Data Compression for Disc Files and Communication Networks

Klaus Holtz
Omni Dimensional Networks
631 O'Farrell, Suite 710
San Francisco, CA 94109
415 / 474 - 4860

Introduction

Data compression has suddenly become big business. Software packages or plug in modules can now more than double the storage capacity in disc drives while the new CCITT V.42bis standard can double the transmission speed in modems. Soon inexpensive chip sets will become a standard utility in most personal computer or communications networks. These chips may combine the highest compression ratios with optional unbreakable data encryption at a speed of more than 10 Million characters per second. This may open additional markets for high speed communications networks such as: ISDN, LAN's, WAN's or wireless networks. The optional encryption feature will guarantee absolute communications security even for open wireless networks. This paper will provide a toolbox of ideas, from basic concepts to the latest implementations, to help in the design of custom data compression systems.

Compressed communication or storage.

Storing compressed information in a disc file or transmitting compressed information in a communications network employs nearly identical methods. The storage device acts as a buffer to hold the transmission codes for delayed retrieval. While transmission errors may be corrected by re-transmissions, storage errors may destroy an entire data file.

The basic string compression concept.

The basic string compression system shown in Fig. 1 can be implemented as a dictionary of words stored in a computer memory. Each text word is marked with a unique address code. Both the transmitter and the receiver must contain an identical dictionary. To transmit a word the transmitter would search the dictionary to find that word and then transmit the address code of that word. The receiver would retrieve the word by a look-up in the dictionary under the transmitted address code. Any text word of arbitrary length could be transmitted by a short address code. This compression is "lossless" because the data is not changed or distorted in any way. Words not found in the dictionary can be transmitted by normal ASCII codes, but address codes and ASCII characters must be distinguished from each other. Experiments have shown that English words including the "Space" contain an average of 5.4 characters. Assuming an 8k word dictionary then the average compression ratio would be 3.3 for that simple system.

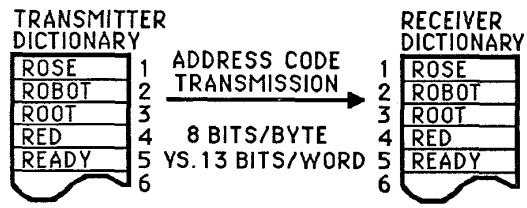

Figure 1: The basic string compression method

While the basic string compression concept is very simple, implementing such a system in hardware is very difficult. The following seven problems must be solved for a commercial system. These problems are more or less common to all data compression systems.

1 **Hardware Complexity** is high in Fig.1 because the word memory must be able to contain text words of any length. This would require a very large dictionary memory.

2 **Compression Ratios** are very high if the data type or language is known in advance. A stored English dictionary could not transmit French text. A library which could learn and adapt itself to the language and data type would be preferable. If a fixed generic dictionary is used then it must be specified in a file preamble code. If a secret library is supplied to authorized users only then the transmitted address codes would present a virtually unbreakable encryption code.

3 **Data Expansion** is a problem for random bit files. Because normal 8 bit ASCII codes must be distinguished from 13 bit address codes, an additional bit is required for each transmitted code. This would require 9 bits per byte for random data files which would make the compressed files larger than the original uncompressed data file. Some systems switch to ASCII transmissions for incompressible data files.

4 **Library Efficiency and Maintenance** is important for high compression ratios. The most common words should be selected and stored in the library where each word should be stored only once.

5 **Encoding Speed** should be very high to keep up with disc file storage or high speed modems. Searching the whole memory for each transmitted word is far too slow for practical systems.

6 **Error Propagation** can be a problem with dynamic library systems. A single storage or transmission error may destroy the entire remaining data file.

7 **Patent Royalties** may be required for implementing some data compression schemes.

The Ziv-Lempel Codes

The Ziv-Lempel codes shown in Fig. 2 are in the public domain and require no patent royalties. The input data character are stored in a shift register type buffer. In subsequent transmissions the buffer is searched to find "The longest matching strings" which are identical to the input data strings. Each string found in the buffer is identified by a "String Start" code and a "String Length" code. These codes are then transmitted or stored. The receiver would contain an identical buffer to accumulate the output data. The transmitted code is used to identify the string in the buffer and to copy the string as output data. The output data is then shifted into the buffer so that both dictionaries would remain identical at all times.

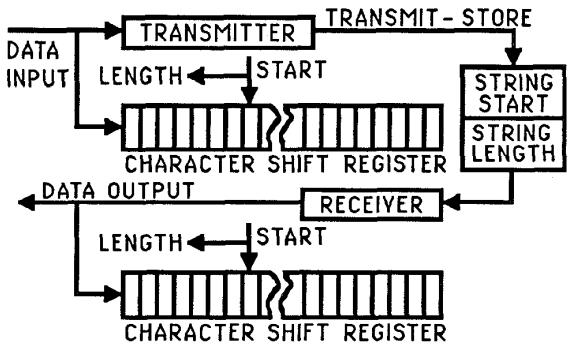

Figure 2: The Ziv-Lempel method.

Data compression results because most data transmissions contain duplicate words or strings from prior transmissions. The library in the buffer is assembled automatically from prior transmissions so that no prior exchange of libraries is required. This method can compress any kind of data in any language but has the following limitations:

Hardware complexity is very minimal and the system can be implemented by software only. **Compression Ratios** are low because of the need for a "String Length" code which is avoided in other methods. Compression can be very high for very limited string inventories, such as computer screen menus or caned computer messages. **Data Expansion** is a very serious problem especially for short data files. This is because the first part of the file will find no matching strings in a mostly empty buffer. Each string must be at least 3 characters long just to break even. **Library Efficiency** is low because each string or text word may be stored many times in the buffer. **Encoding Speed** is very low because it takes ages to identify the longest matching string using software only. Stac Electronics (Patent Whiting 5 003 307) has found a solution. Each character in the shift register buffer contains a separate comparator and a "Matching so far" flip flop. The input character is broadcast to all characters in the buffer and simultaneously compared in each buffer stage. If the input character is identical to the string in the buffer then the flip flop will remain set, otherwise it is reset. The flip flop which remains set longest will eventually identify the longest matching string in the buffer. Even though very fast, this method requires very large and complex VLSI chips. **Error Propagation** is a very serious problem because any transmission or storage error will cause the two libraries to fall out of step and become different. This may completely scramble the entire remaining data file.

Self-learning Tree Networks

The self-learning data tree networks were originally discovered in 1974 by K. Holtz during mathematical research into the brains learning functions. There are now 6 known types of self-learning networks which may be used to compress other types of data, such as images or databases. A patent license (Patent Holtz 4 366 551) is required for the V.42bis modems and most tree compression applications. Starting from the basic tree network shown in Fig. 3 three distinct systems have evolved which are known as the Autosophy networks, the Ziv-Lempel-Welch (LZW) codes (Patent Unisys Welch 4 558 302) and the British-Telecom-Lempel-Ziv (BTLZ) codes (Patent Pending). The Welch patent is presently being reexamined by the Patent Office and may or may not survive this reexamination. According to extensive research and testing by British Telecom and the CCITT committees tree networks combine the highest compression ratios with the simplest hardware implementations.

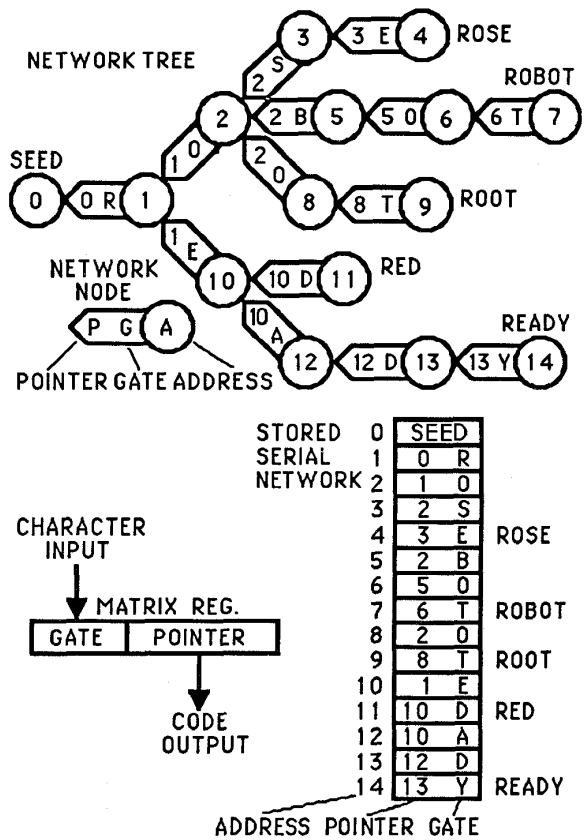

Figure 3: The basic Omni Dimensional Serial Network Tree

A tree network consist of individual nodes stored in a computer memory. The network will assemble itself from the input data according to patented algorithms. Each network node contains a GATE which is the input text character and a POINTER pointing backwards to the previous node ADDRESS. The combination of a GATE and a POINTER is called a MATRIX.

The network starts growing from a predefined SEED node, usually ADDRESS 0, or from a pre-loaded ASCII character set in the first 256 memory locations. The characters in the input text words are used to open GATEs to the following nodes in the tree. Each text word selects a specific trail through the tree maze to a final last node or leaf. This final node ADDRESS will represent the text word for storage or transmission. The text word is later retrieved in reverse order by following the POINTER trail back to the SEED node. If a new text word is entered for the first time then new nodes are created automatically to extend the tree network. This library is very efficient because each text word is only stored once and similar words will share their first nodes. The leaf ADDRESS codes represent the text words without the need for a string length code.

In learning mode the first input character or GATE is combined with the SEED POINTER to form a first MATRIX. The memory is then searched in forward direction to find the MATRIX. If it is found, then the ADDRESS in which it was found becomes the POINTER which is combined with the next input character GATE to form the next MATRIX. If the MATRIX is not found in the memory then it is stored into a next empty memory location and the ADDRESS into which it was stored becomes the POINTER for the next MATRIX. This forward memory search is repeated for each character in the string or word. The string is terminated either if the input character is a "Space" or after a new MATRIX is stored. The last identified POINTER is the output code to represent the string or word.

The string or words can be retrieved from the transmitted ADDRESS codes. The code is used as a memory ADDRESS to fetch a first MATRIX from the library. The GATE character is stored into a First-In-Last-Out stack buffer. The POINTER is used as the next memory ADDRESS to fetch the next MATRIX. Retrieval is complete when the POINTER contains the SEED address. The output string or word is then retrieved from the stack buffer.

Autosophy network systems.

This method combines the highest compression ratios with an optional unbreakable encryption code. The library is either generic, such as for English text, or a secret encryption dictionary. No attempt is made to compress non text data.

As shown in Fig. 4 the library is generated off-line and from random text files. Generating a library is done by software and requires about 30 minutes time. The first 256 locations are pre-loaded with the ASCII character set which may be scrambled for encryption libraries. Data from old text files are then fed to the learning algorithm. Only text words are learned in the library where all characters are converted to lower case. Each text word string is terminated by the next non alphabet character. Each unique text word will generate a 16 bit output code with is send to a bubble sort list. In this list each 16 bit word code will cause a list search to find that code already stored. If the code is not found then it is stored at the end of the list. If the code is found then it is swapped with the code one location towards the top of the list. This will cause the most often used text words to bubble-up or migrate toward the top of the list while rare words will move downwards and are eventually dropped from the list. Eventually this list of the most common words in a language is send to an output file which is then used to generate the library tree network. The library file may be stored as a generic dictionary in the communications software package or it may be stored into a generic Read Only Memory in the hardware. A removable encryption library which is made available only to authorized users provides near absolute communications or file access security. The kind of library to be used can be negotiated during call initiation in the V.42 standard. The specific data type of a file can be specified by a data type specifier such as used in the Macintosh. A communications software package may contain many generic libraries including private encryption libraries.

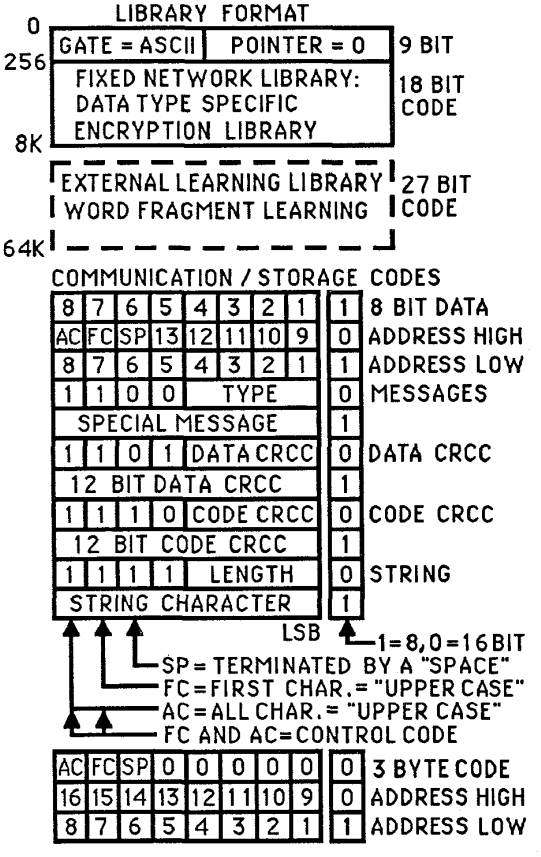

Figure 4: Autosophy library and communications codes

Experiments have shown that an 8k node library can store about 4000 of the most common text words. Even though most people have a passive vocabulary of about 50 thousand words the most common 4000 words constitute about 97% of written text. Words not found in the library are automatically cut into fragments. The encoding routine will follow the tree branches until either a "Space" character is detected or a leaf is not found in the library. The missing leaf will transmit the POINTER to represent a partial word. The routine will then return to the SEED to encode the remaining word in a new string. This fragmentation of unknown words will not greatly reduce the compression ratio because of the relative rarity of such words. An external learning library may be used to learn all fragments without the "Space" bit. This may increase the compression ratio because unknown words are fragmented only once during a transmission or storage.

Dictionary learning routine

MATRIX OR NODE MEMORY WORD

GATE 8 bit	POINTER 13 bit

Start: Fill ADDRESS 0 to 255 with GATE=ASCII, POINTER=0
Loop1: Set POINTER = SEED = 0
Loop2: Move (next) text input character into GATE
If GATE is not "alphabetic" (A to Z) then Go to: Loop1
Convert GATE to lower case ASCII (Set ASCII bit 6 = 1)
Search memory to find a matching MATRIX
If MATRIX is found then move memory ADDRESS
to POINTER. Go to: Loop2
Else store MATRIX in "Empty" memory ADDRESS.
Move storage ADDRESS to POINTER.
(Limit string length to 10 - 16 characters)
If POINTER = 8191 (8k memory is full) then:
End of dictionary learning.
Else Go to: Loop2

Compression routine

MATRIX -- TRANSMISSION

GATE 8 bit	POINTER 13 bit	SPACE bit

Start: Set POINTER = SEED = 0
Loop: Move (next) input character into GATE
If GATE is not "Alphabetic" (A to Z) or: "Space" then:
If POINTER = 0 then output GATE code. Go to: Start
Else output POINTER code + GATE code . Go to: Start
Else If GATE ="Space" then output POINTER code
with SPACE bit set. Go to: Start
Loop1: Else search dictionary memory for matching MATRIX
If MATRIX is found them move ADDRESS to
POINTER. Go to: Loop
Else output POINTER code. Set POINTER = 0
Go to Loop1

Expansion routine

MATRIX - TRANSMISSION

GATE 8 bit	POINTER 13 bit	SPACE bit

Start: Move input code to POINTER
If input is a single character GATE code then
output character as data. Go to: Start
Else If SPACE = 1 then put "Space" character into a
FIRST-IN-LAST-OUT STACK.
Clear SPACE bit from POINTER code.
Loop: Use POINTER as ADDRESS to fetch MATRIX
Store GATE into STACK.
If POINTER = SEED then retrieve data from STACK.
Go to: Start
Else Go to: Loop

The above routines will grow a library file from random text files. These libraries are then loaded both into the transmitter and into the receiver. Input data is separated into either 9 bit single character codes for random bit files or as 18 bit address codes for text. No attempt is made to compress non text characters because such compression is not possible. No compression scheme can compress random bit files or numbers.

Data expansion can be avoided by modified communications or storage methods. Asynchronous communications, such as RS-232, use a 10 bit per character format in which each character is preceded by a "Start" bit and terminated by a "Stop" bit. If the "Stop" bit is inverted into the next "Start" bit then 2 byte codes can be send without data expansion. Storage files on disc or tapes usually contain a "Parity" bit per byte. This parity bit can be modified into a "continue" bit to indicate that the following byte is part of the code. This would allow the storage of mixed single character codes or address codes without any possibility of data expansion. The error checking function is replaced by 2 CRCC code generators. These check code generators would accumulate a separate 12 bit code both for the transmitted codes and for the ASCII code input data. A software command would insert these codes into the data stream and reset the generators. The receiver would detect these special error codes and compare them with its own generated codes. If the codes do not agree then an error message is send to the computer. The two CRCC codes would separate transmission or storage errors from translation errors which may be due to the wrong library.

For a more compact library which contains only lower case words two extra bits are added. The FC bit will indicate that the first character of a word is a capital character. The AC bit indicates that the whole word consists of capital letters. If both the AC and the FC bit is set then a special code is send to the receiver. String length compression is used for strings of up to 16 identical characters. A special 3 byte code may be used for external library learning. Any address code without the SP (Space) bit would indicates a word fragment. These word fragments are retrieved and used to generate a separate external network tree both in the transmitter and in the receiver. If address codes from the external library are used then a 3 byte code is required. External library learning may increase the compression ratios but at the cost of exposing the system to error propagation.

Autosophy compression systems have the following features: **Hardware complexity** is very low. The whole system may be implemented in software or on a single integrated chip. Each separate library requires only 24k bytes of storage. **Compression ratios** for English text has been measured as 2.7 average. This is the highest compression ratio if the correct library is used. **Data Expansion** can be avoided for any type of data by modified communications or storage methods. **Library efficiency** is extreme and the best possible. An 8k node tree library can represent as much as 128k of random text. **Encoding speed** is fairly high. One complete scan of the short library will encode any length text word. Very high speed devices require special hardware which is explained later. **Error propagation** will not occur in fixed library systems.

The Ziv-Lempel-Welch codes

The LZW codes are believed to be the most successful and the most often used codes for data compression. Except for minor alterations these codes use the original autosophy tree algorithms. The main advantage is that any type of data can be compressed without the prior exchange of libraries. This makes it suitable for generic general purpose data compression programs. The original patent Welch 4 558 302 now owned by Unisys is being reexamined by the Patent Office and may or may not survive this reexamination.

The main differences between the autosophy trees and the LZW codes are the way a string is terminated by a not found leaf node and how these codes are transmitted. The library is initialized by loading the first 256 locations with the ASCII codes. This in effect creates 256 different trees, each

starting from the first string character. This first string character is used as the SEED pointer. The next input character is combined with the first character as POINTER to form a first MATRIX. The memory is then searched to find such MATRIX. If it is found then the ADDRESS where it was found is the POINTER which together with the next input character forms the next MATRIX. If it is not found then the MATRIX is stored into a next empty memory ADDRESS. The GATE (character) and the POINTER (prefix) are then transmitted or stored as the compressed output code. The receiver would create a new node by storing the received MATRIX into a next empty memory location. The GATE (character) is stored into a Fist-In-Last-Out stack. The POINTER code is used to retrieve the data string using the autosophy expansion routine.

Ziv-Lempel-Welch encoding routine.

MATRIX OR NODE MEMORY WORD

GATE 8bit	POINTER 13 bit

Start: Use first 256 locations as first character SEED
Loop: Clear POINTER in MATRIX
Loop1: Move (Next) Input data character to GATE
If POINTER = Clear then move GATE to POINTER
Go to Loop1
Else search memory to find MATRIX
If MATRIX is found then move ADDRESS to POINTER
Go to Loop1
Else store MATRIX into next empty memory ADDRESS
Output code = MATRIX (Character - Prefix)
Go to Loop

Ziv-Lempel-Welch expansion routine.

MATRIX OR NODE MEMORY WORD

GATE 8bit	POINTER 13 bit

Start: Use first 256 locations as final SEED code
Loop: Move input code (Character - Prefix) to MATRIX
Store MATRIX into next empty memory ADDRESS
Store GATE into a First-In-Last-Out stack
Loop1: Move POINTER to memory ADDRESS to read MATRIX
Store new GATE into the stack
If POINTER is less than 256 then
retrieve data from the stack
Go to Loop
Else Go to Loop1

The LZW codes have the following unique features: The library will assemble itself from the data input. There is no need for exchanging libraries before transmission. **Hardware complexity** is a problem because of the undetermined library memory size which must be at least equal to the data file size. **Compression ratios** are only very modest because of the disorganized library and the long output code. **Data expansion** is inevitable for short files because there are no strings for comparison in a mostly empty dictionary. Even though each transmitted code represents at least 2 character the minimum code length is 3 bytes. Just to break even about 4k of input text is required. **Library efficiency** is very low because of the disorganized library. While the Autosophy library contains organized strings of text words the LZW library resembles a string salad. This is because text words or any input data is chopped into fragments whenever a leaf node is not found. This disorganized library is known as a "Greedy" algorithm because it gobbles up any data type. **Encoding speed** is a problem because a large memory must be searched for each input character. The original inventor Terry Welch triec to increase encoding speed with Hash coding in which the nodes are sorted and stored according to their content. Very high speed applications require Content Addressable Memories (CAM). **Error propagation** is a very serious problem because any transmission or storage error may destroy the entire remaining data file.

The British-Telecom-Lempel-Ziv codes

The BTLZ codes use the basic LZW method but add some embellishments for a practical system design. The code is now used in the new CCITT V.42bis modems but there will be many more applications in the future. British Telecom is expected to receive a patent for their code.

The network library shown in Fig.6 reserves the first 256 memory locations for the first string character. A number of addresses is then reserved for special control codes to the receiver. The ETM code will switch transmission mode to uncompressed clear communication. The FLUSH code will flush all the remaining untransmitted data from the encoder. The STEPUP code will increase the output code by one more bit. This is because while encoding is proceeding in the 512 memory section only a 9 bit address code is required. When the network grows into higher memory then every time the library memory doubles an extra bit is added to the transmitted address code. Eventually when a specified memory size is reached then network leaf nodes are cleared and recycled in all further encoding. This recycling of network nodes must distinguish between branch nodes and leaf nodes. Branch nodes cannot be recycled because they are used as references in other node pointers.

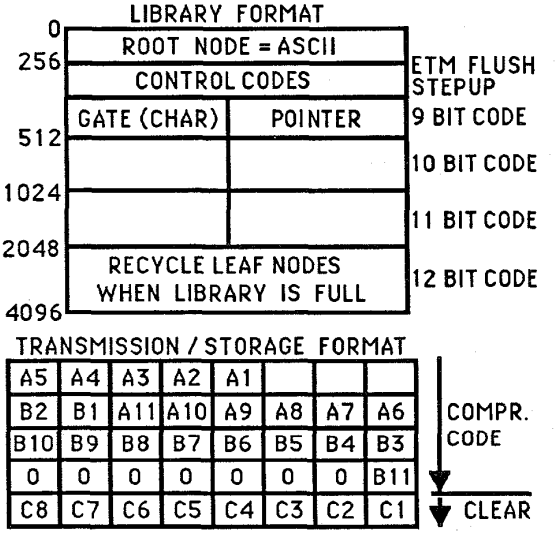

Figure 6: BTLZ library and transmission codes

Data is transmitted in packets where the address codes are stuffed into 8 bit octets. The bit length of the address codes is determined by the STEPUP commands. To avoid data expansion a software routine will monitor the efficiency of the compression process. If the data is incompressible then a switch to clear transmission is made with the ETM command.

Later when data again becomes compressible a switch back to compressed transmission is made using an "Escape Character". While switching between compressed and clear transmissions the encoder must realign clear character codes with the octet boundaries by inserting zeros.

The main difference to the LZW code is "Delayed Innovation". When a MATRIX or leaf is not found in the library then it is stored into a next free memory ADDRESS but only the POINTER code is transmitted. The GATE or unmatched character is used as the first character of the next string to continue encoding. The receiver will retrieve the output string from the POINTER code and then hold the POINTER in a buffer. The next POINTER code received is retrieved to find the GATE as the first character of that second string. The previous POINTER from the buffer and the new GATE are then combined into a MATRIX and stored into a next free memory location in the receiver library. This delayed innovation scheme may cause problems if a node is used in the transmitter which is not yet established in the receiver. Special precaution steps are specified in the V.42bis standard.

The BTLZ codes have the following unique features: The library is kept small by recycling nodes. The library will adapt quickly to changes in the data type. **Hardware complexity** is low because of the limited library memory but intensive data monitoring for compressibility is required. **Compression ratios** are much better than the LZW codes because of the delayed innovation feature but the library is still very disorganized. **Data expansion** is mostly avoided by switching between compressed and clear modes of transmission. **Library efficiency** is low because of the greedy algorithm in the LZW code. **Encoding speed** is much improved because of the limited library memory to be searched. This speed is sufficient for telephone modems but not fast enough for digital networks. **Error propagation** is not a problem because of the error correction feature in the V.42 standard. Data packets containing transmission errors are recognized by the receiver before they are used. Requests for re-transmissions are made until the packets are delivered without errors.

Very high speed encoding methods.

All tree networks may be adapted for very high speed operations by using Content Addressable Memories (CAM), such as the AMD99C10 chips (256 X 48 bit) from Advanced Micro Devices. Encoding and retrieval speed may reach 10 Million characters per second. Even though these devices are still very small and expensive this need not remain so in the future. Content addressable memories can be build almost as cheaply as normal RAMs if a sufficient market quantity is established. Fully integrated data compression system chip sets may contain a novel Content Addressable Read Only Memory (CAROM, Patent pending). This memory consist of a large array of integrated 21 bit AND gates which inputs are selected by external loading, such as EEPROM, FLUSH, blowing fuses such as in ROMs or by setting antifuses such in FPGAs.

Conclusions and outlook.

Data compression is already a very large and lucrative market, but the best is yet to come. Soon inexpensive data compression chip sets will become available which may become a standard utility in most Personal Computer and communications systems.

After all, if significant data compression is possible at bus speeds and without large expenses or data expansion then transmitting or storing redundant data can only be wasteful. Data encryption in fixed library systems may add very welcome communications security or file access barriers. The savings in transmission cost or in expanded storage facilities can be very significant and far outweigh the cost of including data compression into a system.

Soon other data compression applications will appear. Image compression methods, using parallel tree networks, may revolutionize High Definition Television (HDTV) and teleconferencing. Voice or music compression may be implemented in packet telephones, Digital Audio Tapes (DAT) or for storing voice messages in Personal Computer. A self-learning brain-like Autosophy database may store multimedia data in extreme compressed format to replace many applications in the the now dominant data processing computer.

References.

1 CCITT V.42bis THE INTERNATIONAL TELEGRAPH AND TELEPHONE CONSULTATIVE COMMITTEE
Data Communication over the telephone networks.
Data Compression procedures for data circuit terminating equipment (DCE) using error correction procedures.
Recommendation V.42bis Geneva 1990

2 U.S. Patent Holtz 4 366 551
Associative Memory Search System

3 NORTHCON-91, Sess. D4, Portland Oregon. 10 /91
Learning Networks, an Alternative to Data Processing

4 Colin Johnson: Storage technique can "learn"
Electronic Engineering Times. January 6 1992

5 K. Holtz: Text Compression and Encryption
with self-learning Networks
IEEE GLOBECOM-85 Conference.

6 U.S. Patents:
Barnes 3 914 747
Betz 3 976 844
Hoerning 4 021 782
Jackson 4 054 951
Moll 4 412 306
Storer 4 436 422
Schippers 4 038 652, 4 152 582
Eastman (AT&T) 4 464 650 Aug.7 1984
Van Maren (HP) 4 870 415
Whiting (STAC) 5 003 307 Mar. 25 1991
Whiting (STAC) 5 016 009 May 14 1991
Katz(PKWARE) 5 051 745

7 T.A. Welch "A Technique for High-Performance Data Compression" IEEE Computer, June 1984
Patent: Welch (Unisys) 4 558 302
(Patent is being reexamined by Patent Office)

8 J. Ziv, A. Lempel "A Universal Algorithm
for Sequential Data Compression"
IEEE Trans. Information Theory, Vol. IT-23, No 3

9 J. A. Storer. Data Compression Methods and Theory

10 How compressed data travel faster - and cheaper.
Business Week, Nov. 29, 1982, pp 94D-94H

11 G. Pastrick. Double - No - Triple Your Hard Disc Space
with on-the-fly Data Compression
PC Magazine, pp 261-283, Jan. 28, 1992