ALGORITMO PARA OPTIMIZAR EL USO DE ENERGIA EN LA GANADERIA DE PRESICIÓN

Tomás Gaviria
Universidad Eafit
Colombia
tgaviriao@eafit.edu.co

David Ruiz
Universidad Eafit
Colombia
druize@eafit.edu.co

Simón Marín
Universidad Eafit
Colombia
smaring1@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

RESUMEN
En esta ocasión, el problema que se está abordando es cómo podemos a través de un algoritmo de compresión y descompresión de imágenes ayudar a la ganadería tradicional. Es importante debido a que los ganaderos manejan esta información de forma ineficiente como usando cuadernos, de tal manera que entregándoles una nueva forma para esto haría mucho más ameno su trabajo. Uno de los mayores inconvenientes es que este software debe ser económico de sostener, en cuanto a energía y espacio en memoria debido a que en el campo no se encuentran las mejores prestaciones para la energía y los sistemas de computación. El algoritmo que finalmente se utilizó fue el Huffman, este algoritmo permite en tiempo una rápida respuesta

1. INTRODUCCIÓN

Cada día que pasa en el mundo se desarrolla nuevas aplicaciones, sistemas, métodos para hacer más sencilla la vida de las personas, sin embargo, todavía contamos con actividades económicas históricas que cuentan con unos sistemas obsoletos y parecen haber sido olvidadas, a continuación, uno de los mayores ejemplos sobre este tópico.

La ganadería hoy en día es una de las actividades económicas más grandes del planeta, nada más en Estados Unidos de América se pueden encontrar alrededor de 30 millones de cabezas de res a lo largo de todo el país y este número sólo crece con los años. Es increíble que para tal magnitud de ejemplares se usen formas de conteo y control anticuadas, esta es la motivación para crear un software de ganadería de precisión, mejorar el proceso de la ganadería. [2]

1.2 Solución

En este trabajo, utilizamos una red neuronal convolucional para clasificar la salud animal, en el ganado vacuno, en el contexto de la ganadería de precisión (GdP). Un problema común en la GdP es que la infraestructura de la red es muy limitada, por lo que se requiere la compresión de los datos.

El algoritmo elegido para la comprensión con perdida es el de escalado de imágenes, este permite reducir la escala de la imagen, reduciendo consigo los pixeles y por tanto el tamaño de la imagen. El algoritmo utilizado para la compresión y descompresión sin perdida es el Huffman

1.3 Estructura del artículo

En lo que sigue, en la Sección 2, presentamos trabajos relacionados con el problema. Más adelante, en la Sección 3, presentamos los conjuntos de datos y los métodos utilizados en esta investigación. En la Sección 4, presentamos el diseño del algoritmo. Después, en la Sección 5, presentamos los resultados. Finalmente, en la

Sección 6, discutimos los resultados y proponemos algunas direcciones de trabajo futuras.

2. TRABAJOS RELACIONADOS

En lo que sigue, explicamos cuatro trabajos relacionados. en el dominio de la clasificación de la salud animal y la compresión de datos. en el contexto del PLF.

3.1 VIGSA: Software for Animal Health Surveillance

Este software presentó una alternativa a la problemática de monitoreo en la salud de los animales en cualquier campo, desde fincas, granjas hasta zoológicos. Esto lo lograron hacer a través de un software programado principalmente en DojoJS (Dojo Java Script, oriented to objects) el cual puede tener acceso desde un computador en la web o cualquier tipo de dispositivo móvil inteligente. El software que hace el informe y seguimiento de un estado sanitario de la población animal también permite almacenar información para su posterior aplicación a través de una ventana que muestra toda información de la categoría/animal seleccionado. Brinda beneficios como la reducción del uso de papel y la graficación de los datos anteriormente registrados en la población. [3]

3.2 Animal Care Software Solutions that Revolutionize Livestock Management and Pet Care.

Este software permite a los trabajadores de las fincas ganaderas específicamente, deshacerse del papel por completo, presentando un atractiva e innovador software el cual te permite digitalizar la cantidad de ganado, fecha de nacimiento de cada una de estas, procesamiento y reubicación; como también el mantenimiento de los corrales, terneros y comida. Folio 3 tiene un valor agregado que le permite ver al supervisor de la finca como se es el proceso de crianza y cuidado de los animales. Tiene aparte una sección de ayuda veterinaria, la cual le permite interactuar con un veterinario, el cual le brinda una foto del animal para que este haga su debida inspección. Como se puede evidenciar el software no está completamente automatizado para la clasificación

de salud del animal, sin embargo, presenta una idea hacía esto. [5]

3. MATERIALES Y MÉTODOS

En esta sección, explicamos cómo se recogieron y procesaron los datos y, después, diferentes alternativas de algoritmos de compresión de imágenes para mejorar la clasificación de la salud animal.

3.1 Recopilación y procesamiento de datos

Recogimos datos de *Google Images* y *Bing Images* divididos en dos grupos: ganado sano y ganado enfermo. Para el ganado sano, la cadena de búsqueda era "cow". Para el ganado enfermo, la cadena de búsqueda era "cow + sick".

En el siguiente paso, ambos grupos de imágenes fueron transformadas a escala de grises usando Python OpenCV y fueron transformadas en archivos de valores separados por comas (en inglés, CSV). Los conjuntos de datos estaban equilibrados.

El conjunto de datos se dividió en un 70% para entrenamiento y un 30% para pruebas. Los conjuntos de datos están disponibles en https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets.

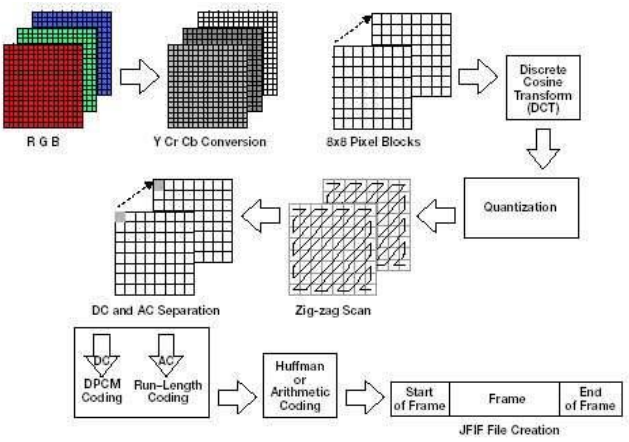
Por último, utilizando el conjunto de datos de entrenamiento, entrenamos una red neuronal convolucional para la clasificación binaria de imágenes utilizando *Teachable Machine* de Google disponible en https://teachablemachine.withgoogle.com/train/image.

3.2 Alternativas de compresión de imágenes con pérdida

En lo que sigue, presentamos diferentes algoritmos usados para comprimir imágenes con pérdida.

3.2.1 JPEG

La estructura del algoritmo JPEG tiene tres partes [Wal89, ARA91]: (1) un sistema que usa la información de la imagen ordenada en líneas, (2) un conjunto de opciones que extienden las capacidades del sistema anterior, y (3) la posibilidad de codificar sin pérdida de información de manera independiente de las anteriores. El sistema basado en líneas es la parte más simple del algoritmo propuesto por .JPEG. Este sistema consta de técnicas que son muy conocidas como la Transformada Discreta del Coseno, la cuantización uniforme y la codificación de Huffman. Además, hace uso también de algoritmos que proporcionan mayor compresión en imágenes. El sistema está construido de tal forma que permite que una imagen codificada sea transmitida secuencialmente, y sus pixeles sean reconstruidos por el decodificador siguiendo el orden de un cuadriculado. [6]



3.2.2 La Transformada Discreta del Coseno

La transformada discreta del coseno (DCT) es una variante de la transformada de Fourier que toma sólo la parte real de los datos [Pet92]. Esta transformada toma un bloque de n x n pixeles como un vector en un espacio de n2 dimensiones. Cada uno de estos bloques se transforma de manera separada. Los n2 valores de los pixeles Sxy son transformados a través de la transformada directa discreta del coseno (FDCT) en n2 nuevas coordenadas Suv. [6]

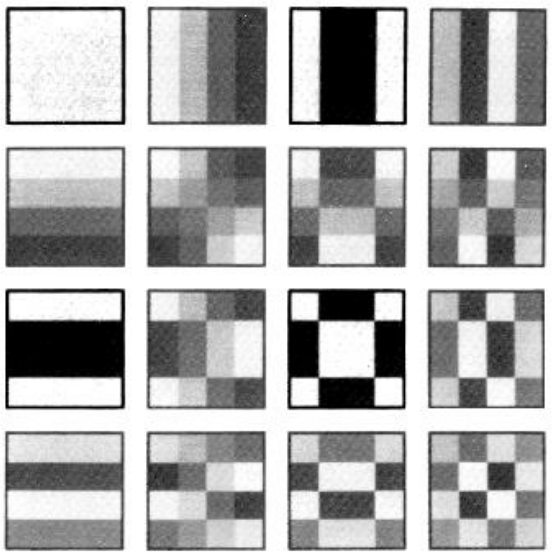


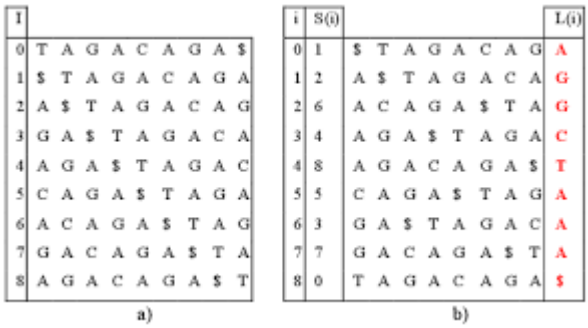
Figure 3.28 *Discrete cosine transform basis functions for N × 4 elements, corresponding to x and y varying from 0 to 3 The its top left. The highest value is shown in white. Other values , meaning smaller.* *Each block consists of 4 origin of each block is at own in grays, with darker*

3.3. Alternativas de compresión de imágenes sin pérdida

En lo que sigue, presentamos diferentes algoritmos usados para comprimir imágenes sin pérdida.

3.3.1 Transformación Borrows y Wheeler.

El algoritmo de transformación Borrows y Wheeler toma un conjunto de datos y lo ordena usando un algoritmo de ordenación. El conjunto de salida resultante contiene exactamente los mismos elementos de datos con los que empezó, difiriendo sólo en el ordenamiento. La transformación es reversible, o sea, la ordenación original puede ser restablecida si pérdida de fidelidad [4].



I									
0	T	A	G	A	C	A	G	A	$
1	$	T	A	G	A	C	A	G	A
2	A	$	T	A	G	A	C	A	G
3	G	A	$	T	A	G	A	C	A
4	A	G	A	$	T	A	G	A	C
5	C	A	G	A	$	T	A	G	A
6	A	C	A	G	A	$	T	A	G
7	G	A	C	A	G	A	$	T	A
8	A	G	A	C	A	G	A	$	T

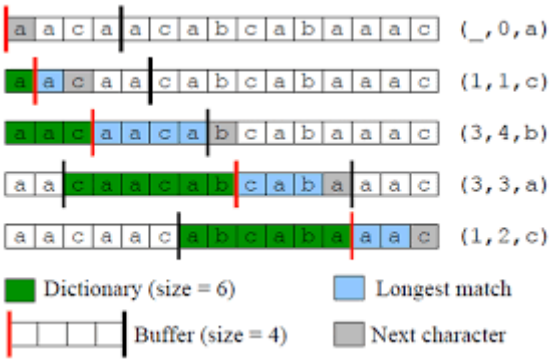
a)

i	S(i)								L(i)
0	1	$	T	A	G	A	C	A	G	A
1	2	A	$	T	A	G	A	C	A	G
2	6	A	C	A	G	A	$	T	A	G
3	4	A	G	A	$	T	A	G	A	C
4	8	A	G	A	C	A	G	A	$	T
5	5	C	A	G	A	$	T	A	G	A
6	3	G	A	$	T	A	G	A	C	A
7	7	G	A	C	A	G	A	$	T	A
8	0	T	A	G	A	C	A	G	A	$

b)

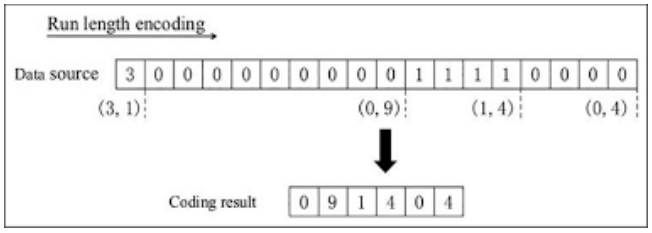
3.3.2 LZ77

Este algoritmo examina el texto ingresado como una cadena de caracteres dividida en dos buffers, de anticipación y de búsqueda. El análisis de este se basa en que busca en la ventana la coincidencia más larga con el comienzo del búfer de búsqueda anticipada y genera un puntero a esa coincidencia. Dado que es posible que ni siquiera se pueda encontrar una coincidencia de un carácter, la salida no puede contener solo punteros. LZ77 resuelve este problema de esta manera: después de cada puntero, genera el primer carácter en el búfer de búsqueda anticipada después de la coincidencia. Si no hay coincidencia, genera un puntero nulo y el carácter en la posición de codificación. [1]



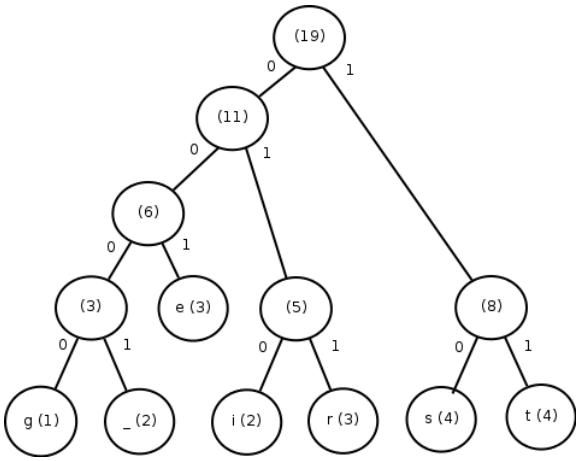
3.3.3 Run-Length.

Esta es la más simple de todas las técnicas de compresión. Es utilizada en cadenas las cuales tenga varios caracteres y repetidos. Por ejemplo, la cadena “abchhhhhhcvvvvabcdef” puede ser codificada como “abc6hc4vabcdef”, donde los números indican la cantidad de veces que las letras posteriores a las mismas deben ser repetidas. En esta técnica es común es utilizar bytes con signo para indicar la presencia o ausencia de repeticiones. Si el byte encontrado es negativo, significa que los siguientes n bytes deben ser escritos de manera normal. Si el byte es positivo, significa que el siguiente byte debe ser repetido n veces. El ejemplo anterior sería entonces codificado como “-3 a b c 6 h -1 c 4 v -6 a b c d e f “. [4]



3.3.4 Codificación de Huffman.

La codificación Huffman es una codificación que pasa un contenido de tamaño fijo a uno de tamaño variable (menor por lo general). Está basada en asignar códigos cortos a caracteres muy repetidos y códigos largos a caracteres menos repetidos. 2[4]



4. DISEÑO E IMPLEMENTACIÓN DE LOS ALGORITMOS

En lo que sigue, explicamos las estructuras de datos y los algoritmos utilizados en este trabajo. Las implementaciones de las estructuras de datos y los algoritmos están disponibles en GitHub[1]

4.1 Estructuras de datos

El codificador Huffman crea una estructura arbórea ordenada con todos los símbolos y la frecuencia con que aparecen. Las ramas se construyen en forma recursiva comenzando con los símbolos menos frecuentes. [5]

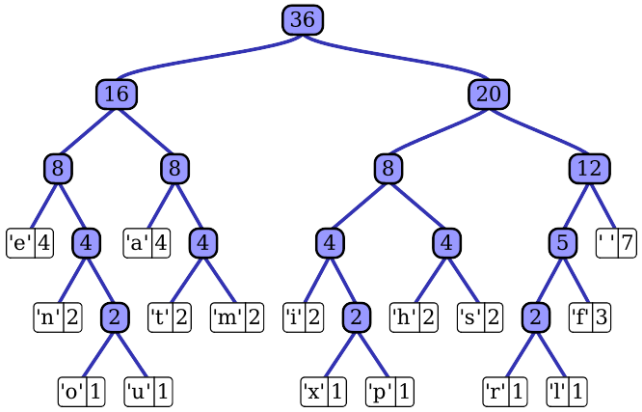


Figura 1: Árbol de Huffman generado a partir de las frecuencias exactas del texto "this".

4.2 Algoritmos

En este trabajo, proponemos un algoritmo de compresión que es una combinación de un algoritmo de compresión de imágenes con pérdidas y un algoritmo de compresión de imágenes sin pérdidas. También explicamos cómo funciona la descompresión para el algoritmo propuesto.

Se utiliza el algoritmo de escalado de imágenes, este lo que hace es agrupar diferentes pixeles cercanos y convertirlos en uno para así reducir la escala, los pixeles y por tanto el tamaño

4.2.1 Algoritmo de compresión de imágenes con pérdida

El algoritmo recibe una imagen como si fuera un archivo CSV y según la solicitud agrupa los datos más cercanos (vistos como pixeles). Debido a que se agrupan los pixeles, simplemente para descomprimir hay que solicitarle la imagen con la misma escala.

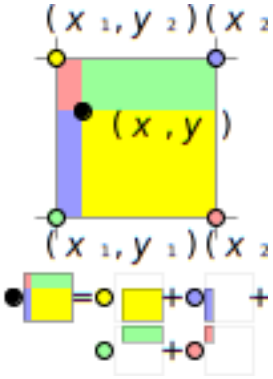


Figura 2: Escalado de la imagen mediante interpolación bilineal. (Por favor, siéntase libre de cambiar esta figura si utiliza una estructura de datos diferente).

4.2.2 Algoritmo de comprensión de imágenes sin pérdida

El algoritmo Huffman en este caso lo que hace es darle el valor de peso a los datos ingresados en el sistema, este valor se asigna según la frecuencia de aparición de los datos, para luego generar un código binario. Luego el código binario se descomprime para así obtener nuevamente los datos ingresados en el sistema sin pérdidas.

4.3 Análisis de la complejidad de los algoritmos

Para entender la complejidad del algoritmo es necesario entender que esta es el uso de recursos utilizados para la función de este. En este caso, nosotros obtuvimos la complejidad analizando temporalmente el tiempo utilizado con respecto a los datos ingresados y en memoria analizando el código y las transformaciones y requerimientos que este necesitase para su funcionamiento. Haciendo esto obtuvimos que:

[1]https://github.com/DavidRuizE/ST0245-001/tree/master/proyecto

Algoritmo	La complejidad del tiempo
Compresión	O (N*M Log(N*M))
Descompresión	O (N*M Log(N*M))

***Tabla* 1:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. N y M son las columnas y filas ingresadas

Algoritmo	Complejidad de la memoria
Compresión	O(K)
Descompresión	O(K)

***Tabla* 2:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. La variable K corresponde a los pixeles únicos que aparecen en las imagenes (los que no se repiten).

4.4 Criterios de diseño del algoritmo

Luego de hacer una investigación profunda sobre los diferentes algoritmos que permitían el comprimir y descomprimir imágenes sin perdida, entre los dos decidimos que el algoritmo Huffman era el ideal para desarrollar el proyecto. Uno de sus mayores rivales era el LZ77 pero los resultado de compresión y descompresión fueron totalmente arrasadores y es que el algoritmo Huffman tomaba menos de la mitad del tiempo que el LZ77.

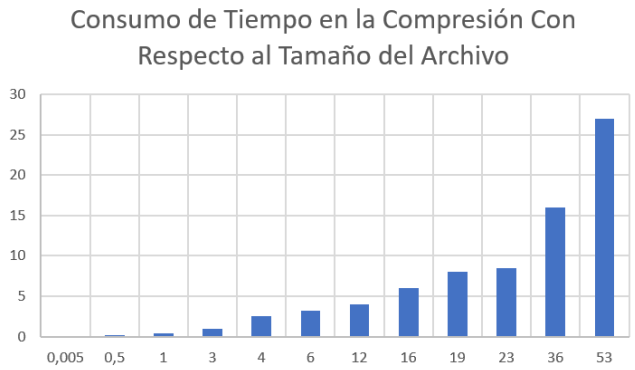
5. RESULTADOS

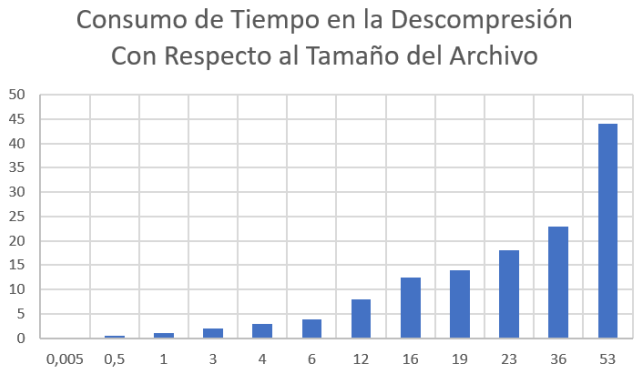
5.1. Evaluación del tiempo consumido

A continuación, presentamos los resultados del tiempo de consumo

	Tiempo Promedio	***Tamaño promedio del archivo***
Compresión	7s	15mb
Descompresión	11s	15mb

Tabla 3. Evaluación del Tiempo consumido por el algoritmo



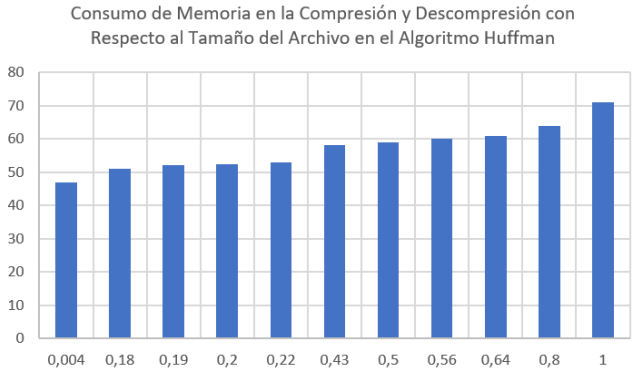


5.1.2 Evaluación del consumo de memoria

A continuación, presentamos el uso de memoria de la compresión y la descompresión del algoritmo sin perdidas

	Consumo promedio de memoria	***Tamaño promedio del archivo***
compresión	72 MiB	0.5MB
descompresión	59 MiB	0.5MB

Tabla 4: Evaluación de la memoria utilizada por el algoritmo



5.3 Tasa de compresión

Presentamos los resultados de la tasa de compresión del algoritmo en la Tabla 8.

	Ganado sano	***Ganado enfermo***
Tasa de compresión promedio	1: 2.6	1: 2.8

Tabla 8: Promedio redondeado de la tasa de compresión de todas las imágenes de ganado sano y ganado enfermo.

6. DISCUSIÓN DE LOS RESULTADOS

Creemos que el algoritmo Huffman es el apropiado para la resolución de este proyecto, debido a su eficiencia en memoria y el poco consumo de tiempo, lo hace el más ocpionado para esto y destaca por encima de los demás. En temas computacionales es el mejor con el que pudimos trabajar.

6.1 Trabajos futuros

Nos gustaría saber si se podría combinar este algoritmo con otro para hacerlo más eficiente y también testearlo con la red neuronal.

RECONOCIMIENTOS

Agradecemos la asistencia con brindar ideas y proponer mejoras a Juan Castro por los comentarios que mejoraron enormemente el manuscrito o la codificación del algoritmo y a demás compañeros que estuvieron presentes en este arduo proceso.

REFERENCIAS

1. Anjali S. Patel, Suman M. LZ77 and LZ78, Choudhary, Sonal J. Parmar. International Journal of Engineering Science and Innovative Technology (IJESIT), 2015.
2. Coneo M. CONMEMORACIÓN ANUAL AL SECTOR QUE APORTA 1,6% AL PIB NACIONAL, LOS GANADEROS, Agronegocios, Colombia, 2019.
3. Delgado Fernández, R., Libera Frómeta, J. y Barreto Argilagos, G. 2018. VIGSA: Software for Animal Health Surveillance. Revista de Producción Animal. 28, 2-3 (ene. 2018), 60-62.
4. Gímenez, R. Algoritmos de Compresión Sin Pérdida de Datos. Universidad Católica Nuestra Señora de la Asunción, Asunción, 2004
5. López C. CCM. ¿Cómo funciona el código Huffman? Francia 2021
6. Folio3. Cattle record keeping software that saves you time, reduces cost and improves performance. Belmont, 2018.
7. Veneces L. COMIPRESIÓN FRACTAL DE IMÁGENES FIJAS Y SECUENCIAS DE IMAGENES UTILIZANDO ALGORITMOS GENETICOS. Instituto Tecnológico y de Estudios Superiores de Monterrey, Estado de México, 1994.