A NEW TECHNIQUE FOR TEXT DATA COMPRESSION

UDITA KATUGAMPOLA
SOUTHERN ILLINOIS UNIVERSITY, CARBONDALE, IL 62901

Abstract. In this paper we use ternary representation of numbers for compressing text data. We use a binary map for ternary digits and introduce a way to use the binary 11-pair, which has never been use for coding data before, and we futher use 4-Digits ternary representation of alphabet with lowercase and uppercase with some extra symbols that are most commonly used in day to day life. We find a way to minimize the length of the bits string, which is only possible in ternary representation thus drastically reducing the length of the code. We also find some connection between this technique of coding data and Fibonacci numbers.

Keywords: Coding, Data Compression, Binary system, Ternary System, Golden Ratio

INTRODUCTION

Ternary or trinary is the base-3 numeral system. A ternary digit, trit contains about 1.58596 \((\log_2 3)\) bit of information. Even though ternary most often refers to a system in which the three digits, 0, 1, and 2, are all nonnegative integers, the adjective also lends its name to the balanced ternary system, which uses -1, 0 and +1 instead, used in comparison logic and ternary converters [1][2][3][4].

Techniques have been developed for text document compression that are semiadaptive, which uses frequency-ordered array of word-number mappings [5][6][7]. Encoding binary digital data in ternary form has applicability to digital data communication systems and magnetic data storage systems [8] and also in high-speed binary multipliers, which uses ternary representations of two numbers [9]. Methods have been developed for converting binary signals into shorter balanced ternary code signals [2].

Variable length coding is a widely-used method in data compression, especially, in the applications of video data communication and storage, for example, JPEG, MPEG, CCITT H.261 and so on. Most of those methods implement the coding with two-field codes. Hsieh [10] introduced a method using three-field representation for each code. Also, we can adopt ternary systems in difference coding in audio compression [11][12]. The node of a Peano Curve can be represented by a base-3 reflected Gray Codes, RGC [13].

Ternary systems have been used in digital data recording systems [14] and precoded ternary data transmission systems as well [15] from which they can send and store more data compared to what binary system does.

Digital data compression is an important tool because it can be utilized, for example, to reduce the storage requirements for files, to increase the rate at which data can be transferred over bandwidth limited communication channels, and to reduce the internal redundancy of data prior to its encryption in order to provide increased security.

In this paper we use standard ternary representation for coding data. We introduce a new representation called Base \(B_{23}\), which uses both ternary and binary representation with the maximum use of both binary and ternary features in a single coded data. Before going further, let us compare the binary and ternary representations of few numbers,

\[
85_{10} = 1010101_2 = 100103 \quad \text{and} \quad 150_{10} = 10010110_2 = 121203
\]

Thus, the binary representation of 85 uses 7 bits while ternary representation uses only 5 trits, and in 150, it uses 5 trits again, which is more than 40% saving in memory. Here we also notice that the 12-pair occurs twice in this representation. Thus if we can use a better representation which can code the 12-pair we can save more memory when saving these data and can save more time when sending to another detector. Thus our main goal is to develop a system which uses ternary representation in a sophisticated manner. That is what we are going to do in the rest of this paper.
The Basic Definitions

We begin with the following two definitions.

**Definition 1.** Let \( n = \epsilon_1 \epsilon_2 \epsilon_3 \ldots \epsilon_k \), where \( \epsilon_i = 0, 1, 2; k \in N \) be the ternary representation of the decimal number \( n \). Then we say \( n \) is in the form \( A_{23} \), if there is a map \( \phi : \{0, 1, 2\} \mapsto \{00, 01, 10\} \) such that,

\[
\phi(\epsilon_i) = \begin{cases} 
00 & \text{if } \epsilon_i = 0; \\
01 & \text{if } \epsilon_i = 1; \\
10 & \text{otherwise.}
\end{cases}
\]

for each \( i = 1, 2, 3, \ldots, k \).

With this definition, we can write \( 85_{10} = 10010_3 = 010000100_{A_{23}} \) and \( 150_{10} = 12120_3 = 0110011000_{A_{23}} \). Thus it double the length of the string which represents the number in \( A_{23} \) base. But in this format we waste the 11 pair. So we need to modify the coding so that we can make use of the binary string 112. Thus we come up with the following definition, with the so called \( Base B_{23} \).

**Definition 2.** Let \( n = \epsilon_1 \epsilon_2 \epsilon_3 \ldots \epsilon_k \), where \( \epsilon_i = 0, 1, 2; k \in N \) be the ternary representation of the decimal number \( n \). Then we say \( n = \epsilon_1 \epsilon_2 \epsilon_3 \ldots \epsilon_l \), where \( \epsilon_i = 00, 01, 10, 11; l \leq k \), is in \( Base B_{23} \), if there is a map \( \psi : \{0, 1, 2\} \mapsto \{00, 01, 10, 11\} \) such that \( \psi(\epsilon_i) = \epsilon_j \), where

\[
\epsilon_j = \begin{cases} 
00 & \text{if } \epsilon_i = 0; \\
01 & \text{if } \epsilon_i = 1 and \epsilon_{i+1} \neq 2; \\
10 & \text{if } \epsilon_i = 2 and \epsilon_{i-1} \neq 1; \\
11 & \text{if } \epsilon_i \epsilon_{i+1} = 12.
\end{cases}
\]

for each \( i = 1, 2, 3, \ldots, k \).

That is, we say a number \( n \) is in \( Base B_{23} \), if each of the trits of its ternary representation is replaced by the following binary bits in that order:

\[
0 \mapsto 00, \\
12 \mapsto 11, \\
1 \mapsto 01, \\
2 \mapsto 10.
\]

So we can write \( 85_{10} = 0100000100_{A_{23}} = 0100000100_{B_{23}} \) and \( 150_{10} = 0110011000_{A_{23}} = 111100_{B_{23}} \). Thus it drastically reduces the length of the bits string if the 123 is present in the coding. The more 123 pairs are present, the more compact the code can be. Therefore, it is natural to seek the availability of 123 pairs in a string of ternary representation of a data. Thus we have the following lemma,

**Lemma. Golden Lemma**

The number \( S_n \) of ways that a string of trits 0, 1 and 2 of length \( n \) with no 12 pairs in the string is

\[
S_n = \frac{1}{\sqrt{5}} \left( \phi^{2n+2} - \phi^{-2n-2} \right)
\]

for \( n \in N \), where \( \phi \) is the Golden Ratio \( \frac{1 + \sqrt{5}}{2} \).

**Proof.** We prove this result using a recurrence relation. First consider the following construction of string of 0,1 and 2 of size \( n \).
According to the diagram we have,

\[
S_n = 2S_{n-1} + S_{n-2} + S_{n-3} + \ldots + S_2 + S_1 + 2,
\]

with \( S_1 = 3 \) and \( S_2 = 8 \) for \( n \geq 3 \). This reduces to \( S_n = 3S_{n-1} - S_{n-2} \) with \( S_1 = 3 \) and \( S_2 = 8 \) for \( n \geq 3 \). Solving the recurrence relation with the given data, we have,

\[
S_n = \frac{1}{\sqrt{5}} \left\{ \left( \frac{1+\sqrt{5}}{2} \right)^{2n+2} - \left( \frac{1-\sqrt{5}}{2} \right)^{2n+2} \right\} \text{ for } n \geq 3
\]

\[
= \frac{1}{\sqrt{5}} \left( \phi^{2n+2} - \phi^{-2n-2} \right), \text{ where } \phi = \frac{1+\sqrt{5}}{2}.
\]

The first few terms of this sequence are \( S_1 =3, S_2 = 8, S_3 = 21, S_4 = 55, S_5 = 144 \). These are the even terms of the Fibonacci sequence defined by, \( F_n = F_{n-1} + F_{n-2} \) with \( F_1 = 2, F_2 = 3 \).

Now, consider a uniformly distributed ternary string of 0, 1, and 2 of size \( n \). Then the following holds:

**Theorem.** The probability of appearing at least one 12– pair of a string of 0, 1, and 2 of length \( n \) is asymptotically 1.

**Proof.** If \( S_n \) is the number of ways that a string of trits 0, 1 and 2 of size \( n \) can be arranged with no 12–pairs, then the probability of getting at least one pair of 12 is

\[
P_n = \frac{3^n - S_n}{3^n},
\]

\[
= 1 - \frac{1}{3^n \sqrt{5}} \left( \phi^{2n+2} - \phi^{-2n-2} \right)
\]

Therefore, \( \lim_{n\to\infty} P_n = \lim_{n\to\infty} \left( \frac{3^n - \frac{1}{\sqrt{5}}\phi^{2n+2}}{3^n} \right) = \lim_{n\to\infty} 1 - \frac{\phi^2}{3^n} \left( \frac{\phi^2}{\sqrt{5}} \right)^n \to 1 \), since \( \phi^2 = \frac{3+\sqrt{5}}{2} < 3 \). \( \square \)

This shows that when \( n \) gets larger, the mapping 12→11, should efficiently reduce the length of the \( B_{23} \) string.

**DATA COMPRESSION WITH \( B_{23} \)**

Here we discuss one of the main usage of \( B_{23} \) in coding data in day to day life. When compared to other benefits of converting data into binary form, word processing takes the leadership. Thus in this paper we discuss a technique that could drastically increase the storing and sending capabilities of data using the so called \( B_{23} \) ternary coding.

**Algorithm.** Let \( A=\{a_1, a_2, a_3, \ldots, a_n\} \) be an alphabet with, \( p(a_i) \) being the probability of \( a_i \) appearing in the language generated by \( A \). Without loss of generality, suppose \( p(a_1) \geq p(a_2) \geq \ldots \geq p(a_n) \). Let \( B=\{b_1, b_2, b_3, \ldots, b_n\} \) be a sequence of integers written in ternary form. Define, \( \delta: B \to \mathbb{Z} \) by

\[
\delta(b_i) = j, \text{ where } b_i = \ldots b_1 \ldots 12_j \ldots 12_{j-1} \ldots
\]
Suppose without loss of generality, \( \delta(b_1) \geq \delta(b_2) \geq ... \geq \delta(b_n) \). Then we define a map \( \Omega : A \mapsto B \) such that \( \Omega(a_i) = b_i \) for \( i = 1, 2, ..., n \).

We notice that this scheme generates a coding for the language generated by \( A \). Now, we turn to a more practical example. That is the English alphabet and the language generated by it.

**Example.** Here we are going to find a \( B_{23} \) code for 26-letter English alphabet, which is different from Huffman coding of 26-letter alphabet and also is different from the ternary Huffman coding [11][16][17][18].

We also include some extra characters, which are frequently used in word processing. Before going further, we need to observe some facts related to the usage of English alphabet. When we use a language, we have to use some letters more frequently than others. Particularly in English the letter ‘e’ has much highest frequency compared to the others characters in the alphabet, while the letter ‘z’ has the lowest frequency. Some of these facts is summaries in the following table accompanied with the two charts 1 and 2.

![Figure 2: Bar-chart of frequency order of 26-letter English alphabet](image-url)

According to the table below, the letters, e, t, a, o, i, n, s, h, and r has comparatively high frequency than the characters p, b, v, k, j, x, q, and z. Therefore if we can use a shorter code for the high frequency letters and comparatively longer code for low frequency characters we can possible reduce the string length for the coding sequence. This is the whole goal in the rest of the paper.

| Letter | Frequency(%) | Letter | Frequency(%) |
|--------|-------------|--------|-------------|
| a      | 8.167       | n      | 6.749       |
| b      | 1.492       | o      | 7.507       |
| c      | 2.782       | p      | 1.929       |
| d      | 4.253       | q      | 0.095       |
| e      | 12.702      | r      | 5.987       |
| f      | 2.228       | s      | 6.327       |
| g      | 2.015       | t      | 9.056       |
| h      | 6.094       | u      | 2.758       |
| i      | 6.966       | v      | 0.978       |
| j      | 0.153       | w      | 2.360       |
| k      | 0.772       | x      | 0.150       |
| l      | 4.025       | y      | 1.974       |
| m      | 2.406       | z      | 0.074       |

Table 1: Frequency order of 26-letter English alphabet

---

1 Cryptographical Mathematics, Robert Edward Lewand. MAA, Washington DC, 2000
A NEW TECHNIQUE FOR TEXT DATA COMPRESSION

Above table force us to reorder the alphabet so that we use a short code for high frequency letter, while longer code for low frequency letters. Before doing this we have to notice few things that are characteristics to the English alphabet. One major thing is alphabet has only 26 characters. Thus if we use a ternary coding with three trits, we can code all the 26 characters, but if we use binary coding we have to use 5 bits to handle these characters. This is the major observation in ternary system compared to binary system.

3 Exchanged Character Map - 3ECM

To represent high frequency characters with short code, we simply exchanged the positions of the English alphabet. Letters e, t, a, o, ... get shorter codes while the letters v, k, x, q, and z get comparatively longer codes as in the Huffman-encoding. What remains is to consider which upper case letters, typically the first letter of a word, appear more frequently as the first letter of a word. The top ten letters with frequencies, which occur at the beginning of words are:

| Letter | Frequency(%) |
|--------|--------------|
| T      | 15.94        |
| A      | 15.50        |
| S      | 8.23         |
| O      | 7.12         |
| C      | 5.97         |
| M      | 4.26         |
| F      | 4.08         |
| P      | 4.00         |
| W      | 3.82         |

Table 2: Frequency of the first letters

Clearly, the order differs from that for lower case (cf. t & e Vs T & E). Thus we propose the following character map with few extra symbols and accompanied ternary codes:

| Dec | Symbol | Ternary | Dec | Symbol | Ternary | Dec | Symbol | Ternary |
|-----|--------|---------|-----|--------|---------|-----|--------|---------|
| 0   | W      | 0000    | 27 | z      | 1000    | 54 | !      | 2000    |
| 1   | N      | 0001    | 28 | p      | 1001    | 55 | $      | 2001    |
| 2   | B      | 0002    | 29 | b      | 1002    | 56 | ^      | 2002    |
| 3   | C      | 0010    | 30 | w      | 1010    | 57 | %      | 2010    |
| 4   | D      | 0011    | 31 | x      | 1011    | 58 | \      | 2011    |
| 5   | T      | 0012    | 32 | e      | 1012    | 59 | \      | 2012    |
| 6   | F      | 0020    | 33 | f      | 1020    | 60 | *      | 2020    |
| 7   | G      | 0021    | 34 | g      | 1021    | 61 | /      | 2021    |
| 8   | H      | 0022    | 35 | v      | 1022    | 62 | =      | 2022    |
| 9   | P      | 0010    | 36 | q      | 1100    | 63 | <      | 2100    |
| 10  | J      | 0101    | 37 | j      | 1101    | 64 | >      | 2101    |
| 11  | K      | 0102    | 38 | k      | 1102    | 65 | @      | 2102    |
| 12  | L      | 0110    | 39 | y      | 1110    | 66 | &      | 2110    |
| 13  | M      | 0111    | 40 | m      | 1111    | 67 | \      | 2111    |
| 14  | A      | 0112    | 41 | n      | 1112    | 68 | "      | 2112    |
| 15  | O      | 0120    | 42 | o      | 1120    | 69 | ?      | 2120    |
| 16  | I      | 0121    | 43 | a      | 1121    | 70 | (      | 2121    |
| 17  | S      | 0122    | 44 | i      | 1122    | 71 | )      | 2122    |
| 18  | R      | 0200    | 45 | r      | 1200    | 72 | {      | 2200    |
| 19  | Q      | 0201    | 46 | s      | 1201    | 73 | }      | 2201    |
| 20  | T      | 0202    | 47 | t      | 1202    | 74 | |      | 2202    |
| 21  | U      | 0210    | 48 | u      | 1210    | 75 | |      | 2210    |
| 22  | V      | 0211    | 49 | h      | 1211    | 76 | \      | 2211    |
| 23  | .      | 0212    | 50 | Space  | 1212    | 77 | :      | 2212    |
| 24  | X      | 0220    | 51 | d      | 1220    | 78 | :      | 2220    |
| 25  | Y      | 0221    | 52 | l      | 1221    | 79 | +      | 2221    |
| 26  | Z      | 0222    | 53 | c      | 1222    | 80 | -      | 2222    |

Table 3: Coding Table
In our table the ternary code of almost all most frequent letters contain at least one pair. Applying the $B_{23}$ scheme results in the following code table.

According to the table, what we achieve here is that high frequent characters has shorter length compared to the others. Let us exemplify the method. For that we use a test string and code it in two ways, using $B_{23}$ and the standed ASCII code and then compare the two bitstrings generated from these techniques. It is done in the following way. The first Algorithm converts the text directly into $B_{23}$ code, and Algorithms 2 and 3 convert it back into human readable characters.

| Symbol | Ternary | $B_{23}$ | Symbol | Ternary | $B_{23}$ | Symbol | Ternary | $B_{23}$ |
|--------|---------|---------|--------|---------|---------|--------|---------|---------|
| W      | 0000    | 00000000 | z      | 1000   | 01000000 | !      | 0200   | 10000000 |
| N      | 0001    | 00000001 | p      | 1001   | 01000001 | $      | 0201   | 10000001 |
| B      | 0002    | 00000002 | b      | 1002   | 01000010 | "      | 0202   | 10000010 |
| C      | 0010    | 00000100 | w      | 1010   | 01000100 | %      | 0210   | 10001001 |
| D      | 0011    | 00000101 | x      | 1011   | 01000101 | /      | 0211   | 10001010 |
| T      | 0012    | 000011   | e      | 1012   | 01010001 | ,      | 0212   | 10001011 |
| F      | 0020    | 00010000 | f      | 1020   | 01100000 | *      | 0220   | 10010000 |
| G      | 0021    | 00010001 | g      | 1021   | 01100001 | /      | 0221   | 10010001 |
| H      | 0022    | 00010100 | v      | 1022   | 01101000 | =      | 0222   | 10010010 |
| P      | 0100    | 00100000 | q      | 1100   | 01110000 | <      | 0223   | 10010011 |
| J      | 0101    | 00100100 | j      | 1101   | 01110001 | >      | 0224   | 10010100 |
| K      | 0102    | 00101000 | k      | 1102   | 01110010 | @      | 0225   | 10010101 |
| L      | 0110    | 00110100 | y      | 1110   | 01110100 | &      | 0226   | 10010110 |
| M      | 0111    | 00111000 | m      | 1111   | 01110101 | '      | 0227   | 10010111 |
| A      | 0112    | 000111   | n      | 1112   | 011111   | “      | 0228   | 10011100 |
| O      | 0120    | 000100   | o      | 1120   | 011100   | ?      | 0229   | 10011000 |
| I      | 0121    | 001110   | a      | 1121   | 011101   | (      | 0221   | 10011100 |
| S      | 0122    | 001110   | i      | 1122   | 011110   | )      | 0222   | 10011110 |
| R      | 0200    | 00100000 | r      | 1200   | 110000   | {      | 0223   | 10011000 |
| Q      | 0201    | 00100001 | s      | 1201   | 110001   | }      | 0224   | 10011001 |
| T      | 0202    | 00100100 | t      | 1202   | 110010   | |      | 0225   | 10011010 |
| U      | 0210    | 00101000 | u      | 1210   | 110100   | }      | 0226   | 10011011 |
| V      | 0211    | 00101010 | h      | 1211   | 110101   | \     | 0227   | 10011012 |
|       | 0212    | 00110111 | Space  | 1212   | 1111     | ;      | 0228   | 10011013 |
| X      | 0220    | 00100100 | d      | 1220   | 111000   | :      | 0229   | 10011000 |
| Y      | 0221    | 00101001 | l      | 1221   | 111010   | +      | 0230   | 10011010 |
| Z      | 0222    | 00101010 | c      | 1222   | 111010   | -      | 0231   | 10011010 |

Table 4: Complete Coding Table

**Algorithm 1. Coding into $B_{23}$ form.** Here we assume that the text string only consists of the characters listed in the above table.

Let $u_{ij} = \{(W, 0000), (N, 0001), (B, 0002), \ldots ,(\, , 2222)\}, i = 1, 2, \ldots, 81; j = 1,2$

Input: Text String S. Let $l=\text{length}(S)$; NewString = $\emptyset$,
for $i=0$ to $l$ do
if $u_{ij} = \text{characterAt}(S, i)$
NewString = NewString + $u_{ij}$,
end do.

Once we received the string to the destination we use the following two algorithm to decode it. First one to transform it back into ternary and then the second one to decode it to human readable code.

**Algorithm 2. Converting into ternary.**
Input received string S. Let $l=\text{length}(S/2)$, NewString = $\emptyset$.
for $i=1$ to $l$
if SubString(S,i,i+1) = '00' then NewString = NewString + '0',
elseif SubString(S,i,i+1) = '01' then NewString = NewString + '1'
end do.
A NEW TECHNIQUE FOR TEXT DATA COMPRESSION

elseif SubString(S,i,i+1) = '10' then NewString = NewString + '2'
else NewString = NewString + '12'
i = i+2;
do

Now we read this string as a four character groups and assign each such group into a single character according to the above table. So we use the following algorithm

Algorithm 3. Decoding ternary into human readable characters
Let \( u_{ij} := \{(W, 0000), (N, 0001), (B, 0002), \ldots, (-, 2222)\}, i = 1, 2, \ldots, 81; j = 1,2 \)
Input: Ternary String S. Let \( l=\text{length}(S); \) NewString = []
for \( i=0 \) to \( l \) do
if \( u_{i2} = \text{substring}(S,i,i+3) \)
NewString = NewString + \( u_{i1} \),
else NewString = NewString + []
i=i+4;
do.

Upperbound for Compression Ratio. To derive an upperbound for the compression ratio, we have to make few assumptions since this compression is dynamic. Let us assume for a longer text at least the space has the highest frequency. In order to get a numerical value we assume the chance of space is 50\% and the other letters \( a_i \) has the probabilities, \( p_i \) half of the table values. Then

\[
\text{Compression Ratio} = \frac{\text{Length of compressed text}}{\text{Length of uncompressed text}}
\]

\[
= \frac{\sum_{i=1}^{n} \text{length of letter } a_i \text{ after compression } \times p_i}{\sum_{i=1}^{n} \text{length of letter } a_i \text{ before compression } \times p_i}
\]

\[
\leq \frac{4 \times 100 + 6 \times (8.167 + \ldots + 2.758) + 8 \times (1.492 + \ldots + 0.074)}{8 \times 200}
\]

\[
= 0.64577
\]

We can achieve much stronger compressions when we applied this scheme with the Huffman encoding. We can directly use the \( B_{23} \) scheme, once the technology develops to a level where we can use ternary bitstrings for data transmission.

Now we use the above algorithms to the following example, and compare the results with the familiar ASCII encoder. We notice that even for a short code we see noticeable reduction of the coded string.

Example. So, consider the text string,

\( S = " \text{This is the test message.}" \)

Once we apply the code into \( B_{23}, \) we get,

\[
\text{CodedString} = 000011110101011110110001111111111001011010100111111001001101100011110101010011011001111010011010010110100110100101010011011101010101001011010110100101010010100100101001100101100110100101011100110111001101100001011001110110010100101110
\]

This can be compared with the corresponding Binary string generated by ASCII coding, which is 25\% larger than the \( B_{23}, \)

\[
\text{BinaryString} = 1010100011011000110100100111001110010110010110100000111000010111100111011001011011101100010000000111010110111001011100101101100101110010000000111010100110010111001011011001011100101101
\]

After using second algorithm, we end up getting,

\[
\text{DecodedString} = "\text{This is the test message."}
\]
Thus if we can adopt this technique in word processing and data compression we can drastically reduce the memory needed to store information and also in data transmission.

We can extend this technique for six-digits ternary system with more characters than in this case. That would be the next task. We can also extend this technique with slight modifications for compressing highly randomized data [20]. We conclude the paper with the following question,

Question: What kind of distributions has more 12− pairs in a string of 0, 1, and 2?

Acknowledgement. Author would like to express his heart felt gratitude to Dr. Jerzy Kocik of Department of Mathematics of Southern Illinois University, for his invaluable suggestions and continued support.

REFERENCES

[1] Gundersen, Henning B., Yngvar, Fast Addition Using Balanced Ternary Counters Designed with CMOS Semi-Floating Gate Devices, University of Oslo, Norway, May 2007, ISSN: 0195-623X
[2] Van et al., US Patent 4003041. System for converting binary signals into shorter balanced ternary code signals. filed Dec 30, 1974
[3] Van et al., US Patent 3599205. Binary to ternary protected code converter. filed Aug 28, 1968
[4] Gilbert et al., US Patent 3866147. Balanced correlated ternary coding system. filed Feb 26, 1973
[5] Kaplan et al., US Patent 5029091. Text-compression technique using frequency-ordered array of word-number mappers. June, 1994. 341/51
[6] Lisle et al., US Patent 4843389. Text compression and expansion method and apparatus. June, 1989
[7] Crandall, G.E. US Patent 3999949. Text file compression system utilizing word terminators. December, 1999
[8] Jacoby et al., US Patent 4506252. Ternary data encoding system. filed July 05, 1983.
[9] Fensch, T., US Patent 4628472. Binary multiplier using ternary code. filed Nov 18, 1983
[10] Hsieh, C., US Patent 5648774. Variable length coding with three-field codes. filed May 08, 1995
[11] Data Compression, 4ed. David Salomon, Springer-Verlag, London, 2007.
[12] Gray, F. US Patent 2632056. Pulse Code Communication, March 17, 1953.
[13] Cole, A.J.(1985) A Note On Peano Polygons and Gray Codes, International Journal of Computer Mathematics, 18:3-13.
[14] Potter et al., US Patent 3226685. Digital recording systems utilizing ternary, n bit binary and other self-clocking forms. December, 1965. 360/40
[15] Howson, R.D. US Patent RE30182. Precoded ternary data transmission. December, 1979. 375/290
[16] David Salomon, Variable-length Codes for Data Compression, Springer-Verlag, London, 2007.
[17] David Salomon, A Concise Introduction to Data Compression, Springer-Verlag, London, 2008.
[18] C.B, John, G. C, Ian, H.W., Text Compression, Timothy, Prentice Hall, Englewood Cliffs, New Jersey, 1990
[19] Robert Edward Lewand, Cryptological Mathematics, The Mathematical Association of America, Washington DC, 2000.
[20] James, D.C. US Patent 5533051. Method for data compression. July 1996, 375/240