebas tales como interferencia.

Las contribuciones principales de este trabajo son: 1) Proveer un marco referencial en torno a la comparación de los tres algoritmos de compresión elegidos; 2) Diferenciación entre el uso de ancho de banda de archivos comprimidos y no comprimidos con tres algoritmos y tipos de archivos diferentes.

Haciendo referencia a las conclusiones y limitaciones encontradas se realizan las siguientes recomendaciones en miras a la continuación del desarrollo de este tema. 1) Realizar pruebas en ambiente no simulado; 2) Evaluar otros factores como tiempo y capacidad de procesamiento necesitados por los algoritmos de compresión; 3) Realizar un estudio de cómo afecta el cifrado de datos a la compresión de los mismos; 4) Hacer pruebas con más algoritmos.

REFERENCIAS BIBLIOGRÁFICAS

Abdmouleh, K., Masmoudi, A., & Bouhlel, S. (2012). A New Method Which Combines Arithmetic Coding with RLE for Lossless Image Compression. *Journal of Software Engineering and Applications, 5*, 41-44.

Abramson, N. (1963). Information Theory and Coding. *Information Theory and Coding*. McGraw-Hill.

Al-Hashemi, R., Al-Dmour, A., Fraij, F., & Musa, A. (2011). A Grayscale Semi-Lossless Image Compression Technique Using RLE. *Journal of Applied Computer Science & Mathematics*(10), 9-13.

Al-laham, M., & El Emary, I. (2007). Comparative Study Between Various Algorithms of Data Compression Techniques. *World Congress on Engineering and Computer Science*.

Alamin Ibrahim, A. M., & Mustafa, M. E. (2015). Comparison Between (RLE and Huffman) Algorithms for Lossless Data Compression. *International Journal of Innovative Technology and Research, 3*(1), 1808-1812.

Bandyopadhyay, S. K., Paul, T. U., & Ratchoudhury, A. (2011). Image Compression using Approximate Matching and Run Length . *International Journal of Advanced Computer Science and Applications, 2*(6), 117-121.

Blelloch, G. E. (31 de Enero de 2013). Introduction to Data Compression. Computer Science Department.

Correa, M. (2008). Fundamentos de la Teoría de la Información. En M. Correa, *Fundamentos de la Teoría de la Información.* ITM. Recuperado el 10 de Agosto de 2016

Dudek, G., Borys, P., & Grzywna, Z. (2007). *Lossy dictionary based image compression method.* Image and Vision Computing.

Fariña, A. (2005). New Compression Codes for Text Databases. *New Compression Codes for Text Databases*. España: Dpto de computación, Universidade da Coruña.

Gil, P. G., Pomares, J., & Candelas, F. (2010). Redes y transmisión de datos. En P. G. Gil, J. Pomares, & F. Candelas, *Redes y transmisión de datos.* Universidad de Alicante.

Herrera. (2003). *Tecnologías y redes de transmisión de datos.* Editoial Limusa.

Hussen, A. H., Mahmud, S. S., & Mohammed, R. J. (2011). Image Compression Using Proposed Enhanced Run Length Encoding Algorithm. *Al-Haitham J. For Pure & Applied Science, 24*(1).

Islam, K., & Arafat, Y. (2014). Redundant Reduced LZW (RRLZW) Technique of Lossless Data Compression. *International Journal of Advanced Research in Computer Science, V*(6), 261-266.

Jagadeesh, B., & Ankitha, R. (Diciembre de 2013). An approach for Image Compression using Adaptive Huffman Coding. *International Journal of Engineering & Technology, II*(12), 3216-3224.

Liew, S.-C., Zain, J., & Liew, S.-W. (Diciembre de 2010). Reversible Medical Image Watermarking for Tamper Detection and Recovery With Run Length Encoding Compression. *World Academics of Science, Engineering and Technology, 4*, 12-29.

Martínez, M. A. (2010). *Estudio y Aplicación de Nuevos Métodos de Compresión de Texto Orientada a Palabras.* Chile: Universidad de Valladolid.

Mondelo, A., & Iglesias, I. (2014). *Sistemas de archivo y clasificación de documentos.* Ideas propias Editorial S.L.

Neapolitan, R., & Naimipour, K. (2011). *Foundations of Algorithms* (4ª edición ed.). Estados Unidos: Jones and Bartlett Publishers.

Pere-Pau, V., Jordi, G., Angela, M., & Xavier, M. (2006). Programación para Ingenieros. En V. Pere-Pau, G. Jordi, M. Angela, & M. Xavier, *Programación para Ingenieros* (pág. 41). Editorial Paraninfo.

Salomon, D. (2002). *A Guide to Data Compresión Methods.* New York: Springer-Verlag.

Sangvikar, J., & Joshi, G. (2011). Compression of Noisy and Noiseless Images through Run Length Encoding Approach. *International Journal of Advances in Embedded Systems, I*(1), 1-3.

Seleznejv, O., & Shykula, M. (2012). Run-length compression of quantized Gaussian stationary signals. *Random Oper. Stoch*(20), 311-328.

Sharma, M. (2010). Compression Using Huffman Coding. *IJCSNS International Journal of Computer Science and Network Security*.

Sklar, B. (2001). *Digital Communications: Fundamentals and Applications* (2ª edición ed.). Los Ángeles: Pearson.

Slaby, R., Hercik, R., & Machacek, Z. (2013). Compression methods for image processing implementation into the low capacity devices.

Yang, L., Guo, Z., Yong, S., Guo, F., & Wang, X. (2015). A Hardware Implementation of Real Time Lossless Data Compression and Decompression Circuits. *Applied Mechanics and Meterials*, 554-560.