ESCUELA POLITÉCNICA DEL EJÉRCITO

SEDE LATACUNGA

CARRERA DE INGENIERÍA ELECTRÓNICA

Proyecto de Grado previo a la obtención del Título en Ingeniería Electrónica en Instrumentación.

"Estudio y análisis de la compresión de datos de almacenamiento de información, diseño e implementación del Código de Huffman Adaptativo"

César Alfredo Naranjo Hidalgo

Latacunga – Ecuador

2005

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el señor: César Alfredo Naranjo Hidalgo, bajo nuestra supervisión.

Ing. Armando Álvarez S.
DIRECTOR DE TESIS

Ing. Eddie Galarza Z.
CODIRECTOR DE TESIS

AGRADECIMIENTO

Primeramente a DIOS quien siempre a guiado nuestros caminos.

Al personal Docente de la Escuela Politécnica del Ejército sede Latacunga, por los conocimientos y experiencias compartidas durante el proceso de formación académica. De manera especial a los señores ingenieros: Armando Álvarez y Eddie Galarza director y codirector respectivamente, quienes colaboraron de una u otra manera para el desarrollo y culminación del presente proyecto.

DEDICATORIA

A mi Esposa ***Sandra*** el amor de mi vida y a mi gran tesoro mis Hijas: ***Grace*** y ***Cecibel***. Quienes con su amor y comprensión supieron darme fuerza para salir adelante

César

INDICE

INTRODUCCIÓN

CAPÍTULO I

CONCEPTOS DE LA TEORÍA DE INFORMACIÓN

1.1 INTRODUCCIÓN 1
1.2 PRINCIPIOS DE LA TEORÍA DE INFORMACIÓN 2
1.3 VARIEDAD DE SÍMBOLOS 7
1.4 VELOCIDAD DE LA SEÑAL 12
1.5 BAUDIO 14
1.6 INFORMACÓN Y SU MEDIDA 16

CAPÍTULO II

GENERACIÓN DE LA INFORMACIÓN

2.1 ENTROPÍA 21
2.1.1 Propiedad de la Entropía 25
2.1.2 Redundancia 28
2.2 FUENTE DE SÍMBOLOS 29
2.2.1 Fuente de Información de memoria nula (FIMN) 31
2.2.1.1 Entropía de la fuente de memoria nula 32
2.2.1.2 Intervalo de valores de la Entropía de una fuente de Q Símbolos 33
2.2.2 Fuentes de Información con memoria o Fuentes Markov 34

2.3 PROBABILIDADES 36
2.4 CAPACIDAD DE CANAL 39
2.4.1 Introducción 39
2.4.2 Teorema de Capacidad Máxima de un Canal 41

CAPÍTULO III

CODIFICACIÓN DE LA INFORMACIÓN

3.1 TRANSMISIÓN DE LA INFORMACIÓN 44
3.1.1 Tipos de Transmisión de Datos 44
3.1.1.1 Transmisión Analógica 44
3.1.1.2 Transmisión Digital 45
3.1.1.3 Transmisión Asíncrona 45
3.1.1.4 Transmisión Sincrónica. 46
3.1.1.5 Transmisión de Datos en Serie 46
3.1.1.6 Transmisión en Paralelo 47
3.1.2 Medios de Transmisión 47
3.1.3 Ventajas de la Transmisión Digital 48
3.1.4 Perturbaciones en la Transmisión 48
3.1.4.1 Atenuación 48
3.1.4.2 Distorsión de retardo 49
3.1.4.3 Ruido 49
3.2 CODIFICACIÓN: DETECCIÓN Y CORRECCIÓN DE ERRORES 49
3.2.1 Codificación 50
3.2.2 Propiedad de los códigos 51
3.2.3 Codificación de la fuente 54
3.2.3.1 Código BCD 54
3.2.3.2 Código EBCDIC 54
3.2.3.3 Código FIELDATA 55
3.2.3.4 Código ASCII 55
3.2.4 Códigos Detectores y Correctores de Error 55
3.3 CÓDIGOS LINEALES 60
3.4 CÓDIGOS HAMMING, REED MULLER 64

3.4.1 Código Hamming 64
3.4.2 Código Reed Muller 67
3.5 CÓDIGOS CÍCLICOS 68
3.5.1 Polinomio Generador 69
3.5.1.1 Códigos Polinómicos 70
3.5.2 Polinomio Chequeo de Paridad 71
3.5.3 Codificación para Códigos Cíclicos 74
3.5.4 Cálculo del Síndrome. 76

CAPÍTULO IV

COMPRESIÓN DE DATOS

4.1 INTRODUCCIÓN 78
4.2 EL PROBLEMA DE LA COMPRESIÓN 79
4.3 COMPRESIÓN ESTADÍSTICA 82
4.3.1 Codificación Huffman 83
4.3.2 Código Shannon-Fano 84
4.3.2.1 Diagrama de árbol 85
4.3.3 Códigos Aritméticos 87
4.3.4 Compresión Predictiva 88
4.4 COMPRESIÓN BASADA EN DICCIONARIO 88
4.4.1 Compresión RLE 89
4.4.2 Código Lempel – Ziv 90
4.5 COMPRESORES EN TIEMPO REAL 92

CAPÍTULO V

IMPLEMENTACIÓN DEL CÓDIGO HUFFMAN ADAPTATIVO

5.1 GENERALIDADES 94
5.2 CÓDIGO DE HUFFMAN 95
5.2.1 Algoritmo Huffman 95

5.2.2 Mecanismo del algoritmo 96
5.3 IMPLEMENTACIÓN DEL SOFTWARE 100
5.3.1 El Interfase Gráfico de Usuario (GUI) 107

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES 117
6.2 RECOMENDACIONES 119

BIBLIOGRAFÍA
ANEXOS

INTRODUCCIÓN

La historia de las comunicaciones es tan antigua como la del hombre. En efecto, desde los primeros registros históricos hay evidencia del deseo y los esfuerzos del ser humano por comunicarse entre sí más allá del alcance de la voz.

El propósito de un sistema de comunicación es, en el más amplio sentido, la transmisión de información desde un punto en el espacio y el tiempo hasta otro. Pero todos los modos de comunicación no se encuentran libre de errores.

Hay gente que habla por los codos, pero dicen muy poco. En cambio hay otros que todo lo que dicen es *útil*. Estos son los dos casos de datos con redundancia y datos sin redundancia. Aunque el primer caso puede ser deseable en alguna circunstancia (actos sociales...), no lo es en la mayoría.

La transmisión de datos no es tan rápida como se quisiera, los dispositivos de almacenamiento no tienen capacidad ilimitada, las cantidades muy grandes de datos son poco manejables. Sería interesante reducir la cantidad de datos, pero sin perder *información*.

La Teoría de la Información muestra, entre otras cosas, el camino a seguir para determinar la cantidad de información útil de unos datos y para comprimir la información de manera que los datos se representen de una manera eficiente.

La Teoría de la Información se desarrolla en términos de probabilidades, ya que la información tiene una naturaleza aleatoria (sí se supone de antemano la información, ¿para qué se la desea?.). Por supuesto, en la realidad no se dispone a priori de las probabilidades necesarias, por lo que habrá que estimarlas de los datos existentes.

Hoy en día los requerimientos de almacenamiento de información lleva a buscar formas eficaces de almacenamiento de información. La necesidad de comprimir información, no es tema de actualidad. A finales de los setenta se dieron los primeros pasos hacia la compresión de datos.

La adquisición de software nuevo o la actualización del mismo, reduce el espacio de almacenamiento de información, y lleva a la búsqueda y utilización de software especializado en la compresión de datos. La elección del mismo depende de las características físicas del hardware, o bien de las necesidades de almacenamiento.

Las necesidades de la compresión, son entre otras, que cuando el medio de almacenamiento llega al límite de su capacidad, si no se cuenta con otro medio para seguir almacenando se puede recurrir al software de compresión que se adapte a estas necesidades, ahorrando espacio de almacenamiento dándole un uso óptimo al mismo; la transmisión de un alto volumen de datos provoca un aumento en el costo del envío que crece en forma considerable y que conlleva a la transmisión de datos en forma comprimida, reduciendo costos de envío; de igual forma el tiempo de transmisión en redes o vía satélite es limitado y costoso.

Las comunicaciones y la diversidad de herramientas de software y hardware llevan a la integración de aplicaciones como camino a seguir en la compresión de información.

En los últimos años se ha dado un aumento espectacular tanto de la capacidad de almacenamiento de los ordenadores como de la velocidad de proceso de éstos. A esto lo acompaña una bajada de los precios de memoria principal y secundaria así como también un aumento de velocidad de estos dispositivos. Esto hace que se pregunte ¿para qué la compresión?.

Sin embargo, el auge que últimamente han tenido las redes de ordenadores hace que cada vez más usuarios pidan más prestaciones a la red sobre la que están conectados.

Prestaciones que, como siempre, están por encima de las posibilidades reales. Cuando se habla de posibilidades se refiere principalmente a la velocidad de transferencia de datos. La compresión de datos observa bloques repetitivos de datos y los envía al modem remoto en forma de palabras codificadas. Cuando el otro modem recibe el paquete lo decodifica y forma el bloque de datos original.

En este entorno, para conseguir mayores prestaciones de velocidad, los implementadores de los programas deben recurrir a técnicas que les permitan superar de alguna manera las deficiencias físicas de la red y de los equipos de conexión (que les permita, por ejemplo, ofrecer videoconferencia a través de *módems* de 14.400 bps).

La técnica más importante en este sentido es la compresión de datos. La compresión de datos es beneficiosa en el sentido de que el proceso de compresión-transmisión-descompresión es más rápido que el proceso de transmisión sin compresión.

Pero no sólo es para la transmisión para lo que se usa la compresión, también para el almacenamiento masivo. La necesidad de almacenamiento también crece por encima de las posibilidades del crecimiento de los discos duros o memoria.

Este proyecto que se realiza en este trabajo, tiene por objetivo el de implementar un sistema que realice la compresión y descompresión de información. Para esto va ser muy útil el empleo del paquete MATLAB, el mismo que se define como un "Laboratorio Matricial". Con la ayuda del paquete MATLAB, se desarrollaran las funciones que permitan realizar la compresión y descompresión de la información a través de la implementación del Código Huffman Adaptativo, las mismas que constituyen los programas para la aplicación.

Para el desarrollo del presente proyecto se elaboraron los siguientes capítulos:

En el capítulo I, se realiza un análisis de la teoría de la información y una introducción a la medida de la misma, dando conceptos básicos que forman la base para los procesos aleatorios.

En el capítulo II, se presenta un estudio de la generación de la información, analizando las fuentes generadoras de símbolos con sus respectivas probabilidad de ocurrencia y términos empleados por las mismos.

En el capítulo III, se realiza un análisis de la codificación de la información, detección, corrección del error presente en la palabra recibida para obtener la decodificación de la misma.

En el capítulo IV, se presenta un estudio de la compresión de datos, dando conceptos básicos así como la aplicación de la generación de diagramas de árbol para la compresión y analizando los algoritmos más empleados, uno de ellos el algoritmo de Huffman y los tipos de compresión de datos sin pérdidas.

En el capítulo V, se realiza una introducción al paquete MATLAB, se presenta el diseño de las funciones y programas necesarios para la compresión y descompresión de información utilizando el algoritmo de Huffman.

En el capítulo VI, se realizan, las conclusiones y recomendaciones sobre el tema desarrollado.

CAPÍTULO I

CONCEPTOS DE LA TEORÍA DE INFORMACIÓN

1.1 INTRODUCCIÓN

El propósito de un sistema de comunicaciones es transmitir información desde un emisor hasta un receptor a través de un canal.

El esquema genérico de un sistema de comunicaciones es el siguiente:

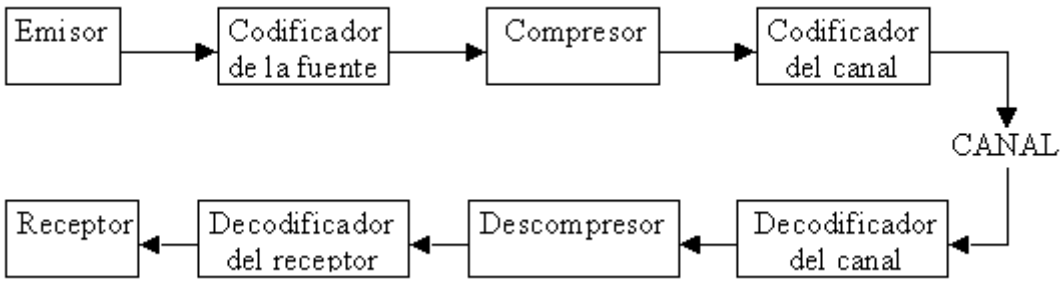


FIGURA. I.1 Sistema de Comunicación

El emisor es una fuente discreta de información desde la que se emiten los distintos símbolos del alfabeto fuente que se quieren transmitir.

Los símbolos emitidos por el emisor llegan al codificador de la fuente donde son transformados en símbolos de un código binario más adecuado, para ser transmitido a través de un canal de comunicaciones. Opcionalmente estos símbolos codificados pueden ser comprimidos con el objetivo de reducir su tamaño para conseguir una transmisión más rápida.

Durante la transmisión de los símbolos a través del canal, pueden producirse alteraciones de los mismos debidas a la presencia de ruido en el canal. A estas alteraciones se las denomina errores. Por ello, antes de enviar los símbolos codificados a través del canal, se realiza una nueva codificación orientada a que el receptor pueda detectar y corregir los errores producidos en el canal.

En la recepción se ejecuta un proceso inverso. Primeramente se realiza una decodificación del canal para detectar y corregir los posibles errores que contengan los símbolos recibidos a través del canal. A continuación se procede a una posible descompresión de los símbolos en el caso de haber sido comprimidos en la fuente. Por último se realiza una decodificación en la que los símbolos codificados se transforman en los símbolos originales que fueron transmitidos por el emisor.

Por lo tanto, el objetivo de un sistema de comunicaciones es transmitir la información de una fuente hasta un destinatario (destino). Además a lo largo del sistema la señal portadora de información puede ser transformada para asumir diferentes formatos, en las que era fundamental que el contenido de la información asociado a un mensaje fuera preservado. De este modo, la tónica de toda la explicación recae sobre el término "información".

El concepto de información es muy reciente y además sumamente sencillo. Fue desarrollado en la década de los 40's por el matemático norteamericano Claude Shannon, para referirse a todo aquello que está presente en un mensaje o señal cuando se establece un proceso de comunicación entre un emisor y un receptor.

1.2 PRINCIPIOS DE LA TEORÍA DE INFORMACIÓN.

La palabra información es comprendida por más de una manera; ya sea en el sentido común (de la vida cotidiana) o en el sentido estrictamente técnico (de la Teoría de la información). En el sentido común, se da mucha importancia al significado y valor de la información. Estos conceptos son muy subjetivos para la interpretación por modelos técnicos.

Por otro lado, desde el punto de vista físico, el canal es solicitado de una forma que no depende del significado o valor atribuidos a la información, sino de las características del fenómeno físico que lo transporta[1].

Desde el punto de vista técnico, la información es analizada en lo que respecta a las características de la diversidad y de la aleatoriedad de los símbolos que selecciona. Interesa evaluar cuantos símbolos distintos hay que reconocer y cual es la probabilidad de ocurrencia de cada uno. Este análisis debe ser hecho tanto en su aspecto estático (propiedades intrínsecas de los mecanismos de producción de mensajes) como en su aspecto dinámico (acompañándose la producción de mensajes a lo largo del tiempo).

Se podría intentar dar una definición amplia sobre el concepto de información, según la cuál, la información es una disciplina matemática que proporciona importantes contribuciones a diversas ciencias como la informática, comunicaciones. Sin embargo, se tratará de dar una definición más intuitiva del concepto de información. La información que transmite un mensaje no está relacionada con su longitud. Se

[1] Naturaleza del concepto de información. Sistemas de Comunicación Análogo Digital. Barradas R.

puede tener dos mensajes con distinta longitud y que transmitan la misma información.

"*El concepto de información está muy relacionado con el concepto de probabilidad. Cuanto más probable es un mensaje menos información contiene"*.

Se definirá como **información** todo aquello que revista interés para todos. La información puede venir representada en formato **analógico**, es decir, pudiendo adoptar cualquier valor entre infinitos en un determinado rango (como por ejemplo la temperatura de un termómetro) o **digital,** esto es, pudiendo adoptar un número finito de valores en un rango determinado (como por ejemplo los números enteros de 0 a 100).

La información analógica es, en principio, más precisa que la digital ya que da un valor exacto, pero está muy sujeta a errores e interferencias como por ejemplo los errores de redondeo debido a su naturaleza continua.

La información digital sin embargo es de tipo discontinuo ,es menos precisa ya que se tiene un conjunto finito de valores en un determinado rango pero también está menos sujeta a errores y a variaciones.

Para esto se cuenta con los símbolos de un alfabeto fuente que son transmitidos por el emisor. Cada uno de estos símbolos tiene asociada una probabilidad.

Toda fuente de información dispone de un conjunto de símbolos. En el mecanismo de selección sucesiva de símbolos para la composición del mensaje se produce la información.

El número distinto de símbolos de un conjunto se denomina variable de ese conjunto. Por lo tanto, la variable de una fuente exprime el número distinto de símbolos que ella posee. Para que un símbolo sea seleccionado en la fuente, de entre todos los de su alfabeto, hay la necesidad de una cierta cantidad de información para que sea escogido.

Por ejemplo, en el alfabeto de letras, una fuente selecciona la letra C para componer la palabra CASA. Cuando el destinatario recibe esta letra le causa una sorpresa por haber sido la letra C y no otra la seleccionada. Esta sorpresa corresponde a una cierta cantidad de información que el destinatario recibe.

Entonces hay dos puntos de vista: uno, el de la fuente, imprimiendo una cantidad de información del símbolo por haberlo escogido; el otro, el del destinatario, que toma esta información al identificar el símbolo enviado. De esta forma, se puede indicar que el proceso de selección o el modo de escoger de un particular símbolo de entre los de un conjunto de símbolos, corresponde a la *generación de información* y que el proceso de identificación de este símbolo, dentro del mismo conjunto de símbolos, corresponde a la *recepción de la información*.

La información es un atributo de un proceso de selección. Así como se dice que el calor es energía y que la temperatura es un atributo de esta energía, se puede también decir que la selección es un proceso y que la información es un atributo de este proceso. Como se verá más adelante, se puede tener una selección con poca o ninguna información, así como una selección con mucha o infinita información.

La naturaleza básica del fenómeno de comunicación consiste en la transferencia de la información de la fuente al destinatario. Como en el punto de vista del destinatario, los símbolos recibidos son imprevisibles y por tanto, para él, hay un carácter de aleatoriedad. Así, la cantidad de información recibida por el destinatario es el que permite cambiar su estado de inseguridad al respecto de cual símbolo será recibido, para el estado de seguridad, cuando el símbolo recibido fue identificado. Ahora, el nivel de inseguridad al respecto de la ocurrencia de un símbolo puede ser expresado por la "probabilidad de ocurrencia" de este símbolo. Esta probabilidad es fundamental para la medida de la cantidad de información que cada símbolo carga para el destinatario.

Siendo la cantidad de información una medida de inseguridad, ella está íntimamente asociada al número de símbolos disponibles en el proceso de la

comunicación. Cuanto mayor el número de símbolos disponibles en la fuente, mayor será el grado de inseguridad sobre cual símbolo será seleccionado. Así, la cantidad de información posible de ser generada indica el grado de libertad que esta fuente posee en el acto de selección.

Si todos los símbolos tienen igual probabilidad de ser seleccionados, esto es, si no hay preferencia por ningún símbolo, la fuente genera un máximo de información asociada a su conjunto de símbolos, pues posee el mayor grado de libertad posible en la selección. Si, por otra lado, hay probabilidades diferentes en el momento de escoger los símbolos, esto es, si hay preferencia para ciertos símbolos que sean seleccionados, el grado de libertad de selección de la fuente disminuye y por tanto, disminuye la cantidad media de información asociada que es posible de ser generada. Cuando la fuente selecciona un símbolo, esta implícito un proceso de escoger este símbolo.

Hay dos métodos básicos para realizar esta selección: o la fuente evidencia el símbolo directamente de su alfabeto (selección directa), o se divide el conjunto de símbolos en subconjuntos de la fuente, por medio de selecciones simples y sucesivas selecciona finalmente el símbolo adecuado. El mecanismo "Si o No", también simbolizado por "0 o 1" es el más fácil y confiable de ser realizado tecnológicamente. Como este mecanismo de selección solo tiene dos posibilidades, se denomina de selección binaria.

Para realizar una determinada selección se necesita de una cierta cantidad de información (punto de vista de la fuente). Al haber sido hecha la identificación de la selección efectuada se recibió esta información (punto de vista del destinatario). Un ejemplo, basado en un modelo simplificado de tránsito, permite una visualización más concreta de esas afirmaciones. Para tal se empleará la figura I.2.

Donde se representan diversos caminos, partiendo de un mismo punto y llevando a varios otros, habiendo puntos intermediarios de bifurcación en la entrada.

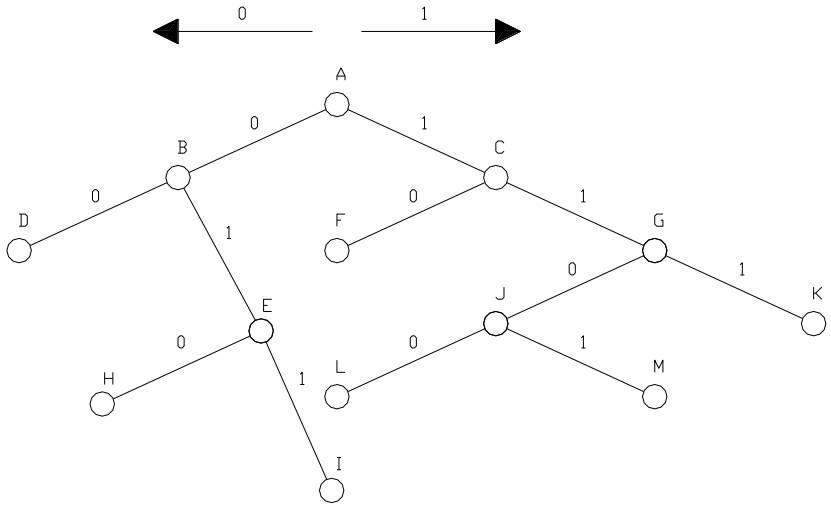


FIGURA. I.2 Modelo de información de tránsito

Si, partiendo de la ciudad **A**, se desea llegar a la ciudad **M**. Cada encrucijada, precisa de

cierta cantidad de información para hacer la selección del camino real. Sean las informaciones "tome la derecha" dada por el símbolo 1 y "No tome la derecha" dada por el símbolo 0. En **A** se recibe la información para la selección 1, en **C** para la selección 1, en **G** para la selección 0 y en **J** para la selección 1. Con esto se llega a M. En cada encrucijada se recibe un valor parcial de la cantidad de información.

Si se recibió en **A** la información total 1101, se va a llegar del mismo modo a **M**. Por lo que se deduce de esto, que la información total es la suma de las informaciones parciales.

Ahora, si el viajante (destinatario) recibe una orden (de una fuente) de viajar por las rutas simbolizadas por 010, por las informaciones parciales tendrá la "sorpresa" de saber que irá para la ciudad **H** (información total). Esta "sorpresa" de haber sido escogida la ciudad **H** y no otra, corresponde a una cierta cantidad de información recibida (punto de vista del destinatario).

Nótese que 010 es un símbolo que indica tres selecciones del camino de hacer en el viaje y cada selección corresponde una cantidad parcial de información. La suma de estos valores forman la cantidad de información total.

Entonces, para ser obtenida la cantidad de información correspondiente a una selección, basta cuantificar el número de selecciones que la fuente debe hacer para seleccionar el símbolo deseado y los valores de la cantidad de información correspondientes de cada selección.

1.3 VARIEDAD DE SÍMBOLOS

Se define variedad de un símbolo como el número total de configuraciones posibles, que

puede asumir este símbolo, definida a su estructura.

Al definir la fuente se refiere a los conceptos de elemento y alfabeto de elementos, así como de símbolo y alfabeto de símbolos.

Por ejemplo, el alfabeto de elementos dispone de *n* elementos y el símbolo está compuesto de una combinación de *m* elementos de entre los *n* citados. Ahora, el número de configuraciones posibles, agrupándose a los *n* elementos tomados *m* a *m* es:

$$N = n^m \qquad \textbf{(I.1)}$$

lo que quiere decir que, con la estructura definida en la ecuación I.1, se puede construir hasta *N* símbolos diferentes.

Por ejemplo, dado el alfabeto de elementos 1 y 0 se puede componer $2^5 = 32$ símbolos de 5 elementos, lo que permite representar todas las letras del alfabeto de la lengua escrita y este esquema es usado en el código del teleimpresor.

La ecuación I.1, indica que el número total de configuraciones posibles de los símbolos varía exponencialmente con la cantidad de elementos que lo constituyen. La ley exponencial, es de manejo incómoda e interesa que la medida de esta grandeza sea simple. Hartley[2] fue el primero al sugerir el empleo de la escala logarítmica, la misma que permite transformar la función exponencial en lineal. De esta forma, es más conveniente definir la variedad como:

$$\log_2 N = \log_2 n^m$$

$$v = \log_2 N = m \log_2 n \qquad \textbf{(I.2)}$$

y por ser n constante, v = k*m (ley lineal).

[2] R:V:L: HARTLEY. "Transmission of information" – Bell System Technical Journal.1928

El valor numérico de esta expresión depende de la base seleccionada para el logaritmo. Usualmente se emplea la base 2 y se dice que la variedad es expresada en una unidad de bit[3], o sea:

$$v = m\log_2 n \qquad \textit{[bit]} \tag{I.3}$$

Se explicará mejor la razón de esta selección si se considera el proceso de codificación. En el proceso de codificación se hace corresponder de una forma biunivoca, los símbolos del alfabeto de la fuente a los símbolos del alfabeto del codificador. Entonces si el alfabeto de la fuente emplea símbolos de m_1 elementos formados a partir de un alfabeto de n_1 elementos y si el codificador emplea símbolos de m_2 elementos formados a partir de un alfabeto de n_2 elementos, se tiene:

$$N = n_1^{m1} = n_2^{m2} \tag{I.4}$$

o sea:

$$v = m_1 * \log_2 n_1 = m_2 * \log_2 n_2 \tag{I.5}$$

[3] El término bit denota unidad de información.

La mejor solución tecnológica es aquella que adopta elementos que presentan 2 estados simples, por ejemplo: presencia o ausencia de pulso, pulso positivo o pulso negativo, etc. Tales estados pueden ser representados por los dígitos 1 y 0. En estas condiciones el receptor solo tiene que realizar una desición para reconocer la señal recibida (si es 1 o 0). Una señal más compleja, por ejemplo, con pulsos de tensión en varios niveles posibles, exigirá diversas decisiones por parte del receptor.

El alfabeto de símbolos más simple de ser producido es aquel en que los símbolos son formados por un elemento escogido de un alfabeto de dos elementos. Un símbolo de este tipo es usualmente designado como elemento binario y tendrá una variedad de:

$$v = 1\log_2 2 = 1\ bit$$

De esta forma, si los símbolos del codificador fueran elementos binarios ($n_2=2$), en la ecuación I.5, se tendrá:

$$v = m_1 * \log_2 n_1 = m_2 \quad [bits] \qquad \textbf{(I.6)}$$

Esto es, la variedad de un símbolo cualquiera es el número de elementos binarios que corresponde a su símbolo binario equivalente.

No se debe confundir el concepto de elemento binario con el de elemento simple. Los dígitos 0 y 1, las letras A, B, C,...,Z representan elementos simples. El elemento binario es aquel que puede presentar dos estados posibles, esto es, el compuesto por 2 elementos simples: puede ser el uno o el otro.

En una moneda, la "cara" es un elemento simple, la "cruz" es el otro elemento simple. La moneda está compuesta de dos elementos simples presentando solamente dos estados posibles y por tanto es un elemento binario. La variedad de una moneda es 1 bit.

Se puede formar un alfabeto de símbolos más complejos, usando símbolos compuestos de *m* elementos binarios, tal como se indica en la figura I.3.

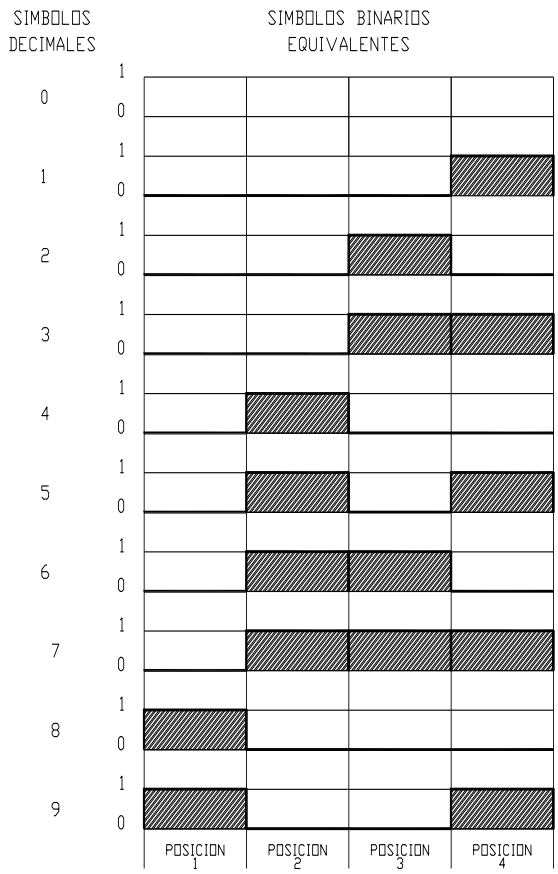


FIGURA. I.3 Alfabeto de 10 símbolos con n = 4 elementos binarios

Una fuente que dispone de símbolos de esta naturaleza es llamada de fuente binaria y la variedad asociada a sus símbolos es de:

$$v = m * \log_2 2 = m \text{ bits} \qquad \textbf{(I.7)}$$

Supóngase, por ejemplo, que los símbolos sean producidos por el lanzamiento de dos monedas. Cada moneda corresponde a un elemento binario, pudiendo asumir cualquiera de las dos configuraciones posibles (cara o cruz). El símbolo posee dos elementos binarios y puede asumir cualquiera de 2*2 = 4 configuraciones posibles. A cada elemento binario corresponde 1 bit de variedad al conjunto de 2 elementos binarios es lógico asociar 2 bits de variedad. De hecho, habiendo 4 configuraciones posibles:

$$v = \log_2 4 = 2\log_2 2 = 2 \text{ bits}$$

De un modo general, el lanzamiento de n monedas corresponden símbolos de *n* elementos binarios. La variedad de lanzamiento simultáneo de n monedas (símbolos) es n bits. Comprobando:

$$v = \log_2 2^n = n \text{ bits} \qquad \textbf{(I.8)}$$

o considerando cada moneda de por si:

$$v = \underbrace{1+1+.....+1}_{n} = n \text{ bits}$$

Estos ejemplos permiten inducir la llamada ley de adición de las variables:

$$v = \sum_i V_i \qquad \textbf{(I.9)}$$

que puede ser enunciada de la siguiente forma: si, a partir de un mismo alfabeto de elementos, es posible construir N_1 símbolos de m_1 elementos, al que corresponde la variedad v_1 y N_2 símbolos de m_2 elementos, al que corresponde v_2, es posible construir ($N_1 x N_2$) símbolos de (m_1+m_2) elementos, al que corresponde la variedad ($v_1+ v_2$).

Como la adición de elementos corresponde la adición de las variedades, se ve la ventaja de la adopción de la escala logarítmica para la medida de la variedad. Por lo tanto se tiene:

$$m_2 = m_1 * \log_2 n_1 \qquad \textbf{(I.10)}$$

o sea, la expresión de la variedad indica con cuantos elementos binarios(m_2) se pueden construir los símbolos codificados correspondientes a los símbolos originales.

Siempre que al referirse a la variedad asociada a una fuente se estará refiriéndose a su fuente binaria equivalente.

De un modo general, se consigue más rendimiento en la presentación de los símbolos aumentando el número de elementos constitutivos del símbolo que aumentando el número de elementos del alfabeto de elementos.

Cuanto más eficientemente se aumenta la variedad en el alfabeto de la fuente, mayor es el grado de libertad para la selección por la fuente y mayor es la disponibilidad en la generación de la información.

1.4 VELOCIDAD DE LA SEÑAL

La variedad mide una característica estática, inherente a la fuente, por su estructura de formación de los símbolos, es realmente la posibilidad de variación de los símbolos por unidad de tiempo la que interesa, porque ella mide la capacidad máxima de generación de información de esta fuente, necesaria para el dimensionamiento del canal.

La fuente para generar un mensaje, produce símbolos sucesivamente a lo largo del tiempo. La realización física de estos símbolos, es la señal introducida en el canal.

Esto define a la variedad V_s de una fuente, como la relación entre la variedad v de los símbolos producidos por la fuente y el intervalo de tiempo τ en que son producidas. Se tiene:

$$V_s = \frac{v}{\tau} bits/seg \qquad \textbf{(I.11)}$$

Si la fuente produce *S* símbolos a lo largo del tiempo *T*, a cada símbolo corresponde el intervalo de tiempo:

$$\tau = \frac{T}{S} \qquad \textbf{(I.12)}$$

por lo tanto de las ecuaciones I.11 y I.12, se tiene:

$$V_S = \frac{S * v}{T} bits/seg \qquad \textbf{(I.13)}$$

Si la variabilidad indica la velocidad con que una fuente selecciona sus símbolos, se dice también que ella los puede emitir con esta variabilidad y para el sistema de telecomunicaciones, esto caracteriza lo que está siendo transmitido. Si estos símbolos son señales eléctricas, la variabilidad de V_s de una fuente equivale a la velocidad de la señal V_s en el sistema.

En la realidad, la velocidad de la señal está referida a una escala binaria. De hecho, la variedad indica cuantos elementos la fuente binaria equivalente produce en correspondencia a cada símbolo de la fuente original. Si todavía cada elemento binario fuera representado físicamente por pulsos eléctricos, la velocidad de la señal puede también ser interpretada como la cantidad de pulsos eléctricos que la fuente binaria equivalente producirá en la unidad de tiempo.

Para el caso particular en que la señal considerada es de naturaleza binaria, esto es, formado por elementos binarios, se emplea también la expresión tasa de señalización binaria para designar esta velocidad en bits/seg.

1.5 BAUDIO

En telegrafía y en transmisión de datos, la señal eléctrica puesta en la línea es discreta, caracterizándose por tener en cada intervalo de tiempo, un valor discreto de amplitud de señal. Además, estos intervalos son usualmente de la misma duración.

Ahora, dado el intervalo τ, la banda de paso mínimo que el sistema debe poseer para permitir el reconocimiento de una secuencia de pulsos entrantes de duración τ tiene que dejar pasar, por lo menos, la fundamental del patrón donde haya transiciones a cada intervalo, esto es:

$$B = \frac{1}{2 * \tau} \qquad \textbf{(I.14)}$$

En la técnica telegráfica, las señales que eran transmitidas, llegan con una cierta distorsión, la misma que era función de la banda de paso del sistema. Por esto, la forma considerada más práctica para definir los requisitos de la banda de paso del sistema, era por medio de la llamada velocidad de modulación V_M, definida como el inverso del menor intervalo de tiempo presente en la señal, medida en la unidad referida como **bauds**[4].

$$V_M = \frac{1}{\tau} baud \qquad \textbf{(I.15)}$$

Para las aplicaciones telegráficas, el uso consagró la expresión de velocidad telegráfica para denominar esta grandeza. Por lo tanto al relacionar las dos ecuaciones I.14 y I.15 se tiene :

$$V_M = 2 * B \qquad \textbf{(I.16)}$$

En el estudio actual de la tecnología de transmisión de datos a través del canal telefónico, se usan señales de pulsos con hasta 16 niveles de percepción, conduciendo a la condición:

[4] Dado en homenaje a J.M.E. Baudot, oficial del servicio telegráfico francés, cuyas contribuciones fueron fundamentales en la técnica telegráfica.

$$\frac{V_s}{V_M} = 4\frac{bits/seg}{baud}$$

Por ejemplo si se tiene una señal de 9600 bits/seg equivale a producir una señal con una velocidad de modulación V_M = 2400 bauds.

Se puede generalizar el concepto de un modo general. Dado que la señal polibit tendrá n niveles, se puede relacionar V_S y V_M por la ecuación:

$$V_S = V_M * \log_2 n \qquad \textbf{(I.17)}$$

de las ecuaciones I.11 y I.15

$$V_S = \frac{v}{\tau} \qquad V_M = \frac{1}{\tau}$$

$$V_S = \frac{1}{\tau} v$$

$$V_S = V_M * v \; ; \qquad v = m * \log_2 n$$

reemplazando:

$$V_S = V_M * m \log_2 n$$

como es un pulso entonces m = 1, entonces se tiene:

$$V_S = V_M \log_2 n$$

Para un alfabeto binario en donde $n = 2$, la relación de la velocidad digital y la velocidad de modulación es:

$$V_S = V_M \log_2 2$$

$$V_S = V_M$$

lo que representa que el baudio es igual al bit cuando a cada pulso eléctrico se le hace coincidir un pulso binario.

Frecuentemente se dispone de un medio de transmisión cuyo ancho de banda B define una V_M máxima, inferior a las necesidades de velocidad de la señal para la comunicación de la información deseada, de modo de alcanzar las necesidades de transmisión de información con las posibilidades de transmisión de la señal en el medio.

Se puede entonces concluir que hay necesidad de adaptar la fuente al canal y éste al destinatario sobre el aspecto de la variabilidad, esto es, la velocidad con que estos niveles son modificados.

1.6 INFORMACIÓN Y SU MEDIDA

Como se indicó anteriormente, la *Teoría de la Información* es la disciplina que se encarga del estudio y cuantificación de los procesos que se realizan sobre la *información*. Por lo tanto, es obvio que se necesita de una manera de *medir* la *información*. Se tiene que pasar de nuestra intuición a una definición matemática que:

- esté de acuerdo con nuestra visión intuitiva
- nos permita medir y comparar los procesos sobre ella.

Para llegar a una medida de la información, es necesario repasar la idea intuitiva que se tiene de ésta. En primer lugar se ve que la cantidad de información que proporciona cierto dato es menor cuanto más se espera ese dato.

Si una persona comenta el clima que está en la playa, al decir "soleado" no se obtiene casi información, ya que es lo que se esperaba con mayor probabilidad. Si por el contrario se dice "lluvioso", se recibe más información, ya que la probabilidad de que esto ocurra es menor.

Se puede considerar las informaciones acerca del clima en la playa como datos que se recibe de cierta variable aleatoria que, cada vez que se pregunta, se dice el clima que hace en la playa. Esta variable aleatoria se convierte en una *fuente de información*. La *probabilidad de ocurrencia* de cierto evento entonces da información modelada por una variable aleatoria y su conjunto de mensajes y probabilidades asociadas. Para el caso anterior, se tiene, por ejemplo:

Tiempo	**Probabilidad**
Soleado	0.70
Lluvioso	0.30

Con este modelo se tendrá algo más concisa la idea de *información*: una fuente de información puede ser modelada como una variable aleatoria; al recibir un mensaje de esa fuente, se obtiene una cantidad de información, que depende ***sólo*** de la probabilidad de emisión de ese mensaje; además, la cantidad de información es una función creciente con la inversa de la probabilidad (cuanta menor probabilidad, mayor cantidad de información se recibe, como se vio en el ejemplo).

La *ganancia de información* que se experimenta, al recibir un mensaje, se puede entender como la *reducción de incertidumbre sobre el estado de la fuente* tras recibir el mensaje.

La última consideración que puede hacerse es una que, por un lado es intuitiva y por otro parece intencionada para llegar a la cuantificación final de la *información*. Se refiere a que la cantidad de información que se recibe con n mensajes de una fuente de m posibles mensajes debe ser la misma que al recibir *un* mensaje de una fuente con m^n mensajes. Esto sugiere una función logarítmica para la cantidad de información, ya que si se tiene dos fuentes equiprobables de m y m^n mensajes respectivamente, la información al recibir n mensajes de la primera fuente es:

$$n * f(m) \qquad \textbf{(I.18)}$$

donde *f* es la función creciente por ahora sin determinar (nótese que al ser la fuente equiprobable, la probabilidad de cualquier mensaje es *1/m*. Pero *f* es función de la *inversa* de la probabilidad, por lo que depende de *m*). Por otro lado, la recepción de un mensaje de la segunda fuente da una información:

$$f(m^n)$$

Igualando, se tiene que: $n*f(m) = f(m^n)$ sugiriendo como se dijo, una función logarítmica.

En 1948, **Claude Shannon** propuso una medida para la información que cumplía todas la expectativas anteriores, describiendo la información de un mensaje de una manera más científica como el logaritmo del número de posibles secuencias de símbolos que puedan haber sido seleccionadas y demostró que:

$$I = n * \log(N) \qquad \textbf{(I.19)}$$

Siendo:

n = número de símbolos del mensaje.
N = número de símbolos del alfabeto elegido.

Esta definición matemática se puede aceptar si los símbolos se eligen independientemente y si cualquiera de los *N* símbolos tiene igual probabilidad de ser elegido, o sea los símbolos del alfabeto son equiprobables.

En general se puede decir que si un suceso *x* que posee una probabilidad de aparición *P(x)* la información que se ha recibido es(en el siguiente capítulo se verá más detallado el concepto de cantidad de información):

$$I(x) = \log\left(\frac{1}{P(x)}\right) \qquad \textbf{(I.20)}$$

unidades de información. Siendo la base del logaritmo la que determina las unidades de información. Si se usa el logaritmo de base 2, esta medida estará en "*bits"*. Si se utiliza el logaritmo natural, se obtendrá "*nats"*. Y si se emplea el logaritmo de base 10, la unidad es el *"hartley",* en honor a R.V. Hartley, quien sugirió por primera vez el uso de las medidas logarítmicas.

Como la característica de equiprobabilidad es relativamente difícil de conseguir en la mayoría de las fuentes de la vida real, se puede hallar la cantidad de información para símbolo de la fuente como se definió anteriormente para un suceso aislado, pero para calcular la cantidad media de información de la fuente se debe recurrir a una medida nueva que es la cantidad media de información por símbolo de la fuente y que se denomina entropía H(x) de la fuente S, su definición se verá en el siguiente capítulo:

$$H(x) = \sum_{i=1}^{n} P(x_i) * I(x_i) \qquad \textbf{(I.21)}$$

Las unidades son bits / símbolo.

CAPÍTULO II

GENERACIÓN DE LA INFORMACIÓN

En la teoría de la información hay que diferenciar claramente el significado de dos palabras que en la vida cotidiana se usa con frecuencia e incluso como sinónimos sin percatarse de su importancia y su diferencia, estas dos palabras son *"comunicación e información"*. Se podría decir que la comunicación engloba a la información ya que siempre que hay paso de información hay comunicación pero no siempre que hay una comunicación hay percepción de información, es decir, se puede comunicar sin informar ya sea porque el receptor conoce el mensaje o porque este es indescifrable, sin embargo siempre que hay paso de información hay una comunicación debido a que el emisor transmite un mensaje desconocido para el receptor.

Como se explicó anteriormente el objetivo de la comunicación es el intercambio de información entre el emisor y el receptor. Así mismo se puede definir la información como la transmisión a un ser consciente de una idea, una significación, por medio de un mensaje más o menos convencional y por un soporte espacio-temporal.

Por lo tanto se entiende por *generación de la información* al proceso de selección de un símbolo particular dentro de un conjunto de símbolos llamados alfabeto, por ejemplo:

$$x = \{A, B, C, \ldots\ldots Z\} \quad \text{alfabeto de símbolos}$$

$$y' = \{0,1\} \quad \text{alfabeto binario}$$

El concepto de información supone la existencia de duda o incertidumbre. La incertidumbre implica que existen diferentes alternativas que deberán ser elegidas, seleccionadas o discriminadas. Las alternativas se refieren a cualquier conjunto de signos

construidos para comunicarse, sean estos letras, palabras, números, ondas, etc. En este contexto, las señales contienen información en virtud de su potencial para hacer elecciones. Estas señales operan sobre las alternativas que conforman la incertidumbre del receptor y proporcionan el poder para seleccionar o discriminar entre algunas de estas alternativas.

Existen diversos tipos de situaciones de elección. Las más sencillas son aquellas en que la fuente escoge entre un número de mensajes concretos. Otras situaciones más complejas son aquellas en que la fuente realiza una serie de elecciones sucesivas de un conjunto de símbolos elementales tales como letras o palabras. En este caso, el mensaje estará constituido por la sucesión de símbolos elegidos. El ejemplo más típico aquí es el del lenguaje.

2.1 ENTROPÍA

Al medir cuánta información proporciona la fuente al receptor al enviar un mensaje, se parte del supuesto que cada elección está asociada a cierta probabilidad, siendo algunos mensajes más probables que otros. Uno de los objetivos de esta teoría es determinar la cantidad de información que proporciona un mensaje, la cual puede ser calculada a partir de su probabilidad de ser enviada.

Por ejemplo, si X es un suceso cualquiera que se presenta con probabilidad $P(x)$. Cuando ocurre el suceso X, la cantidad de información $I(x)$ que se recibe depende de lo probable que sea ese suceso; a más probable es menos la información recibida.

Efectivamente, si se lanza un dado de 6 caras y se obtiene un 5 se recibe más información que si se lanza al aire una moneda y se obtiene cara. Esto es así porque en el caso del dado la probabilidad de obtener un 5 es P(5)=1/6 y en el caso de la moneda P(cara)=1/2. O dicho de otra forma, es más raro sacar un 5 con un dado que una cara con una moneda. Consecuentemente, la medida de la información está basada en la noción intuitiva de la misma palabra. Los acontecimientos más improbables proporcionan más información que los que son más probables.

La cantidad de información recibida respecto a la ocurrencia de un evento es inversamente proporcional a su probabilidad. Asimismo un suceso seguro proporciona información nula ya que es conocido que ocurrirá, (P(1)); y un hipotético suceso imposible daría infinita información (P(0)), (propiedades de la información).

La fórmula matemática que indica la cantidad de información *I* que da un suceso *X* (en bits o unidades de información) es la ecuación I.20 indicada en el capítulo anterior, es decir:

$$I(x) = \log\left(\frac{1}{P(x)}\right)$$

donde la base del logaritmo es 2 por tratarse de código binario.

Nótese que si $P(X) = ½$, se tendrá una cantidad de información $I(X) = 1$, es decir un 1 bit de cantidad de información al especificar una de las dos alternativas posibles igualmente probables, como lanzar una moneda.

Si la cantidad de Información $I(x_i)$ revelada por un resultado $X = x_i$, las propiedades deseables de $I(X)$ son:

- ***Si $P(x_i) = 1$,*** ***entonces $I(x_i) = 0$*** ***(certeza absoluta)***
- ***Si $P(x_i) = 0$,*** ***entonces $I(x_i) = \infty$*** ***(incertidumbre)***
- Si $0 < P(x_i) < 1$, entonces $I(x_i) > 0$
- Si $P(x) < P(y) < 1$, entonces $I(x) > I(y)$
- Si x e y son acontecimientos independientes, entonces I(x,y) = I(x)+I(y)

En definitiva la cantidad de información I(x) es una función continua de *p(x)*.

$$I_x = f(P(x))$$

$$I(X_i) = -\log_2 P(X_i) \qquad \textit{(unid. de inform.) ó [bits]} \qquad \textbf{(II.1)}$$

La misma expresión puede ser representada de la siguiente manera:

$$I(X_i) = \log_2 \left[\frac{1}{P(X_i)} \right]$$

La ecuación II.1 cumple con las propiedades anteriores; expresada en términos de logaritmo de base 10, queda:

$$I(X_i) = -3.3219 * \log_{10} P(X_i) \qquad \text{ó}$$

$$I(X_i) = 3.3219 * \log_{10} \left[\frac{1}{P(X_i)} \right]$$

En la Teoría de la Información la **entropía** es la magnitud que mide la información contenida en un flujo de datos, es decir, lo que aporta sobre un dato o hecho concreto. Por ejemplo, que digan que las calles están mojadas, sabiendo que acaba de llover, aporta poca información, porque es lo habitual. Pero si dicen que las calles están mojadas y se sabe que no ha llovido, aporta mucha información (porque no las riegan todos los días). Nótese que la cantidad de información es diferente, pese a tratarse del mismo mensaje: "Las calles están mojadas". En ello se basan las **técnicas de compresión de datos**, que permiten empaquetar la misma información en mensajes más cortos. La medida de la entropía puede aplicarse a información de cualquier naturaleza, y permite codificarla adecuadamente, indicando los elementos de código necesarios para transmitirla, eliminando toda redundancia.

La entropía indica el límite teórico para la compresión de datos. Su cálculo se realiza mediante la siguiente fórmula:

$$H = p_1 * log(1/p_1) + p_2 * log(1/p_2) + ... + p_m * log(1/p_m) \qquad \textbf{(II.2)}$$

donde *H* es la entropía, las *p* son las probabilidades de que aparezcan los diferentes códigos y *m* el número total de códigos. Si se refiere a un sistema, las *p* se refieren a

las probabilidades de que se encuentre en un determinado estado y *m* el número total de posibles estados.

Se utiliza habitualmente el logaritmo en base 2. La tabla II.1 muestra las unidades de la entropía, entonces la entropía se mide en bits/símbolo. Ejemplo: El lanzamiento de una moneda al aire para ver si sale cara o cruz (dos estados con probabilidad 0,5) tiene una entropía:

$H(x) = 0.5*\log_2(1/0.5)+0.5*\log_2(1/0.5)$

$H(x) = 0.5*\log_2(2)+0.5*\log_2(2)$

$H(x) = 0.5+0.5$

$H(x) = 1 \text{ bit/símbolo}$

TABLA. II.1 Unidades de la Entropía

BASE DEL LOGARITMO	UNIDAD
Decimal (10)	Hartley
Binario (2)	Bits
Número e	Nats

Si la presencia del símbolo *xi* se corresponde con una cantidad de información I(*xi*)=log(1/P(*xi*)), donde P(*xi*) es la probabilidad de que *xi* aparezca, se entiende que la cantidad media de información por símbolo de la fuente S es la siguiente:

$$H(S) = P(xi) * I(xi) \text{ bits/símbolo}$$

O, lo que es lo mismo:

$$H(S) = P(xi) * \log\left(\frac{1}{P(xi)}\right) \text{ bits/símbolo}$$

En ambos casos se extiende la sumatoria a los n símbolos del alfabeto

$$H(X) = \sum_{i=1}^{n} P(xi) * I(xi) \qquad \textbf{(II.3)}$$

$$H(X) = \sum_{i=1}^{n} P(xi) * \log\left(\frac{1}{P(xi)}\right)$$

De la expresión anterior se deduce que la entropía *H(x)* de una fuente significa el número medio de símbolos binarios (bits) que representan un símbolo, sea por ejemplo, si una fuente presenta una entropía *H(x) = 5 bits/símbolos* significa que, ese símbolo puede ser representado por 5 bits.

2.1.1 Propiedades de la Entropía

Se define Entropía como la cantidad de información media por símbolo que emite una fuente; se verá la relación entre la entropía y la probabilidad de los símbolos de la fuente, especialmente en los casos extremos. Ejemplo, si *S* es una fuente binaria tal que *p(0) = p y p(1) = 1 - p = q*: entonces la entropía será:

$$H(S) = -\left[p(0) * \log_2 p(0) + p(1) * \log_2 p(1)\right]$$

$$H(S) = -\left[p * \log_2 p + (1-p) * \log_2 (1-p)\right]$$

como se puede apreciar en la figura II.1, en donde se muestra algunas de las propiedades de la entropía, se ve que H(**X**) es cero cuando P(xi) = 0 ó P(xi) = 1 porque entonces la variable deja de ser aleatoria (variable que es capaz de asumir cualquier valor *xi*, de acuerdo con la probabilidad asociada *P(xi)*, ya que no hay incertidumbre sobre el valor que tomará **X**. Por otra parte la incertidumbre es máxima (difícil el determinar que símbolo se recibirá) cuando P(xi)=1/2, lo que consecuentemente coincide con el máximo valor de H(**X**) (esto ocurrirá cuando en la fuente los símbolos seleccionados sean equiprobables).

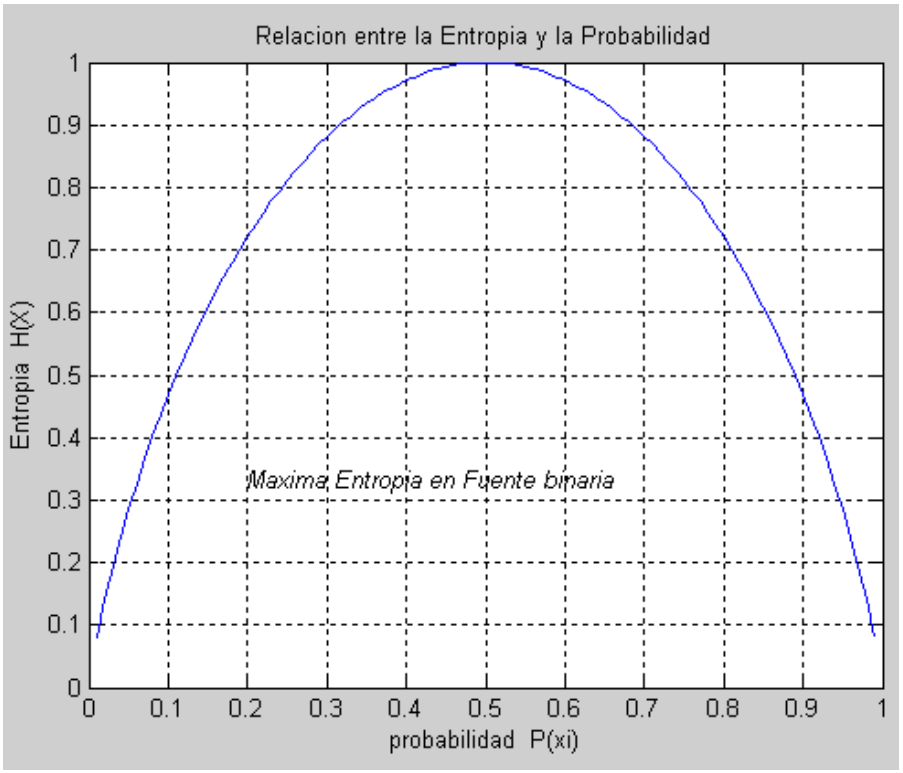


FIGURA. II.1 Relación entre la Entropía y la Probabilidad

Cuando los símbolos sean equiprobables, *P(x1) = P(x2) = ... = P(xn)* la entropía se determina directamente por la siguiente expresión:

$$H(X) = \log_2 n \qquad \Leftrightarrow \qquad H_{máx}(X) = \log_2 n \tag{II.4}$$

donde *n* es el número de símbolos.

La velocidad de la fuente está dada por:

$$V(s) = \frac{H(x)}{T}\left[bits/seg\right] \tag{II.5}$$

donde *H(x)* se evalúa con la ecuación II.3, y *T* es el tiempo requerido para enviar el mensaje.

Basado en la entropía, la teoría de la información muestra: Cómo calcular la probabilidad de cualquier caracter del alfabeto y predecir su mejor compresión, es decir, el mínimo número de bits necesarios, en promedio, para representar el caracter.

Ejemplo, dado el siguiente conjunto de símbolos emitidos por una fuente con sus respectivas probabilidades de ocurrencia A B C D E: 0.5, 0.2, 0.1, 0.1, 0.1., la cantidad de información será:

$$I(X_i) = \log_2(1/P(X_i))$$

$I(A) = \log_2(1 / 0.5) = 1$ bit
$I(B) = \log_2(1 / 0.2) = 2.3219$ bits
$I(C) = \log_2(1 / 0.1) = 3.3219$ bits
$I(D) = \log_2(1 / 0.1) = 3.3219$ bits
$I(E) = \log_2(1 / 0.1) = 3.3219$ bits

$$H(X) = \sum_{i=1}^{n} P(xi) * \log(1/P(xi))$$

$H(X) = 0.5*\log_2(1/0.5) + 0.2*\log_2(1/0.2) + 0.1*\log_2(1/0.1) + 0.1*\log_2(1/0.1) + 0.1*\log_2(1/0.1)$
$H(X) = 0.5 + 0.4643 + 0.3321 + 0.3321 + 0.3321$
$H(X) = 1.9606$ bits / símbolo

Dado el siguiente conjunto de símbolos emitidos por una fuente (S) AAAAABBCDE con sus respectivas probabilidades de ocurrencia A B C D E: 0.5, 0.2, 0.1, 0.1, 0.1., la cantidad de información será:

$P(S) = 5*0,5*\ 2*0,2*\ 0,1*\ 0,1*\ 0,1$
$P(S) = 1.25\ *10\text{-}6$
$I(X) = \log_2(1 / 1.25*10\text{-}6)$
$I(X) = 19.609 \approx 20$ bits

2.1.2 Redundancia

la redundancia se refiere a que las posibilidades dentro de un mensaje se repiten, y se repiten de una cierta manera predecible. Mientras mayor sea, entonces, la redundancia de un mensaje, menor será su incertidumbre y menor la información que contenga.

La redundancia de los idiomas permite que si se pierde una fracción de un mensaje sea posible completarlo en forma muy aproximada al original. Este hecho se puede observar al eliminar varias letras de una oración sin que ello impida al lector completar las omisiones y rehacer la oración. Por ejemplo, en la siguiente frase han sido omitidas las vocales:

CMPLT ST FRS ⇔ *mensaje original* ⇔ COMPLETE ESTA FRASE

Otra función importante de la redundancia es que permite ahorrar tiempo en la decodificación de los mensajes.

Resumiendo la redundancia de los mensajes permite, entonces, corregir con facilidad los errores u omisiones que hayan podido ocurrir durante la transmisión. La redundancia equivale a la reducción informativa respecto a la cantidad de información que podría haberse transmitido mediante la misma cantidad de señales si todas ellas hubieran sido elegidas como igualmente probables (información máxima de una fuente). Siendo 'H' la información efectiva de un mensaje y 'H0' la información máxima, la redundancia se expresa así:

$$R = \frac{H_O - H}{H_O} \qquad \textbf{(II.6)}$$

y se mide en un porcentaje. La redundancia asegura las condiciones de transmisión de un mensaje contrarrestando el *ruido*, es decir, las perturbaciones o distorsiones no intencionadas que afectan al *canal* (el sistema físico- técnico que sirve de vehículo a las señales). En un sentido más general que el puramente probabilística, y próximo al significado común de la palabra, la redundancia es una repetición tendente a hacer inteligible, o más fácilmente inteligible, un mensaje.

2.2 FUENTE DE SÍMBOLOS

En teoría de la información, se denomina a los emisores como fuentes de información (capaz de emitir o generar *n* símbolos), siendo una fuente de información un conjunto de símbolos que se denominará alfabeto de la fuente y unas probabilidades asociadas a esos símbolos siguiendo unas características de la fuente. Las fuentes de información emiten símbolos de acuerdo a las probabilidades asociadas a esos símbolos y según estas probabilidades asociadas a los símbolos estos aparecen más o menos frecuentemente en la generación de la fuente.

Una fuente de información es un elemento que entrega información, como pueden ser una persona hablando, un ordenador entregando datos... La visión de la persona hablando (por ejemplo), puede servir para ver los elementos más importantes en la emisión de la información. La información viaja sobre la voz de la persona (como una onda de presión), que es el soporte de la información. Pero es el hombre quien emite la voz, y es el hombre la verdadera *fuente de información*.

Esto se puede formalizar con unas definiciones más rigurosas. Una **fuente de información** es un elemento que entrega una señal, y una **señal** es una *función* de una o más *variables* que contiene información acerca de la naturaleza o comportamiento de algún fenómeno. Es decir, se considera señal tanto al fenómeno físico que transporta la información como a la función matemática que representa a ese fenómeno. Cualquiera de las dos formas sirve como soporte a la información.

Las fuentes de información se clasifican basándose en el tipo de señal que entregan. Se pueden clasificar, *según el tipo de variable independiente* (tiempo) en:

- **Fuentes de tiempo continuo**: la función está definida para cualquier valor de la variable independiente.
- **Fuentes de tiempo discreto**: la función sólo está definida para un conjunto contable de instantes de tiempo.

Pero se pueden clasificar también *según el rango de valores* que cubren las señales. En este caso los tipos de fuentes de información serán:

- **Fuentes continuas o de amplitud continua**: el valor de la función toma un rango continuo de valores.
- **Fuentes discretas o de amplitud discreta**: el valor de la función sólo toma un conjunto finito de valores. A cada uno de estos valores se lo llamas *símbolo*. El conjunto de todos los símbolos se suele llamar *alfabeto*. La elección del alfabeto

  es, en cierto modo, arbitraria, ya que se puede variar los símbolos para crear otros, por ejemplo.

En la práctica sólo se encuentran dos tipos: las llamadas fuentes analógicas, que son fuentes continuas de tiempo continuo; y las llamadas fuentes digitales, que son fuentes discretas de tiempo discreto.

Las fuentes digitales se suelen clasificar según la relación que tenga un símbolo con los que le preceden de la siguiente manera:

- **Fuentes sin memoria**: los símbolos son estadísticamente independientes entre sí. De esta manera, los símbolos que hayan aparecido hasta el momento no van a condicionar al símbolo presente ni a posteriores.
- **Fuentes con memoria**: la aparición de los símbolos no es estadísticamente independiente. Es decir, si han aparecido *n-1* símbolos, el símbolo *n-ésimo* está condicionado por los anteriores.

De lo anterior se puede definir que fuente es un componente de naturaleza humana o mecánica que determina el tipo de mensaje que se transmitirá y su grado de complejidad.

2.2.1 Fuente de información de memorias nulas (FIMN)

Una fuente de información de memoria nula es aquella en la que los símbolos son independientes estadísticamente, es decir, la aparición de un símbolo no depende de la aparición de otros símbolos y tampoco limita la aparición de otros símbolos. Son las fuentes de información más sencillas que se recogen en la teoría de la información.

La descripción matemática de una fuente de información se puede representar gráficamente como en la figura II.2.

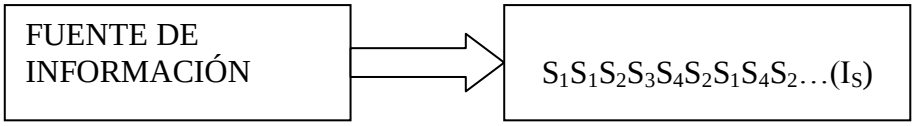


FIGURA. II.2 Fuente de información típica

Esta fuente emite secuencias de símbolo pertenecientes a un conjunto completo denominado *Alfabeto Fuente*: $S = \{S_1, S_2, S_3, \ldots, S_q\}$

Donde *q* es el número de símbolos de la fuente. Una fuente de información, además de estar caracterizada por el alfabeto fuente, también lo está por la probabilidad con que se emite cada símbolo. Sí una fuente emite secuencias como la (I_S) de la figura II.2, donde el número de símbolos es muy grande cumpliéndose

$$P(S_i S_j) = P(S_i)P(S_j)$$
$$P(S_i S_j S_k) = P(S_i)P(S_j)P(S_k)$$

Entonces los símbolos S_i , S_j y S_k son estadísticamente independientes y se dice que la fuente es de memoria nula**. Esto es posible solo en virtud de la probabilidad condicional y se expresa como:**

$$P(\frac{Si}{Sj}) = P(S_i) \qquad \text{(II.7)}$$

Una fuente de memoria nula queda descrita completamente mediante el alfabeto de la fuente y las probabilidades de ocurrencia de cada símbolo: S_1 con $P(S_1)$, S_2 con $P(S_2)$,... y S_q con $P(S_q)$

2.2.1.1 Entropía de la fuente de memoria nula

Teniendo en cuenta que:

q cantidad de símbolos de la fuente

N Número de símbolos emitidos

con N>>q

la información del símbolo ***j*** será:

$$I_j = \log_2 \frac{1}{P_j} \qquad \textbf{(II.8)}$$

La información total del mensaje será:

$$Np_1I_1 + Np_2I_2 + \ldots\ldots\ldots + Np_qI_q = \sum_{j=1}^{q} Np_jI_j \text{ Shanon} \qquad \textbf{(II.9)}$$

La información promedio por símbolo será:

$$H(S) = \frac{1}{N}\sum_{j=1}^{q} Np_jI_j = \sum_{j=1}^{q} p_jI_j = \sum_{j=1}^{q} p_j \log_2 \frac{1}{p_j} \text{ Shanon/ símbolo}$$

(II.10)

Es de notar que H es un promedio del conjunto de símbolos. Sí, la fuente no es estacionaria, las probabilidades de los símbolos pueden cambiar con el tiempo y la entropía no es significativa.

- Un proceso se dice estacionario cuando las leyes de distribución coinciden independientemente del corte en el tiempo en que se observan, esto es: $p(x,t_1)\ p(x,t_2) = p(x)$.

En teoría de la Información se consideran las fuentes de información ergódicas, tal que el promedio del conjunto y del tiempo son idénticos.

- Un proceso aleatorio estacionario se considera ergódico cuando es posible calcular cualquier promedio estadístico a partir do un promedio equivalente en el tiempo de una realización cualquiera

2.2.1.2 Intervalo de valores de la entropía de una fuente de q símbolos

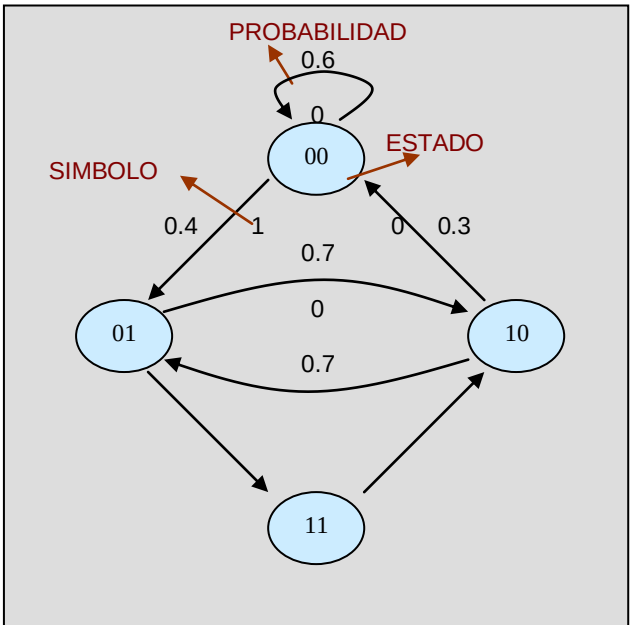
Ya se ha manifestado que la información no puede ser negativa y la peor información se da cuando el evento tiene la probabilidad máxima de ocurrencia, es decir cuando no hay incertidumbre. Esto es cuando algún símbolo de la fuente asume toda la probabilidad de ocurrencia, haciendo que los demás símbolos nunca ocurran. En este caso la entropía será H(S) = 0.

$$0 \leq H \leq \log q$$

2.2.2 Fuentes de información con memoria o fuentes MARKOV

Una fuente de información de Markov es aquella en la que la aparición de un símbolo depende de la aparición anterior de un número m determinado de símbolos anteriores, lo que significa que la probabilidad de aparición de un símbolo está condicionada a la aparición anterior de otros símbolos. Por eso la fuente se denomina fuente de Markov de orden m. Se puede indicar la situación de la fuente en cualquier momento indicando el estado en el que se encuentra, definiéndose este estado por los m símbolos precedentes, teniendo en cuenta que el estado puede cambiar con la emisión de cada símbolo. Para estudiar la fuente de información se puede realizar un diagrama de estados relacionados entre sí por unas transiciones entre sí que indican las probabilidades de pasar de un estado a otro de la fuente.

Donde la sumatoria de los estados es: $P(00)+P(01)+P(10)+P(11)=1$, para el caso de la figura siguiente.



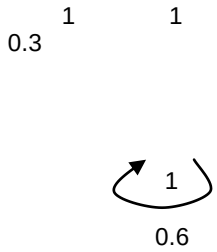


FIGURA II.3 Diagrama de estados

Se define la entropía conjunta de *X,Y*:

$$H(X,Y)=\sum_{i=1}^{I}\sum_{j=1}^{J}P(x_{i,}y_j)\log_2\frac{1}{P(x_i,y_j)} \qquad \textbf{(II.11)}$$

Propiedad. La entropía conjunta es aditiva para variables aleatorias

$H(X,Y)=H(X)+H(Y) \quad \Leftrightarrow P(x_i,y_j)=P(x_i)P(y_j) \quad \forall\, i=1...I; j=1...J$ independientes:

Se define la entropía condicional de *X* dado *Y=yj*:

$$H(X/Y=y_j)=\sum_{i=1}^{I}P(x_i/Y=y_j)\log_2\frac{1}{P(x_i/Y=y_j)} \qquad \textbf{(II.12)}$$

Se define la entropía condicional de *X* dado *Y* como la media de los valores de la entropía condicional de *X* dado *Y=yj*:

$$H(X/Y)=\sum_{j=1}^{J}P(y_j)\sum_{i=1}^{I}P(x_i/Y=y_j)*\log_2\frac{1}{P(x_i/Y=y_j)}$$

$$H(X/Y)=\sum_{i=1}^{I}\sum_{j=1}^{J}P(x_i,y_j)*\log_2\frac{1}{P(x_i/Y=y_j)} \qquad \textbf{(II.13)}$$

Generalizando la entropía para una fuente Marcov está definida por la siguiente expresión:

$$H(S) = \sum P(Sj_1, Sj_2, ..., Sj_m, Si) * \log_2 \frac{1}{P(Si / Sj_1, Sj_2, ..., Sj_m)} \qquad \textbf{(II.14)}$$

Hay que tener en cuenta que estos dos tipos de fuentes de memoria pueden admitir extensiones de orden n que son agrupaciones de los símbolos de la fuente original en paquetes de símbolos de tamaño n.

En ambos tipos de fuente definidos anteriormente se puede calcular la entropía, sin embargo en las fuentes de memoria de Markov se presenta dificultad para calcular las probabilidades de los estados.

2.3 PROBABILIDADES

Un experimento se llama aleatorio si su resultado no puede predecirse con exactitud porque las condiciones bajos las cuales se realiza no pueden ser determinadas completamente y con suficiente exactitud.

Lanzar una moneda al aire, arrojar un dado y sacar una carta de un mazo son algunos ejemplos de experimentos aleatorios. Un experimento aleatorio puede tener varios resultados posibles identificables.

El concepto de probabilidad se usa para medir (numéricamente) los resultados favorables de determinado experimento.

De acuerdo con la definición clásica, la probabilidad *P(A)* de un evento A es determinada por:

$$P(A) = \frac{N_A}{N} \qquad \textbf{(II.15)}$$

Donde: N es el posible número de resultados, y N_A es el número de resultados que son
favorables al evento A. Sin embargo, la definición clásica adolece de ambiguedades ya que el significado de los N y N_A no es claro.

Por lo tanto, la definición clásica es modificada a una versión mejorada como sigue: "**La probabilidad de un evento:** es igual a la relación de sus resultados favorables para el número total de posibles resultados permitiendo que todos los eventos sean igualmente probables".

La definición puede ser aplicada a una limitada clase de problemas. Si en el caso de un dado se carga una de sus caras, evidentemente que la condición de equiprobabilidad no se cumple.

Para que la medida sea válida, es necesario examinar antes las bases de la medición. Por ejemplo si se efectúa un experimento cuyo resultado no es constante, uno de los posibles resultados se marca *A*. Lanzar una moneda es un experimento cuyos posibles resultados son cara o cruz. Si el experimento se repite N veces, se supone que el resultado A ocurrirá N_A veces, "la frecuencia de ocurrencia" de A es N_A / N. Esta razón no es muy predecible cuando N es pequeño. Sin embargo, si el experimento tiene regularidad estadística, la frecuencia relativa[5] de un resultado particular puede tender a un límite cuando el número de repeticiones N se hace muy grande. Este valor límite se llama **probabilidad del resultado *A*** y se escribe *P(A)*:

$$P(A) = \lim_{N \to \infty} \left(\frac{N_A}{N} \right) \qquad \textbf{(II.16)}$$

[5] Frecuencia relativa es la relación entre el número de veces que aparece un dato en una sucesión de datos respecto al total de resultados observados.

El límite en esta ecuación no lo es en el sentido funcional usual pero se usa para indicar que si N es muy grande, la desviación esperada de la razón N_A /N de una constante, es muy pequeña. Esto se llama en ocasiones " ley empírica de los números grandes " y concuerda con la idea intuitiva de probabilidad. Por ejemplo si se lanza una moneda un número grande de veces, se espera que las caras aparezcan en cerca de la mitad de los lanzamientos y que esta aproximación se hace mayor mientras más lanzamientos se efectúen.

De la definición de probabilidad, proporcionada mediante la ecuación II.15, se ve que todas las funciones de probabilidad tienen la propiedad:

$$0 \leq P(A) \leq 1$$

donde: P(A) = 0 si el evento A es nulo (no ocurre) y,

P(A) = 1 si el evento A es seguro (siempre ocurre)

De la propiedad anterior, se tiene:

$$\sum_{i=1}^{n} P(Xi) = 1$$

es decir que la probabilidad de *P(Xi)* es un número positivo que varía entre 0 y 1.

Para la probabilidad conjunta y condicional, considere los posibles eventos *A* y *B*. Estos eventos pueden o no ocurrir juntos (si son mutuamente excluyentes, no pueden). ***La probabilidad de ocurrencia conjunta de A y B*** es: *P(AyB)* o, más simplemente, *P(AB)* (equivale a la notación de conjuntos P(A∩ B). Supóngase que el experimento se repite *N* veces y que N_{AB} es el número de veces que *A* y *B* ocurren juntos. Usando el concepto de frecuencia relativa, la probabilidad conjunta de su ocurrencia es:

$$P(AB) = \lim_{N\to\infty}\left(\frac{N_{AB}}{N}\right) \qquad \textbf{(II.17)}$$

Recuérdese que N_A es el número de veces que ocurre A y N_B el número de veces que ocurre B. Como A y B pueden no ocurrir juntos siempre, se tiene que:

$$N_{AB} \leq N_A \Leftrightarrow N_{AB} \leq N_B$$

Pueda ser que la ocurrencia del evento A dependa de alguna manera de la ocurrencia de B. La probabilidad de ocurrencia de A dado que se sabe que ha ocurrido B se llama ***probabilidad condicional de A dado B,*** se escribe ***P(A|B),*** y se define como:

$$P(A \mid B) = \lim_{N_B \to \infty} \left(\frac{N_{AB}}{N_B} \right) \qquad \textbf{(II.18)}$$

Del concepto de frecuencia relativa para las N_A pruebas en que ocurra A, y se tiene:

$$P(A \mid B) = \frac{N_{AB}}{N_B} = \frac{N_{AB}/N}{N_B/N} = \frac{P(AB)}{P(B)} \qquad \textbf{(II.19)}$$

de forma similar:

$$P(B \mid A) = \frac{N_{AB}}{N_A} = \frac{N_{AB}/N}{N_A/N} = \frac{P(AB)}{P(A)} \qquad \textbf{(II.20)}$$

de la combinación de las ecuaciones II.19 y II.20, la probabilidad conjunta *P(AB)* puede expresarse como:

$$P(AB) = P(A \mid B) * P(B) = P(B \mid A) * P(A) \qquad \textbf{(II.21)}$$

o:

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)} \qquad \textbf{(II.22)}$$

la ecuación II.22 se denomina **TEOREMA DE BAYES.**

2.4 CAPACIDAD DE CANAL

2.4.1 Introducción

Los resultados básicos de la teoría de la información son los que relacionan la capacidad de transmisión del canal con el contenido de información presente en el mensaje a ser transmitido. La capacidad de transmisión de un canal es función del poder de recuperación de los símbolos por el receptor al final del proceso de

transmisión. Las señales portadoras de información al transmitirse están sujetas a distorsión, ruidos y las alteraciones sufridas, pueden venir a modificar en el punto de recepción, la interpretación de las señales para la recuperación correcta del símbolo.

Por otro lado, el mensaje es producido por la sucesión ordenada de los símbolos y las fuentes generalmente presentan predilección en la selección de los símbolos que va a usar. La información es asociada a la probabilidad de ocurrencia de cada símbolo y a cada mensaje se asocia una cierta cantidad de información de cada símbolo constituyente.

El canal debe ser dimensionado para transmitir la peor configuración posible de la señal (símbolos), independientemente de como éste se relaciona con el contenido de la información que transporta. El análisis del contenido de la información de la señal permitirá mostrar cuan eficiente es el mecanismo de producción de esta señal. Si la eficiencia es baja, se estará empleando mucha señal para poca información y, en consecuencia, el canal[6] será dimensionado de forma poco económica, en términos de la información que va a transmitir.

Dado un canal con una capacidad de *C* unidades por segundo que recibe señales de una fuente de información de *H* unidades por segundo, se dice que la capacidad de un canal es la medida de la cantidad de información que un canal puede transferir por unidad de tiempo, sus unidades son: $^{bits}/_{seg}$.

Son estas dos cantidades, la tasa de transmisión *H* por la fuente de información y la capacidad *C* del canal, 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.

[6] Canal definido como modelo para expresar el vehículo; conjunto de medios materiales de la transmisión y todos los fenómenos que tienden a restringir la transmisión de un punto A a un punto B.

Si $H \leq C$ será posible transmitir la información con eficiencia. La información, entonces, puede ser transmitida por el canal solamente si H no es mayor que C.

2.4.2 Teorema de Capacidad máxima de un canal

En 1928 Harry Nyquist, un investigador en el área de telegrafía, publicó una ecuación llamada la *Razón Nyquist* que media la razón de transmisión de la señal en bauds. " *La razón de Nyquist es igual a 2B símbolos (o señales) por segundo"*, donde B es el ancho de banda del canal de transmisión. Así, usando la expresión anterior, el ancho de banda de un canal telefónico de 3.000 Hz puede transmitir hasta 6.000 bauds o Hz.

Claude Shannon[7] después de la investigación de Nyquist estudio como el ruido afecta a la transmisión de datos. Shannon tomo en cuenta la razón *señal-a-ruido(S/N)* del canal de transmisión (medido en decibeles o dB) y derivó el teorema de Capacidad de Shannon.

$$C = B * \log_2\left(1 + \frac{S}{N}\right) \Rightarrow bps \qquad \textit{Ley de Shannon} \qquad \textbf{(II.23)}$$

Por ejemplo , un típico canal telefónico de voz tiene una razón de señal a ruido de 30 dB y un ancho de banda de 3.000 Hz. Si se sustituye esos valores en el teorema de Shannon:

$$\#dB = 10\log_{10}\left(S/N\right)$$

$$30dB = 10\log_{10}\left(S/N\right)$$

$$3 = \log_{10}\left(S/N\right)$$

$$S/N = 1000$$

$$C = B * \log_2\left(1 + \frac{S}{N}\right) \Rightarrow bps$$

$$C = 3000 * \log_2\left(1 + 1000\right)$$

$$C = 3000 * \log_2\left(1001\right)$$

$$C = 3000 * \frac{\log_{10}(1001)}{\log_{10}(2)}$$

$$C = 29901678\ bps$$

[7] Claude E. Shannon, reconocido como creador de la teoría de la Información. Físico americano especialista en telecomunicaciones que formuló en 1948 los elementos fundamentales de dicha teoría dentro de un artículo titulado "Una teoría matemática de la comunicación"

lo que indica que la capacidad máxima de un canal telefónico es aproximadamente 30.000 bps.

Debido a que los canales de comunicación no son perfectos, ya que están delimitados por el ruido y el ancho de banda. "El teorema de Shannon - Hartley dice que *es posible transmitir información libre de ruido siempre y cuando la tasa de información no exceda la Capacidad del Canal"* [8].

Así, si el nivel de *S/N* es menor, o sea la calidad de la señal es más cercana al ruido, la capacidad del canal disminuirá. Esta capacidad máxima es inalcanzable, ya que la fórmula de Shannon supone unas condiciones que en la práctica no se dan. No tiene en cuenta el ruido impulsivo[9], ni la atenuación ni la distorsión. Representa el límite teórico máximo alcanzable.

Ejemplo: ¿Cuanto nivel de S/N se requiere para transmitir sobre la capacidad del canal telefónico, 56.000 bps?. De la fórmula de Shannon:

$$C = B * \log_2\left(1+\frac{S}{N}\right) \Rightarrow bps$$

$$\frac{C}{B} = \frac{\log_{10}\left(1+S/N\right)}{\log_{10}(2)}$$

$$0.301029 * \frac{56.000}{3.000} = \log_{10}\left(1+S/N\right)$$

$$5{,}619 = \log_{10}\left(1+\frac{S}{N}\right)$$

$$5{,}619 = \log_{10}\left(\frac{S}{N}\right)$$

$$\left(\frac{S}{N}\right) = 415.910{,}61$$

$$\#dB = 10\log_{10}\left(S/N\right)$$

$$\#dB = 10\log_{10}(41591061)$$

$$\#dB = 56{,}19$$

Lo que significa que si se quiere rebasar el límite de Shannon se debe aumentar el nivel de S/N.

[8] Channel capacity by Shannon

[9] Ruido impulso es aquel se produce por una corta duración de tiempo, dados por picos cortos de energía, aparece por descargas eléctricas, procesos de conmutación, los mismos que producen errores en ráfaga.

CAPÍTULO III

CODIFICACIÓN DE LA INFORMACIÓN

3.1 TRANSMISIÓN DE LA INFORMACIÓN

El término datos, se refiere a la información que puede haber sido tomada de documentos originales, como pedidos de venta, registro de producción, entre otras; de algún medio de almacenamiento, como son las cintas magnéticas o la memoria de una computadora. El traslado de estos datos entre máquinas situadas a cierta distancia es la ***transmisión de datos.***

3.1.1 Tipos de Transmisión de Datos

3.1.1.1 Transmisión Análoga

En un sistema analógico de transmisión se tiene a la salida de éste una cantidad que varía continuamente.

En la transmisión analógica, la señal que transporta la información es continua, en la señal digital es discreta. La forma más sencilla de transmisión digital es la binaria, en la cual a cada elemento de información se le asigna uno de dos posibles estados.

Para identificar una gran cantidad de información se codifica un número específico de bits, el cual se conoce como caracter. Esta codificación se usa para la información escrita. Ejemplo: Teletipo = Servicio para la transmisión de un telegrama.

La mayor de las computadoras en servicio hoy en día utiliza u opera con el sistema binario por lo cual viene más la transmisión binaria, ya sea de terminal a computadora o de computadora a computadora.

3.1.1.2 Transmisión Digital

En la transmisión digital existen dos notables ventajas lo cual hace que tenga gran aceptación cuando se compara con la analógica. Estas son:

- El ruido no se acumula en los repetidores.
- El formato digital se adapta por si mismo de manera ideal a la tecnología de estado sólido, particularmente en los circuitos integrados.

La mayor parte de la información que se transmite en una red portadora es de naturaleza analógica, por ejemplo: La voz y el vídeo.

Al convertir estas señales al formato digital se pueden aprovechar las dos características anteriormente citadas.

Para transmitir información digital (binaria 0 ó 1) por la red telefónica, la señal digital se convierte a una señal analógica compatible con la el equipo de la red y esta función se realiza en el Módem.

Es ventajoso transmitir datos en forma binaria en vez de convertirlos a analógico. Sin embargo, la transmisión digital está restringida a canales con un ancho de banda mucho mayor que el de la banda de la voz.

3.1.1.3 Transmisión Asíncrona.

Ésta se desarrolló para solucionar el problema de la sincronía y la incomodidad de los equipos.

En este caso la temporización empieza al comienzo de un caracter y termina al final, se añaden dos elementos de señal a cada caracter para indicar al dispositivo receptor el comienzo de este y su terminación.

Al inicio del caracter se añade un elemento que se conoce como "Start Space" (espacio de arranque), y al final una marca de terminación.

Para enviar un dato se inicia la secuencia de temporización en el dispositivo receptor con el elemento de señal y al final se marca su terminación.

3.1.1.4 Transmisión Sincrónica.

Este tipo de transmisión se caracteriza porque antes de la transmisión propia de datos, se envían señales para la identificación de lo que va a venir por la línea, es mucho mas eficiente que la Asíncrona pero su uso se limita a líneas especiales para la comunicación de ordenadores, porque en líneas telefónicas deficientes pueden aparecer problemas.

Por ejemplo una transmisión serie es síncrona si antes de transmitir cada bit se envía la señal de reloj y en paralelo es síncrona cada vez que se transmite un grupo de bits.

3.1.1.5 Transmisión de Datos en Serie

En este tipo de transmisión los bits se trasladan uno detrás del otro sobre una misma línea, también se transmite por la misma línea.

Este tipo de transmisión se utiliza a medida que la distancia entre los equipos aumenta a pesar que es más lenta que la transmisión paralelo y además menos costosa. Los transmisores y receptores de datos serie son más complejos debido a la dificultad en transmitir y recibir señales a través de cables largos.

La conversión de paralelo a serie y viceversa se lleva a cabo con ayuda de registro de desplazamiento.

La transmisión serie es *síncrona* si en el momento exacto de la transmisión y recepción de cada bit, está determinada antes de que se transmita y reciba, y *asíncrona* cuando la temporización de los bits de un caracter no depende de la temporización de un caracter previo.

3.1.1.6 Transmisión en Paralelo.

En la transmisión de datos en paralelo cada bit de un caracter se transmite sobre su propio cable. En la transmisión de datos en paralelo hay un cable adicional en el cual se envía una señal llamada strobe ó reloj; esta señal le indica al receptor cuando están presentes todos los bits para que se puedan tomar muestras de los bits o datos que se transmiten y además sirve para la temporización que es decisiva para la correcta transmisión y recepción de los datos.

La transmisión de datos en paralelo se utiliza en sistemas digitales que se encuentran colocados unos cerca del otro, además es mucho más rápida que la serie, pero además es mucho más costosa.

3.1.2 Medios de transmisión.

Los medios de transmisión pueden ser:

- ***Guiados*** si las ondas electromagnéticas van encaminadas a lo largo de un camino físico.
- ***No guiados*** si el medio es sin encauzar (aire, agua, etc..).

La forma de transmisión puede ser:

- ***Simplex*** si la señal es unidireccional;

- ***Half-duplex*** si ambas estaciones pueden trasmitir pero no a la vez;
- ***Full-duplex*** si ambas estaciones pueden transmitir a la vez.

Los campos donde pueden aplicarse ventajosamente la comunicación de datos han aumentado de tal modo y son tantos que es más seguro describirlos en términos generales.

3.1.3 Ventajas de la Transmisión Digital

La transmisión digital tiene el problema de que la señal se atenúa y distorsiona con la distancia, por lo que cada cierta distancia hay que introducir repetidores de señal.

Últimamente se utiliza mucho la transmisión digital debido a que:

- La tecnología digital se ha abaratado mucho.
- Al usar repetidores en vez de amplificadores, el ruido y otras distorsiones no es acumulativo.
- La utilización de banda ancha es más aprovechada por la tecnología digital.
- Los datos transportados se pueden encriptar y por tanto hay más seguridad en la información.
- Al tratar digitalmente todas las señales, se pueden integrar servicios de datos analógicos (voz, vídeo, etc..) con digitales como texto y otros.

3.1.4 Perturbaciones en la transmisión

3.1.4.1 Atenuación

La energía de una señal decae con la distancia , por lo que hay que asegurarse que llegue con la suficiente energía como para ser captada por la circuitería del receptor y además, el ruido debe ser sensiblemente menor que la señal original (para mantener la energía de la señal se utilizan amplificadores o repetidores).

Debido a que la atenuación varía en función de la frecuencia, las señales analógicas llegan distorsionadas, por lo que hay que utilizar sistemas que le devuelvan a la señal sus características iniciales (usando bobinas que cambian las características eléctricas o amplificando más las frecuencias más altas).

3.1.4.2 Distorsión de retardo

Debido a que en medios guiados, la velocidad de propagación de una señal varía con la frecuencia, hay frecuencias que llegan antes que otras dentro de la misma señal y por tanto los diferentes componentes en frecuencia de la señal llegan en instantes diferentes al receptor. Para atenuar este problema se usan técnicas de ecualización.

3.1.4.3 Ruido

El ruido es toda aquella señal que se inserta entre el emisor y el receptor de una señal dada. Hay diferentes tipos de ruido:

- ***Ruido Térmico*** debido a la agitación térmica de electrones dentro del conductor.
- ***Ruido de Intermodulación*** cuando distintas frecuencias comparten el mismo medio de transmisión.
- ***Diafonía*** se produce cuando hay un acoplamiento entre las líneas que transportan las señales.
- ***Ruido Impulsivo*** se trata de pulsos discontinuos de poca duración y de gran amplitud que afectan a la señal .

3.2 CODIFICACIÓN: DETECCIÓN Y CORRECCIÓN DE ERRORES

Los principales conceptos a desarrollar en la Teoría de la Información son:

- La medida de la información.
- La capacidad de un canal de comunicación para transferir la información

- La codificación como medio de utilizar el canal a plena capacidad.

Estos tres conceptos básicos se encuentran reunidos en lo que puede llamarse *"Teorema fundamental de la teoría de la información"*.

La codificación se emplea para adaptar la fuente al canal para una transferencia de información con un máximo de confiabilidad.

Por tal razón la información, para poder ser transmitida, necesita ser adaptada al medio de transmisión. Para ello, generalmente, será preciso codificarla de tal forma que pueda asegurarse una recepción adecuada y segura.

Si se tiene la información en un determinado alfabeto fuente y se desea transformarla a otro alfabeto destino, se puede definir codificación como a la realización de dicha transformación, siendo el código la correspondencia existente entre cada símbolo del alfabeto fuente y el conjunto de símbolos del alfabeto destino.

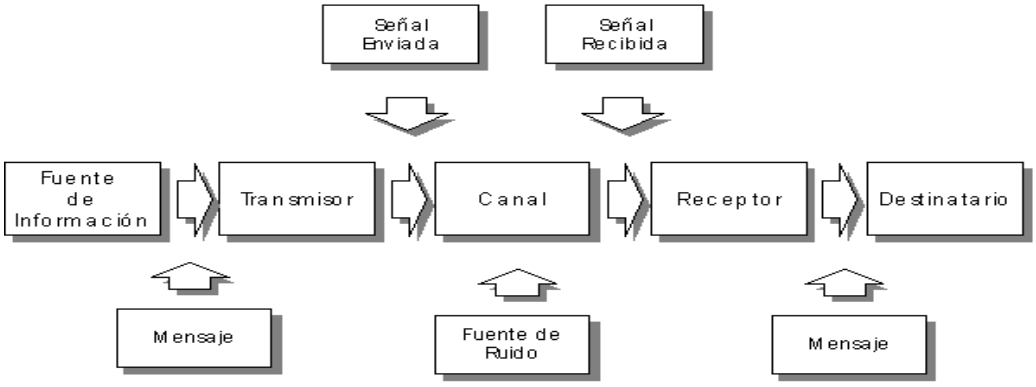


FIGURA III.1 Modelo de comunicación

3.2.1 Codificación.

La codificación consiste en establecer una correspondencia entre cada uno de los símbolos de un alfabeto fuente y una secuencia de símbolos de una alfabeto destino. Al alfabeto destino se le denomina alfabeto código y a cada una de las secuencias de símbolos de este alfabeto que se corresponda con un símbolo del alfabeto fuente se denomina palabra de código.

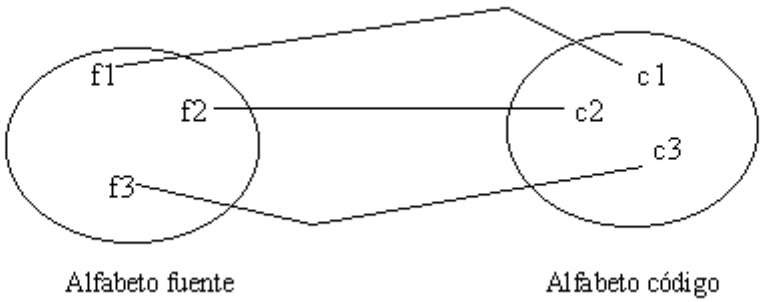


FIGURA III.2 Correspondencia de alfabeto fuente y alfabeto código

El alfabeto fuente contiene los símbolos originales que se quiere codificar. El alfabeto código contiene las palabras de código equivalentes en que se codificarán los símbolos originales. Estas palabras de código son aptas para ser transmitidas por un sistema de comunicaciones.

Se tiene 3 tipos de codificación: *codificación en la fuente, codificación de compresión y codificación del canal*.

3.2.2 Propiedades de los códigos [10]:

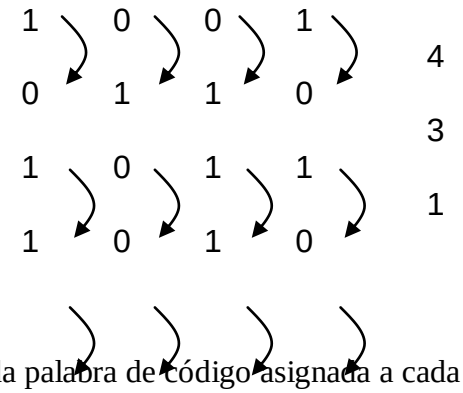
Antes de comenzar con la descripción de algunos de estos códigos es conveniente dar unas definiciones y ciertas propiedades de los códigos:

Tasa de error: Se define como la relación entre el número de bits erróneos recibidos respecto al número total de bits transmitidos. Una tasa de error aceptable para una transmisión es 10^{-6}.

Tasa residual de error: Se define como la relación entre el número de bits erróneos no detectados sobre el total de bits emitidos. Mide la capacidad de detectar errores.

Peso de Hamming: El peso de Hamming *H(c)* de una palabra de código ***c*** se define como el número de bits de esa palabra diferentes de cero.

Distancia de Hamming: Es la distancia entre dos palabras de código de igual longitud y se define como el número de bits (posición a posición) en los que se diferencian las dos palabras. Como se puede observar la distancia mínima de Hamming es 1 para las siguientes secuencias de bits dadas.



Longitud media: Cada palabra de código asignada a cada símbolo del alfabeto fuente tiene una longitud I_K . A partir de aquí se define la longitud media de un código como:

[1] http://www.isa.cie.uva.es/proyectos/codec/teoria1.html

$$\bar{L} = \sum_{K=1}^{n} p_K * I_K \qquad \textbf{(III.1)}$$

Donde: I_K representa el número de dígitos asignados a cada probabilidad y p_K es la probabilidad. La longitud media representa el número medio de bits por símbolo del alfabeto fuente que se utilizan en el proceso de codificación.

Eficiencia: A partir del concepto de longitud media la eficiencia de un código se define como:

$$\eta = \frac{L_{\min}}{\bar{L}} \qquad \textbf{(III.2)}$$

Siendo: $\bar{L} \geq L_{\min}$, $L_{\min}$ representa el valor mínimo de la longitud media ($\bar{L}$).

Para calcular L_{min} es necesario tener en cuenta el primer teorema de Shanon o teorema de la codificación de la fuente: Dada una fuente discreta de entropía *H*, la longitud media de la palabra de código está acotada inferiormente por *H*. Teniendo esto en cuenta L_{min} se fija como el valor de la entropía con lo que la eficiencia puede escribirse como:

$$\eta = \frac{H}{\bar{L}} \qquad \textbf{(III.3)}$$

Redundancia: Se denomina redundancia de un código a la información superflua o innecesaria para interpretar el significado de los datos originales. Se define como:

$$\sigma = 1 - \eta \qquad \textbf{(III.4)}$$

También es necesario hacer una diferenciación entre los distintos tipos de códigos:

Códigos sistemáticos: aquellos códigos en los que la palabra de información aparece de forma explícita en la palabra codificada.

Códigos no sistemáticos: aquellos códigos en los que la palabra de información no aparece de forma explícita en la palabra codificada.

Códigos de bloque: (tienen el mismo significado que en el caso de la codificación de la fuente) aquellos códigos en los que todas las palabras tienen la misma longitud y la codificación se hace de forma estática. Dentro de estos códigos se distinguen:

- ***Código singular***: a cada símbolo del alfabeto fuente le corresponde una única palabra de código.

- ***Código no singular***: a cada símbolo del alfabeto fuente le corresponde uno o más palabras de código.

Códigos lineales: aquellos en los que cualquier combinación lineal de palabras de código válida (por ejemplo la suma módulo $2^{[11]}$) produce otra palabra válida.

Códigos cíclicos: aquellos en los que cualquier desplazamiento cíclico de una palabra de código da lugar a otra palabra de código.

3.2.3 Codificación de la fuente:

El objetivo de la codificación es obtener una representación eficiente de los símbolos del alfabeto fuente. Para que la codificación sea eficiente es necesario tener un conocimiento de las probabilidades de cada uno de los símbolos del alfabeto fuente.

El dispositivo que realiza esta tarea es el codificador de la fuente. Este codificador debe cumplir el requisito de que cada palabra de código debe decodificarse de forma única, de forma que la secuencia original sea reconstruida perfectamente a

[2] Aritmética de módulo 2 sólo existen dos cantidades, 0 y 1. se puede sumar y multiplicar como si fueran enteros ordinarios, excepto que 1 + 1 = 0.

partir de la secuencia codificada. Se tienen los siguientes códigos de codificación en la fuente:

3.2.3.1 Código BCD

Uno de los primeros códigos utilizados para representar datos en notación binaria para poder ser manejados por una computadora fue el código BCD (Binary Coded Decimal), esta técnica de codificación permite que un conjunto de caracteres alfanuméricos pueda ser representado mediante 4 bits.

3.2.3.2 Código EBCDIC

Este código surge como una ampliación del código BCD. En las transmisiones de datos es necesario utilizar un gran número de caracteres de control para la manipulación de los mensajes y realización de otras funciones. De ahí que el código BCD se extendiera a una representación utilizando 8 bits dando origen al código EBCDIC (Extended Binary Coded Decimal Interchange Code).

3.2.3.3 Código FIELDATA

Es un código utilizado en transmisiones de datos de algunos sistemas militares y está orientado al lenguaje máquina

3.2.3.4 Código ASCII

ASCII son las siglas de American Standar Code for Information Interchange. Su uso primordial es facilitar el intercambio de información entre sistemas de procesamiento de datos y equipos asociados y dentro de sistemas de comunicación de datos.

En un principio cada carácter se codificaba mediante 7 dígitos binarios y fue creado para el juego de caracteres ingleses más corrientes, por lo que no contemplaba ni caracteres especiales ni caracteres específicos de otras lenguas. Esto hizo que posteriormente se extendiera a 8 dígitos binarios

3.2.4 Códigos detectores y correctores de error

Existen tres funciones fundamentales que se realizan con sistemas digitales: *almacenamiento de información, transmisión de información y procesamiento de información*. Al mover los bits de un lugar a otro para cualquiera de estas funciones, se pueden producir errores en la información. La principal causa de estos errores es la presencia del ruido eléctrico, que consiste en fluctuaciones en el voltaje o corriente en un sistema eléctrico y se presenta a manera de picos. Este ruido eléctrico se encuentra presente en todos los sistemas eléctricos en mayor o menor medida.

Suponga un sistema de comunicación, en el que el transmisor envía a través de la línea una señal binaria sin ruido. Cuando dicha señal llega al receptor, la señal original tiene sumada una señal de error. En ocasiones, esta señal de error es lo suficientemente grande como para alterar el nivel lógico de la señal. Cuando esto ocurre el receptor interpreta incorrectamente un bit.

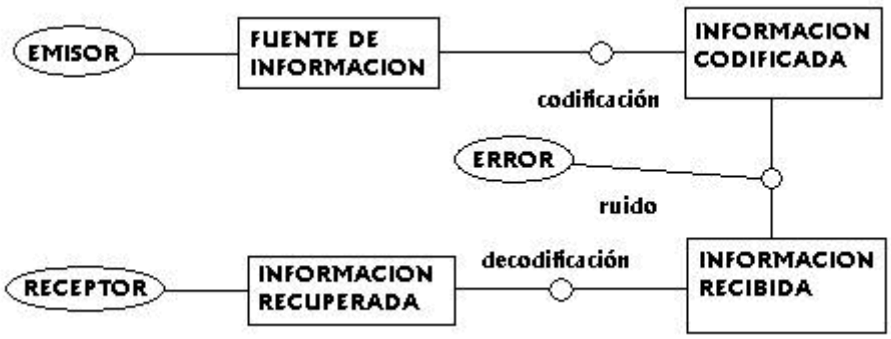


FIGURA III.3 Proceso de la transmisión de la información con error

Por esta razón, muchos sistemas digitales utilizan métodos para detectar o corregir errores. Antes de analizar estos métodos de detección o corrección de errores, es necesario entender algunas definiciones.

En primera instancia están los Tipos de Palabra. En términos generales, la ***palabra*** se refiere a la información que se desea codificar junto con el código que se está utilizando. Se puede decir que la palabra es la información codificada.

Palabra de Código. Combinación de bits que representa un símbolo válido en el código.

Palabra de Error. Combinación de bits que representa un símbolo no válido en el código.

Como se puede apreciar, sólo se tienen dos tipos de palabra, o bien ésta pertenece al código o se trata de un error. El error se produce cuando el receptor interpreta incorrectamente algún bit, esto es que un bit cambió su valor.

Al tratarse de bits, estos sólo pueden tener 2 valores, por lo que en ocasiones también se dice que el bit se invirtió. Dentro de lo que son los errores existen 3 tipos:

Error Detectable. El cambio en los bits produce una palabra que no existe en el código.
Error Indetectable. El cambio en los bits produce una palabra que si existen en el código.

Error Corregible. El cambio en los bits produce una palabra que sólo puede asociarse con una palabra de entre todas las del código. Cuando se produjo un error en el que se formó una palabra que si existe en el código, el sistema leería esa palabra como si no existiera el error.

Un concepto que tiene gran relevancia en lo que son los códigos detectores y correctores de errores es el de la "*distancia*".

Distancia entre dos combinaciones. Es el número de bits que se necesitan cambiar para convertir una combinación en la otra.

Distancia Mínima, Natural o de código. Es la distancia mínima de entre todas las distancias posibles entre 2 palabras del código. Se le conoce como " ***d*** "**.**

Para calcular la distancia de código, es necesario calcular la distancia que existe entre todas las posibles combinaciones de 2 en 2 en el código y después encontrar el mínimo. De hecho, es la distancia ***d*** la que determina el comportamiento del código en cuánto a los errores que se pueden detectar o corregir.

Código Detector. Un código puede detectar *i* errores si y sólo si su distancia es:

$$d \geq i+1 \qquad \textbf{(III.5)}$$

Estos códigos detectores están basados en no aprovechar todas las combinaciones posibles de ceros y unos para formar palabras pertenecientes al código. Así, si se produce algún error en la transmisión, la palabra resultante podría no pertenecer al código y de esta forma se determinaría que se ha producido un error.

Por ejemplo, si se tiene una cierta palabra de código formada por los dígitos $b_{n-1}\ b_{n-2}...b_j...b_1\ b_0$. asuma que también que al ser transmitido el dígito binario j-ésimo sufre alteración. La combinación recibida en el destino sería: $b_{n-1}\ b_{n-2}...\neg b_j...b_1\ b_0$, donde se tiene que $\neg b_j$ es el complementario del dígito b_j. En este caso se verá que la combinación recibida no pertenece al código puesto que dista de la primera una unidad pudiendo concluirse que ha habido un error. Si en lugar de un error hubiese habido dos, por ejemplo en el j-ésimo y en el i-ésimo dígitos la combinación recibida en el destino sería $b_{n-1}\ b_{n-2}...\neg b_j...\neg b_i...\ b_1\ \ b_0$ que pertenecería al código, con lo cual no se podría concluir que ha habido error.

$b_{n-1}\ b_{n-2}...\ b_j...\ b_1\ b_0$ llega Ok

$b_{n-1}\ b_{n-2}...\neg b_j...\ b_1\ b_0$ llega error

$b_{n-1}\ b_{n-2}...\neg b_j...\neg b_i...\ b_1\ b_0$ llega Ok

Visto el ejemplo se puede decir que la condición necesaria y suficiente para que un código sea detector de error en un dígito binario, es que su distancia mínima sea dos. De igual forma, aplicando un razonamiento similar combinado con el método de inducción completa, se llegaría a probar que: *la condición necesaria y suficiente para detectar un error en k dígitos binarios, es que la distancia mínima sea de i+1.*

Código Corrector. Un código puede corregir i errores si y sólo si su distancia es:

$$d \geq 2i + 1 \qquad \textbf{(III.6)}$$

Los códigos vistos en el apartado anterior están diseñados para detectar el error en un dígito binario, pero no se puede determinar en qué dígito binario se ha producido. La razón se encuentra en que su distancia mínima es dos. Si se recibiese una combinación con un dígito erróneo se la compararía con las palabras que sí pertenecen al código para hallar la más parecida, esto es, la más cercana. En estas circunstancias, aparece la posibilidad de encontrar dos combinaciones equidistantes en una unidad. Consecuentemente, no se podría precisar el dígito erróneo.

Por ejemplo: Supóngase que (b) y (c) pertenecen al código. Si en una transmisión se recibe (a), el código está pensado para suponer que se ha producido un error en un dígito. Al comparar (a) con (b) y (c), resulta que dista de estas una unidad, no pudiéndose determinar el dígito que ha sufrido variación. Lo mismo se puede decir al comparar con la combinación (d), obteniéndose los mismos resultados.

TABLA III.1: Comparación de código

(a) 00	Error en un dígito binario.

(b) 01	Pertenece al código.
(c) 10	Pertenece al código.
(d) 11	Error en un dígito binario.

Como se puede ver, entre mayor sea la distancia que presenta un código, mayor es el número de errores que se pueden detectar o corregir y a fin de cuentas, la distancia del código depende de los bits de paridad que se añadan.

Bit de Paridad. Es un bit adicional que se añade a la información para hacer que el número total de unos o ceros sea par o impar respectivamente. Su propósito es ampliar la distancia del código.

Estos códigos se forman a partir de uno de distancia mínima uno o superior, al cual se le añade el denominado dígito de paridad (normalmente como dígito binario más significativo). Este dígito deberá tomar el valor apropiado (0 o 1) para que el número de unos de la combinación resultante sea par o impar dependiendo de si se quiere construir un código de paridad par o un código de paridad impar.

A continuación un sencillo ejemplo aclaratorio de esto:

TABLA III.2: Bits de paridad par o impar

Códigos de partida:	0001	0000	1001

Códigos de paridad par:	10001	00000	01001
Códigos de paridad impar:	00001	10000	11001

Nota: Se debe aclarar que el que no haya ningún '1' en la combinación se lo considera como un número par de unos.

3.3 CÓDIGOS LINEALES

En los códigos de bloques, un bloque de ***k*** dígitos de datos se codifican mediante una palabra de código de ***n*** dígitos ($n>k$). Para cada sucesión de *k* dígitos de datos, existe una palabra de código distinta de *n* dígitos.

Si k dígitos se transmiten mediante una palabra de código de n dígitos, el número de dígitos de comprobación es $m = n - k$. Este código se conoce como (n,k).

Considere un código de bloques lineales (n,k) en el cual la primera posición de los bits *k* es siempre idéntico a la secuencia del mensaje que se transmiten. Los bits $n-k$ en la segunda posición son computarizados desde los bits del mensaje de acuerdo con la regla de codificación que determina la estructura matemática de los códigos.

De acuerdo a esto, estos bits $(n-k)$ son referidos como " bits de chequeo de paridad generalizada " o simplemente **bits de paridad.** Los códigos de bloques en el cual los bits del mensaje son transmitidos en forma inalterada son llamados **códigos sistemáticos.** Para aplicaciones que requieren detección de error y corrección de error, el uso de códigos de bloques sistemáticos simplifican la implementación del decodificador.

Si $m_0\, m_1 \ldots m_{k-1}$ constituyen un bloque de *k* bits de mensaje arbitrario. Entonces se tiene 2^k bloques de mensajes distintos. Por lo tanto esta secuencia de bits de mensaje es aplicada a un codificador de bloques lineales, produciendo una palabra codificada de n-bit cuyos elementos son denotados por $x_0\ x_1 \ldots x_{n-k-1}\, x_{n-k}\, x_{n-1}$. Así $b_0\, b_1 \ldots b_{n-k-1}$ denotan los bits de paridad (*n-k*, agregados como redundancia) en la palabra codificada.

Para que el código posea una estructura sistemática, una palabra código es dividida en dos partes, una de las cuales es ocupada por los bits del mensaje y la otra por los bits de paridad. Se tiene la opción de enviar los bits del mensaje de la palabra código antes de los bits de paridad, o viceversa. La opción formal es indicada en la figura III.4, y su uso es asumido en la secuencia.

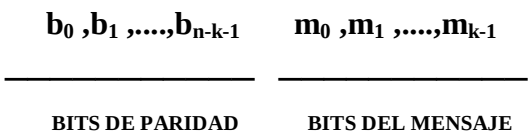


FIGURA III.4 Estructura de la palabra código

De acuerdo a la representación de la figura III.4, se puede escribir:

$$x_i = \begin{cases} b_i \ldots\ldots\ldots\ldots\ldots \Rightarrow i = 0,1,\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, n-k-1 \\ m_{i+k-n} \ldots\ldots\ldots \Rightarrow i = n-k, n-k+1,\ldots\ldots\ldots\ldots\ldots\ldots, n-1 \end{cases} \qquad \textbf{(III.7)}$$

Estas ecuaciones pueden ser reescritas en forma compacta usando una notación de matriz. Para proceder con esta reformulación, hay que definir el vector de mensaje (*1*,*k*) **m**, el vector paridad (*1*,(*n-k*)) **b** y el vector código (*1*,*n*) **x** como sigue respectivamente.

$$\boldsymbol{m = [\, m_0 \;\; m_1 \;\; .... \; m_{k-1} \,]} \qquad \textbf{(III.8)}$$

$$\boldsymbol{b = [\, b_0 \;\; b_1 \;\; .... \; b_{n-k-1} \,]} \qquad \textbf{(III.9)}$$

y

$$\boldsymbol{x = [\, x_0 \; x_1 \;\; .... \;\; x_{n-1} \,]} \qquad \textbf{(III.10)}$$

Note que los tres vectores son vectores de una fila común. De acuerdo a esto se puede escribir el set de ecuaciones simultáneas en una forma de matriz compacta:

$$\boldsymbol{b = m.P} \qquad \textbf{(III.11)}$$

donde **P** es el coeficiente de la matriz *k* por (*n*-*k*) definido por:

$$P = \begin{pmatrix} P_{00} & P_{01} & \cdot & \cdot & \cdot & P_{0,n-k-1} \\ P_{10} & P_{11} & \cdot & \cdot & \cdot & P_{1,n-k-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ P_{k-1,0} & \cdot & \cdot & \cdot & \cdot & P_{k-1,n-k-1} \end{pmatrix} \qquad \textbf{(III.12)}$$

De las definiciones dadas por las ecuaciones de **m** (III.8) y **x** (III.10), se ve que **x** podría ser expresado como un vector fila particionado en términos de vectores **m** y **b** como sigue:

$$\boldsymbol{x = [b \mid m]} \qquad \textbf{(III.13)}$$

Entonces sustituyendo el valor de **b** en la ecuación anterior y factorando la salida del vector mensaje **m**, se tiene:

$$\boldsymbol{x = m[P \mid I_k]} \qquad \textbf{(III.14)}$$

donde I_k es el (k,k) de la matriz identidad:

$$I_k = \begin{matrix} 1 & 0 & ..... & 0 \\ 0 & 1 & ..... & 0 \\ . & . & . & . \\ . & . & . & . \\ 0 & 0 & ..... & 1 \end{matrix}$$

Para realizar la ***codificación*** se utiliza la matriz generadora G (k,n), y que consta de una partición de una matriz identidad I_k (k,k) y una matriz P (k,m):

$$\boldsymbol{G = [P \mid I_k]} \qquad \textbf{(III.16)}$$

De esta forma se puede obtener cada palabra de código a partir de cada palabra de mensaje original realizando la siguiente multiplicación:

$$\boldsymbol{x = m \cdot G} \qquad \textbf{(III.17)}$$

Para realizar la decodificación, en el destino se recibe un vector (x) de tamaño n y lo que se puede hacer es repetir la operación realizada en la codificación: se toman los primeros k bits y se calcula la redundancia usando la matriz generadora y se comprueba si la redundancia obtenida es igual a la redundancia recibida.

Otra opción más eficiente es la basada en el concepto de síndrome. En el proceso de decodificación basado en el síndrome se utiliza la ***matriz de chequeo de paridad***, que se define de la siguiente forma:

$$\boldsymbol{H = [I_{n-k} \mid P^t]} \qquad \textbf{(III.18)}$$

donde P^t es una matriz de ((n-k),k), representando la transpuesta del coeficiente de la matriz P, y I_{n-k} es la matriz identidad de [(n-k),(n-k)].

H ((*n*-*k*),*n*) tiene la propiedad de que sólo las palabras de código verifican que al multiplicarlas por H^t el resultado es el vector nulo. Esta propiedad será utilizada para la detección y corrección de errores. A cada palabra que el receptor recibe a través del canal se la denominará palabra recibida y se la denominará con ***r***. Una palabra recibida se la puede expresar como:

$$\boldsymbol{r = x + e} \qquad \textbf{(III.19)}$$

Donde *x* es la palabra de código enviada por el emisor y *e* es una palabra de error. Cada componente e_i de la palabra de error llamado vector error o patrón de error podrá valer 1 si hay un error en esa posición y 0 si no lo hay.

El receptor, para realizar la codificación utiliza la matriz *H* para calcular el vector de síndrome de error a partir de la palabra recibida. El vector de síndrome de error se obtiene de la siguiente forma:

$$\boldsymbol{s = r \cdot H^t} \qquad \textbf{(III.20)}$$

El vector de síndrome tiene tantos elementos como bits de paridad se estén usando. El vector de síndrome sólo depende de la secuencia de error y no de la palabra de código transmitida.

Si en la transmisión no se ha producido un error, el síndrome es el vector nulo:

$$\boldsymbol{s = r \cdot H^t = 0} \qquad \textbf{(III.21)}$$

Si se ha producido un error la multiplicación de la palabra recibida por H^t da un vector que es igual a una de las filas de H^t. La posición que ocupa esa fila es la posición donde hay un error. Todas estas operaciones se hacen en módulo-2 [12] .(sin acarreo).

3.4 CÓDIGOS HAMMING, REED MULLER

[12]Teoría y práctica moderna de las comunicaciones digitales. Degem System.

3.4.1 CÓDIGO HAMMING

Un código de Hamming es un código de bloque lineal. Estos códigos son empleados en aplicaciones donde ocurre un solo error por bloque, por ejemplo en el proceso de escritura y lectura de una memoria RAM.

Al igual que en los códigos de bloque lineales sistemáticos, se puede denotar un código de Hamming mediante un par (*n*,*k*).

Considere una familia de código de bloque lineal de (7,4) que tienen los siguientes parámetros:

Longitud del bloque:	$n = 2^m - 1$
Número de bits del mensaje:	$k = 2^m - m - 1$
Número de bits de paridad:	$n - k = m$

donde $m \geq 3$. Estos son los llamados códigos Hamming. Todos los códigos Hamming tienen una distancia Hamming de 3, por lo que puede detectar dos errores o corregir uno.

Considere, por ejemplo, el código Hamming (7,4) con n = 7 y k = 4, correspondiendo para m = 3. La matriz generadora de el código debe tener una estructura que conforme la ecuación III.16. La siguiente matriz representa la matriz generadora apropiada para el código Hamming (7,4).

$$G = \left[\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \qquad \text{(III.22)}$$

P $\qquad$ I_K

la correspondiente matriz de chequeo de paridad esta dada por:

$$H = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right] \qquad \text{(III.23)}$$

I_{n-k} $\qquad$ P^T

con k = 4, hay $2^k = 16$ distintas palabras de mensaje como se indica en la siguiente tabla III.3:

TABLA III.3: Código de palabras de un Código Hamming de (7,4)

PALABRA DEL MENSAJE	CÓDIGO DE PALABRA	PALABRA DEL MENSAJE	CÓDIGO DE PALABRA
0000	0000000	1000	1101000
0001	1010001	1001	0111001

0010	1110010	1010	0011010
0011	0100011	1011	1001011
0100	0110100	1100	1011100
0101	1100101	1101	0001101
0110	1000110	1110	0101110
0111	0010111	1111	1111111

Para una palabra de mensaje dado, el correspondiente código de palabra es obtenido usando la ecuación III.17. Entonces la aplicación de esta ecuación resulta en 16 palabras códigos listadas en la tabla III.3.

Los códigos Hamming tienen la propiedad que la distancia mínima es $d_{min} = 3$, independientemente del valor asignado a **m**.

Este código satisface exactamente la frontera de Hamming, y se puede construir un código apropiado. En este caso, m=3 y hay siete síndromes diferentes de cero, y como n=7, hay exactamente siete patrones de un solo error. Por lo tanto, se pueden corregir todos los patrones de un solo error y nada más.

Si los siete síndromes son distintos, es posible decodificar todos los patrones de un solo error. Esto significa que el único requisito sobre H^T (transpuesta de H) es que sus siete renglones sean distintos y diferentes de cero. Observe que H^T es una matriz $(n,(n-k))$, es decir, en este caso (7*3). Ya que existen siete patrones diferentes de cero de tres dígitos, es posible encontrar siete renglones diferentes de cero de tres dígitos cada uno. Existen muchas maneras en las cuales se pueden

ordenar renglones. Pero se debe recordar que los tres renglones inferiores deben formar la matriz identidad I_k , tal como se indica en la ecuación III.23.

Asumiendo un patrón de error simple, se podría desarrollar confiablemente las entradas presentadas en la tabla III.4 para la decodificación.

En la tabla III.4, cada patrón de error simple es asociado con un síndrome único que corresponde a la columna particular de la matriz **H**, El síndrome cero significa ningún error de transmisión.

TABLA III.4: Decodificación para el código Hamming definido en la tabla III.3

SÍNDROME	PATRÓN ERROR
000	0000000
100	1000000
010	0100000
001	0010000
110	0001000
011	0000100
111	0000010
101	0000001

3.4.2 CÓDIGO REED MULLER

Son una familia de códigos que cubre un amplio rango de tasas y distancias mínimas. Para cualquier valor de m, y fijando un r < m, hay un código Reed-Muller con $n=2^m$,

$$\sum_{i=0}^{r} \frac{m}{i} \quad \text{y} \quad \text{dmin}=2^{(\text{m-r})} . \qquad \textbf{(III.24)}$$

La matriz generadora de r – ésimo orden se define como sigue.

V0 = vector con n *unos.*

V1, V2, V3, ..., Vm: son las filas de una matriz con todas las posibles m-tuplas como columnas. Por tanto, las filas de la matriz G serán V0, V1, ..., Vm y todos los productos cruzados de 2, 3, ..., r filas.

Por ejemplo, para m=3, se puede elegir un código bien (8,4) si r=1, con dmin=4, o bien uno (8,7) si r = 2, con dmin = 2.

La matriz generadora de estos dos códigos es la siguiente

TABLA III.5:Matriz generadora para m = 3

R=1	V0 V1 V2 V3	1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1	Vector de unos Todas las combinaciones De tres elementos En binario
R=2	V1*V2 V1*V3 V2*V3	0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0	Todos los productos cruzados (vi,vj) hasta r=2

3.5 CÓDIGOS CÍCLICOS.

Los códigos cíclicos forman una subclase de los códigos lineales de bloque. Un procesamiento para elegir una matriz generadora es relativamente fácil para los códigos de corrección de un solo error. Los códigos cíclicos contienen una cantidad razonable de estructura matemática que permite diseñar códigos de corrección de órdenes mayores. En segundo lugar, para los códigos cíclicos, los cálculos de codificación y de síndrome pueden implementarse fácilmente utilizando simples registros de corrimiento.

Esto quiere decir que la palabra código con elementos $(x_0,x_1,.....,x_{n-1})$ pueden ser representados en la forma de un código de palabra polinomial de la siguiente manera:

$$x(D) = x_0 + x_1 D + ...+ x_{n-1} D^{n-1} \qquad \textbf{(III.25)}$$

donde D es una variable real arbitraria. Naturalmente, para códigos binarios, los coeficientes son **ceros o unos.** Cada potencia de D en el polinomio $x(D)$ representa a un desplazamiento cíclico de un bit a su tiempo. Entonces la multiplicación del polinomio $x(D)$ por D podría ser visto por un desplazamiento cíclico de rotación a la derecha sugerido por turno $D^n = 1$.

La aplicación de $D^n = 1$ tiene dos objetivos: El primero, restaurar el polinomio $D \cdot x(D)$ de orden $(n - 1)$. Segundo, el bit más a la derecha del coeficiente x_{n-1} es alimentado a la derecha. Esta forma especial de multiplicación de polinomio es referido como módulo de multiplicación $(D^n - 1)$. Para un desplazamiento cíclico simple, se podría escribir:

$$D \cdot x(D) mod(D^n - 1) = x_{n-1} + x_0 D + ...+ x_{n-2} D^{n-1} \qquad \textbf{(III.26)}$$

donde ***mod*** es la abreviatura para "módulo". El código de la palabra polinomial en la ecuación III.26 es una representación polinomial de la palabra código $(x_{n-1}, x_0, \ldots, x_{n-2})$. Para dos desplazamientos cíclicos, se puede escribir:

$$D^2 x(D) \bmod (D^n - 1) = x_{n-2} + x_{n-1}D + \ldots + x_{n-3}D^{n-1}$$

la cual es una representación polinomial de la palabra código $(x_{n-2}, x_{n-1}, \ldots, x_{n-3})$ y así sucesivamente. En general, se describe la propiedad cíclica en notación polinomial por medio de sostener que si $x(D)$ es una palabra código polinomial, entonces el polinomio $D^i x(D) \bmod [13] (D^n - 1)$ es también una palabra código polinomial para cualquier desplazamiento cíclico.

[13] mod, representa módulo de multiplicación.

3.5.1 POLINOMIO GENERADOR

Un código cíclico (n,k) es especificado por el set completo de la palabra de códigos polinomial de grado (n-1) o menos, los cuales contienen un polinomio de grado mínimo (n - k) como un factor. Este factor especial, denotado por ***g(D)***, es llamado **polinomio generador del código**. El grado de *g(D)* es igual al número de bits de paridad en el código. El polinomio generador *g(D)* es equivalente a la matriz generadora ***G*** como una descripción del código.

3.5.1.1 CÓDIGOS POLINÓMICOS

También denominados de redundancia cíclica o CRC. Se basan en el tratamiento de polinomios que sólo tienen como coeficientes 0 y 1, y que representan cadenas de bits. Una trama de k bits se considera como el conjunto de coeficientes de un polinomio de orden k-1, el bit más significativo es el coeficiente de x^{k-1}, y el menos significativo el de x^0.Ejemplos:

10010110 ..... $x^7 + x^4 + x^2 + x^1$

11101..... $x^4 + x^3 + x^2 + 1$

10000001 ..... $x^7 + 1$

Cuando se emplea el método de código polinómico el receptor y el transmisor deben acordar de antemano un ***polinomio generador***, G(x). Tanto los bits mayor como menor del polinomio deben ser 1. Para calcular la ***suma de comprobación*** para una trama con m bits (que constituye l mensaje), correspondiente al polinomio M(x), el polinomio generador debe ser de grado menor que M(x).

La suma de comprobación es una operación que se efectúa sobre los bits del mensaje que permite saber si hay alguno erróneo. La idea es anexar una suma de comprobación al final del marco, de manera que el polinomio-mensaje más la suma de comprobación sean divisibles entre G(x).

Cuando se recibe el mensaje se realiza la división y si el resto, E(x), es distinto de cero es que se ha producido algún tipo de error en la transmisión.

El algoritmo para calcular la suma de comprobación es el siguiente:

- Si G(x) es de grado r, entonces se colocan r ceros al final del polinomio que representa el mensaje, M(x), para que ahora su longitud sea m + r, y corresponda al polinomio $x^rM(x)$.
- Se divide $x^rM(x)$ entre G(x) usando división modulo 2.

$$x^rM(x)/G(x) = C(x) + FCS \qquad \textbf{(III.27)}$$

- El resto (FCS, *frame check sequence*), que será de orden r o menor, se coloca al final del polinomio M(x) original. Al resultado de unir estos dos polinomio le llamamos T(x).

$$T(x) = x^rM(x) + FCS \qquad \textbf{(III.28)}$$

El polinomio T(x) es divisible entre G(x), de forma que si no resulta alterado durante la transmisión el resto de la división debe ser cero (E(x)=0), ya que un número binario sumado en módulo 2 con sigo mismo da cero. Sin embargo, un error E(x) no se detectará si es divisible por G(x).

Hay tres polinomios que están estandarizados internacionalmente:

TABLA III.6:Polinomios estandarizados

$X^{12} + X^{11} + X^3 + X^2 + X$	CRC – 12

$^{1} + 1$	
$X^{16} + X^{15} + X^{2} + 1$	CRC – 16 de la IBM
$X^{16} + X^{12} + X^{5} + 1$	CRC de CCITT V.41

3.5.2 POLINOMIO CHEQUEO DE PARIDAD

Un código cíclico (*n*,*k*) es únicamente especificado por su generador polinomial *g(D)* de orden (*n*-*k*). Tal código es también únicamente especificado por otro polinomio de orden k, el cual es llamado **el polinomio de chequeo de paridad**. Anteriormente se indicó que el polinomio generador *g(D)* es una representación equivalente de la matriz generadora ***G***. Correspondientemente, el polinomio de chequeo de paridad, denotado por ***h(D)***, es una representación equivalente de la matriz de chequeo de paridad ***H***. Encontrando que la relación de matriz $\mathbf{HG^T = 0}$, se tiene:

$$h(D)\, g(D) \bmod (D^n - 1) = 0 \qquad \textbf{(III.29)}$$

Cualquier múltiplo del polinomio generador *g(D)* de un código cíclico (*n*,*k*) es un polinomio de la palabra código. De acuerdo a esto, se puede encontrar de la ecuación III.29 que cualquier polinomio de palabra código *x(D)* en el código satisface la siguiente relación fundamental:

$$h(D)\, x(D) \bmod (D^n - 1) = 0 \qquad \textbf{(III.30)}$$

El polinomio generador *g(D)* y el polinomio chequeador de paridad *h(D)*, son factores del polinomio $1 + D^n$, como se muestra:

$$\mathbf{h(D)g(D) = 1 + D^n} \qquad \textbf{(III.31)}$$

"Esta propiedad provee la base para seleccionar el generador o polinomio chequeador de paridad de un código cíclico".

Ejemplo: Para ilustrar la validez de la representación del polinomio de los códigos cíclicos, se va a considerar la generación de un código cíclico (7,4). En el cual la longitud del bloque n = 7, se empezará por factorizar $1 + D^7$ en tres polinomios irreducibles.

$$1 + D^7 = (1 + D)(1 + D^2 + D^3)(1 + D + D^3)$$

Por un ***polinomio irreducible***, se quiere decir un polinomio que no pueda ser factorado usando sólo polinomios con coeficientes del campo binario. Un polinomio irreducible de grado ***m*** se dice ser ***primitivo*** si el integrador positivo más pequeño ***n*** para el cual el polinomio se divide $(1 + D^n)$ es $n = (2^m - 1)$. Para el ejemplo puesto, solamente dos polinomios, nombrados son primitivos, y son:

$$(1 + D^2 + D^3) \; ; \; (1 + D + D^3)$$

entonces se elige la última posibilidad:

$$g(D) = (1 + D + D^3)$$

como el polinomio generador, cuyo grado es igual al número de bits de paridad, significa que el polinomio de chequeo de paridad esta dado por:

$$h(D) = (1 + D)(1 + D^2 + D^3)$$

$$h(D) = (1 + D + D^2 + D^4)$$

cuyo grado es igual al número de bits del mensaje k = 4.

Ejemplo: Se determinará un polinomio generador *g(D)* para un código cíclico (7,4), y se encontrará el vector código para el siguiente vector de dato:

1010

en este caso n = 7 y (n-k) = 3

$$1 + D^7 = (1 + D)(1 + D^2 + D^3)(1 + D + D^3)$$

para un código(7,4), el polinomio generador debe ser del orden (*n-k*) = 3. En este caso se escoge:

$$g(D) = D^3 + D^2 + 1$$

como un posible generador. Para:

$$d = [1\ 0\ 1\ 0]$$

$$d(D) = D^3 + D$$

y el polinomio de código es:

$$c(D) = d(D)\, g(D)$$
$$c(D) = (D^3 + D)(D^3 + D^2 + 1)$$
$$c(D) = D^6 + D^5 + D^4 + D$$

por lo tanto,

$$c = 1110010$$

en forma similar, las palabras códigos para otras palabras datos se pueden encontrar, como se indica en la tabla III:7

TABLA III.7: Palabras codificadas

PALABRA DATO	PALABRA CÓDIGO
1010	1110010
1111	1001011
0001	0001101
1000	1101000

Observe la estructura de las palabras códigos (tabla III.7). Los primeros *k* dígitos no necesariamente son los dígitos de datos. Por consiguiente, éste no es un código sistemático.

En un código sistemático, los primeros *k* dígitos son de datos y los últimos m = n-k son los dígitos de comprobación de paridad.

3.5.3 CODIFICACIÓN PARA CÓDIGOS CÍCLICOS

El procedimiento para la codificación de códigos cíclicos (*n,k*) en forma sistemática envuelve tres pasos:

1. Multiplicación del polinomio de mensaje *m(D)* por D^{n-k}.

2. División de D^{n-k} *m(D)* por el polinomio generador *g(D)* para obtener el residuo *b(D)*, y.

3. Adición de *b(D)* hasta D^{n-k} *m(D)* para formar la palabra código del polinomio deseado.

Entonces estos tres pasos pueden ser implementados por medio del codificador mostrado en la figura III.5, que consiste en un registro de desplazamiento lineal realimentado con (n-k) estados.

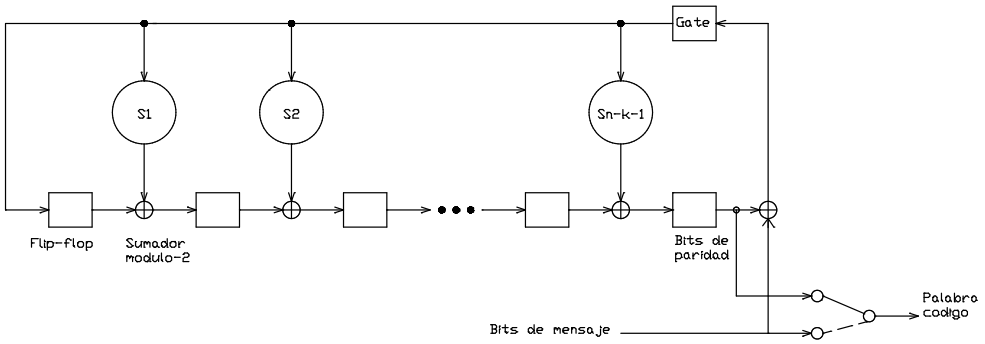


FIGURA III.5 Codificador para un código cíclico de (n,K)

La operación del codificador de la figura III.5 procede de la siguiente manera:

- La gate es activada. Entonces, los bits del mensaje k son desplazados en el canal. Al mismo tiempo los bits del mensaje k ingresan al registro de desplazamiento, el resultado de los bits (n-k) en el registro forman los bits de paridad (los bits de paridad son los mismos del coeficiente del residuo *b(D)*).

- La gate es desactivada, de ese modo desconecta las conecciones de retroalimentación.
- El contenido del registro de desplazamiento son desplazados fuera del canal.

3.5.4 CÁLCULO DEL SÍNDROME

Suponga que la palabra código ($x_0,x_1,.....,x_{n-1}$) es transmitido sobre el canal de ruido, resultando en la palabra recibida ($y_0,y_1,.....,y_{n-1}$). Recuerde que el primer paso para la decodificación de un código de bloque lineal es calcular el síndrome para la palabra recibida. Si el síndrome es cero, no hay errores de transmisión en la palabra recibida. Si por otro lado, el síndrome no es cero entonces la palabra recibida contiene errores de transmisión, lo cual requiere corrección.

En el caso de un código cíclico en forma "sistemática", el síndrome puede ser calculado fácilmente. La palabra recibida puede ser representada por un polinomio de grado (n-1) o menor, como es indicado por:

$$y(D) = y_0 + y_1 D + ...+ y_{n-1} D^{n-1} \qquad \textbf{(III.32)}$$

Permita que *a(D)* denote el cociente y *s(D)* denote el residuo, los cuales son el resultado de dividir *y(D)* por el polinomio generador *g(D)*. Entonces se puede expresar *y(D)* como sigue:

$$y(D) = a(D)\, g(D) + s(D) \qquad \textbf{(III.33)}$$

El residuo *s(D)* es un polinomio de grado (n-k-1) o menor. Este es llamado el síndrome polinomial que sus coeficientes se forman [(*n-k*),*1*] del síndrome *s*. Donde el síndrome polinomial *s(D)* no es cero, la presencia de errores en la transmisión es detectada en la palabra recibida.

La figura III.6 muestra un calculador del síndrome que es idéntico al codificador de la figura III.5 excepto por el hecho de que los bits recibidos son alimentados en los estados (n-k) de la realimentación del registro de desplazamiento desde la izquierda. Tan pronto como todos los bits recibidos han sido desplazados en el registro de desplazamiento, su contenido define el síndrome deseado ***s***. Una vez que se conoce ***s*** se puede determinar el correspondiente error patrón ***e*** y así realizar la corrección apropiada.

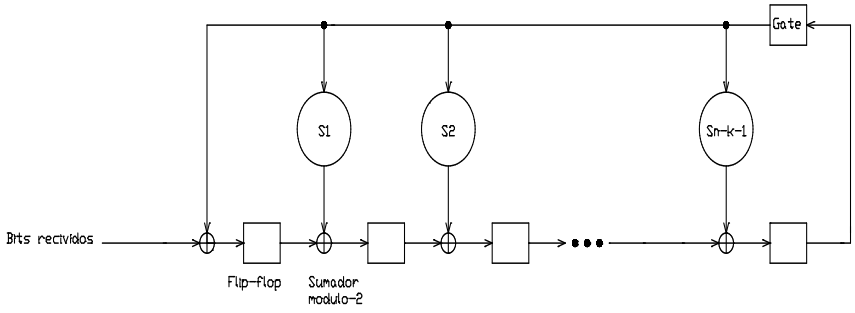


FIGURA III.6 Síndrome calculador

CAPÍTULO IV

COMPRESIÓN DE DATOS

4.1 INTRODUCCIÓN

Una de las particularidades más notorias del desarrollo de la informática es la proporción directa que existe entre el aumento de la capacidad de almacenamiento de la información y el tamaño (o peso) de los archivos que se producen.

Esta característica puede ser inmediatamente percibida por cualquier usuario, porque se trata de un proceso permanente: los discos rígidos albergan cada vez más datos, los programas (procesadores de texto, planillas de cálculo, para diseño gráfico, para dibujo, etc.) son cada vez más completos y complejos, y los archivos resultan cada vez más "pesados".

A principios de la década de 1980, las computadoras XT solían tener un disco rígido de 360 Kb. Un archivo realizado con procesador de textos (Wordstar, por ejemplo) de unas 10 páginas podía pesar 1.500 bytes.

Tanto en la época de las XT como ahora, los usuarios han tenido la necesidad de **transportar la información** que producen o utilizan (archivos de trabajo, programas) de una computadora a otra. Y tarde o temprano, se han encontrado con que el espacio de los dispositivos "portátiles" resulta limitado.

En la actualidad, no siempre es necesario transportar la información en unidades almacenadoras, porque las computadoras pueden estar conectadas entre sí a través de redes (locales o Internet), pero en estos casos, el tamaño de los archivos sin duda afectará el tiempo de **conexión** y **transmisión**.

Los **programas de compresión** permiten básicamente disminuir el tamaño de los archivos, ahorrando espacio de almacenamiento en disco y tiempo de conexión en una red.

Los programas de compresión reordenan la información de un archivo mediante *algoritmos* y generan un nuevo archivo que ocupa menos lugar. El nuevo archivo no tiene las características y funcionalidad del original, por lo que para utilizarlo hay que **descomprimirlo** en principio, con el mismo programa que se usó para comprimirlo.

FIGURA IV.1 Modelo de compresión - descompresión

4.2 EL PROBLEMA DE LA COMPRESIÓN

Hoy en día los requerimientos de almacenamiento de información lleva a buscar formas eficaces de almacenamiento de información. La necesidad de comprimir información, no es tema de actualidad. A finales de los setenta se dieron los primeros pasos hacia la compresión de datos.

La adquisición de software nuevo o la actualización del mismo, reduce el espacio de almacenamiento de información, y lleva a la búsqueda y utilización de software especializado en la compresión de datos. La elección del mismo depende de las características físicas del hardware, o bien de las necesidades de almacenamiento.

Las causas de darse la compresión son entre otras que cuando nuestro medio de almacenamiento llega al límite de su capacidad, si no se cuenta con otro medio para seguir almacenando se puede recurrir al software de compresión que se adapte a las

necesidades, ahorrando espacio de almacenamiento dándole un uso óptimo al mismo; la transmisión de un alto volumen de datos provoca un aumento en el costo del envío que crece en forma considerable y que conlleva a la transmisión de datos en forma comprimida, reduciendo costos de envío; de igual forma el tiempo de transmisión en redes o vía satélite es limitado y costoso.

Se denomina ***compresión*** de datos al conjunto de técnicas que permiten que un conjunto
de datos de una determinada longitud pueda ser reducido en su tamaño, sin alterar el significado de la información que contiene. La compresión modifica la velocidad de transferencia de información y además reduce la probabilidad de que se produzcan errores durante la transmisión a través de un canal con ruido.

Se puede decir que existen dos tipos de compresión:

- la *compresión lógica* la cual se refiere a reducir la información redundante de los campos de información en las bases de datos, mediante una metodología utilizada en el análisis de sistemas (reducción de datos desde el momento del diseño); y.

- la *compresión física* que es acomodar grandes cantidades de datos en pequeños espacios. Tiene en cuenta la frecuencia de ocurrencia de los caracteres. Es decir permite reducir la cantidad de datos antes de ser colocados en el canal de comunicación y expandirlos en su punto de destino. Son varias las técnicas utilizadas, como por ejemplo la estadística.

La compresión se puede medir mediante el índice de compresión:

$$X = \frac{Longitud\_de\_un\_conjunto\_de\_datos}{Longitud\_comprimida\_del\_conjunto\_de\_datos}$$

(IV.1)

En ocasiones también se utiliza el factor de mérito, que es el inverso del índice de compresión:

$$P = \frac{1}{X} \qquad \textbf{(IV.2)}$$

Es muy difícil clasificar los distintos tipos de compresión de datos que existen, debido a que, por un lado se tiene muchos algoritmos: LZ77, LZ78, LZW, Huffman, aritméticos, fractales, MPEG, JPEG, etc.

Además, hay muchas aplicaciones que utilizan la compresión con distintas expectativas: compresión de datos, vídeo y voz, compresión en tiempo real, etc. A su vez, cada uno de estos usos requiere unas características de velocidad, reversibilidad (que el algoritmo pueda ser aplicado de forma reversible para obtener los datos originales), pérdida mínima de información (en el caso de los algoritmos con pérdida de información, etc.).

Debido a esto, se ha elegido una clasificación muy general tomando como característica de división la reversibilidad del algoritmo. Así, los algoritmos de compresión se pueden dividir en dos tipos:

- Compresores ***lossless*** o sin pérdidas, en el sentido de que guarda absolutamente toda la información original (es reversible). Se utilizan para la compresión de datos, en los que no se puede dar pérdida de información, es decir se conserva el original.

- Compresores ***lossly*** o con pérdidas. La compresión de pérdidas se usa con frecuencia para reducir archivos de audio o de imágenes en las que no se requiere de la exactitud absoluta de los datos, basada en que se asume que la pérdida de los datos no se percibirán. Esta técnica de compresión se aprovecha en dispositivos de hardware o sea en chips, los cuales son más sofisticados que los de software y se aplica mejor a tarjetas gráficas, los cuales se basan en el grupo de expertos

fotográficos (JEPG), grupo de expertos de fotos en movimiento (MPEG) y ahora en el formato parcial de imágenes (TIF) que son muy usados en aplicaciones gráficas.

Este trabajo está centrado solo en la compresión "**lossless**" sin pérdida, es decir, técnicas que garantizan que no habrá ningún tipo de pérdida de información al comprimirlos (factor fundamental para comprimir y recuperar programas, ficheros de bases de datos...etc).

Hay básicamente dos tipos de compresores / algoritmos de compresión lossless:

- Compresores **estadísticos.**
- Compresores **basados en diccionario** ó **sustitucionales**.

4.3 COMPRESION ESTADÍSTICA

La compresión estadística no es una técnica de compresión propiamente dicha. Se trata de una técnica en la que se realiza la codificación en la fuente y la compresión simultáneamente. De ahí que esta técnica sea también conocida con el nombre de codificación estadística.

Su objetivo consiste en realizar una codificación en la fuente para obtener códigos tales que la longitud media de los datos codificados sea menor que la obtenida con códigos de longitud fija. Por este motivo, para la construcción de estos códigos es necesario tener un conocimiento previo de la frecuencia de ocurrencia de cada uno de los caracteres del código original (aprovechar la redundancia de información de la fuente para conseguir esa compresión). Se usarán codificaciones más cortas para representar los caracteres con mayor frecuencia de aparición.

Este tipo de compresores parten de:

- una fuente de información de *n* mensajes.
- las probabilidades de aparición de cada mensaje de la fuente (que pueden ser extraídas a priori de forma experimental o pueden ser dadas y fijas).

- un *alfabeto* de salida que consta de una serie de símbolos (por ejemplo, el alfabeto binario consta de los símbolos 0 y 1).

En esta sección se examina los compresores que utilizan la información de las probabilidades de los mensajes de la fuente para construir una codificación. Entre los compresores estadísticos se puede encontrar varios tipos:

- Compresores del tipo Huffman
- Compresores de Shannon-Fano
- Compresores aritméticos
- Compresores predictivos

La construcción de este tipo de códigos se basa en la propiedad del prefijo, según la cual, ninguna secuencia de bits que represente a un carácter del código podrá aparecer como subsecuencia inicial de otra secuencia de longitud mayor que represente a otro carácter del código.

4.3.1 Codificación Huffman

Este código es un código óptimo dentro de los códigos de codificación estadística, ya que es el código de menor longitud media.

La construcción de este código se fundamenta en asignar a cada símbolo del alfabeto fuente una secuencia de bits cuya longitud esté relacionada de forma directa con la probabilidad de aparición de ese símbolo. De esta forma, a los símbolos con mayor frecuencia de aparición se les asignarán las palabras de código de menor longitud.

En el proceso de construcción de este código, lo primero que se hace es ordenar el conjunto de símbolos del alfabeto fuente en orden decreciente de probabilidades de aparición. A continuación se juntan los dos símbolos con menor probabilidad de aparición en un único símbolo cuya probabilidad será la suma de las probabilidades de los símbolos que dieron origen a este nuevo símbolo.

Se repite este proceso hasta que sólo se tenga dos símbolos.

A continuación se realiza el proceso de codificación. Primeramente se asigna un 1 a uno de los dos símbolos que se tiene y un 0 al otro. Posteriormente se recorre la estructura que se ha construido hacia atrás de forma que cuando dos símbolos hayan dado origen a un nuevo símbolo, estos dos símbolos "heredarán" la codificación asignada a este nuevo símbolo y a continuación se le añadirá un 1 a la codificación de uno de los símbolos y un 0 a la del otro símbolo.

Generalmente los descompresores de este tipo no tiene posibilidad de conocer previamente las probabilidades de los mensajes, pues sólo recibe los códigos asignados a los mensajes; en consecuencia el árbol ya procesado ha de ser pasado al descompresor, junto con los datos. Esto representa una carga adicional al fichero comprimido que resta en parte la eficiencia de esta técnica. Por ello una de las soluciones es hacer que estos algoritmos sean "adaptativos": se construye el árbol dinámicamente tanto por el compresor como por el descompresor, y así no estaremos obligados a pasarle al descompresor el árbol. Dependiendo del compresor, se suelen utilizar diferentes implementaciones "adaptativas" para los compresores.

En el siguiente capítulo se detallará el algoritmo de Huffman el cual es tema central para la implementación del compresor de datos, que es el propósito fundamental de este proyecto..

4.3.2 Código de Shannon-Fano

Para la construcción de este código, el primer paso consiste en ordenar el conjunto de símbolos del alfabeto fuente en orden decreciente de probabilidad de aparición.

A continuación se divide el conjunto en dos subconjuntos de forma que la suma de probabilidades de los símbolos de cada subconjunto sea igual o aproximadamente igual en cada subconjunto.

A los símbolos del primer subconjunto se les asigna un 1 y a los del segundo un 0 (o al revés).
En cada subconjunto se repite el proceso hasta que se obtienen subconjuntos de un solo símbolo.

4.3.2.1 Diagrama de Árbol

Las distintas posibilidades pueden ser representadas mediante un árbol binario.

Ambos algoritmos indicados anteriormente terminan construyendo un árbol que representa la codificación que de los mensajes de la fuente se ha realizado, de manera que los nodos hoja contienen cada uno de los mensajes emitidos por la fuente. En el caso más simple, el alfabeto de salida en el que se realiza la codificación es binario. Esto quiere decir que de cada nodo partirán dos ramas, una para el "0" y otra para el "1". El código para cada mensaje se construye siguiendo el camino desde el nodo raíz hasta la hoja que representa el mensaje.

Nótese que este esquema permite que si, a la hora de descomprimir, el decodificador Huffman o Shannon-Fano poseen el mismo árbol que se hizo para comprimir, la decodificación es tan sencilla como leer *bits* de la fuente a descomprimir y seguir el camino desde la raíz hacia abajo bifurcando hacia un lado o hacia otro dependiendo del valor del bit. Eventualmente se llegará a una hoja, que representa al mensaje que se estaba recibiendo.

Pero esto todavía no tiene nada de particular. Lo verdaderamente importante es que la codificación resultante de la aplicación de estos algoritmos asigna longitudes de codificación **inversamente proporcionales** a la probabilidad de aparición de cada mensaje. Es decir, el mensaje que más aparezca tendrá una codificación más corta, con lo que se ahorrará espacio en la transmisión. Recuerde que los mensajes con más probabilidad son los que menos información tendrán, por eso, se asigna menos símbolos del alfabeto de salida en su codificación.

Se ha analizado el objetivo de ambos algoritmos: la construcción del *árbol de códigos* que representa a la codificación obtenida. En donde difieren es en la manera de conseguir el fin. Ambos construyen el árbol de manera que los mensajes con menor probabilidad quedan más abajo en el árbol, sin embargo, Huffman lo hace *botton-up* y Shannon-Fano *top-down*.

La figura IV.2 representa un modelo de generación del árbol binario a manera de ejemplo.

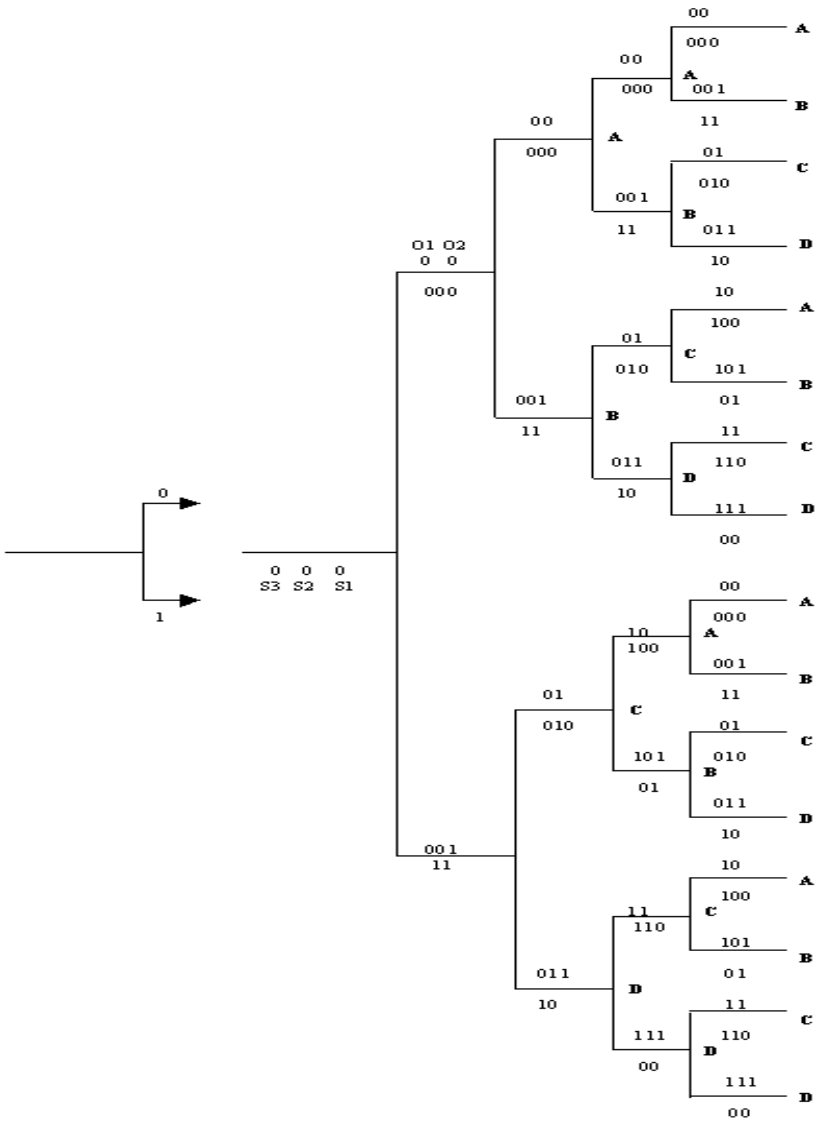


FIGURA IV.2 Modelo de la generación del árbol binario

La interpretación del árbol del código es la siguiente: Hay dos ramas en cada nodo:

- La rama superior corresponde a una entrada de un 0.
- La rama inferior corresponde a la entrada de un 1.

- En la parte exterior de cada rama se muestra el valor de salida.
- El número de ramas se va multiplicando por dos con cada nueva entrada.
- A partir del segundo nivel el árbol se vuelve repetitivo. En realidad, solo hay cuatro tipos de nodos: A,B,C,D. Estos tipos de nodos en realidad son estados del codificador.
- A partir de estos nodos, se producen los mismos bits de salida y el mismo estado.

Por ejemplo, de cualquier nodo etiquetado como C se producen el mismo par de ramas de salida:

Salida 10 y estado A
Salida 01 y estado B

4.3.3 Códigos aritméticos

Se basan también en las probabilidades de repetición de los mensajes a la entrada, aunque su metodología es muy distinta. Lo que hacen es representar un valor del intervalo [0,1] con mayor número de decimales, se tiene mayor precisión cuanto mayor sea la información de los datos a comprimir. Básicamente va dividiendo el intervalo [0,1] sucesivamente hasta obtener un número dentro de ese intervalo que utilice menos bits para representar toda la entrada. Así el descompresor podrá reconstruir la entrada con ese número más la información del número de elementos codificados y sus probabilidades correspondientes. Es un proceso bastante eficiente, aunque como se acaba de decir las probabilidades deben ser incluidas en el fichero comprimido. Por ello se suelen utilizar también modelos adaptativos, que cambian dinámicamente con la entrada.

4.3.4 Compresión Predictiva

Los compresores predictivos son, al contrario que los anteriores, totalmente adaptativos. Procuran **predecir** el siguiente mensaje de la entrada tomando como base de conocimiento la entrada procesada hasta ese momento (en el fondo, también probabilidades). Si el mensaje que se encuentra a la entrada coincide con el predicho, su codificación se podrá hacer con menos bits. Si no, su codificación se hará con más bits, que permitirán entonces sincronizar al descompresor para que mantenga sus tablas internas idénticas a las del compresor sin pasárselas explícitamente.

Poseen ciertas ventajas sobre los anteriores algoritmos, entre ellas su velocidad: al actuar sobre un mensaje cada vez y realizar una predicción que generalmente suele ser de cálculo sencillo, son capaces de dar una alta velocidad de compresión/descompresión. Velocidad y sencillez son dos conceptos que generalmente van unidos, por lo que estos algoritmos, a la vez que rápidos, resultan sencillos de programar, por lo que se pueden convertir en una solución barata para sistemas de compresión transparente en tiempo real, con unas relaciones de compresión aceptables.

Sin embargo, para algunas aplicaciones, no son tan aceptables las relaciones de compresión. Además, su mejora es sustancialmente difícil: la predicción es muy escurridiza en cualquier ámbito. Son muy malos cuando hay mucha redundancia, así que en la práctica se suelen usar en conjunción con otras técnicas de compresión para tratar los casos en que hay excesiva redundancia (por ejemplo el Run-Length).

4.4 COMPRESIÓN BASADA EN DICCIONARIO

Un enfoque diferente a los algoritmos anteriores es el que presentan los compresores basados en diccionario. Su idea es construir un diccionario tomando como referencia la entrada procesada hasta ese momento. El diccionario contiene las cadenas de

mensajes (un ejemplo práctico las cadenas de caracteres ASCII o bytes). Estas cadenas están identificadas por un índice, de manera que índice y cadena son de correspondencia biunívoca. El resultado práctico a todo esto es que si en algún momento la entrada que se está procesando actualmente es una cadena que está presente en el diccionario, el compresor puede dar como salida el índice que identifica esa cadena en el diccionario. Teniendo en cuenta que las cadenas pueden ser arbitrariamente largas, la producción del índice en lugar de la cadena representa un ahorro de información, y por lo tanto, compresión.

Incluidos dentro de este grupo se considera a los algoritmos *RLE*, que puede ser considerado como poseyendo un diccionario de longitud 1 byte, el *LZW* y otros derivados de LZ78 y las variantes de LZ77.

4.4.1 COMPRESIÓN RLE

Denominado también *Run Length Encoding*. Es el más simple y a la vez el más ineficiente. Se podría considerar que utiliza un diccionario deslizante para predecir el siguiente carácter de la entrada. Realmente se le considera ya algo "primitivo". Además hay diferentes formas de implementarlo. Busca repeticiones consecutivas de un mismo símbolo y lo que hace es almacenar en un byte el número de esas repeticiones consecutivas y en el segundo byte se escribe el símbolo. Como ejemplo:

17 48 (el byte 48 se repite 17 veces)

Demuestra gran eficiencia cuando hay un alto número de repeticiones consecutivas de un determinado byte. La unidad básica serían dos bytes, el primero indica el número de veces que se repite el segundo. Básicamente se utiliza para crear archivos tipo BMP o PCX sin gradaciones de color. Era utilizado por el ARC[14], entre otros.

[14] ARC (Automatic Repeat Request), consiste en la transmisión de la información afectada por errores y solo retransmite si hay errores, este no permite la transmisión en tiempo real (voz y video).

4.4.2 CÓDIGO LEMPEL – ZIV

En esta técnica, durante el proceso de codificación el emisor va construyendo una tabla o diccionario con la información que va tratando. Cada entrada al diccionario consiste en un par (*n,Ai*), donde *n* es un 'puntero' a otra localización del diccionario y *Ai* es un símbolo del alfabeto fuente. Todas las palabras de código (a las que también se denominan bloques codificados) tendrán la misma longitud previamente determinada.

Esta longitud será conocida por el emisor y por el receptor. Si la longitud de las palabras de código se ha fijado en *m* bits, los *m-1* bits primeros se utilizan para buscar elementos que ya han aparecido en la tabla y al último bit se le denomina bit de innovación.

El número de entradas o posiciones de la tabla es función de la longitud de palabra de código y será de 2^{m-1} .

En la cadena de entrada se buscarán subsecuencias que no coincidan con ninguna de las subsecuencias que ya se han almacenado en la tabla, y se añadirán a la tabla.

Cuando la tabla ya se ha llenado, las nuevas subsecuencias que se busquen deberán ser igual a alguna que ya esté en la tabla concatenándole un bit 0 o 1.

La idea básica consiste en dividir la cadena original en subcadenas que no hayan sido encontradas anteriormente. Esto se repite hasta completar todas las entradas de la tabla. Una vez llena la tabla, se empiezan a repetir las codificaciones.

Es muy importante el tamaño que se escoge para la palabra codificada. Cuanto mayor sea (dentro de un rango) mayor índice de compresión se logrará. Podría

ocurrir que si inicialmente se repiten mucho unas secuencias y con ellas se llena la tabla, cuando ya no se puedan introducir más secuencias en la tabla se habrá llegado al límite y se podrá obtener malos resultados.
La tabla que se construye durante el proceso de codificación está formada por 4 columnas:

- La primera columna de la tabla se denomina "Posiciones Numéricas" e indica las posiciones numéricas de las subsecuencias individuales de la tabla de códigos.
- La segunda columna de la tabla se denomina "Subsecuencias" e indica las subsecuencias individuales de la tabla de códigos. El último símbolo de cada subsecuencia es el bit de innovación: la presencia de ese bit es lo que diferencia la subsecuencia de las anteriores. El resto de los bits son la subsecuencia raíz.
- La tercera columna de la tabla se denomina "Representaciones Numéricas" e indica las posiciones numéricas de las subsecuencias que componen la nueva subcadena.
- La cuarta columna de la tabla se denomina "Bloques Binarios Codificados" e indica la codificación para la subsecuencia correspondiente a esa fila. Para obtener la codificación, primero se codifica el primer dígito que compone la representación numérica mediante *m-1* bits, y a continuación se le concatena el bit asociado a la posición numérica especificada por el segundo dígito que compone la representación numérica. De esta forma se obtiene la palabra de código que se transmitirá.

En el proceso de decodificación el receptor recibe las palabras de código y a partir de ellas construye su propia tabla de forma similar a como lo hacía el emisor. La tabla creada por el receptor coincidirá con la elaborada por el emisor en el proceso de codificación.

En la tabla del receptor, las dos primeras entradas se inicializan con las subsecuencias 0 y 1. A continuación se procesa cada palabra de código recibida.

Para cada palabra de código se toma la subsecuencia raíz, se interpreta como si fuera binario natural obteniendo un número decimal que indicará una de las entradas de la tabla. A continuación se toma la subsecuencia asociada a esa entrada de la tabla y se concatena el bit de innovación. De esta forma se obtiene la subsecuencia original enviada por el emisor.

Cada vez que se realice una decodificación, si la tabla no está llena, el receptor añadirá esta decodificación a la tabla.

4.5 Compresores en tiempo real

Con el aumento de velocidad de los ordenadores la compresión se está aplicando cada vez más a entornos en tiempo real o de compresión transparente. Esta técnica es llamada también " On the fly " tiene por objeto ampliar la capacidad de los discos duros de las microcomputadoras, laptops y note-books por medio de un programa de software o de un dispositivo de hardware.

Este esquema de compresión se basa principalmente en el algoritmo de compresión de Lempel-Ziv; los continuos procesos de tiempo real de reducir los tamaños de los archivos y de restaurarlos a su forma original, se ejecutan durante los requerimientos del sistema al disco duro. En cuanto a su lectura y escritura éstos ocurren de tal manera que son transparentes al usuario.

Con el tiempo, además de reducir el tamaño de los archivos, los programas de compresión han incorporado otras operaciones orientadas a hacer más transportable la información:

- Compresión de varios archivos en uno solo.

- Segmentación de un archivo muy grande en varios más pequeños que puedan entrar en sendos disquetes.

- Creación de archivos comprimidos que se descomprimen por sí mismos (autoejecutables), sin necesidad de pasar por el programa de compresión.

Entre los programas de compresión más comunes se tiene:

PKZIP, PKUNZIP utilidades compartidas de compresión y descompresión de ficheros. El PKZIP para Windows, el más rápido que hay, soporta línea de comandos para programas como Windows Commander. Para quienes quieren velocidad y compatibilidad.

WIN ZIP es el comprimidor y descomprimidor más conocido del momento, se podría decir el más popular manejador de archivos comprimidos ZIP para Windows. El Winzip es un programa que sirve para comprimir cualquier tipo de archivos. Usualmente se utiliza para compactar texto; reduciendo, considerablemente, el tamaño que mide un archivo.

WIN RAR es mas práctico que el Winzip y comprime más datos con menos espacio que cualquier otro. Comprime cualquier tipo de archivo. Posee una excelente compresión en archivos multimedia. Soporte interno de archivos ZIP, LZH, etc. Crea archivos autoextractables, multivolumen, autorecuperación de datos. Muy buena integración con Windows.

WinRAR y WinZip son dos de los programas de compresión más utilizados. Ambos se consiguen en forma gratuita a través de Internet o en los CD-Rom que suelen traer las revistas especializadas en informática

CAPÍTULO V

IMPLEMENTACIÓN DEL CÓDIGO HUFFMAN ADAPTATIVO

5.1 GENERALIDADES

Uno de los aspectos fundamentales en el manejo de información es transportarla de forma eficiente, el costo de transportar información a través de grandes distancias es muy alto si se lo mide en tiempo así que conviene hallar alguna forma de "empaquetar" la información de alguna forma en que el volumen de bits de la misma disminuya y así también el tiempo de transmisión. Es también conveniente, de hecho indispensable, contar con algún medio para "desempacar" la información nuevamente, una vez llegada a su destino, de forma que vuelva al estado en que en el contenido del mensaje original tenga sentido en algún contexto apropiado. Este es sólo un ejemplo de la importancia de estos mecanismos, que en la práctica son desarrollados e implementados a partir de un "modelo" que finalmente produce un *algoritmo* para la compresión y descompresión de datos.

Básicamente la compresión consiste en tomar una trama de símbolos y transformarlos en códigos/claves. Si la compresión es eficiente, las claves resultantes ocuparán menor espacio que los símbolos originales. La decisión de obtener una codificación a partir de de ciertos símbolos (o conjunto de ellos) está basada en un modelo. El modelo es

simplemente una colección de datos y reglas usados para procesar a la entrada símbolos y determinar su correspondiente codificación a la salida. Por ejemplo un programa usa el modelo para definir aproximadamente las probabilidades para cada símbolo y el codificador para producir una codificación apropiada basada en esas probabilidades.

En éste capítulo se presentará una de las técnicas importantes para la compresión de datos, esperando que después de haber establecido la importancia de este tipo de programas se aprecie el valor de las técnicas de compresión, la cual es la que se desarrolla como parte de este trabajo.

5.2 CÓDIGO DE HUFFMAN

La codificación de Huffman fue descrita por primera vez por David A. Huffman en 1952 en un artículo llamado *A Method for the Construction of Minimum Redundancy Codes*. Debido a su facilidad de cómputo, es ampliamente utilizada en programas de compresión como gzip, máquinas de fax y en esquemas de compresión de imágenes como JPEG.

Se trata de un algoritmo que puede ser usado para compresión o encriptación de datos.

Este algoritmo se basa en asignar códigos de distinta longitud de bits a cada uno de los caracteres de un fichero. Si se asignan códigos más cortos a los caracteres que aparecen más a menudo se consigue una compresión del fichero.

Esta compresión es mayor cuando la variedad de caracteres diferentes que aparecen es menor. Por ejemplo: si el texto se compone únicamente de números o mayúsculas, se conseguirá una compresión mayor.

Para recuperar el fichero original es necesario conocer el código asignado a cada carácter, así como su longitud en bits, si ésta información se omite, y el receptor del

fichero la conoce, podrá recuperar la información original. De este modo es posible utilizar el algoritmo para encriptar ficheros.

5.2.1 Algoritmo Huffman

Se basa en crear un árbol binario completo, que representa la codificación de los mensajes de la fuente, en el que cada nodo intermedio es menor que sus hijos (y la raíz el menor de todos). Los nodos-hoja contienen cada uno de los mensajes emitidos por la fuente. El código para cada mensaje se construye siguiendo el camino desde el nodo raíz hasta la hoja que representa el mensaje. Además, si el decodificador implementa el mismo árbol usado para comprimir, la decodificación no será más que leer bits e ir siguiendo el camino desde la raíz del árbol hasta las hojas en función del valor de esos bits. Al llegar a la hoja se habrá llegado al mensaje.

La codificación es inversamente proporcional a la probabilidad de aparición del mensaje. Al mensaje más redundante, que se repita más, se le dará una codificación más corta asignándole menos símbolos del alfabeto de salida y ahorrando así espacio.

En cada paso se recogen los dos nodos con menor probabilidad del árbol y se crea un nodo padre para ambos que contendrá la probabilidad sumada de los dos. Los nodos con menor probabilidad irán quedando al fondo.

Generalmente los descompresores de este tipo no tiene posibilidad de conocer previamente las probabilidades de los mensajes, pues sólo recibe los códigos asignados a los mensajes; en consecuencia el árbol ya procesado ha de ser pasado al descompresor, junto con los datos. Esto representa una carga adicional al fichero comprimido que resta en parte la eficiencia de esta técnica. Por ello una de las soluciones es hacer que estos algoritmos sean "adaptativos": se construye el árbol dinámicamente tanto por el compresor como por el descompresor, y así no se estará obligado a pasarle al descompresor el árbol. Dependiendo del compresor, se suelen utilizar diferentes implementaciones "adaptativas" para los compresores.

5.2.2 Mecanismo del algoritmo

Para implementar el algoritmo, debe considerarse el siguiente procedimiento:

- Contar cuantas veces aparece cada carácter en el fichero a comprimir. Y crear una lista enlazada con la información de caracteres y frecuencias.
- Ordenar la lista de mayor a menor en función de la frecuencia.
- Convertir cada elemento de la lista en un árbol.
- Fusionar todos estos árboles en uno único, para hacerlo se continúa con el siguiente procedimiento, mientras la lista de árboles contenga más de un elemento:

    - Con los dos primeros árboles formar un nuevo árbol, cada uno de los árboles originales en una rama.
    - Sumar las frecuencias de cada rama en el nuevo elemento árbol.
    - Insertar el nuevo árbol en el lugar adecuado de la lista según la suma de frecuencias obtenida.

- Para asignar el nuevo código binario de cada carácter sólo hay que seguir el camino adecuado a través del árbol. Si se toma una rama cero, se añade un cero al código, si se toma una rama uno, se añade un uno.
- Se recodifica el fichero según los nuevos códigos.

El siguiente es un ejemplo de codificación en el que se tiene el siguiente texto corto:

ata la vaca a la estaca

1. Se cuenta las veces que aparece cada carácter y se hace una lista enlazada:

' '(5), a(9), c(2), e(1), v(1), l(2), s(1), t(2)

2. Se ordena por frecuencia de menor a mayor

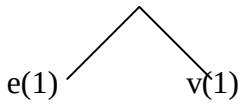
e(1), v(1), s(1), c(2), l(2), t(2), ' '(5), a(9)

3. Considerar ahora que cada elemento es el nodo raíz de un árbol.

e(1) → v(1) → s(1) → c(2) → l(2) → t(2) → ' '(5) → a(9)

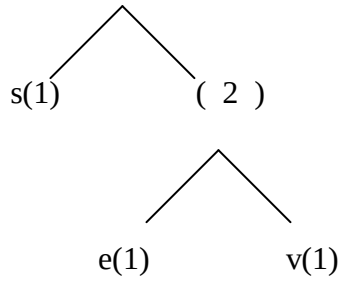
4. Se unen los dos primeros nodos (árboles) en un nuevo árbol, se suman sus frecuencias y se lo coloca en el lugar correspondiente:

s(1) → (2) → c(2) → l(2) → t(2) → ' '(5) → a(9)

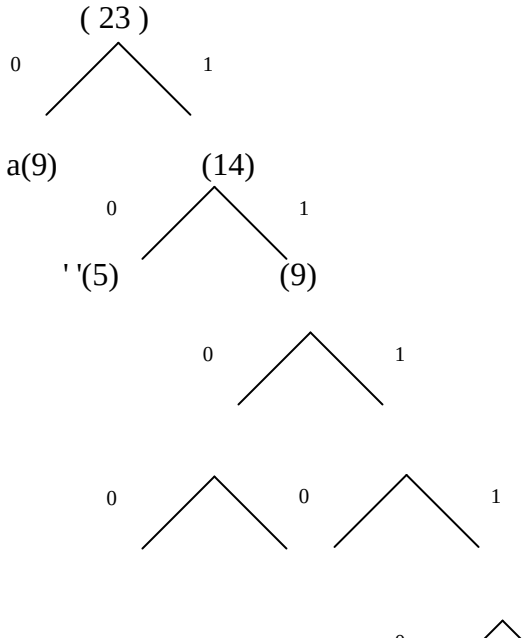


Y sucesivamente:

c(2) → l(2) → t(2) → (3) → ' '(5) → a(9)



El resultado final es:



(4) (5)

c (2) l(2) t(2) (3)

s(1) (2)

e(1) v(1)

5. Se asigna los códigos, las ramas a la izquierda son ceros, y a la derecha unos (por ejemplo), es una regla arbitraria.

a	' '	c	l	t	S	e	j
0	10	1100	1101	1110	11110	111110	111111

6. Y se traduce el texto:

a	t	a	' '	l	a	' '	j	a	c	a	' '	a	' '	l	a	' '	e	s	t	a	c	a
0	1110	0	10	1101	0	10	111111	0	1100	0	10	0	10	1101	0	10	111110	11110	1110	0	1100	0

Y sólo queda empaquetar los bits en grupos de ocho, es decir en bytes:

01110010	11010101	11111011	00010010	11010101	11110111	10111001	10000000
0x72	0xD5	0xFB	0x12	0xD5	0xF7	0xb9	0x80

En total ocho bytes, y el texto original tenía 23.

Considere una fuente S, de símbolos s1,s2,...,sq, y probabilidades P1, P2, ...Pq. determinar los códigos huffman para el siguiente grupo de caracteres:

S	a	B	c	d	e	f	g	h	i	j
P(X)	0.171	0.031	0.057	0.092	0.274	0.052	0.042	0.130	0.149	0.002

El resultado, siguiendo todos los pasos para la codificación es:

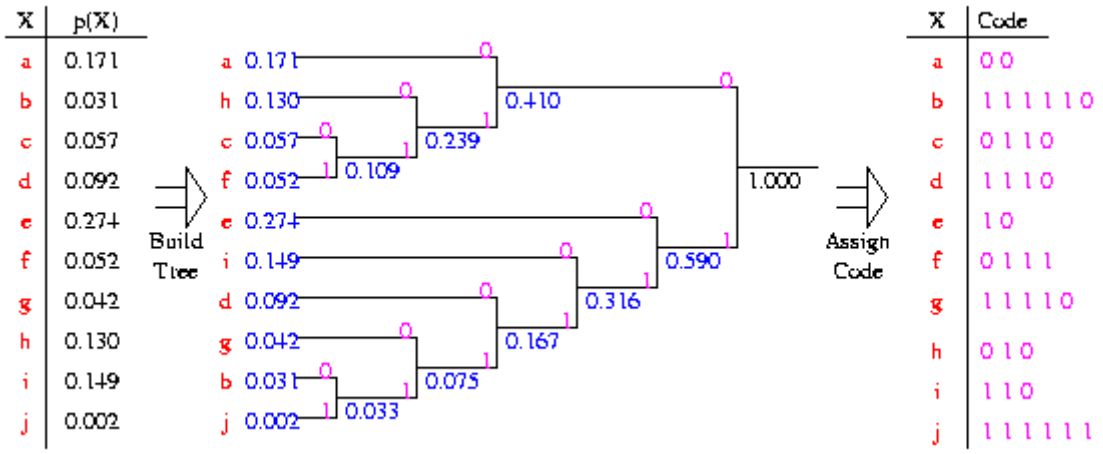


FIGURA V.1 Generación del árbol y los códigos asignados

5.3 IMPLEMENTACIÓN DEL SOFTWARE

Para la realización del presente proyecto se ha optado por el programa MATLAB por sus características y diversas aplicaciones relacionados con la ingeniería y la investigación científica.

Esta sección contiene las funciones creadas en MatLab, los archivos tipo m, que hacen la compresión – descompresión implementados para el código de Huffman Adaptativo.

MATLAB es un paquete de software interactivo de alto rendimiento para ingeniería en computación numérica y científica. El Matlab integra análisis numérico, computarización de matrices, procesamiento de señales y gráficos, en un medio ambiente fácil de usar. Por otra parte, MATLAB presenta un lenguaje de programación de muy alto nivel basado en vectores, arreglos y matrices.

Otra de las bondades es que MATLAB cuenta con un constructor de Interfaz Gráfico de Usuario GUI [15], con objetos gráficos como botones, cuadros de texto, deslizadores, menús, etc., permitiendo al usuario invocar la ejecución de ese u otro código desarrollado.

La función *norm2huff*

Ésta función calcula el vector comprimido y la información necesaria para el proceso de descompresión. Tiene un argumento de entrada llamado *vector*, que contiene los datos del archivo que se desea comprimir; y dos argumentos de salida denominados *zip* e *info* que son el vector comprimido y la estructura de información respectivamente.

La estructura de información *info* contiene toda la información necesaria para el proceso de descompresión. Está compuesta por los siguientes campos:

INFO.pad

Este campo indica cuántos bits se han añadido al vector comprimido para completar el último byte, su valor puede estar entre 1 y 7.

INFO.huffcod

Es un vector disperso (sparse) que contiene los códigos de Huffman, expresados como índices y los símbolos correspondientes, expresados como elementos de la matriz.

[15] MATLAB, para ciencias e ingeniería. Herón Morales Marchena

Debido a que un vector disperso está diseñado para manipular elementos diferentes de cero, los símbolos contenidos en este vector están incrementados en uno, de manera que podrán tener valores entre 1 y 256.

INFO.long

Indica la longitud del vector de datos original (sin comprimir).

INFO.max_lon

Contiene la longitud máxima de los códigos de Huffman generados.

INFO.rel_comp

Es la relación de compresión, se calcula dividiendo la longitud del vector original por la longitud del vector comprimido.

Descripción del funcionamiento

La primera operación que se realiza, consiste en calcular la probabilidad de cada símbolo, luego se eliminan los símbolos con probabilidad igual a cero y se ordenan con probabilidad ascendente.

El árbol de Huffman se genera utilizando un vector celda. Primero se crean nodos de valor 0 y de valor 1 con los dos elementos de menor probabilidad, para esto se utiliza la función *nuevo_nodo* que agrega un 1 ó un 0 al elemento actual y a los elementos precedentes. Luego se suman las probabilidades de estos dos símbolos y se reordena nuevamente el vector con probabilidad ascendente. Este procedimiento se repite hasta terminar con todos los elementos.

Los códigos calculados son asignados a la celda *codigo* de forma tal que exista una correspondencia símbolo – código, es decir, los valores de los símbolos son los índices del arreglo y los códigos de Huffman son los elementos, expresados como vectores de unos y ceros.

Utilizando los códigos generados anteriormente, se toma cada uno de los elementos del vector de datos y se concatena el código de Huffman correspondiente a la variable denominada *string* al final, si es necesario, se agregan ceros hasta que la longitud de la variable sea un múltiplo de 8. Al terminar el proceso, la variable *string* constituirá el vector comprimido expresado como una cadena de unos y ceros.

El siguiente paso consiste en convertir en un valor numérico los códigos de la celda *codigo* expresados como un vector de unos y ceros, pero es necesario notar que pueden existir ambigüedades en ciertos códigos, por ejemplo los códigos 110, 0110, 00110, etc., tienen el mismo equivalente numérico que es 6. Para evitar esta ambigüedad es necesario agregar un 1 en la posición más significativa de cada código, Así, los códigos 110, 0110, 00110 se convierten en 1110, 10110, 100110 y sus equivalentes numéricos serán: 14, 22 y 38 respectivamente.

Los valores numéricos de los códigos de Huffman se utilizan como índices del vector disperso *huffcod*. Los elementos correspondientes están conformados por los símbolos del vector de datos con su valor previamente incrementado en uno.

Se transforma el vector comprimido en la variable *string* en bytes, es decir un vector en formato *uint8*, esta operación se realiza con funciones orientadas al manejo de bits

a fin de lograr un menor tiempo de procesamiento. Finalmente se crea la estructura de información que contiene los campos ya explicados.

La función *huff2norm*

Realiza la función inversa a la función *norm2huff*. Tiene dos argumentos de entrada que son el vector comprimido *zip* y la estructura de información *info* y devuelve un argumento que es el vector original.

La estructura *info* es idéntica a la estructura *info* de la función *norm2huff*.

Descripción del funcionamiento

El vector comprimido *zip* se transforma en una cadena de unos y ceros almacenada en la variable *string*. Se eliminan los elementos excesivos indicados en el campo *pad* de la estructura *info.*

Sé inicializa el vector descomprimido con la longitud indicada en el campo *info.long*. Este procedimiento reduce considerablemente el tiempo de procesamiento porque no es necesario asignar memoria adicional al vector durante el proceso de descompresión.

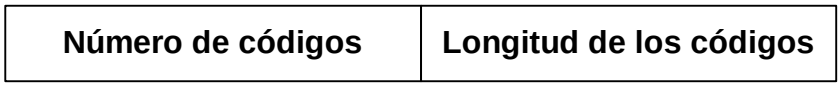
Para el proceso de descompresión se toma un elemento del vector y se verifica si corresponde a un código almacenado en el campo *info.huffcod*, para esto primero se fija en uno la posición más significativa. El proceso de búsqueda del símbolo equivalente se realiza con la función *decodificar*, que devuelve cero cuando el código no es válido y un valor diferente de cero cuando se encontró un código válido. Si es un código no válido, el procedimiento se repite pero tomando dos elementos, luego tres y así sucesivamente. Si es un código válido, se repite el procedimiento empezando con un elemento.

Al finalizar el proceso de decodificación, la variable *vector* contendrá los elementos del vector original.

La función *comprimir*

Ésta función da la información de la estructura *info* y los datos comprimidos en un solo vector, de tal forma que la información ocupe el menor espacio posible.

El primer paso consiste en comprimir los datos utilizando la función *norm2huff*. Luego hay que eliminar el elemento más significativo de cada código Huffman y se agrupan los códigos de Huffman de acuerdo a su longitud. La variable *num_len* contiene el número de códigos de una determinada longitud y su correspondiente longitud y la variable *ind_cod* contiene los símbolos existentes incrementados en uno ordenados de forma que exista correspondencia con la variable *num_len*.



Variable num_len

FIGURA V.2 Formato de la variable num_len

Por ejemplo, si existen 3 códigos de longitud 4, 10 de longitud 7, 4 de longitud 10 y 14 de longitud 11, el vector *num_len* contendrá los siguientes valores:

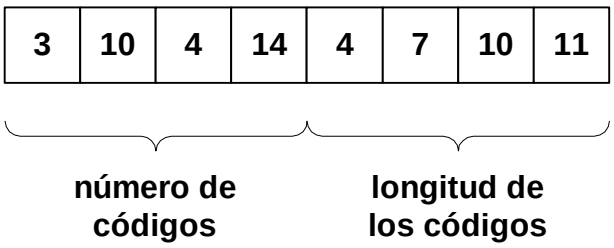


FIGURA V.3 Códigos de longitud variable

A continuación se representan los códigos como una cadena de unos y ceros y posteriormente como una cadena de bytes almacenada en la variable *byte_cod*. Se expresa la longitud del vector original como un dato de 8 bytes en la variable *lenv*.

La función comprimir tiene un argumento de salida, *datos*, que contiene los siguientes valores.

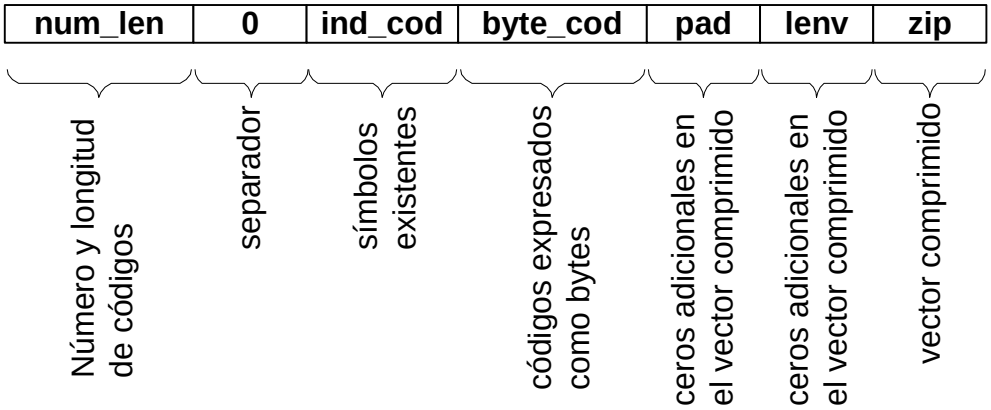


FIGURA V.4 Formato de la función comprimir

La función *informacion*

Tiene como argumento de entrada los datos de un archivo comprimido y como argumentos de salida el vector comprimido *zip* y la estructura de información, es decir, funciona de forma inversa a la función *comprimir*.

Descripción del funcionamiento

Primero se extrae el vector *num_len*, que está conformado por los elementos que preceden al separador (primer valor cero). De ésta variable se extraen el número de códigos de una determinada longitud y la longitud de los códigos.

Se recupera el vector *ind_cod*, cuyo número de elementos es igual a la suma del número de códigos, a los valores obtenidos hay que sumarles uno.

A continuación se obtiene el vector *byte_cod* que tiene un número de elementos igual al producto entre la longitud y el número de códigos dividido para 8, en el caso en que la división no sea exacta se toma el inmediato superior. Se transforma el vector *byte_cod* en una cadena de unos y ceros y luego se extrae el valor de cada código empleando los valores del vector *num_len*.

El siguiente elemento constituye el valor del *pad* y los 8 siguientes la longitud del vector original *lenv*. Todos los valores hasta ahora recuperados se anexan a la estructura *info* con los procedimientos descritos anteriormente. Los elementos restantes constituyen el vector comprimido.

El proceso de compresión de archivos

Para comprimir un archivo se siguen los siguientes pasos.

1. Se abre el archivo a ser comprimido con formato *uint8* y se almacena su contenido en un vector.

2. Se llama a la función comprimir, el argumento de entrada es el vector que contiene los datos del archivo abierto anteriormente.

3. Se almacenan los datos devueltos por la función comprimir en un archivo que constituye el vector comprimido.

El archivo comprimido puede tener cualquier nombre y extensión, sin embargo, en este programa el nombre del archivo comprimido se forma con el mismo nombre y extensión del archivo original y una extensión adicional.cmp.

El proceso de descompresión de archivos

Para descomprimir un archivo se siguen los siguientes pasos

1. Se abre el archivo comprimido con formato *uint8* y se almacena su contenido en un vector.

2. Se obtiene el vector comprimido y la estructura de información utilizando la función *informacion*.

3. Se descomprime el vector utilizando la función *huff2norm* y se guarda el vector descomprimido en un archivo.

El nombre del archivo descomprimido es el mismo del archivo comprimido pero sin la extensión .cmp.

Los códigos fuentes de cada una de estas funciones se muestran en los anexos (Anexo B).

5.3.1 EL INTERFASE GRÁFICO DE USUARIO (GUI)

Está conformado por dos archivos, un archivo .fig que constituye la parte gráfica del programa y un archivo .m denominado *archivo m de aplicación*. Que contiene las funciones necesarias para la operación del GUI.

El GUI de este programa tiene la apariencia mostrada a continuación.

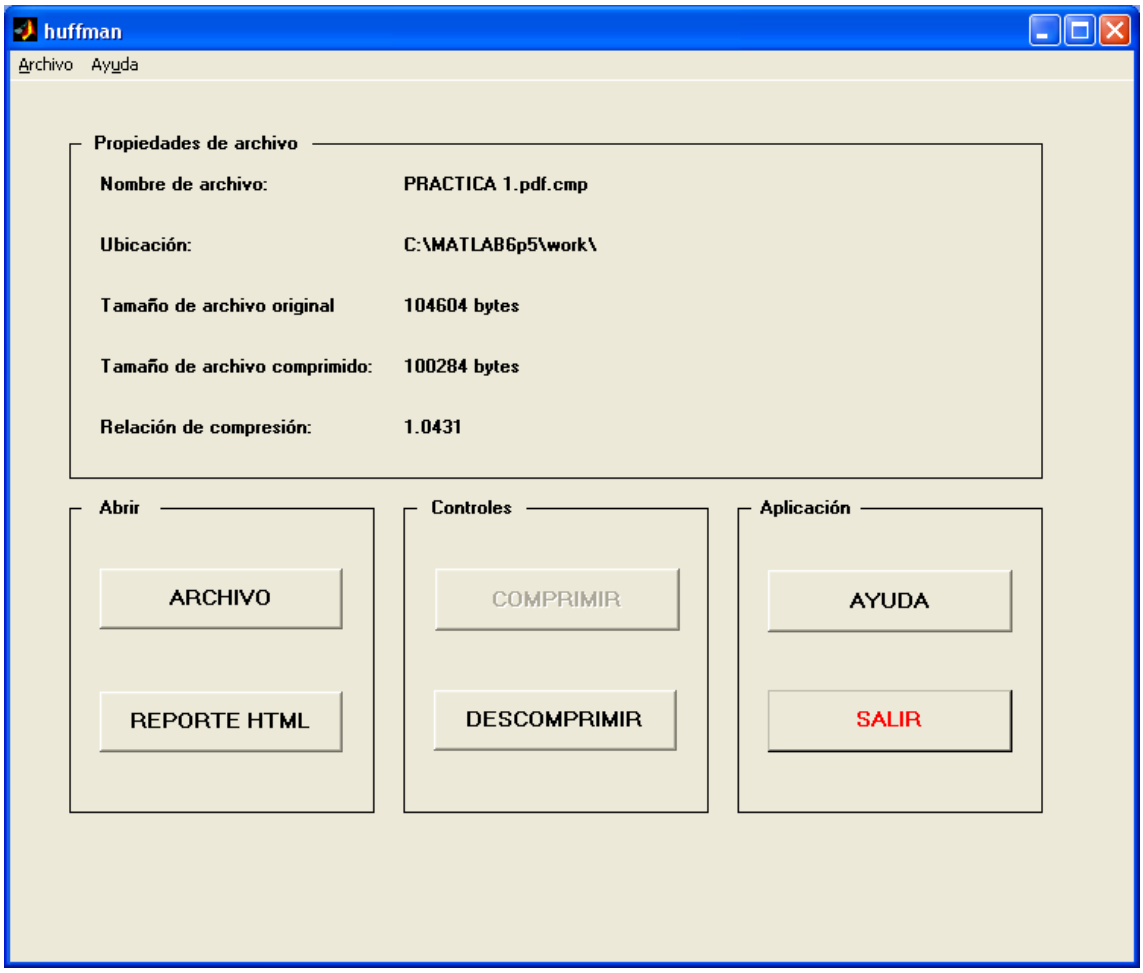


FIGURA V.5 Pantalla principal del GUI

La Barra de menús

Contiene las funciones necesarias para el manejo de archivos y ayudas.

MENÚ ARCHIVO

Abrir

Presenta el diálogo estándar de Windows para abrir archivos. Este diálogo soporta cualquier tipo de extensión o la extensión .cmp para archivos comprimidos.

Comprimir

Abre el diálogo estándar de Windows para selección de directorio (carpeta). Aquí se selecciona la carpeta de destino donde se desea comprimir el archivo. Si ya existe un archivo con el mismo nombre pregunta con una caja de diálogo si desea reemplazarlo.

Esta opción sólo está disponible si se ha abierto un archivo no comprimido (con una extensión diferente a .cmp).

Esta opción comprime los archivos utilizando el procedimiento de compresión descrito anteriormente.

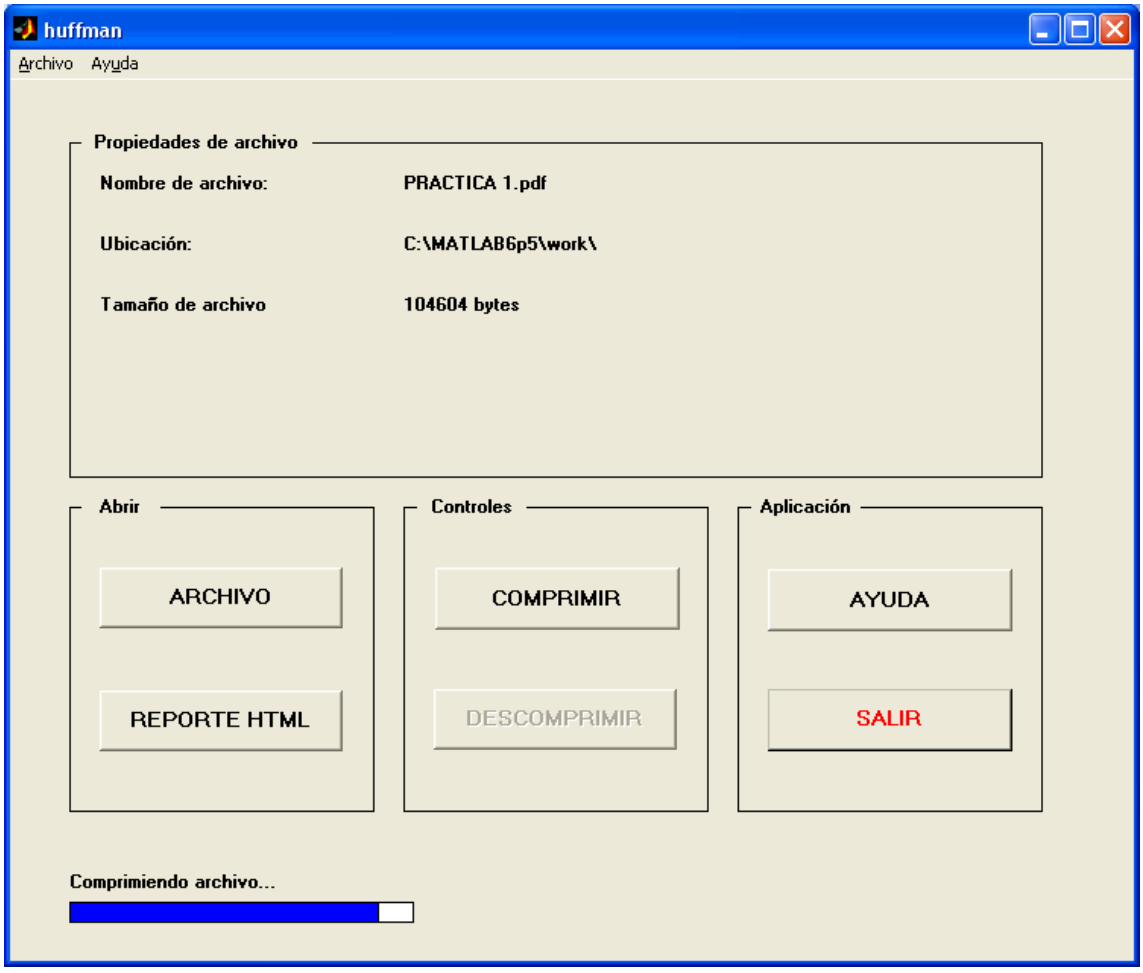


FIGURA V.6 Pantalla durante el proceso de compresión

Descomprimir

Abre el diálogo estándar de Windows para selección de directorio. Aquí se selecciona la carpeta de destino donde se desea descomprimir el archivo. Si ya existe un archivo con el mismo nombre pregunta con una caja de diálogo si desea reemplazarlo.

Esta opción sólo está disponible si se ha abierto un archivo comprimido (con una extensión .cmp).

Esta opción descomprime los archivos utilizando el procedimiento de descompresión descrito anteriormente.

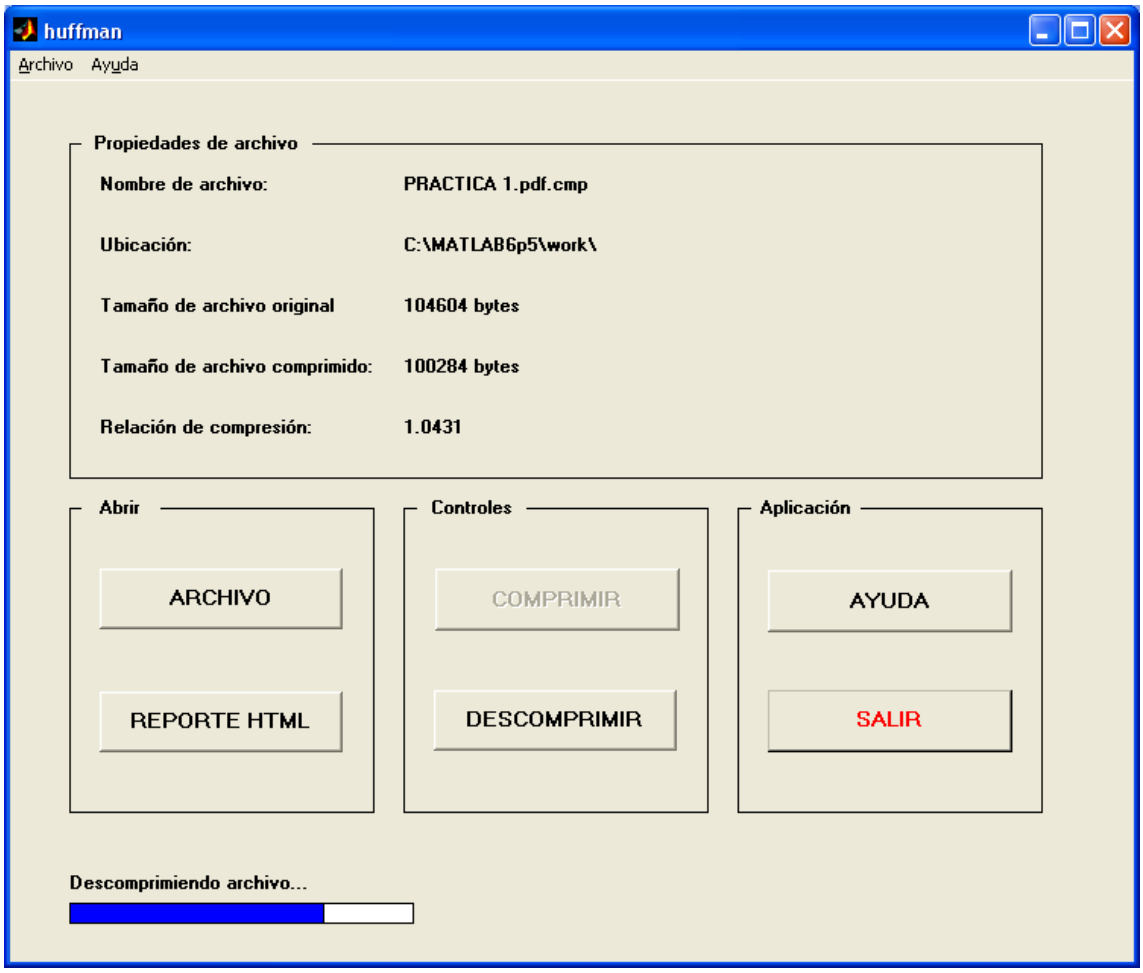


FIGURA V.7 Pantalla durante el proceso de descompresión

Reporte HTML

Genera un reporte del archivo actual y lo muestra en el navegador de Internet disponible en el computador. Proporciona un interfase amigable para ver, copiar e imprimir reportes. Esta opción hace uso del generador de reportes de Matlab.

Salir

Presenta un cuadro de diálogo en el que se pregunta al usuario si desea cerrar el programa. Si se selecciona No se continúa con la ejecución normal del programa, si se selecciona Sí se cierra detiene la ejecución del programa y se cierra Matlab.

MENÚ AYUDA

Temas de ayuda

Abre la ayuda de la aplicación en la ventana de ayuda de Matlab.

Acerca de

Muestra información específica de la aplicación: Versión de software, Autor y Fecha de creación.

CONTROLES

Controles de texto estático Propiedades de archivo

Muestran la información del archivo actual: Nombre, Ubicación, Tamaño, etc.

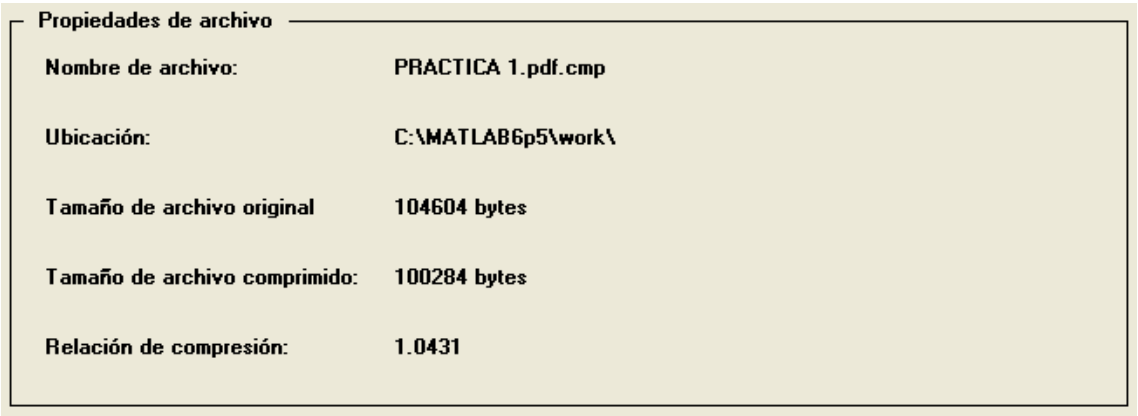


FIGURA V.8 Datos informativos del archivo

Grupos de botones

Permiten acceder de forma sencilla a las opciones de la barra de menús.

FIGURA V.9 Botones de acceso a los archivos

5.4 PRUEBAS Y ANÁLISIS DE RESULTADOS

Comprimir significa reducir el tamaño. Pero, ¿por qué se podría querer reducir el tamaño de un fichero de datos?.

La primera razón en obvia: para que quepa en un espacio menor. Actualmente los discos duros tienen una gran capacidad de almacenamiento pudiendo guardar en ellos muchos datos de gran tamaño. Sin embargo, si quiere transportar los datos en otro medio (disquete, CD-ROM…) el espacio está más limitado, y tal vez no quepan los datos que se necesita llevar con nosotros. Por ello hay que comprimirlos.

La segunda razón aparece fundamentalmente con Internet, y es el tiempo (que también se traduce en dinero) que se necesita para acceder o descargar ficheros de gran tamaño por la red. Por tanto, si esos datos están comprimidos se tardará menos (y se gastará menos) en enviarlos o recibirlos (por correo electrónico, web, ftp…).

Lógicamente la compresión suele llevar asociada una descompresión, con la que se pueda recuperar el formato original de esos datos para poder utilizarlos.

Esta compresión es posible porque normalmente en un fichero, además de información, hay también redundancia, es decir, datos que no aportan información, en general porque pueden obtenerse a partir de los datos anteriores. Lo que suelen

hacer los algoritmos de compresión es deshacerse de esa redundancia y optimizar así el espacio que ocupan los datos.

Otra de las razones es que para comprimir unos datos se requiere un tiempo de cálculo que no siempre estaremos dispuestos a esperar (si cada vez que guardo un documento en Word tengo que comprimirlo, tardaré bastante más que si lo guardo en el formato original). Lo mismo ocurre cuando quiera ver o utilizar esos datos, ya que se tendrá que descomprimirlos para recuperar su formato original y eso vuelve a requerir más tiempo.

Para comprimir los datos se utilizan distintos algoritmos, estando estos especialmente diseñados para cada tipo de dato. El resultado de la compresión será un fichero de menor tamaño (o igual, si no se ha conseguido comprimir nada) que el original y con un formato específico.

Hay programas que pueden comprimir y descomprimir distintos formatos de ficheros (por ejemplo, WinZip puede manejar ficheros del tipo ZIP sin considerar el tipo de fichero de ingreso,TAR, gzip y CAB).

Se puede dividir los formatos de compresión (o algoritmos) según el tipo de datos que manejan. De esta forma se tendrá formatos de compresión de archivos de datos como documentos, texto, programas, etc.; formatos de compresión de audio, de imágenes, de video…

5.4.1 PRUEBA DE FUNCIONAMIENTO DEL COMPRESOR

Para determinar el correcto funcionamiento del Compresor con el código Adaptativo Huffman, se realizaron e implementaron pruebas con algunos tipos de formatos, de cada prueba se obtuvo un archivo .cmp, el cual es generado por el compresor.

Tabla V.1 Parámetros obtenidos de las pruebas de compresión en diferentes tipos de archivos

NOMBRE DEL ARCHIVO ORIGINAL	FORMATO	TAMAÑO ORIGINAL bytes	NOMBRE DEL ARCHIVO COMPRIMIDO	FORMATO	TAMAÑO COMPRIMIDO bytes	RELACIÓN DE COMPRESIÓN
Prueba_1	.doc	308736	Prueba_1.doc	.cmp	211162	1.4621
Prueba_2	.doc	115712	Prueba_2.doc	.cmp	56826	2.0363
Prueba_3	.xls	78336	Prueba_3.xls	.cmp	37188	2.1065
Prueba_4	.xls	30720	Prueba_4.xls	.cmp	14076	2.1824
Prueba_5	.ptt	504832	Prueba_5.ptt	.cmp	482986	1.0452
Prueba_6	.ptt	2264064	Prueba_6.ptt	.cmp	2156059	1.0499
Prueba_7	.pdf	351317	Prueba_7.pdf	.cmp	328318	1.0701
Prueba_8	.pdf	946886	Prueba_8.pdf	.cmp	909557	1.041
Prueba_9	.jpeg	130072	Prueba_9.jpeg	.cmp	129681	1.003
Prueba_10	.jpeg	581450	Prueba_10.jpeg	.cmp	580874	1.001
Prueba_11	.zip	422492	Prueba_11.zip	.cmp	420934	1.0037
Prueba_12	File	2041023				
Prueba_13	.cmp	483526	Prueba_13	.cmp	482225	1.0027
CDEngine	.dll	958464	CDEngine.dll	.cmp	714323	1.3418
Si te vas	.mp3	2623488	Si te vas.mp3	.cmp	2620340	1.0012

Para analizar los resultados obtenidos, se realizaron pruebas de compresión de datos con el mismo entorno tecnológico, el formato más extendido en entornos Windows es el ZIP. Para este análisis comparativo se utilizó el programa de compresión de ficheros Win Zip, ya que es el más utilizado en el mundo. Es muy fácil de usar y permite comprimir, descomprimir, crear, editar, comprobar, gestionar y encriptar en los formatos de compresión más populares de Internet, como: ZIP, TAR, UUencode, XXencode, entre otros. Y su obtuvo los siguientes resultados, que se muestran en la tabla V.2.

Tabla V.2 Parámetros obtenidos de las pruebas de compresión con el Win Zip

NOMBRE DEL ARCHIVO ORIGINAL	FORMATO	TAMAÑO ORIGINAL bytes	NOMBRE DEL ARCHIVO COMPRIMIDO	FORMATO	TAMAÑO COMPRIMIDO bytes	RELACIÓN DE COMPRESIÓN
Prueba_1	.doc	308736	Prueba_1	.zip	103942	2.9703
Prueba_2	.doc	115712	Prueba_2	.zip	18804	6.1536
Prueba_3	.xls	78336	Prueba_3	.zip	15863	4.9383
Prueba_4	.xls	30720	Prueba_4	.zip	7537	4.0759
Prueba_5	.ptt	504832	Prueba_5	.zip	395494	1.2764
Prueba_6	.ptt	2264064	Prueba_6	.zip	1889195	1.1984
Prueba_7	.pdf	351317	Prueba_7	.zip	248846	1.4118
Prueba_8	.pdf	946886	Prueba_8	.zip	718863	1.3172

Prueba_9	.jpeg	130072	Prueba_9	.zip	128544	1.0119
Prueba_10	.jpeg	581450	Prueba_10	.zip	580025	1.0024
Prueba_11	.zip	422492				
Prueba_12	File	2041023	Prueba_12	.zip	1414050	1.4434
Prueba_13	.cmp	483526	Prueba_13	.zip	435537	1.1102
CDEngine	.dll	958464	CDEngine	.zip	346826	2.7635
Si te vas	.mp3	2623488	Si te vas	.zip	2603377	1.0077

La tabla V.3 presenta un cuadro comparativo entre el compressor desarrollado y el Win Zip.

Tabla V.3 Cuadro comparativo entre los compresores HUFFMAN ADAPTATIVO – WIN ZIP

NOMBRE DEL ARCHIVO ORIGINAL	TAMAÑO ORIGINAL Bytes	TAMAÑO COMPRIMIDO Bytes .cmp	TAMAÑO COMPRIMIDO Bytes .zip	RELACIÓN DE COMPRESIÓN HUFFMAN	RELACIÓN DE COMPRESIÓN WIN ZIP
Prueba_1.doc	308736	211162	103942	1.4621	2.9703
Prueba_2.doc	115712	56826	18804	2.0363	6.1536
Prueba_3.xls	78336	37188	15863	2.1065	4.9383
Prueba_4.xls	30720	14076	7537	2.1824	4.0759
Prueba_5.ptt	504832	482986	395494	1.0452	1.2764
Prueba_6.ptt	2264064	2156059	1889195	1.0499	1.1984
Prueba_7.pdf	351317	328318	248846	1.0701	1.4118
Prueba_8.pdf	946886	909557	718863	1.041	1.3172
Prueba_9.jpeg	130072	129681	128544	1.003	1.0119
Prueba_10.jpeg	581450	580874	580025	1.001	1.0024
Prueba_11.zip	422492	420934		1.0037	
Prueba_12 - file	2041023		1414050		1.4434
Prueba_13.cmp	483526	482225	435537	1.0027	1.1102
CDEngine.dll	958464	714323	346826	1.3418	2.7635
Si te vas.mp3	2623488	2620340	2603377	1.0012	1.0077

De las pruebas realizadas se comprobó que el compresor con el código Huffman Adaptativo funciona correctamente en un tiempo adecuado y con una capacidad de compresión significativa siempre y cuando los archivos de entrada sean archivos de tamaño relativamente pequeños mientras que para entradas de archivos de gran tamaño la compresión se lo realiza pero el tiempo de respuesta se eleva, asumiendo el autor que se debe al entorno tecnológico que se utiliza pues en este caso se requiere mayor capacidad de memoria que el entorno de prueba no proporciona.

Realizando una evaluación de ejecución del software a consideración se puede también resaltar sus limitaciones:

- **Huffman Adaptativo entre sus entradas no considera a directorios o también denominados carpetas, puesto que es creado específicamente para archivos.**
- **Huffman Adaptativo no excluye como archivos de entrada a aquellos que ya están comprimidos, ejecutando su acción pero sin ganar mucho peso de capacidad.**
- **Huffman Adaptativo como salida de proceso de aquellos archivos que se encuentran involucrados con fidelidad, alta resolución o similares no proporciona su ventaja de compresión en relación con el parámetro de compresión Win Zip.**

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

Éste trabajo está centrado solo en la compresión "**lossless**", es decir, técnicas que garantizan que no habrá ningún tipo de pérdida de información al comprimir los datos (factor fundamental para comprimir y recuperar programas, ficheros de bases de datos, etc). Las conclusiones y recomendaciones obtenidas del presente trabajo se indican a continuación.

6.1 CONCLUSIONES

- Se diseñó e implementó un compresor/descompresor de archivos con el Código Hufmman Adaptativo, el mismo que a más de realizar la compresión/descompresión, presenta un reporte de los datos con sus respectivas probabilidades de ocurrencia.

- Por medio de la investigación se ha incursionado en el campo de la compresión de datos, la cual es una técnica que, mediante procesos y algoritmos matemáticos, permite reducir el tamaño de los archivos para así facilitar la transferencia de los mismos, o su almacenamiento en discos duros, o cualquier otro soporte, aprovechado las herramientas tecnológicas, cumpliendo de esta manera con los objetivos trazados en el presente proyecto de tesis.

- El compresor/descompresor con el código Huffman Adaptativo se ha implementado en base al entorno computacional MATLAB, que es un software en continuo crecimiento y muy adaptable a los avances científicos, permitiendo

resolver los problemas que presenta la ingeniería en el desarrollo de productos innovadores.

- El algoritmo de compresión de Huffman se basa en asignar códigos de distinta longitud de bits a cada uno de los caracteres de un fichero. Si se asignan códigos más cortos a los caracteres que aparecen más a menudo se consigue una compresión del fichero.

- MATLAB al tener una amplia colección de Toolboxes y poseer una interfaz gráfica, resultó muy versátil para el desarrollo del presente proyecto, pues a partir de una determinada aplicación se puede extender su utilidad a proyectos que contengan características similares.

- Una ventaja del compresor con Código Huffman adaptativo es que puede volver a comprimir archivos comprimidos que tengan la extensión .cmp, lo que no sucede con los otros compresores conocidos.

- Se demostró que la codificación Huffman es el método de compresión más efectivo de entre los de su clase; cuando los símbolos aparecen en el texto con las mismas frecuencias que las que se utilizaron para construir el código

- Al realizar las pruebas con el compresor/descompresor, los resultados obtenidos muestran una similitud con los generados por otros programas de compresión existentes.

- De las pruebas realizadas se comprobó que el compresor con el código Huffman Adaptativo funciona correctamente en un tiempo adecuado y con una capacidad de compresión significativa siempre y cuando los archivos de entrada sean archivos de tamaño relativamente pequeños mientras que para entradas de archivos de gran tamaño la compresión se la realiza pero el tiempo de respuesta se eleva. Hay que tener en cuenta que MATLAB es un lenguaje interpretado y no compilado, por lo que su ejecución es lenta.

- El compresor/descompresor Huffman cuenta con las características necesarias que requiere un programa para comprimir especialmente texto, por lo que puede ser utilizado en los computadores personales, brindando de esta manera una aplicación en el área de la compresión de datos.

- En base a los resultados obtenidos en las pruebas de compresión se puede concluir que la relación de compresión es aceptable.

Realizando una evaluación de ejecución del software a consideración se puede también resaltar sus limitaciones:

- **Huffman Adaptativo entre sus entradas no considera a directorios o también denominados carpetas, puesto que es creado específicamente para archivos.**

- **Huffman Adaptativo no excluye como archivos de entrada a aquellos que ya están comprimidos, ejecutando su acción pero sin ganar mucho peso de capacidad.**

- Huffman Adaptativo como salida de proceso de aquellos archivos que se encuentran involucrados con fidelidad, alta resolución o similares no proporciona su ventaja de compresión, en relación con el parámetro de compresión Win Zip.

- La compresión es posible porque normalmente en un fichero además de información hay también redundancia (en ésta tesis no se aplica el concepto de redundancia), es decir, datos que no aportan más información, en general porque pueden obtenerse a partir de los datos anteriores. Lo que suelen hacer los algoritmos de compresión es deshacerse de esa redundancia y optimizar así el espacio que ocupan los datos.

- Es importante tener los conocimientos de la teoría de compresión y de sus distintas técnicas de compresión que emplean diferentes algoritmos para la realización de la misma.

6.2 RECOMENDACIONES

- El proyecto desarrollado realizó la compresión y descompresión empleando el código Huffman Adaptativo, se recomienda continuar con la investigación y aplicarlo en otro campo de la teoría de información como es el caso de seguridad ó encriptación de archivos de texto.

- Utilizar máquinas con procesadores de alta velocidad de procesamiento, para evitar los tiempos largos del proceso de compresión.

- Leer el manual de operación antes de realizar la compresión de archivos con el fin de no cometer errores en la realización del proceso de compresión/descompresión.

- Es importante considerar los tamaños de los archivos al momento de realizar la compresión del mismo, debido a que puede demorarse más tiempo en el proceso de compresión, y puede causar que la aplicación no responda de una manera adecuada.

- A quienes estén interesados en el área de la compresión integrarse a foros virtuales para poder compartir información y novedades relaciones con este campo.

- Realizar trabajos de investigación relacionados con la compresión de datos empleando el mismo software Matlab o crear variantes de la aplicación desarrollada con el fin de hacer ejecutables los programas desarrollados.

BIBLIOGRAFÍA

- COUCH II LEON W. “Sistemas de Comunicación Digitales y Analógicos”, Prentice Hall, México. 1998.
- WAYNE TOMASI. “Sistemas de Comunicaciones Electrónicas”, Prentice may, México. 1996.
- SMITH STEVEN W. “Digital Signal Proccessing a Practical Guide for Engineers and Scientists”, Newnes, USA. 2003.
- NARANJO H. CÉSAR A. “Simulación de Codificación de Canal”, ESPE – Latacunga. 1996.
- GRIVET MATTOSO M. A. “Curso de Comunicacões Digitais”, Río de Janeiro.
- MORALES MARCHENA HERÓN. “MATLAB 7 para ciencias e ingeniería, Métodos numéricos y visualización gráfica” Megabyte, Lima. 2005.
- ENCICLOPEDIA “Técnico en Telcomunicaciones”, Cultutal, España. 2002.
- BLELLOCH GUY E. “Introduction to Data Compresion”, manuscrip.pdf.2001.
- WILLIAMS R.N. “Adaptive Data Compression”. Kluwer Academic, 1991.

ENLACES

- http://www.it.uc3m.es/~jmoreno/telematica/servidor/apuntes/tema2/tema02.htm
- http://www.tecnociencia.es/mediawiki/index.php/Teoría_de_la_información/htm
- http://www.tecnociencia.es/mediawiki/index.php/Compresion_de_datos/htm
- http://www.ux.his.no/~karlsk/proj99/htm
- www.guia academica teoria de la informacion.htm
- www.Data Compression/Section 10.htm
- http://www.tecnova.es/ti/historias05.htm. 2001
- www.Data_Compression-info.com

ANEXO A

MANUAL DE OPERACION

MANUAL DE OPERACIÓN

REQUERIMIENTOS DEL HARDWARE

En cuanto al hardware, el programa exige características básicas como un PC-Compatible con microprocesador Pentium III o IV para trabajar sobre Windows 95, 98, Me, NT, XP o Windows 2000, con tarjeta gráfica VGA y monitor a color. Son imprescindibles 128 Mbytes de memoria RAM, disco duro con un espacio libre de 1Gbyte si se va a utilizar todo el sistema, un ratón y unidad de CD-ROM.

INSTALACION DEL SOFTWARE

- Verificar si tiene instalado MATLAB 6P5, caso contrario realizar la instalación del software.

- Grabar la carpeta Huffman en el directorio Work de MATLAB.

- Ejecute el programa MATLAB desde el icono que se encuentra en el escritorio, seleccione la opción **File** a continuación seleccione **Set Path** luego de un clic sobre el botón **Add Fólder,** seleccione la carpeta **Huffman** y finalmente de un clic en **aceptar** y seleccione **save** y **close**, de esta manera Ud. a cargado todos los archivos de la carpeta Huffman.

INSTRUCCIONES DE USO.

Para ejecutar el programa huffman, se puede realizar de dos maneras:

- A través de la ventana de comandos de MATLAB (Command Window), digite huffman presione **Enter.**
- Ó desde el escritorio de un clic sobre el icono huffman.

Huffman

Este icono tiene las siguientes características:

- Tipo de destino : Aplicación
- Ubicación : winn32
- Destino : C:\MATLAB6p5\bin\win32\matlab.exe /r huffman /automation
- Iniciar en: C:\MATLAB6p5\work

Lo que hará que aparezca la ventana principal.

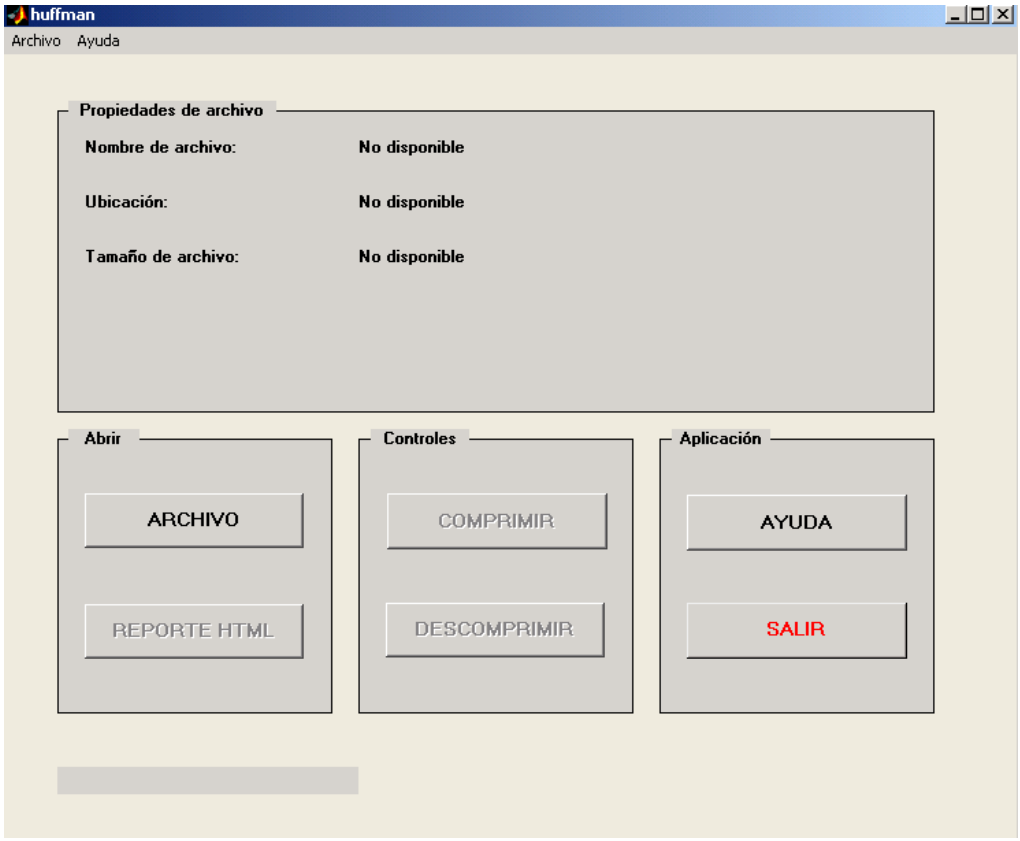


FIGURA 1. Ventana principal

En la parte superior de la ventana principal, se presenta la barra de menús de la aplicación, la cual contiene las funciones necesarias para el manejo de archivos y ayudas.

Menú Archivo:

- Abrir.
- Comprimir
- Descomprimir
- Reporte HTML
- Salir

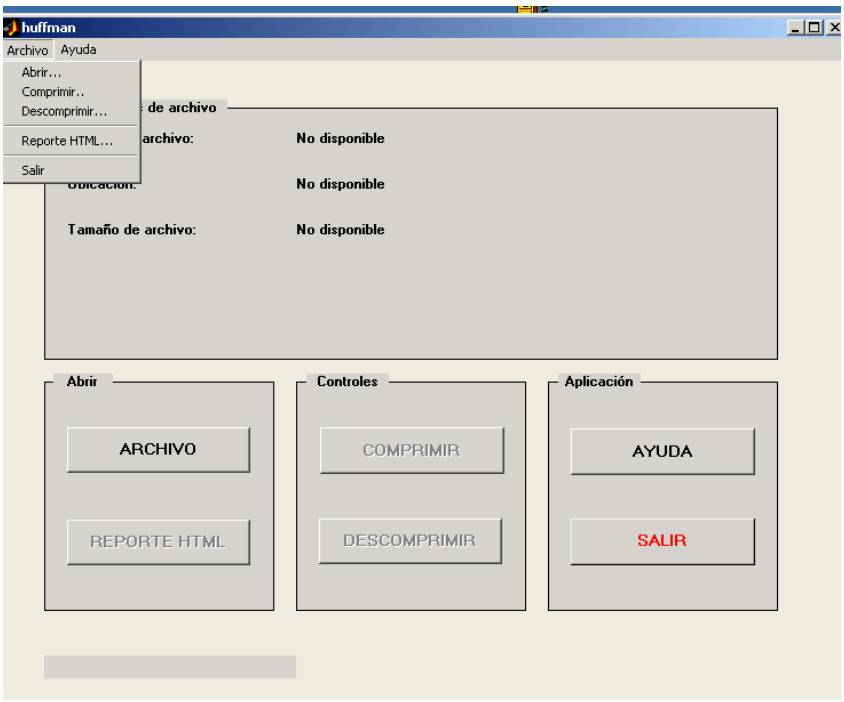


FIGURA 2. Menú Archivo

Menú Ayuda:

- Temas de ayuda
- Acerca de

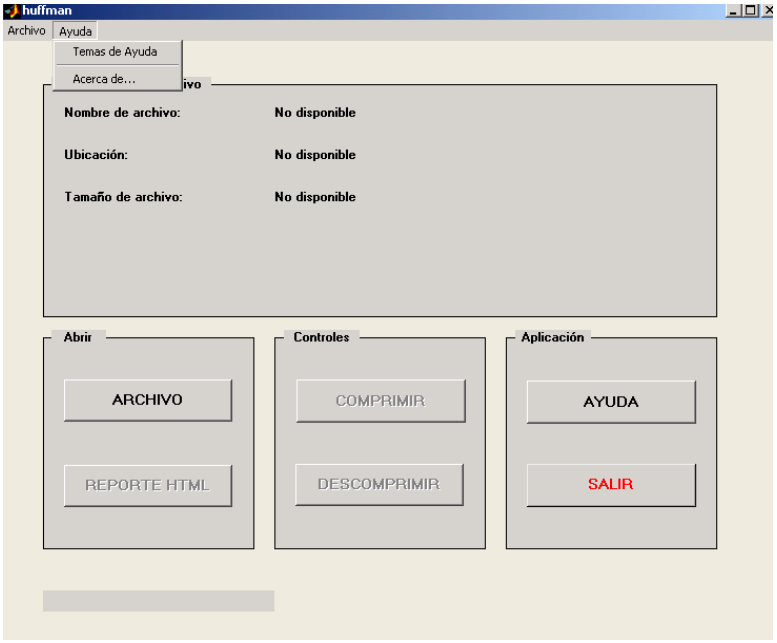


FIGURA 3. Menú Ayuda

ABRIR ARCHIVO

Al seleccionar esta opción se presenta el diálogo estándar de Windows para abrir archivos. Este diálogo soporta cualquier tipo de extensión o la extensión .cmp para archivos comprimidos, observe que los controles comprimir y descomprimir se encuentran deshabilitados, antes de que seleccione el archivo.

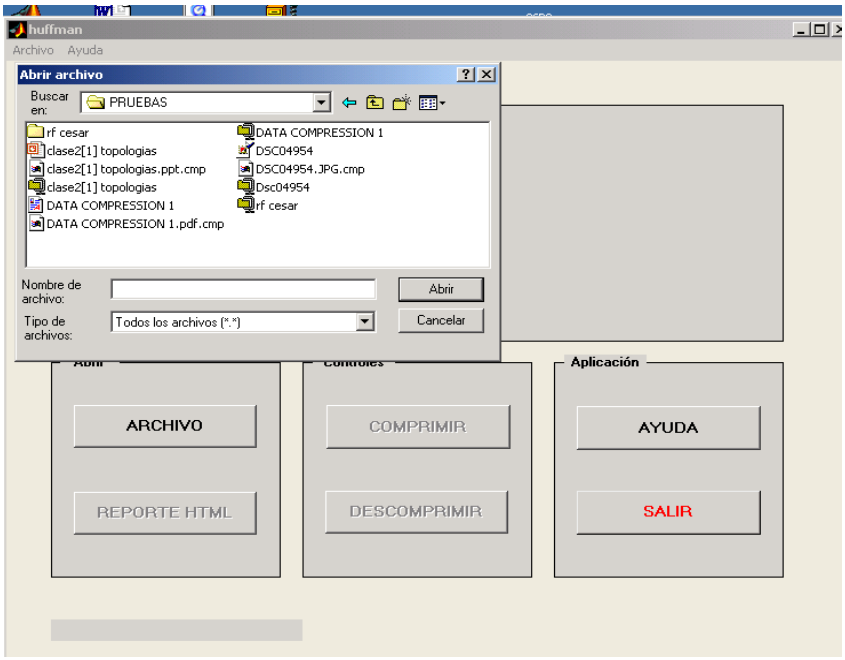


FIGURA 4. Opción Abrir

Cuando se tenga seleccionado el archivo presionar *abrir*, la ruta de acceso del archivo abierto se indica en la pantalla (almacenadas en las variables globales *pn* y *fn*) y los datos del archivo como es el tamaño (se almace en la variable global y). Ahora se habilita el control comprimir.

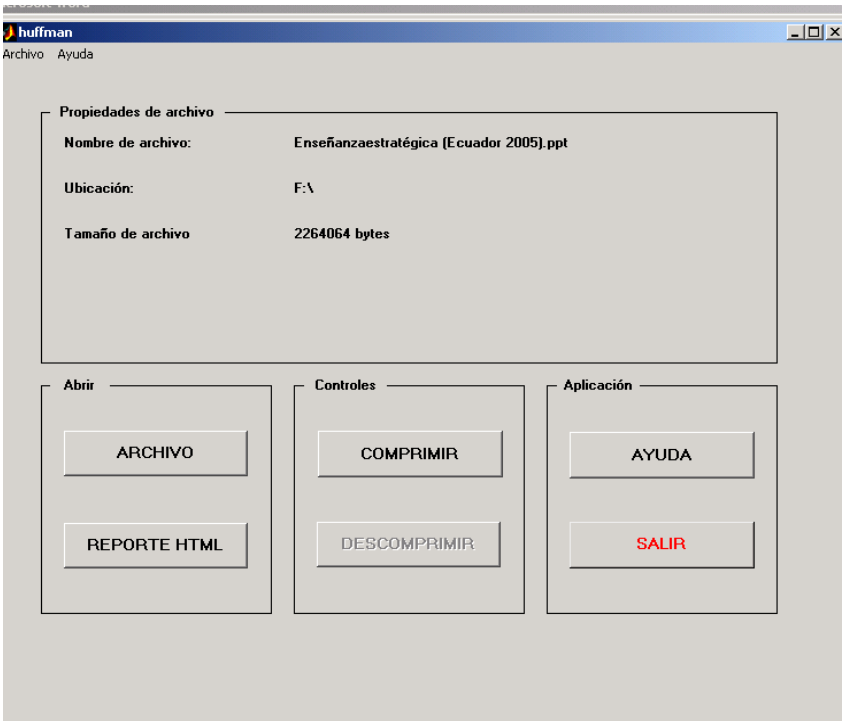


FIGURA 5. Opción Abrir con selección de archivo

COMPRIMIR ARCHIVO

Abre el diálogo estándar de Windows para selección de directorio (carpeta). Aquí se selecciona la carpeta de destino donde se desea comprimir el archivo. Si ya existe un archivo con el mismo nombre pregunta con una caja de diálogo si desea reemplazarlo. Esta opción sólo está disponible si se ha abierto un archivo no comprimido (con una extensión diferente a .cmp).

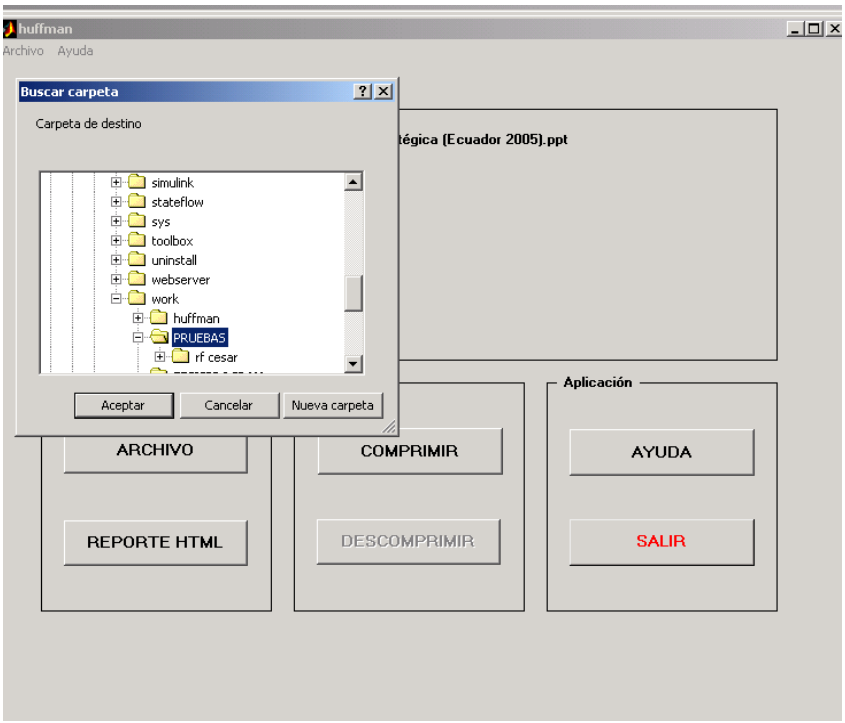


FIGURA 6. Opción Comprimir

Una vez seleccionada la carpeta de destino se presiona Aceptar. Comprime los archivos utilizando el procedimiento de compresión descrito anteriormente.

FIGURA 7. Proceso de compresión

Una vez terminado el proceso de compresión, el archivo comprimido se almacena en la carpeta de destino seleccionada y genera el mensaje después de unos segundos (o más, según el tamaño del archivo) "*El archivo fue comprimido*". En este momento, se tiene tanto el archivo original como el comprimido, como se puede observar en la siguiente figura.

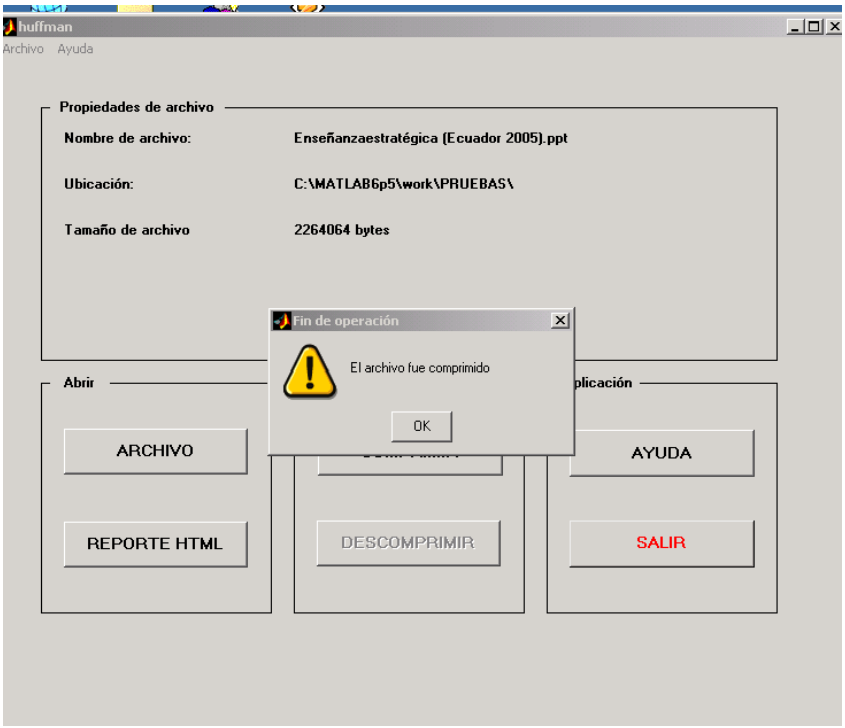


FIGURA 8. Aviso de finalización del proceso de compresión

DESCOMPRIMIR ARCHIVO

Abre el diálogo estándar de Windows para selección de directorio. Aquí se selecciona la carpeta de destino donde se desea descomprimir el archivo. Si ya existe un archivo con el mismo nombre pregunta con una caja de diálogo si desea reemplazarlo.

Esta opción sólo está disponible si se ha abierto un archivo comprimido (con una extensión .cmp).

Esta opción descomprime los archivos utilizando el procedimiento de descompresión descrito anteriormente.

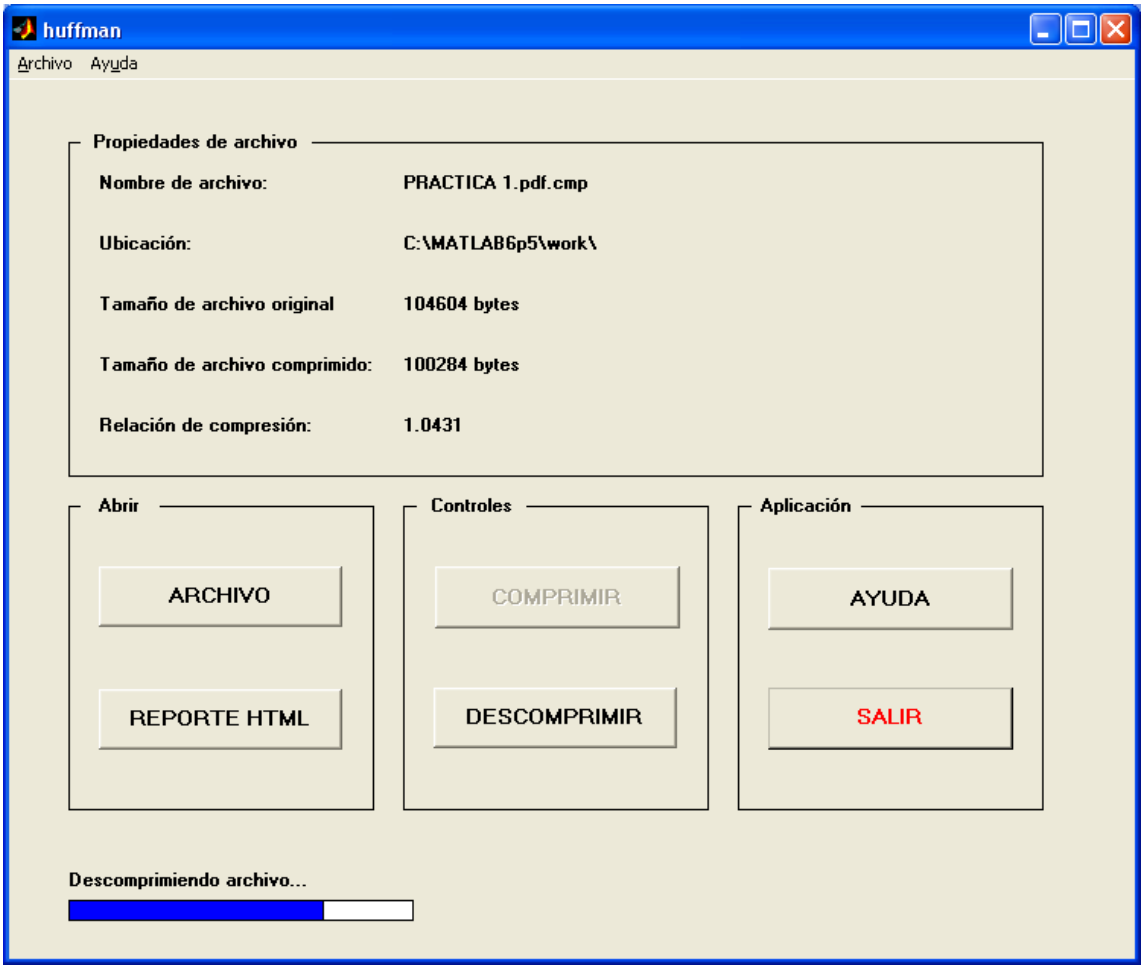
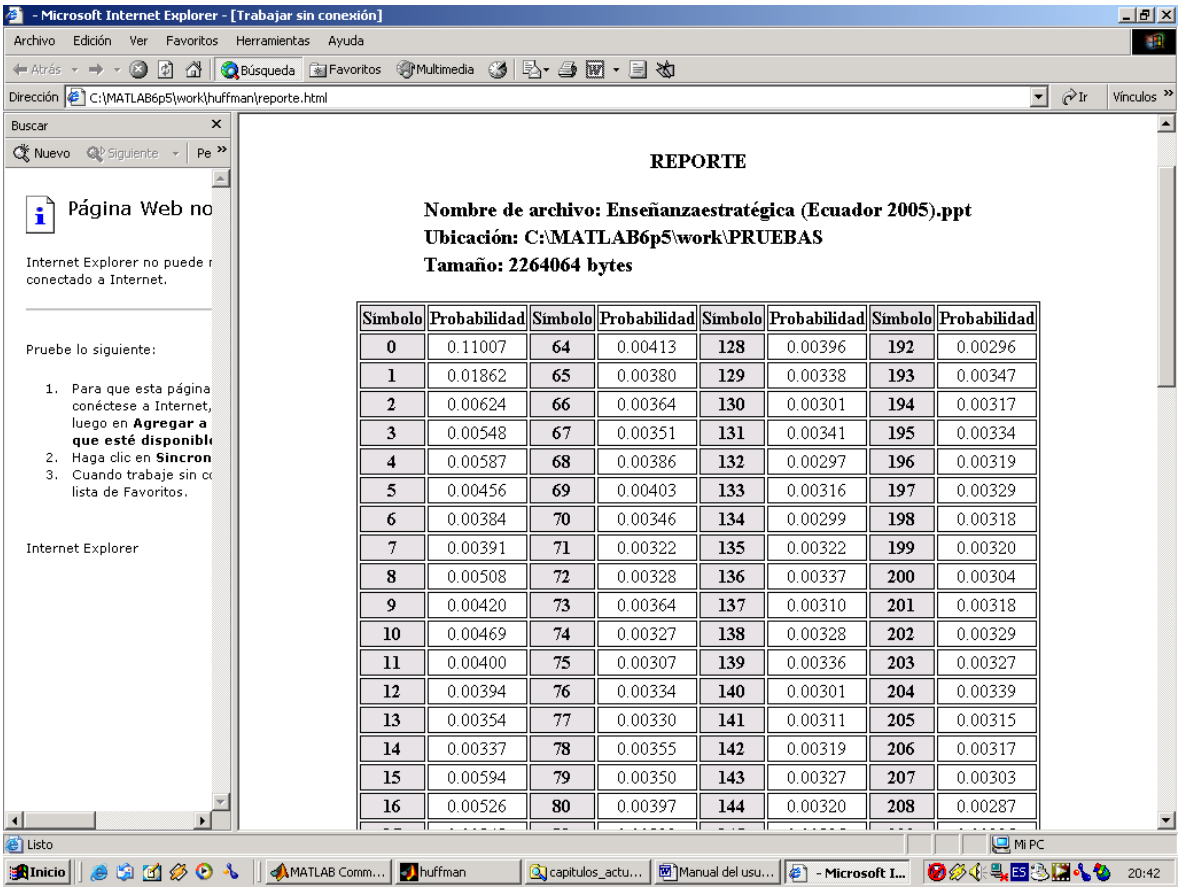


FIGURA 9.Opción Descomprimir

REPORT HTML DEL ARCHIVO

Genera un reporte del archivo actual y lo muestra en el navegador de Internet disponible en el computador. Proporciona un interfase amigable para ver, copiar e imprimir reportes. Esta opción hace uso del generador de reportes de Matlab.



Símbolo	Probabilidad	Símbolo	Probabilidad	Símbolo	Probabilidad	Símbolo	Probabilidad
0	0.11007	64	0.00413	128	0.00396	192	0.00296
1	0.01862	65	0.00380	129	0.00338	193	0.00347
2	0.00624	66	0.00364	130	0.00301	194	0.00317
3	0.00548	67	0.00351	131	0.00341	195	0.00334
4	0.00587	68	0.00386	132	0.00297	196	0.00319
5	0.00456	69	0.00403	133	0.00316	197	0.00329
6	0.00384	70	0.00346	134	0.00299	198	0.00318
7	0.00391	71	0.00322	135	0.00322	199	0.00320
8	0.00508	72	0.00328	136	0.00337	200	0.00304
9	0.00420	73	0.00364	137	0.00310	201	0.00318
10	0.00469	74	0.00327	138	0.00328	202	0.00329
11	0.00400	75	0.00307	139	0.00336	203	0.00327
12	0.00394	76	0.00334	140	0.00301	204	0.00339
13	0.00354	77	0.00330	141	0.00311	205	0.00315
14	0.00337	78	0.00355	142	0.00319	206	0.00317
15	0.00594	79	0.00350	143	0.00327	207	0.00303
16	0.00526	80	0.00397	144	0.00320	208	0.00287

FIGURA 10. Pantalla de reporte

SALIR DEL PROGRAMA

Presenta un cuadro de diálogo en el que se pregunta al usuario si desea cerrar el programa. Si se selecciona No se continúa con la ejecución normal del programa, si se selecciona Sí se cierra detiene la ejecución del programa y se cierra Matlab.

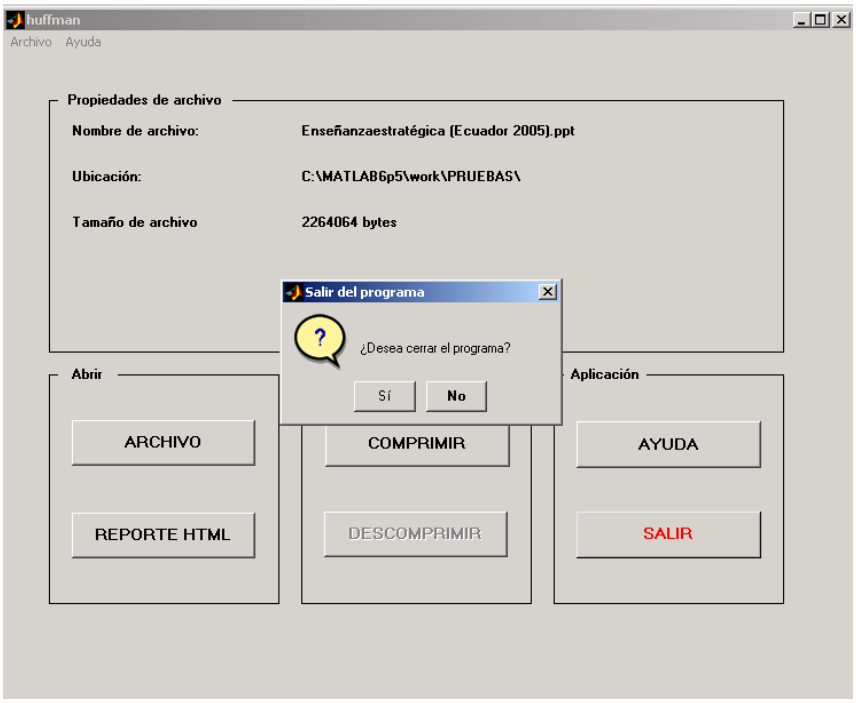


FIGURA 11. Opción Salir

MENÚ AYUDA

Temas de ayuda

Abre la ayuda de la aplicación en la ventana de ayuda de Matlab.

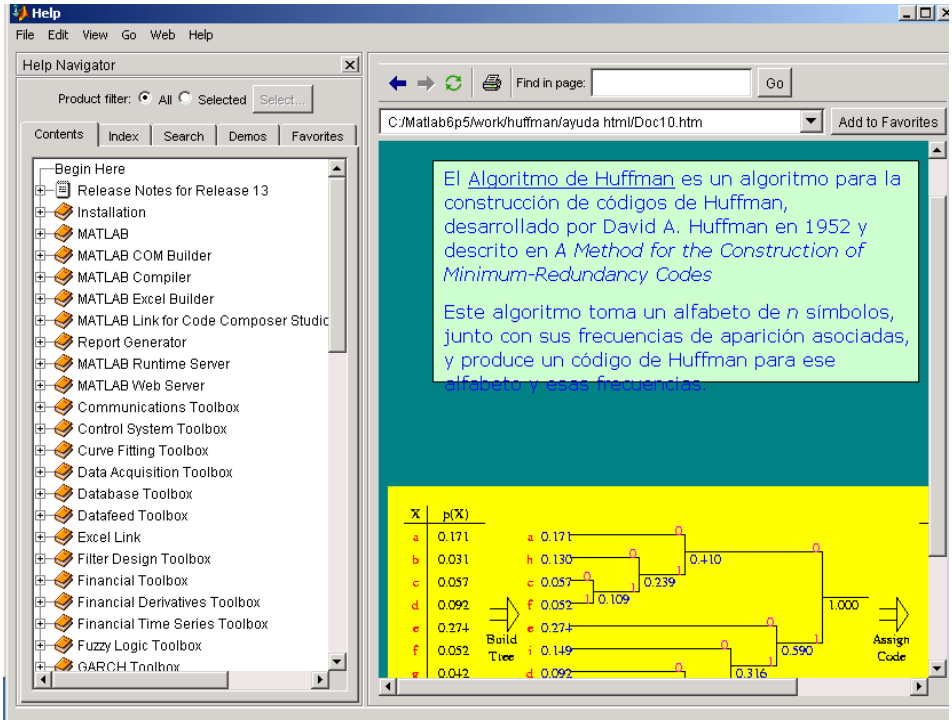


FIGURA 12. Opción Salir

Acerca de...

Muestra información específica de la aplicación: Versión de software, Autor y Fecha de creación.

FIGURA 13. Opción Acerca de

CONTROLES

Controles de texto estático Propiedades de archivo

Permite ver las propiedades del archivo actual procesado o a procesar: Nombre, Ubicación, Tamaño, etc.

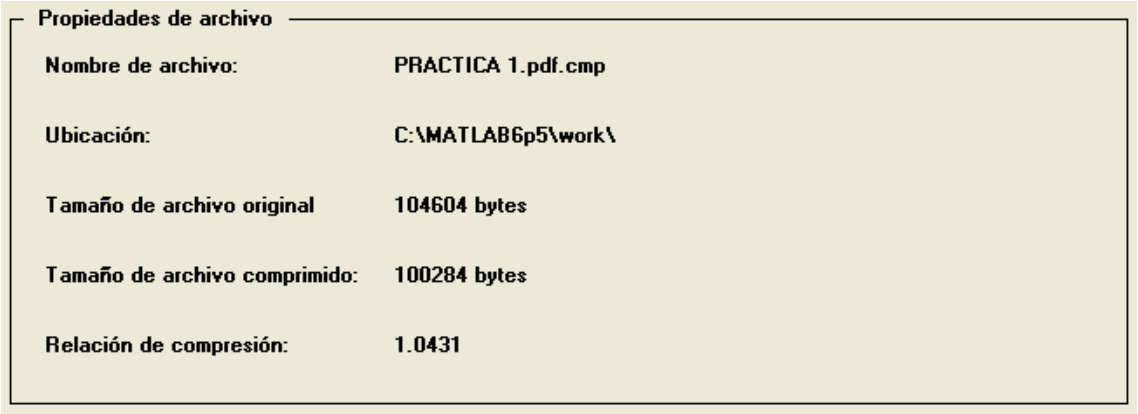


FIGURA 14. Formato de los controles de texto

Grupos de botones

Permiten acceder de forma sencilla a las opciones de la barra de menús.

FIGURA 15. Formato del grupo de botones

ANEXO B

PROGRAMACIÓN. CÓDIGOS FUENTE

PROGRAMA VENTANA PRINCIPAL

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @huffman_OpeningFcn, ...
   'gui_OutputFcn',  @huffman_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end


function huffman_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


%%%%%% CENTRAR LA VENTANA EN LA PANTALLA%%%%%%

FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');
set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');
FigWidth = OldPos(3);
FigHeight = OldPos(4);
ScreenUnits=get(0,'Units');
set(0,'Units','pixels');
ScreenSize=get(0,'ScreenSize');
set(0,'Units',ScreenUnits);

FigPos(1)=1/2*(ScreenSize(3)-FigWidth);
FigPos(2)=2/3*(ScreenSize(4)-FigHeight);
FigPos(3:4)=[FigWidth FigHeight];
```

```
set(hObject, 'Position', FigPos);
set(hObject, 'Units', OldUnits);


%%%%%% CONFIGURACION INICIAL DE LOS ITEMS DEL MENÚ%%%%%%

set(handles.abrir,...
   'Accelerator',  'A')
set(handles.comprimir,...
   'Enable',       'off',...
   'Accelerator',  'C')
set(handles.descomprimir,...
   'Enable',       'off',...
   'Accelerator',  'D')
set(handles.reporte,...
   'Enable',       'off',...
   'Accelerator',  'R')
set(handles.salir,...
   'Accelerator',  'Q')
set(handles.temas,...
   'Accelerator',  'H')


%%%%%% PROGRAMA PARA ABRIR ARCHIVOS%%%%%%

function abrir_Callback(hObject, eventdata, handles)

global y
global zip
global info
global pn
global fn

[fn_aux pn_aux]=uigetfile({...
       '*.*','Todos los archivos (*.*)'
   '*.cmp','Archivos comprimidos (*.cmp)'},...
   'Abrir archivo');

pf=[pn_aux fn_aux];

if fn_aux == 0
   return
end

pn = pn_aux;
fn = fn_aux;

fid = fopen(pf);
y = fread(fid,'uint8');
fclose(fid);


set(handles.txt_nombre,'String',fn)
set(handles.txt_ubicacion,'String',pn)
```

```
set(handles.reporte,'Enable','on')
set(handles.reporte2,'Enable','on')

s = fliplr(fn);
n = strfind(s,'.');
tipo = fliplr(s(1:n));

if strcmp(tipo,'.cmp')
   [zip,info] = informacion(y);


%%%%%% CONFIGURACION DE CONTROLES DE TEXTO%%%%%%

   set(handles.tamano1,'String','Tamaño de archivo original')
   set(handles.txt_tamano2,'Visible','on')
   set(handles.txt_relacion,'Visible','on')
   set(handles.tamano2,'Visible','on')
   set(handles.relacion,'Visible','on')

%%%%%%INICIALIZACIÓN DE CONTROLES DE TEXTO%%%%%%

   set(handles.txt_tamano1,'String',[num2str(info.long) ' bytes'])
   set(handles.txt_tamano2,'String',[num2str(length(zip)) ' bytes'])
   set(handles.txt_relacion,'String',num2str(info.rel_comp))

%%%%%% CONFIGURACION DE ITEMS DE MENU%%%%%%

   set(handles.comprimir,'Enable','off')
   set(handles.descomprimir,'Enable','on')

   set(handles.comprimir2,'Enable','off')
   set(handles.descomprimir2,'Enable','on')

else

%%%%%% CONFIGURACION DE CONTROLES DE TEXTO%%%%%%

   set(handles.txt_tamano2,'Visible','off')
   set(handles.txt_relacion,'Visible','off')
   set(handles.tamano2,'Visible','off')
   set(handles.relacion,'Visible','off')

%%%%%%INICIALIZACIÓN DE CONTROLES DE TEXTO%%%%%%

   set(handles.tamano1,'String','Tamaño de archivo')
   set(handles.txt_tamano1,'String',[num2str(length(y)) ' bytes'])

%%%%%% CONFIGURACION DE ITEMS DE MENU%%%%%%

   set(handles.comprimir,'Enable','on')
   set(handles.descomprimir,'Enable','off')

   set(handles.comprimir2,'Enable','on')
   set(handles.descomprimir2,'Enable','off')
```

```
end
%%%%%% PROGRAMA PARA COMPRIMIR%%%%%%

function comprimir_Callback(hObject, eventdata, handles)

global y
global pn
global fn

pn_aux=uigetdir(pn,'Carpeta de destino');

if pn_aux==0
   return
elseif length(pn_aux)==3
   pf=[pn_aux fn];
else
   pf=[pn_aux '\' fn];
end
pn = pn_aux;

pf=[pf '.cmp'];
fid = fopen(pf);
if fid>0
   qd = questdlg(['El archivo ',fn,'.cmp',' ya existe. ¿Desea reemplazarlo?'],'Archivo ya
existente','Si','No','No');
   fclose(fid);
   if ~strcmp(qd,'Si')
      return
   end
   set(handles.figure1,'Pointer','watch')
   set(handles.estado,'String','Comprimiendo archivo...')
   datos = compresion(y,handles);
   fid = fopen(pf,'w');
   fwrite(fid,datos,'uint8');
   fclose(fid);
   set(handles.estado,'String','')
   set(handles.figure1,'Pointer','arrow')
   msgbox('El archivo fue comprimido','Fin de operación','warn','modal')

else
   set(handles.figure1,'Pointer','watch')
   set(handles.estado,'String','Comprimiendo archivo...')
   set(handles.progreso,'Visible','on')
   datos = compresion(y,handles);
   fid = fopen(pf,'w');
   fwrite(fid,datos,'uint8');
   fclose(fid);
   set(handles.estado,'String','')
   set(handles.figure1,'Pointer','arrow')
   msgbox('El archivo fue comprimido','Fin de operación','warn','modal')

end
```

```
%%%%%% PROGRAMA PARA GENERAR EL REPORTE HTLM%%%%%%

function reporte_Callback(hObject, eventdata, handles)

global y
global info
global pn
global fn

s = fliplr(fn);
n = strfind(s,'.');
tipo = fliplr(s(1:n));

if strcmp(tipo,'.cmp')
   crear_rep2(y,fn,pn,info)
else
   crear_rep(y,fn,pn)
end

set(handles.figure1,'Pointer','watch')
set(handles.estado,'String','Generando Reporte...')
report('reporte');
set(handles.estado,'String','')
set(handles.figure1,'Pointer','arrow')

%%%%%% PROGRAMA PARA SALIR DE LA APLICACION%%%%%%

function salir_Callback(hObject, eventdata, handles)

close(handles.figure1)

%%%%%% PROGRAMA PARA DESCOMPRIMIR%%%%%%

function descomprimir_Callback(hObject, eventdata, handles)

global zip
global info
global pn
global fn

pn_aux=uigetdir(pn,'Carpeta de destino');

if pn_aux==0
   return
elseif length(pn_aux)==3
else
   pf=[pn_aux fn];
   pf=[pn_aux '\' fn];
end
pn = pn_aux;
```

```
pf=pf(1:end-4);
fid = fopen(pf);
if fid>0
   qd = questdlg(['El archivo ',fn(1:end-4),' ya existe. ¿Desea reemplazarlo?'],'Archivo ya
existente','Si','No','No');
   fclose(fid);
   if ~strcmp(qd,'Si')
      return
   end
   pause(0.25)
   set(handles.figure1,'Pointer','watch')
   set(handles.estado,'String','Descomprimiendo archivo...')
   datos = huff2norm(zip,info,handles);
   fid = fopen(pf,'w');
   fwrite(fid,datos,'uint8');
   fclose(fid);
   set(handles.estado,'String','')
   set(handles.figure1,'Pointer','arrow')
   msgbox('El archivo fue descomprimido','Fin de operación','warn','modal')
else
   set(handles.figure1,'Pointer','watch')
   set(handles.estado,'String','Descomprimiendo archivo...')
   datos = huff2norm(zip,info,handles);
   fid = fopen(pf,'w');
   fwrite(fid,datos,'uint8');
   fclose(fid);
   set(handles.estado,'String','')
   set(handles.figure1,'Pointer','arrow')
   msgbox('El archivo fue descomprimido','Fin de operación','warn','modal')
end

%%%%%% PROGRAMA PARA LA AYUDA DE LA APLICACION%%%%%%

function ayuda_Callback(hObject, eventdata, handles)


%%%%%% PROGRAMA PARA LOS TEMAS DE INFORMACION%%%%%%

function temas_Callback(hObject, eventdata, handles)

web('C:/Matlab6p5/work/huffman/ayuda html/doc1TESIS.htm')

function acerca_Callback(hObject, eventdata, handles)

acerca

function comprimir2_Callback(hObject, eventdata, handles)

comprimir_Callback(hObject, eventdata, handles)

function descomprimir2_Callback(hObject, eventdata, handles)

descomprimir_Callback(hObject, eventdata, handles)
```

```
function reporte2_Callback(hObject, eventdata, handles)

reporte_Callback(hObject, eventdata, handles)

function abrir2_Callback(hObject, eventdata, handles)

abrir_Callback(hObject, eventdata, handles)

function ayuda2_Callback(hObject, eventdata, handles)

temas_Callback(hObject, eventdata, handles)

function salir2_Callback(hObject, eventdata, handles)

salir_Callback(hObject, eventdata, handles)


function figure1_CloseRequestFcn(hObject, eventdata, handles)

btn = questdlg('¿Desea cerrar el programa?','Salir del programa','Sí','No','No');
if strcmp(btn,'Sí')
   delete(hObject);
   quit
end
```

ANEXO C

GLOSARIO

GLOSARIO

Ancho de Banda. Es la diferencia entre las frecuencias máximas y mínimas que es capaz de transmitir el canal. El canal transmite todas las señales cuyo espectro está incluido dentro del ancho de banda del canal. Pero el ancho de banda tiene un limite según el canal y a la hora de transmitir hay que tener en cuenta tanto el espectro de la señal como el del canal para que la transmisión sea buena.

ARQ. (Siglas de automatic retransmisión request, respuesta de transmisión automática). Técnica de corrección de error donde el mensaje se retransmite cuando el recptor detecta un error.

ASCII. (Código Estándar de Estados Unidos para el intercambio de Información). Código de 7 bits más 1 bit de paridad.

Baud. Tasa de transmisión de datos, en unidades de símbolos por segundo, donde un símbolo es contado para cada nivel posible de modulación. Un baud es los mismo que un bit por segundo, si cada símbolo representa un bit.

Bit. Empleado para denotar unidades de información.

Canal. Modelo para expresar el vehículo; conjunto de medios materiales de la transmisión y todos los fenómenos que tienden a restringir la transmisión de un punto A a un punto B.

Capacidad de un canal. Es la velocidad de transmisión máxima que se puede alcanzar en el canal. La capacidad de un canal va a estar limitada por el ancho de banda. La capacidad es el doble del ancho de banda. Cuando por un canal se pueden transmitir n estados de señalización.

Esta formula se aplica a los canales sin ruido:

$C = 2AB * \log_2 N$

Para los canales con ruido, la capacidad es:

$C = AB * \log_2 (1 + S/R)$ bits/seg.

Código. Conjunto de todas las posibles palabras código.

Código lineal. Un código es lineal si y solo si para cualquier par de palabras código del mismo, la combinación lineal de las mismas produce otra palabra código.

Código prefijo. Se trata de un código que no tiene un tamaño en bits determinado, sino en el que cada palabra código puede tener un tamaño variable, y en el que es posible determinar la palabra código recibida en cuanto se recibe su último símbolo.

Diafonía. Acoplamiento entre las líneas que transportan las señales.

Entropía de una fuente. Es la cantidad de información generada por una fuente. Trabajando en base 2 (binario), la entropía indica el número de bits, en media (en momentos puntuales se puede estar por encima o por debajo de ese valor "medio"), que se necesita para transmitir fielmente los datos generados por una fuente.

GUI.- Interfase Gráfica de Usuario.

Palabra código. Cada vector legal de un código.

Ruido. Cualquier señal indeseable que se inserta entre el emisor y el receptor.

Ruido de intermodulación. Cuando distintas frecuencias comparten el mismo medio de transmisión.

Ruido impulsivo. Pulsos discontinuos de poca duración y de gran amplitud.

Ruido térmico. Debido a la agitación térmica de electrones dentro del conductor.
Velocidad de Transmisión. Es el máximo número de elementos binarios que se pueden transmitir durante 1 segundo. Su unidad bit/seg.

Velocidad de Modulación. Es el número máximo de veces por segundo que puede cambiar el estado de la señal en la línea (inversa de la duración del intervalo significativo mínimo medido en segundos). Unidad Baudio.

Latacunga, agosto del 2005.

Elaborado por:

César Alfredo Naranjo Hidalgo
CI.: 0501498505

Ing. Nancy Guerrón Paredes
DIRECTORA DE LA CARRERA DE ELETRÓNICA.

Ab. Eduardo Vásquez
SECRETARIO ACADÉMICO