UNIVERSIDAD DE ESPECIALIDADES ESPÍRITU SANTO

Facultad de Sistemas, Telecomunicaciones y Electrónica

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

Trabajo de Titulación que se presenta como requisito para los títulos de:

Ingeniero en Telecomunicaciones

Ingeniero en Sistemas

Autores:

Gabriel José Ortega Jouvin

Karla María Samaniego Villarroel

Tutor: Ing. Luis Enrique Benavides Castillo, MBA

Samborondón, marzo 2017

APROBACIÓN DEL TUTOR

En mi calidad de Tutor de los estudiantes Gabriel José Ortega Jouvin, que cursa estudios en el programa de tercer nivel Ingeniería en Telecomunicaciones; y Karla María Samaniego Villarroel, que cursa estudios en el programa de tercer nivel Ingeniería en Sistemas con concentración en Telecomunicaciones; dictados en la Facultad de Sistemas, Telecomunicaciones y Electrónica de la UEES, en modalidad presencial.

CERTIFICO

Que he revisado el Trabajo de Titulación denominado: “Optimización del uso del ancho de banda en los enlaces de transmisión de datos por medio de algoritmos de compresión”, presentado por los estudiantes Gabriel José Ortega Jouvin y Karla María Samaniego Villarroel, como requisito previo para optar por el Grado Académico de Ingeniero en Telecomunicaciones e Ingeniero en Sistemas. CERTIFICO que el Trabajo de Titulación ha sido analizado y reúne todos requisitos para ser presentado y sometido a los procesos de revisión estipulados por la Facultad.

Atte.

Ing. Luis Enrique Benavides Castillo
CI: 0908815806

RECONOCIMIENTOS

Nos gustaría agradecer principalmente a nuestro tutor, Ing. Luis Benavides. Por haber sido una guía en todo el tiempo de desarrollo de este trabajo. Gracias por el apoyo, tiempo y dedicación. Sin sus conocimientos y ayuda hubiera sido muy difícil desarrollar el tema.

También quisiéramos agradecer a nuestras familias y amigos, sin su apoyo incondicional no podríamos haber logrado esto.

Índice General

1 INTRODUCCIÓN 1
1.1 Antecedentes 5
1.2 Descripción del problema 6
1.3 Alcances y limitaciones 7
1.4 Justificación 8
1.5 Objetivos 9
1.5.1 Objetivo general: 9
1.5.2 Objetivos específicos: 9
2 MARCO TEÓRICO 10
2.1 Algoritmo 10
2.2 Modelo de comunicación 11
2.3 Tipos de señales y parámetros de una señal 11
2.4 Capacidad del canal 12
2.5 Ancho de banda 13
2.6 Transmisión de datos 14
2.7 Información y entropía 15
2.8 Redundancia 17
2.9 Compresión de datos 18
2.10 Clasificación por integridad de información 20
2.10.1 Métodos de compresión sin pérdida de información 21
2.10.2 Métodos de compresión con pérdida de información 21
2.11 Clasificación por técnica de compresión 21
2.11.1 Algoritmos de compresión 22
2.11.2 Lempel Ziv Welch – LZW 23
2.11.3 Algoritmo RLE 25
2.11.4 Codificación Huffman 27
2.11.5 Modelo Dinámico o Adaptivo 34
3 METODOLOGÍA 38
3.1 Descripción de la metodología 38
3.2 Justificación de la metodología 39

4 DESARROLLO DE PROPUESTA Y RESULTADOS 41
4.1 Ambiente de pruebas 41
4.2 Fuentes de datos 46
4.3 Formato de entrada 49
4.4 Formato de salida de la compresión 50
4.5 Modulación de datos 51
4.6 Análisis de datos 52
4.7 Resultados generales 55
4.8 Resultados específicos 57
4.9 Ancho de banda 61
5 CONCLUSIONES Y RECOMENDACIONES 66
5.1 Conclusiones 66
5.2 Recomendaciones 69
6 REFERENCIAS 70

Índice de Tablas

Tabla 1. Clasificación de tipos de señales. 12

Tabla 2. Ejemplo aplicado del algoritmo LZW. 25

Tabla 3. Tasa de compresión promedio. 55

Índice de Figuras

Figura 1. Ejemplo de Huffman estático 31

Figura 2. Ejemplo de actualización de árbol en Huffman Adaptativo 36

Figura 3. Diagrama de flujo de algoritmo RLE..... 43

Figura 4. Diagrama de flujo de algoritmo LZW. 44

Figura 5. Diagrama de flujo de algoritmo Huffman adaptativo..... 45

Figura 6. Diagrama de flujo de proceso de actualización de árbol Huffman Adaptativo. 46

Figura 7. Captura de la entrada de imagen blanco y negro en Matlab. ... 47

Figura 8. Captura de la entrada de una imagen RGB en Matlab..... 47

Figura 9. Captura de la entrada de archivo de texto. 48

Figura 10. Captura de entrada de archivo de audio. 49

Figura 11. Formato de salida de RLE en Matlab. 51

Figura 12. Formato de salida de LZW en Matlab. 51

Figura 13. Formato de salida de Huffman adaptativo en Matlab. 51

Figura 14. Comparación general de la eficiencia de compresión 57

Figura 15. Comparación de RLE, LZW y Huffman Adaptativo en imágenes. 59

Figura 16. Comparación de RLE, LZW y Huffman Adaptativo en texto. .. 60

Figura 17. Comparación de RLE, LZW y Huffman Adaptativo en audio. . 62

Figura 18. Comparación de uso de ancho de banda antes y después de la compresión RLE..... 63

Figura 19. Comparación de uso de ancho de banda antes y después de la compresión LZW. 64

Figura 20. Comparación de uso de ancho de banda antes y después de la compresión en Huffman Adaptativo. 64

Resumen

La evolución de la tecnología ha hecho que el tamaño de los datos se incremente considerablemente para obtener una mejor calidad. Con la finalidad de mitigar este crecimiento, la compresión de datos es esencial para lograr mejoras como tiempos de transmisión más pequeños o velocidades más altas en la transferencia de información. Esta compresión se puede llevar a cabo haciendo uso de varias técnicas. La elección de la técnica más adecuada es de gran importancia para ahorrar recursos como ancho de banda.

Este trabajo de titulación se lleva a cabo mediante un diseño experimental, enfoque cuantitativo y alcance descriptivo. 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. En Matlab se obtienen el tamaño de los archivos después de las compresiones. Estos valores se usan para realizar los cálculos de tasa de compresión y ancho de banda.

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. De esta manera, este trabajo aporta con un marco referencial en torno a la comparación de los tres algoritmos elegidos desde el punto de vista de la tasa de compresión y el uso de ancho de banda.

1 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). 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.

Para llevar a cabo este trabajo se utilizó una metodología de diseño experimental, enfoque cuantitativo y alcance descriptivo. Existen tres etapas en el desarrollo de este proyecto. La primera etapa implica la selección de los tipos de archivos a comprimir donde se eligieron archivos de texto, audio e imágenes. En esta etapa se escogió por conveniencia los

archivos a procesar intentando obtener una gran variedad en su distribución de bits.

Posteriormente, en la segunda etapa, denominada de compresión, se realizó la simulación de los algoritmos *Run Length Encoding* (RLE), Lempel-Ziv Welch (LZW) y Huffman Adaptativo mediante el programa matemático Matlab[1]. El desarrollo del código de simulación de los algoritmos se realizó en base a la fundamentación teórica de los mismos pudiendo imitar por completo su funcionamiento. Los archivos se procesaron sólo una vez por algoritmo dado que no había variación en el resultado de salida entre una compresión y otra bajo las mismas condiciones, es decir archivo y algoritmo.

Finalmente, en la tercera etapa se realiza el proceso de modulación, para esto se utilizó una técnica de modulación por amplitud de cuadratura denominada 4-QAM donde dos bits representaban un símbolo. Una vez hecho esto, el ancho de banda se lo calculó a partir de la relación más básica del mismo donde se indica que 1 Hertz es igual a 1 símbolo por segundo de velocidad de transmisión. No se hizo referencia a otros factores de una transmisión, así como redundancia, separación entre símbolos, entre otros. Esto se debe a que por ser una simulación se consideran condiciones totalmente ideales para el análisis de la compresión logrando

[1] Versión R2015b

así un control total de las pruebas, además, no tiene influencia su ausencia o presencia en los resultados de este trabajo.

Este trabajo de titulación está distribuido en seis capítulos. El Capítulo 1, Introducción, presenta información sobre los antecedentes que marcaron un precedente para este trabajo. Luego, se expone la temática a investigar dentro de la descripción del problema. A continuación, se encuentra el alcance y la delimitación del trabajo. La importancia de la realización de este trabajo se resalta en la sección de justificación. Finalmente, se podrán encontrar los objetivos tanto general como específicos.

Seguidamente, el Capítulo 2, denominado Marco Teórico, conlleva una revisión literaria de todos los elementos que rodean este trabajo. Se presenta el modelo de transmisión de información, conceptos de compresión de datos y una revisión de los tres algoritmos elegidos para este trabajo.

Posteriomente, el Capítulo 3, llamado Metodología, define el contexto metodológico siendo así el diseño, alcance y enfoque del trabajo. Adicionalmente, se expone una justificación sobre el uso de estos parámetros mencionados anteriormente.

La metodología en detalle de este trabajo se explica en el Capítulo 4 denominado Implementación de la Metodología. En este punto se abarcan temas específicos sobre el ambiente de pruebas, la simulación de la compresión, la modulación de los archivos y medición y análisis de resultados.

Una vez descrita la metodología en su totalidad, el siguiente capítulo es el 5 llamado Pruebas. En esta sección se presentan los resultados obtenidos a partir de las simulaciones.

Finalmente, el Capítulo 6 expone las conclusiones y contribuciones principales obtenidas a partir de este trabajo. Adicionalmente, se describen las limitaciones que se presentaron en el desarrollo de este trabajo. Finalmente, se redactan recomendaciones para ser tomadas en cuenta para trabajos futuros.

1.1 Antecedentes

Al-laham & El Emary (2007) compararon los algoritmos RLE, RLL, Huffman estático, LZ, LZW y HLZ mediante el uso de diferentes archivos con extensiones .doc, .txt, .bmp, .tif, .gif y .jpg. El objetivo de este trabajo fue determinar el algoritmo más eficiente para cada tipo de archivo. Los resutlados obtenidos en este trabajo fueron destinados a recomendar el uso de cada uno de los algoritmos para los tipos de archivos con las extensiones mencionadas anteriormente.

En 2007, Agulhari, Rocha & Bonatti midieron el rendimiento ante diferentes secuencias de bits de los algoritmos Lempel Ziv Welch (LZW), Huffman estático y RLE con el objetivo de determinar el tamaño de la secuencias de bits que mejor se adapta a cada uno. Adicionalmente, inclinaron su investigación en torno a variaciones de los algoritmos mencionados anteriormente o incluso combinaciones de los mismos.

En 2015, una investigación realizada por Alamin & Mustafa tuvo el objetivo de comparar los algoritmos de compresión Run Length Encoding (RLE) y Huffman estático. Este trabajo se basó en la ejecución de programas realizados en C++ para comparar archivos de diferentes tamaños y así obtener la tasa de compresión. El resultado de esta comparación mostró al algoritmo Huffman estático como más eficiente al comparar el tamaño original de los archivos con los tamaños luego de las

compresiones. La tasa promedio obtenida con Huffman fue del 40%, mientras que con RLE fue del 23%.

Estos antecedentes muestran el constante intento por obtener formas eficientes de compresión. Cada uno de ellos toma sus estudios desde puntos de vista diferentes. El primero destina su investigación a buscar el tipo de compresión más adecuado para cada tipo de archivo. Por otro lado, el segundo estudia cómo afecta la variación en el tamaño de las secuencias de bits en la efectividad de los algoritmos. Finalmente, el tercero se encarga de hacer una prueba total de eficiencia, estudiando así la funcionalidad general de los algoritmos sin basarse en ningún tipo de archivo en específico.

1.2 Descripción del problema

El incremento en la calidad de la información es directamente proporcional a la capacidad esta ocupa tanto en almacenamiento como en transmisión. Un claro ejemplo son los programas especializados para gráficos, los cuales generan archivos de gran tamaño debido a la complejidad de sus caracteríticas; causando que sean complejos de administrar. Asimismo, se vuelve inminente la necesidad de transmitir información a alta velocidad en las redes, esto ha motivado el desarrollo de una amplia gama de algoritmos de compresión para reducir el tamaño de los archivos (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.

1.3 Alcances y limitaciones

Este trabajo está enfocado en la comparación de los algoritmos de compresión LZW, RLE y Huffman Adaptativo con la intención de conocer cual 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.

La principal limitación de este trabajo es que las pruebas se realizan en un ambiente simulado, es decir que no se establece una transmisión real

de datos. Se toman en cuenta condiciones ideales, donde no existe ni interferencia ni variación alguna.

1.4 Justificación

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.

Los argumentos mostrados en los párrafos anteriores denotan la gran importancia de tener un marco referencial en lo que respecta a la comparación de varios algoritmos de compresión. Siendo así que, la

elección correcta de uno de estos puede significar un ahorro de recursos para la transmisión de datos, más aún cuando estos recursos son limitados como el ancho de banda.

1.5 Objetivos

1.5.1 Objetivo general

Realizar el estudio comparativo de la eficiencia de uso de ancho de banda en transmisiones de datos, mediante la implementación de los algoritmos de compresión RLE, LZW y Huffman Adaptativo bajo condiciones simuladas.

1.5.2 Objetivos específicos

1. Desarrollar programas en Matlab que simulen el funcionamiento de los algoritmos referidos en objetivo general.
2. Analizar rendimiento de compresión en ambiente simulado de los algoritmos mencionados en el objetivo general.
3. Relacionar el rendimiento de compresión mencionado en el objetivo específico dos con el uso de ancho de banda para una transmisión.

2 Marco teórico

En este capítulo se hace una revisión literaria de todos los elementos en los que son relevantes para este trabajo. Se revisan temas como el modelo de transmisión de información, conceptos importantes de transmisión de datos y además se hace una descripción de los algoritmos elegidos para este trabajo.

2.1 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.

2.2 Modelo de comunicación

El modelo de comunicación desarrollado por Shannon & Weaver (1963) se basa en un sistema de comunicación general donde la fuente de información selecciona el mensaje deseado en un conjunto de mensajes posibles los cuales son enviados a través de un transmisor.

El transmisor transforma o codifica esta información en una forma apropiada al medio de transmisión denominado canal. También convierte el mensaje en una señal que será enviada por un canal de comunicación hasta llegar al receptor, el cual se encargará de convertirla nuevamente en una aproximación del mensaje original. Durante la etapa de envío de la señal a través del canal pueden existir fuentes de ruido, donde se encontrará un conjunto de distorsiones o adiciones no deseadas por la fuente de información que afectan a la señal que representa el mensaje codificado por el transmisor. Las distoriones pueden afectar el producto final de la transmisión, resultando en distorsiones de sonido como en radio o teléfono, de imágenes, entre otros (Lazar, 1995).

2.3 Tipos de señales y parámetros de una señal

Se considera como señal a la variación en el tiempo de una magnitud física por ejemplo señal electromagnética, óptica, etc. Es posible clasificar las señales atendiendo a distintos criterios como los expresados a continuación en la tabla 1.

Además las señales pueden clasificarse atendiendo a la magnitud física de la señal en: señales eléctricas, que son aquellas en las que su magnitud hace referencia a voltaje o amperaje, señales ópticas, en las que su magnitud es intensidad luminosa, y señales electromagnéticas, en las que varía con el tiempo la fase, amplitud y/o la frecuencia (Gil, Pomares, & Candelas, 2010).

Tabla 1.

Clasificación de tipos de señales.

Tipo de señal	Descripción
Analógica	Señal continua que varía suavemente en el medio tanto en tiempo como en amplitud.
Digital	Señal discreta que sólo puede tener un número definido de valores en instantes de tiempo determinados.
Periódica	Formada por un patrón que se repite continuamente.
Aperiódica	No posee un patrón repetitivo.
Simples	Basadas en funciones senoidales o cosenoidales.
Compuestas	Basadas en una composición de señales simples.

Fuente: Elaboración propia.

2.4 Capacidad del canal

El canal de transmisión es el medio físico por el cual viaja la señal procedente del emisor con destino al receptor. Los primeros establecen una comunicación punto a punto entre emisor y receptor, mientras que los segundos permiten enviar información de modo de difusión.

Shannon (1948) considera que dado un canal con una capacidad de C unidades por segundo que recibe señales de una fuente de información de H unidades por segundo, la tasa de transmisión de H por la fuente de información y la capacidad C del canal, son las que determinan la efectividad del sistema para transmitir información. Si H>C será ciertamente imposible transmitir toda la información de la fuente, no habrá suficiente espacio disponible. Si H≤C será posible transmitir la información con eficiencia. La información entonces, puede ser transmitida por el canal solamente si H no es mayor que C.

Sin embargo, el problema de calcular la capacidad del canal se complica por la presencia de ruido. La presencia de ruido durante la transmisión provocará que el mensaje recibido contenga ciertos errores que contribuirán con la incertidumbre.

2.5 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).

2.6 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).

2.7 Información y entropía

La Teoría de la Información es la disciplina que se encarga del estudio y cuantificación de los procesos que se realizan sobre la información. La idea de la probabilidad de ocurrencia de cierto evento que nos da información puede modelarse por una variable aleatoria y su conjunto de mensajes y probabilidades asociadas. Al recibir un mensaje de esa variable aleatoria, se obtiene una cantidad de información, que depende sólo de la probabilidad de emisión de ese mensaje. La cantidad de información es una función creciente e inversa a la probabilidad, es decir, un menaje con menor probabilidad proporciona mayor cantidad de información (Abramson, 1963).

La definición básica de entropía en teoría de la información, mantiene una relación con la incertidumbre que hay en una fuente de información. Así como también la cantidad de ruido o desorden que contiene o libera un sistema. La entropía también se puede definir

como la cantidad de información promedio que contienen los símbolos usados. Los símbolos con menor probabilidad son los que aportan mayor cantidad de bits (Johansen, 1982).

En 1948, Claude Shannon propuso una medida para la información. En la cual, si se supone una fuente F de información de n mensajes Fi, cada uno con probabilidad pi, la información al recibir un mensaje está dado por la ecuación (1), mostrada a continuación. El uso de un logaritmo en base dos hace que la medida esté dada en bits.

$$I\ (Fi) = -\log_2 pi \qquad (1)$$

La medida media de información o entropía de F está dada por la esperanza matemática de la información de cada símbolo que se expresa en la ecuación (2).

$$H(F) = \sum_{i=1}^{n} pi * I(F = Fi) = -\sum_{i=1}^{n} pi * \log(pi) \quad (2)$$

Donde:

I = Cantidad de información del mensaje

F = Fuente de información.

Fi = Mensajes

pi = Probabilidad individual de los mensajes.

n = Número de mensajes.

H(F) = Entropía de la fuente de información.

2.8 Redundancia

La mayoría de los mensajes se constituyen a partir de un número limitado de posibilidades, por ejemplo, sólo 29 letras en el caso del idioma español. La probabilidad de ocurrencia de una estas posibilidades dentro de un mensaje depende de diferentes factores; por ejemplo la probabilidad que ocurra la letra “q” luego de una “p” es 0, estas combinaciones son impropias del español pero pueden llegar a ser probables en algún otro idioma.

Son estos dos hechos los que en conjunto determinan que todo mensaje contenga cierto grado de redundancia. En otras palabras, la redundancia se refiere a que las posibilidades dentro de un mensaje se repiten. Otra función muy importante de la redundancia es que nos permite ahorrar tiempo en la decodificación de los mensajes.

Si se pierde parte del mensaje por el ruido que afecta al canal, la redundancia permite rehacer en forma aproximada el mensaje. Por el contrario, la fracción de un mensaje no redundante que se pierde por el ruido es imposible de ser recuperada. En pocas palabras, la redundancia de los mensajes permite corregir con facilidad los errores u omisiones que hayan podido ocurrir durante la transmisión (Correa, 2008).

2.9 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 3 y 4. Entre mayor redundancia exista en los datos, mejor razón o tasa de compresión será obtenido (Mondelo & Iglesias, 2014).

$$Razon\ de\ compresión = \frac{N°\ Bytes\ archivo\ comprimido}{N°\ Bytes\ archivo\ original} \quad (3)$$

$$Factor\ de\ compresión = \frac{N°\ Bytes\ archivo\ original}{N°\ Bytes\ archivo\ comprimido} \quad (4)$$

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).

2.10 Clasificación por integridad de información

La compresión de datos puede dividirse en dos tipos principales: compresión con pérdida y compresión sin pérdida. En compresión sin pérdida, es posible reconstruir exactamente los datos originales D dado D'. Al proceso de reconstrucción se le denomina descompresión. Por otra parte, en compresión con pérdida, la descompresión produce solamente una aproximación D* a los datos originales D. La compresión de datos sin pérdida es comúnmente usada en aplicaciones como compresión de texto, donde la pérdida de un solo bit de información es inaceptable. La compresión con pérdida es usada a menudo para aplicaciones de compresión de imágenes, audio y video destinadas al entretenimiento. No es el caso en aplicaciones de compresión de imágenes médicas como por ejemplo la identificación de tumores o anomalías (Fowler & Yagel, 1994).

2.10.1 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).

2.10.2 Métodos de compresión con pérdida de información

Los métodos de compresión con pérdida de información también denominados *lossy* logran alcanzar tasas de compresión más elevadas a costa de sufrir una pérdida de información sobre el archivo original. Para la compresión de imágenes se emplean métodos *lossy*, ya que busca alcanzar una tasa de compresión considerable, pero que se adapte a la calidad deseada que la aplicación exige sobre la imagen objeto de compresión (Chavez, 2003).

2.11 Clasificación por técnica de compresión

Los médotos de compresión pueden clasificarse como adaptables, no adaptables, sustitucionales y estadísticos. Un método de compresión no adaptable es rígido y no modifica su operación o sus parámetros en respuesta de los datos particulares que se van a comprimir. Un método

adaptable examina los datos y modifica su operación o parámetros de acuerdo a ellos. Un método semi-adaptable o de dos pasadas, lee los datos que van a comprimirse para determinar sus parámetros de operación internos (primera pasada) y a continuación realiza la compresión (segunda pasada) de acuerdo a los parámetros fijados en la primera pasada (Salomon, 2002). Los métodos de compresión sin pérdida pueden clasificarse en dos tipos: sustitucionales y estadísticos. La principal distinción entre ambos es que en el caso de los métodos estadísticos, la codificación de un símbolo se basa en el contexto en el que este ocurre, mientras que los métodos sustitucionales agrupan símbolos creando un tipo de contexto implícito (Witten, Moffat, & Bell, 1999). En general, un compresor estadístico logra mejores razones de compresión que un compresor sustitucional pero la complejidad computacional y requerimientos de memoria para los compresores estadísticos son mucho mayores que en los compresores sustitucionales.

2.11.1 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).

2.11.2 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).

Una gran ventaja que presenta LZW es que el diccionario no tiene que transmitirse hacia el receptor para poder ser decodificado el mensaje. El algoritmo permite que la descompresión pueda realizarse únicamente a

partir de la cadena transmitida. El proceso para la compresión se demuestra de forma práctica en la tabla 2.

Tabla 2.

Ejemplo aplicado del algoritmo LZW.

Cadena de entrada = ABCDABCABCCABCEABCF			
Carácter de entrada	**Código de salida**	**Nuevo código**	**Nueva cadena**
AB	A	256	AB
C	B	257	BC
D	C	258	CD
A	D	259	DA
BC	256	260	ABC
A	C	261	CA
BCC	260	262	ABCC
AB	261	263	CAB
CE	257	264	BCE
A	E	265	EA
BCF	260	266	ABCF
EOF	F	-	-

Fuente: Elaboración propia

2.11.3 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 Abmouleh, 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.

2.11.4 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).

El algoritmo se inicia mediante la creación de una lista de todos los símbolos en orden descendente de sus probabilidades. A continuación, se construye un árbol, con un símbolo en cada hoja, de abajo arriba. Esto se hace por etapas, donde a cada paso se seleccionan los dos símbolos con las probabilidades más pequeñas; se añaden a la parte superior del árbol parcial, se eliminan de la lista y se reemplazan por un símbolo auxiliar que

representa a los dos símbolos originales. Cuando la lista se reduce a sólo un símbolo auxiliar que representa a todo el alfabeto, el árbol está completo. Posteriormente, se recorre el árbol para determinar los códigos de los símbolos (Salomon, 2002).

Este proceso se ilustra mejor con un ejemplo. Dados cinco símbolos a1, a2, a3, a4, a5, con probabilidades de aparición 0,4, 0,2, 0,2, 0,1 y 0,1, respectivamente, podemos agruparlos como se muestra en la Figura 1a, siguiendo este orden:

1. a4 se combina con a5 y ambas se sustituyen por el símbolo a45, cuya probabilidad es de 0,2.
2. Ahora quedan cuatro símbolos disponibles: a1 con una probabilidad de 0,4 y a2, a3, a45 con probabilidades de 0,2 cada uno. Se seleccionan de forma arbitraria a3 y a45, combinándolos y reemplazándolos por el símbolo auxiliar a345 cuya probabilidad es de 0,4.
3. En este momento tenemos tres símbolos, a1, a2 y a345 con probabilidades 0,4, 0,2 y 0,4, respectivamente. Se seleccionan de forma arbitraria a2 y a345, combinándolos y reemplazándolos por el símbolo auxiliar a2345 cuya probabilidad es de 0,6.
4. Finalmente, combinamos los dos símbolos restantes, a1 y a2345, sustituyéndolos por a12345 con una probabilidad de 1,0.

El árbol ahora está completo. Se muestra en la Figura 1a: la raíz principal, en la parte superior; y según se desciende, sus ramas izquierda y derecha; cada elemento o combinación de elementos está acompañado por su probabilidad de aparición; en cada rama del árbol se indica el código que se le ha asignado donde cada par determina dos posibles caminos que se diferencian de forma única marcando uno de ellos con un 0 y el otro con un 1. Esto se traduce en los códigos 0, 10, 111, 1101 y 1100, para a1, a2, a3, a4 y a5, respectivamente. La asignación de los bits a las ramas es arbitraria. El tamaño medio de este código es:

$$0{,}4 \times 1 + 0{,}2 \times 2 + 0{,}2 \times 3 + 0{,}1 \times 4 + 0{,}1 \times 4 = 2{,}2 \text{ bits/símbolo}$$

Es importante aclarar que el código Huffman no es único. Algunos de los pasos anteriores han sido elegidos arbitrariamente, ya que había más de dos símbolos con probabilidades pequeñas. La Figura 1b: muestra cómo los mismos cinco símbolos pueden combinarse para obtener otro código Huffman diferente 11, 01, 00, 101 y 100, para a1, a2, a3, a4 y a5, respectivamente. El tamaño medio de este código es:

$$0{,}4 \times 2 + 0{,}2 \times 2 + 0{,}2 \times 2 + 0{,}1 \times 3 + 0{,}1 \times 3 = 2{,}2 \text{ bits/símbolo,}$$

la misma que el código anterior.

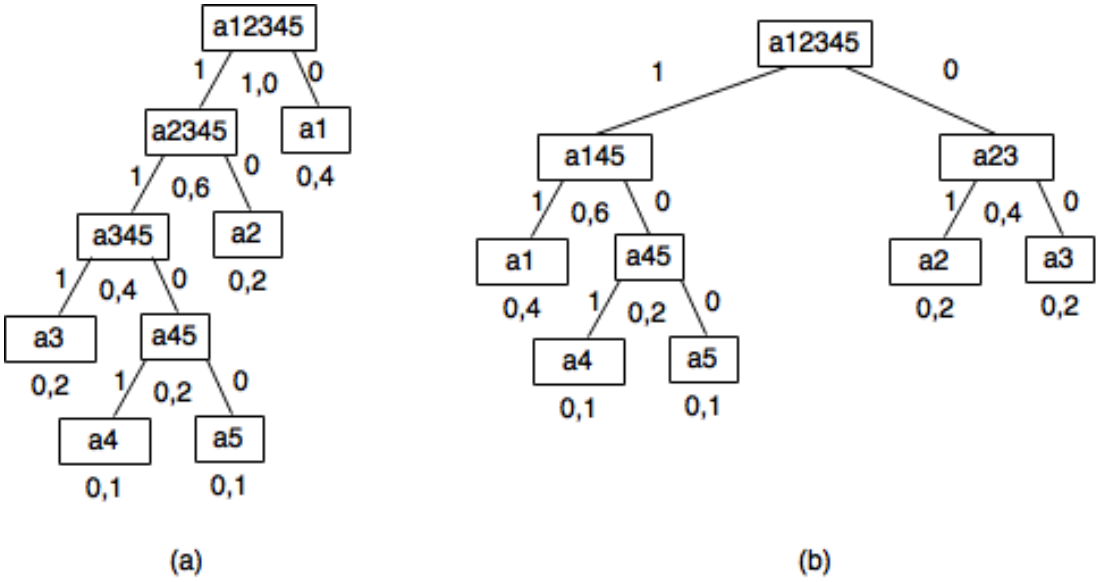

Figura 1. Ejemplo de Huffman estático

Fuente: Elaboración propia

De todo esto se infiere que las decisiones arbitrarias en la construcción del árbol de Huffman afectan a los códigos individuales, pero no al tamaño medio del código. Aun así, existe la siguiente pregunta: ¿cuál de todos los diferentes códigos Huffman para un determinado conjunto de símbolos es el mejor? La respuesta, aunque no obvia, es simple: el mejor código, es el que tiene la varianza más pequeña. La varianza de un código mide cuánto se desvían los tamaños de los códigos individuales del tamaño medio.

La varianza del código de la Figura 1a es:

$$0{,}4 \cdot (1 - 2{,}2)2 + 0{,}2 \cdot (2 - 2{,}2)2 + 0{,}2 \cdot (3 - 2{,}2)2 + 0{,}1 \cdot (4 - 2{,}2)2 + 0{,}1 \cdot (4 - 2{,}2)2 = 1{,}36,$$

mientras que la varianza de la Figura 1b es:

$$0{,}4 \cdot (2 - 2{,}2)2 + 0{,}2 \cdot (2 - 2{,}2)2 + 0{,}2 \cdot (2 - 2{,}2)2 + 0{,}1 \cdot (3 - 2{,}2)2 + 0{,}1 \cdot (3 - 2{,}2)2 = 0{,}16$$

El código “b” es, por lo tanto, preferible al “a”.

Una mirada cuidadosa a los dos árboles muestra cómo seleccionar el que queremos. En el árbol de la Figura 1a, el símbolo a45 se combina con a3, mientras que en el árbol de la figura 1b se combina con a1. La regla es: cuando hay más de dos nodos con la probabilidad más pequeña, se seleccionan —para fusionarlos— el nodo más bajo y el más alto del árbol. Esto combina los símbolos de menor probabilidad con los de mayor probabilidad, lo que reduce la varianza total del código. Si el codificador simplemente escribe la secuencia comprimida en un archivo, la varianza del código no produce ninguna diferencia. Un código Huffman de varianza pequeña es preferible sólo en aquellos casos donde el codificador transmite la secuencia comprimida, ya que se está generando sobre una línea de comunicaciones. En tal caso, un código con una varianza grande provoca que el codificador genere bits a una velocidad que varía continuamente. Dado que los bits tienen que ser transmitidos a una velocidad constante, el codificador tiene que usar una memoria intermedia. Los bits de la secuencia comprimida se introducen en el búfer a medida que se van generando y salen de él a una velocidad constante, para ser transmitidos. Es fácil ver intuitivamente que un código Huffman con varianza cero introducirá bits en

el búfer a una velocidad constante, por lo que sólo necesitará un buffer pequeño. Cuanto mayor es la varianza del código, más variable es la velocidad con la que entran los bits en el búfer, obligando al codificador a utilizar un buffer mayor (Solomon, 2002).

Este algoritmo de compresión en archivos permite reducir el tamaño de la imagen sin pérdida de información. Este método calcula el grado de repetición o frecuencia de ciertos pixeles y con base en eso asigna una codificación más larga a los elementos que se repitan de forma poco frecuente y más corta a los que se repitan frecuentemente.

Este método tiene el inconveniente de ser poco adecuado en el caso que se tenga un conjunto de entrada con muchos símbolos, ya que se requiere N-2 reducciones y N-2 asignaciones, siendo N el número de símbolos. Existen variaciones al método de Huffman que intentan reducir el tiempo de codificación. En el código Huffman truncado, solo se aplica el método de Huffman clásico a los símbolos más probables, distinguiéndose las palabras codificadas de las que no lo están, mediante la incorporación de un prefijo (Solomon, 2002).

Este es el codificador estadístico más popular, y erróneamente se tiende a pensar que su funcionamiento es óptimo, este algoritmo es capaz

de producir un código óptimo en el sentido de Mínima Redundancia para el código de entrada. Esta compresión sólo será óptima si las probabilidades de todos los símbolos de entrada son potencias enteras de ½, y el peor de todos los casos se presentará cuando alguno de los símbolos posea una probabilidad cercana al 100% (López, 1995).

2.11.5 Modelo Dinámico o Adaptivo

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. Esta actualización se representa gráficamente con el árbol adaptativo en la figura 2 mediante la compresión de la cadena “aardv” (Jagadeesh & Ankitha, 2013).

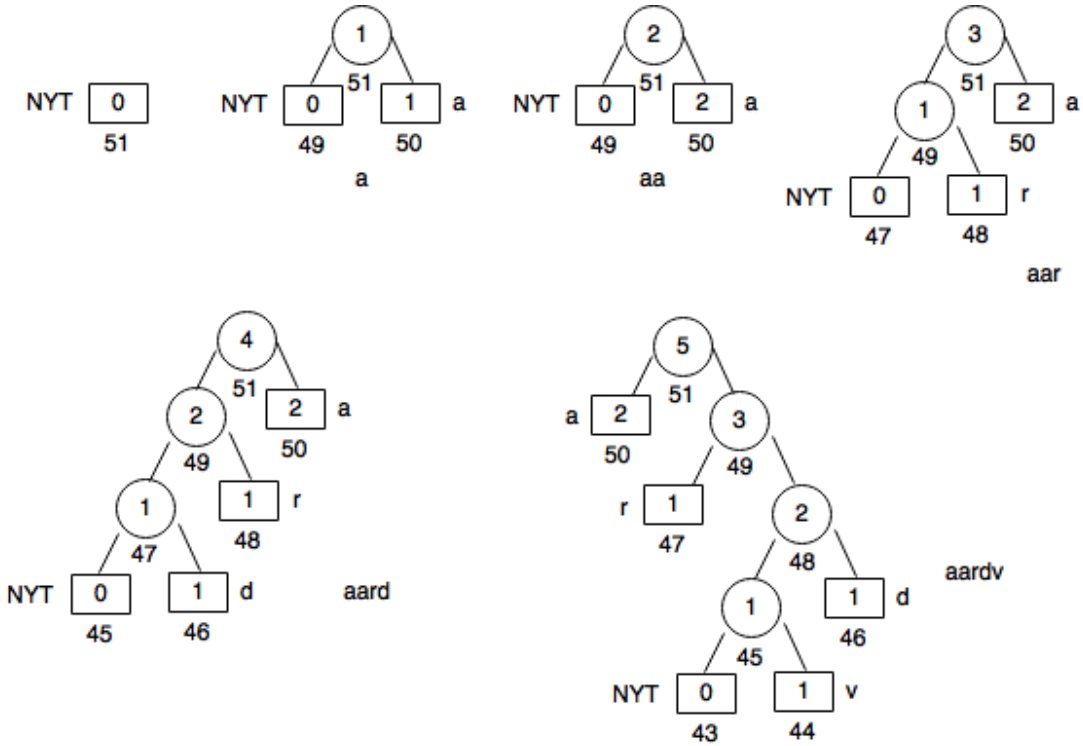

Figura 2. Ejemplo de actualización de árbol en Huffman Adaptativo

Fuente: Elaboración propia

Como se aprecia en el ejemplo de la figura 2 la distribución del árbol cambia continuamente debido a que se añaden nuevos elementos en la cadena a comprimir. En este caso sólo se puede apreciar un cambio grande que es en el último paso donde toda una rama se intercambia con un nodo individual.

3 Metodología

3.1 Descripción de la metodología

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.

3.2 Justificación de la metodología

Como indican Hernández, Fernández & Baptista (2010) un enfoque cuantitativo implica la medición numérica de las variables que rodean a la investigación. Puesto que esta investigación se centra en la medición tanto de ancho de banda como de eficiencia de compresión de los algoritmos. Se contempla que el enfoque elegido es el que más se aproxima a la naturaleza de esta investigación.

Asimismo, en lo que respecta al alcance descriptivo y diseño experimental. Hernández, Fernández & Baptista (2010) indican una investigación con alcance descriptivo se enfoca en especificar las propiedades, características y rasgos importantes del fenómeno que se analice, en este caso el análisis se encuentra en los algoritmos de compresión seleccionados. Además, un diseño experimental implica principalmente que debe existir manipulación intencional de las variables independientes, medición de variables dependientes y mecanismos de control. En este caso se usan archivos con diferentes tamaños de

diccionario y se mide el ancho de banda. Además, el punto de control es la revisión del tamaño de archivo antes de realizar la compresión.

4 Desarrollo de propuesta y resultados

4.1 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. Estos fundamentos se representan mediante los

diagramas de flujo de la figura 3 para RLE, figura 4 en el caso de LZW y también la figura 5 para el algoritmo Huffman Adaptativo. 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.

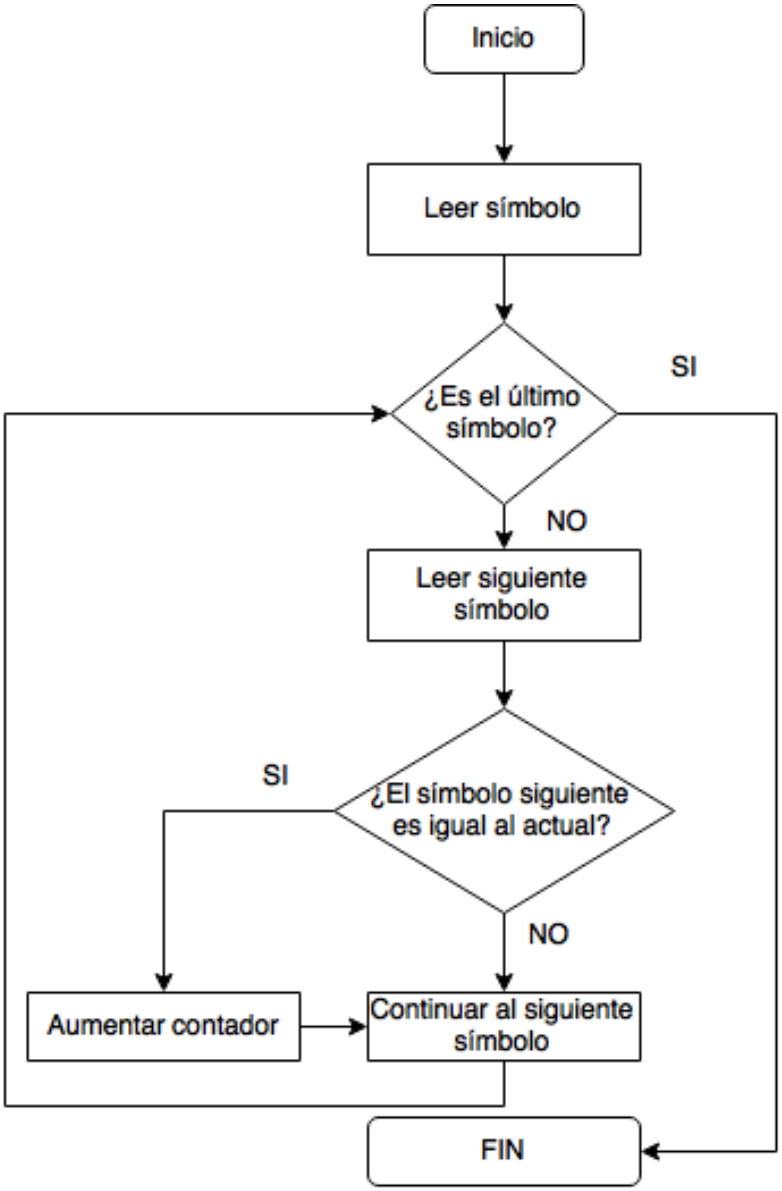

Figura 3. Diagrama de flujo de algoritmo RLE.

Fuente: Elaboración propia

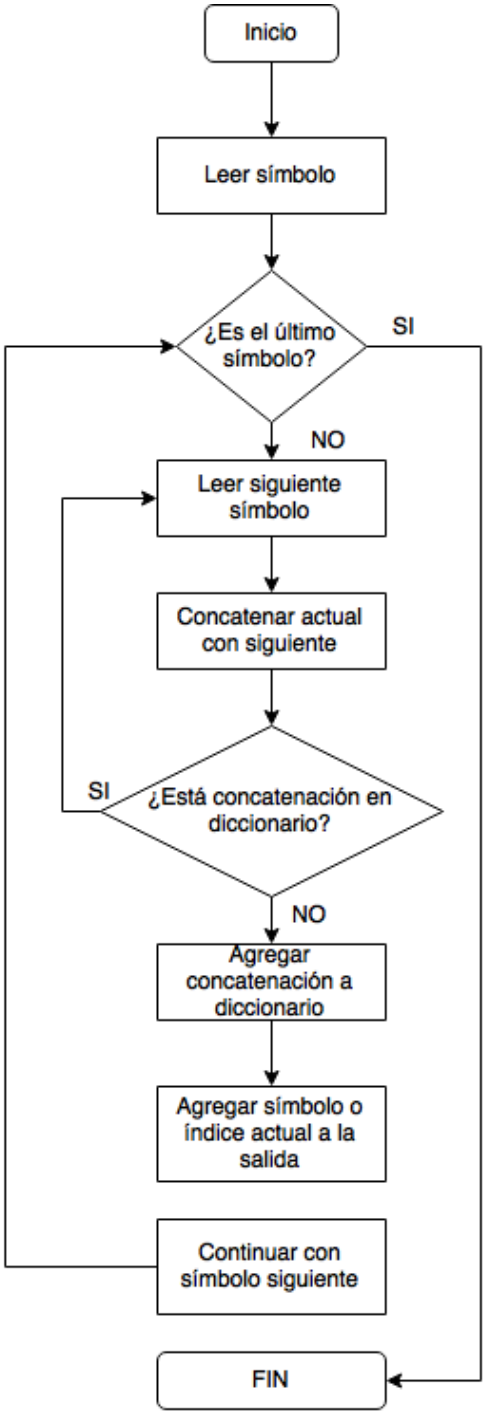

Figura 4. Diagrama de flujo de algoritmo LZW.

Fuente: Elaboración propia

Para el caso del algoritmo Huffman Adaptativo, puesto que es un algoritmo más complejo de programar. Se ha separado su diagrama de flujo en dos partes. Una de esas partes representa el proceso general apreciado en la figura 5, mientras que, la figura 6 representa únicamente el proceso de actualización del árbol que es una de las partes esenciales de esta técnica de compresión.

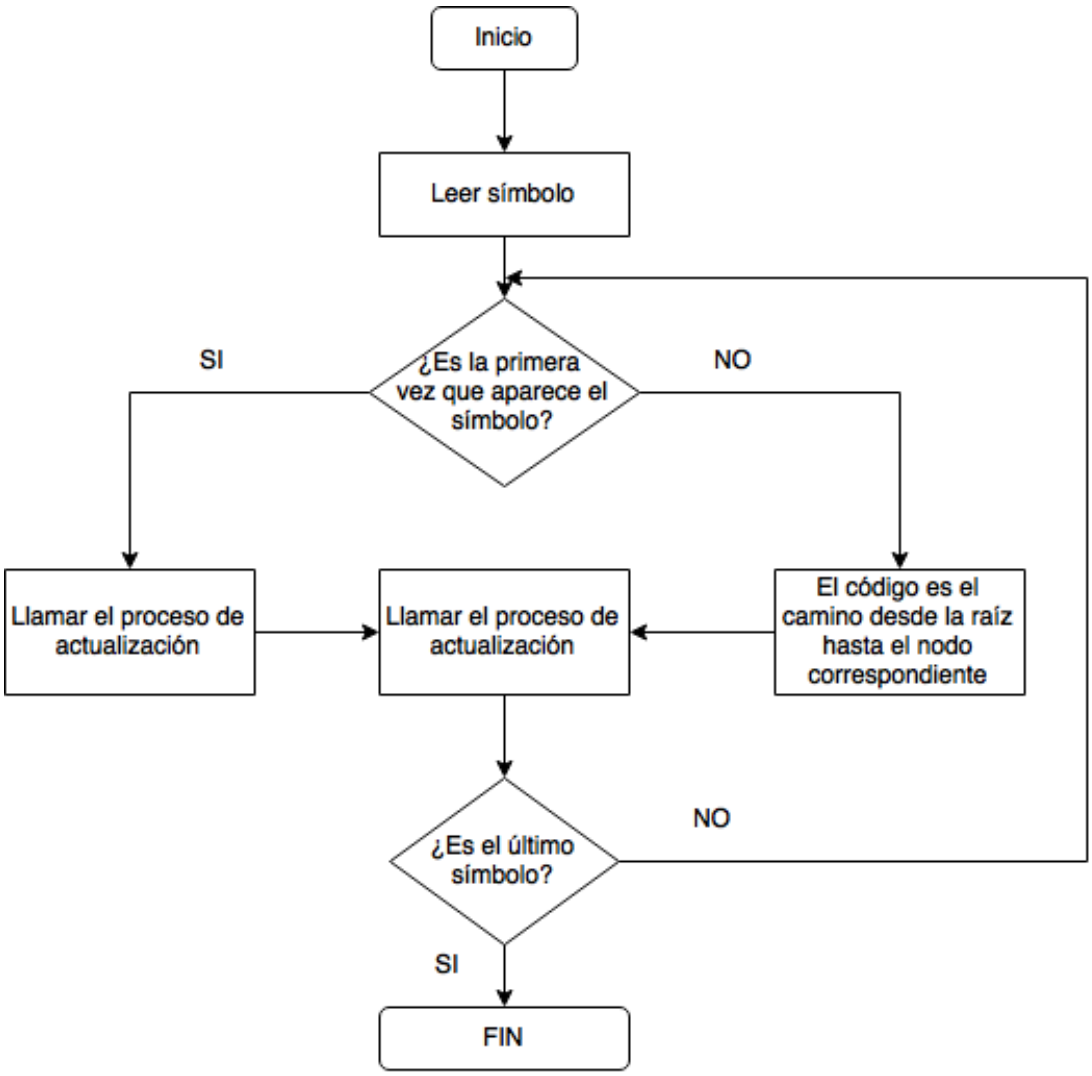

Figura 5. Diagrama de flujo de algoritmo Huffman adaptativo.

Fuente: Elaboración propia

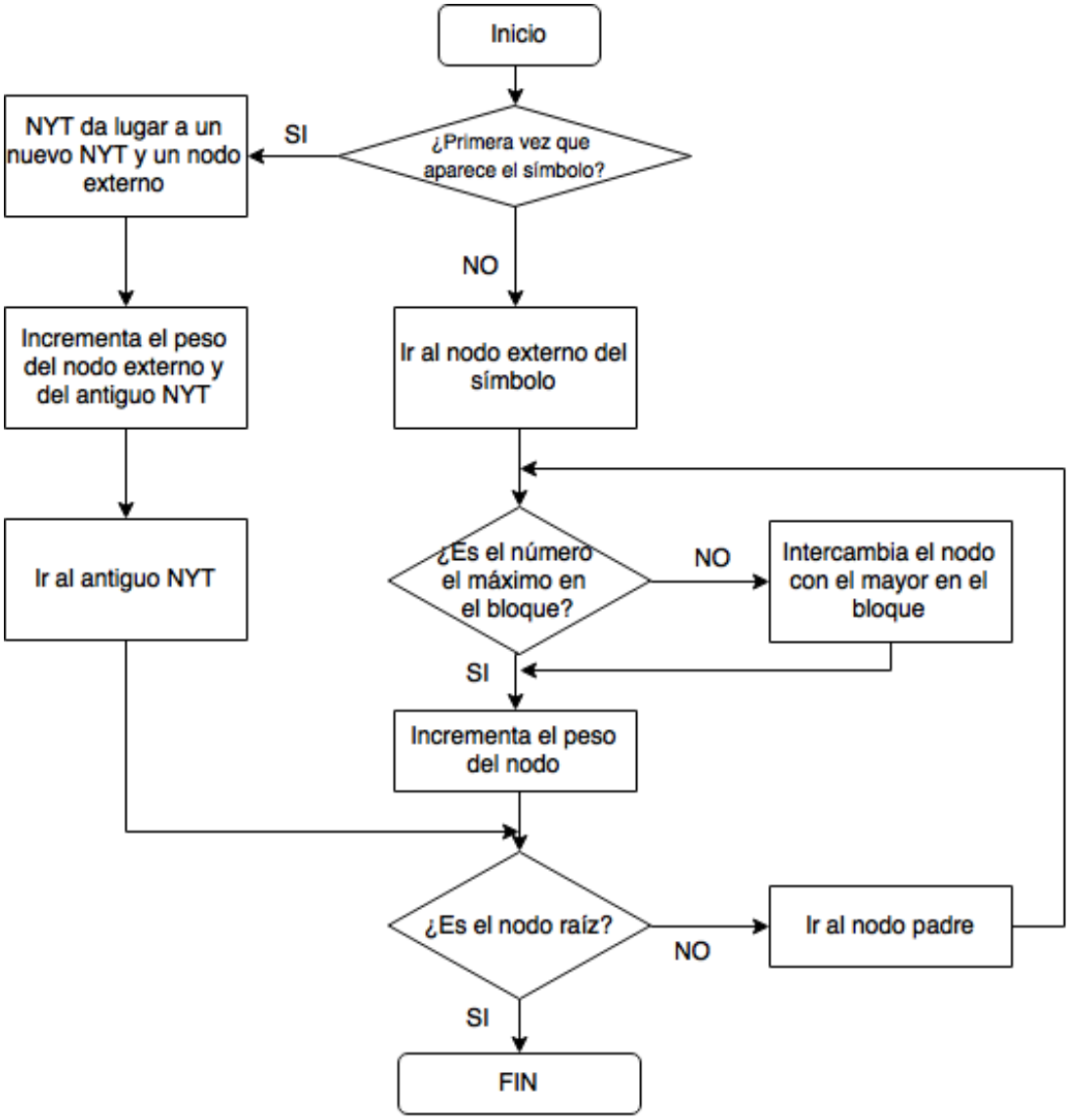

Figura 6. Diagrama de flujo de proceso de actualización de árbol Huffman Adaptativo.

Fuente: Elaboración propia

4.2 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 como se puede ver en la figura 7. 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.

Name	Size	Bytes	Class
img	100x100	10000	uint8

Figura 7. Captura de la entrada de imagen blanco y negro en Matlab.
Fuente: Captura de pantalla realizada en Matlab.

Name	Size	Bytes	Class
img	100x100x3	30000	uint8

Figura 8. Captura de la entrada de una imagen RGB en Matlab.
Fuente: Captura de pantalla realizada en Matlab.

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. Un ejemplo de la entrada en archivos de texto se puede observar en la figura 9.

Name	Size	Bytes	Class
texto	1x25000	50000	char

Figura 9. Captura de la entrada de archivo de texto.
Fuente: Captura de pantalla realizada en Matlab.

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. La representación del audio en forma de variable realizada por Matlab al abrirlo se observa en la figura 10. 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.

```
Name          Size            Bytes  Class

audio      13988x1            13988  uint8
```

Figura 10. Captura de entrada de archivo de audio.
Fuente: Captura de pantalla realizada en Matlab.

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.

4.3 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.

4.4 Formato de salida de la compresión

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, esto se aprecia en la figura 11. 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, esto puede ser visible en la figura 12. 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, esto se puede ver en la figura 13.

```
Name          Size          Bytes  Class

respuesta     1x39826       39826  uint8
```

Figura 11. Formato de salida de RLE en Matlab.

Fuente: Captura de pantalla realizada en Matlab.

```
Name          Size          Bytes  Class

output_lzw    1x7987        15974  uint16
```

Figura 12. Formato de salida de LZW en Matlab.

Fuente: Captura de pantalla realizada en Matlab.

```
Name          Size          Bytes  Class

out_huffman   1x113695      227390 char
```

Figura 13. Formato de salida de Huffman adaptativo en Matlab.

Fuente: Captura de pantalla realizada en Matlab.

4.5 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.

4.6 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 5. 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.

$$TC = \frac{TD \times 100}{TA} \qquad (5)$$

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 6.

$$\delta = \frac{CT}{D} \qquad (6)$$

Donde:

δ = Densidad de diccionario.

CT = Número total de caracteres del archivo.

D = Diccionario.

De acuerdo con la ecuación 7, la tasa de transferencia es inversamente proporcional al tiempo que toma una transmisión y directamente proporcional a la cantidad de símbolos a transferirse. Esta

ecuación se utilizó con la intención de medir la transmisión de datos desde el siguiente caso. Se calcula la tasa de transferencia de una transmisión en base a un tiempo fijo, en este caso igual a 0,05 segundos, y una cantidad de símbolos variable que se obtiene como resultado de la compresión de cada uno de los archivos.

$$SR = \frac{SIMBS}{t} \qquad (7)$$

Donde:

SR = Tasa de transferencia, dada en símbolos por segundo.

Símbolos = Cantidad de símbolos a transmitirse.

t = Tiempo de transmisión.

Siguiendo la metodología explicada en el capítulo anterior se realizaron pruebas y obtuvieron resultados para cada uno de los algoritmos de compresión en estudio ante los tres tipos de archivos propuestos. Los resultados brindados por las pruebas se direccionan tanto a la compresión como al ancho de banda.

4.7 Resultados generales

La tabla 3 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 3.

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 3 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 14. 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%.

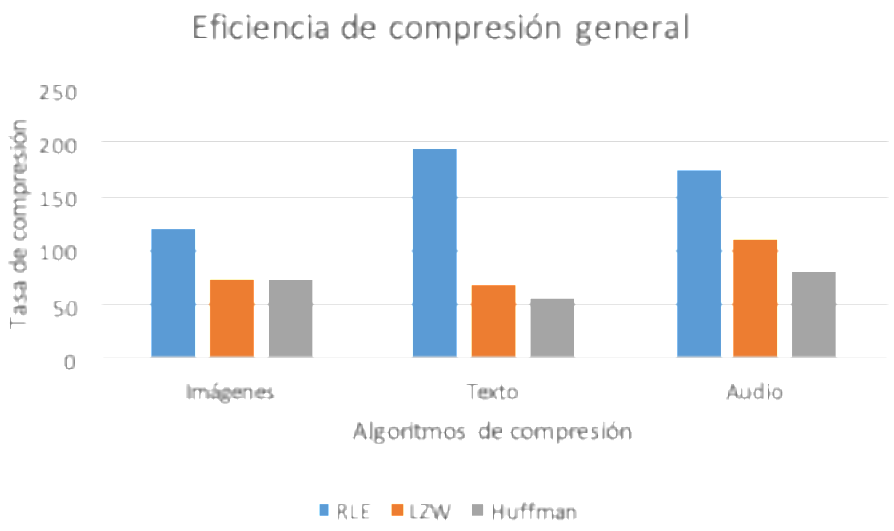

Figura 14. Comparación general de la eficiencia de compresión
Fuente: Elaboración propia

4.8 Resultados específicos

La figura 15, 16 y 17 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 15, 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.

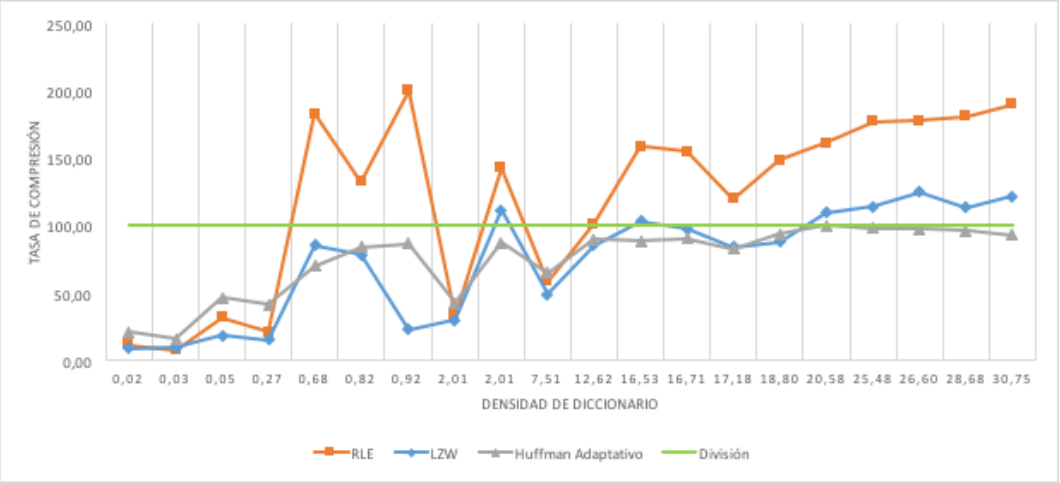

Figura 15. Comparación de RLE, LZW y Huffman Adaptativo en imágenes.

Fuente: Elaboración propia

En la figura 16, 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.

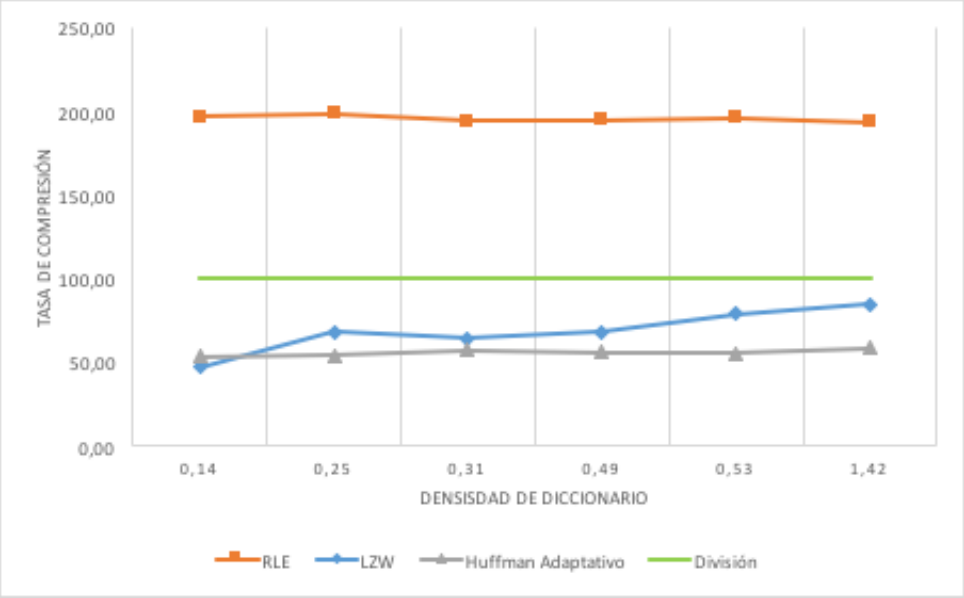

Figura 16. Comparación de RLE, LZW y Huffman Adaptativo en texto.
Fuente: Elaboración propia

En el caso de la figura 17, 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.

4.9 Ancho de banda

En lo que respecta al uso de ancho de banda, las figuras 18, 19 y 20 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.

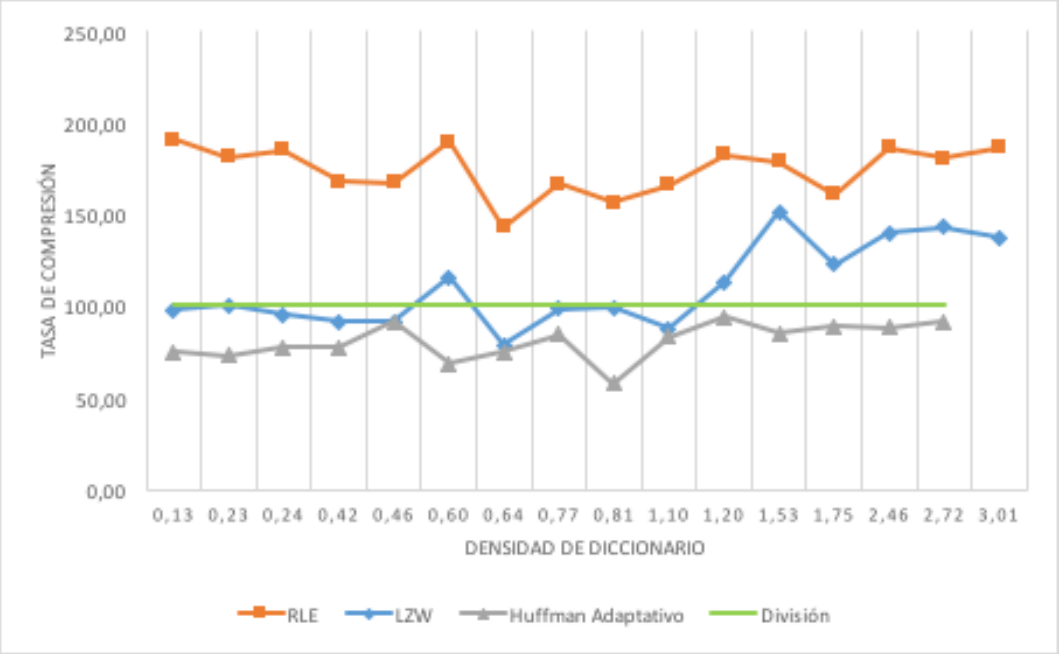

Figura 17. Comparación de RLE, LZW y Huffman Adaptativo en audio.

Fuente: Elaboración propia

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 18 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.

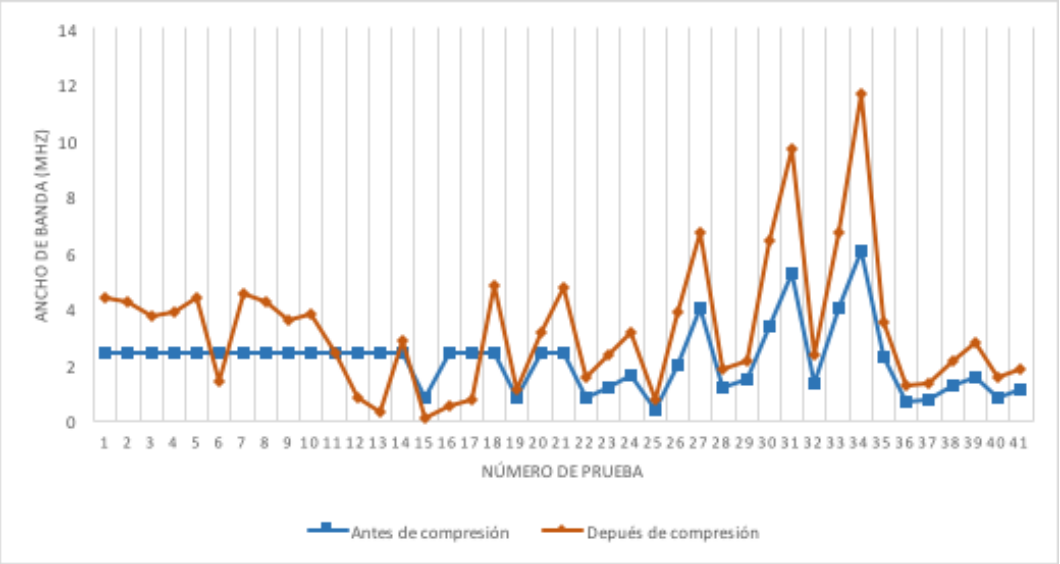

Figura 18. Comparación de uso de ancho de banda antes y después de la compresión RLE.

Fuente: Elaboración propia

El caso de LZW se notan ciertas diferencias en comparación al referido en la figura anterior. En la figura 19 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.

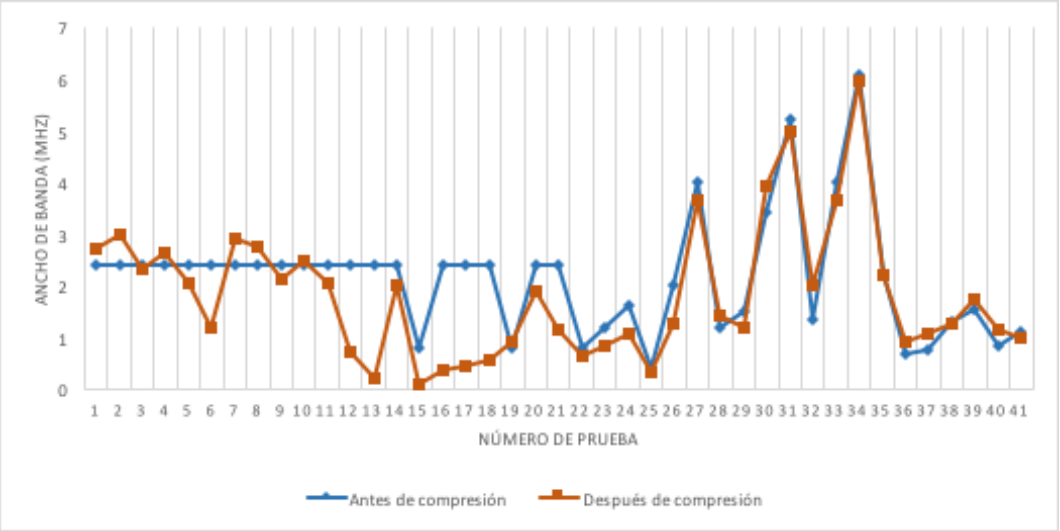

Figura 19. Comparación de uso de ancho de banda antes y después de la compresión LZW.

Fuente: Elaboración propia

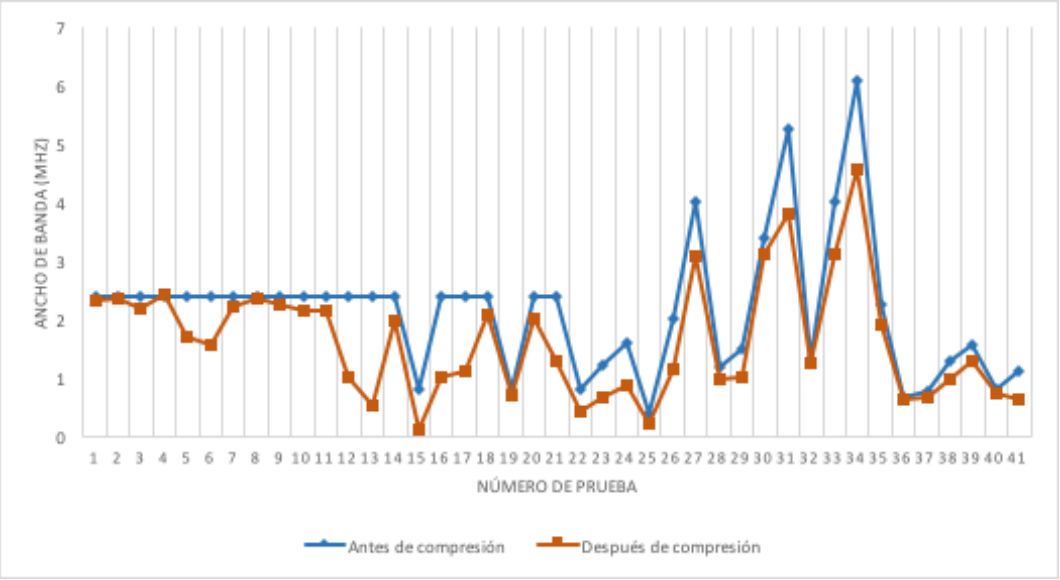

Figura 20. Comparación de uso de ancho de banda antes y después de la compresión en Huffman Adaptativo.

Fuente: Elaboración propia

Finalmente, para Huffman Adaptativo, esquematizado en la figura 20 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 20. En relación al resto de los casos, se observan reducciones importantes de ancho de banda para los archivos comprimidos.

5 Conclusiones y recomendaciones

5.1 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 18, 19 y 20, 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.

5.2 Recomendaciones

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.

6 Referencias

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

Agulhari, C. M., Rocha Silveira, R. M., & Bonatti, I. S. (14 de Septiembre de 2007). Lossless compression applied to sequences of bits.

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.

Chavez, D. (Diciembre de 2003). Implementación del algoritmo de Huffman para compresión y descompresión aplicado a datos provenientes de imágenes telemétricas. *Implementación del algoritmo de Huffman para compresión y descompresión aplicado a datos provenientes de imágenes telemétricas*. Quito, Ecuador.

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

De Moura, E., Navarro, G., & Ziviani, N. (1997). Indexing compressed text. En E. De Moura, G. Navarro, & N. Ziviani. 4th South American Workshop on String Processing.

Drozdek, A. (2002). Elements of Data Compression. En A. Drozdek, *Elements of Data Compression.* Thomson Learning.

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.

Fowler, J., & Yagel, R. (1994). *"Lossless Compression of Volume Data".* Symposium on Volume Visualization.

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.

Hernandez, R., Fernandez, C., & Baptista, M. (2010). *Metodología de la Investigación.* México: Mc Graw Hill.

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.

Johansen, O. (1982). Introducción a la teoría general de sistemas. Editorial Limusa.

Lazar, J. (1995). La ciencia de la comunicación. Publicaciones Cruz O.

López, M. (1995). *Notas de visión y apuntes sobre ingeniería del software.* Univ de Castilla La Mancha.

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

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

Morales, M. (2003). *Notas sobre compresión de datos.* Recuperado el 10 de Noviembre de 2016, de http://www.tamps.cinvestav.mx/~mmorales/documents/Compre.pdf

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.

Roncagliolo, P. (2007). *Departamento de Electrónica .* Recuperado el 10 de 10 de 2016, de Departamento de Electrónica de la Universidad Técnica Federico Santa María: http://www2.elo.utfsm.cl/~elo328/pdf1dpp/PDI15_Compresion_1dpp.pdf

Salomon, D. (2002). *A Guide to Data Compression 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.

Shannon, C. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal, 27*, 379-423.

Shannon, C., & Weaver, W. (1963). *The Mathematical Theory of Communication.* Illinois.

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.

Witten, I., Moffat, A., & Bell, T. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition.* San Francisco: Morgan Kaufmann Publishers.

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.