In Search of a Suitable Indian Language for Huffman Data Compression Algorithm

Satyendra Nath Mandal[1], Md. Iqbal Quraishi[2], Kuntal Bhowmick[3] and J. Pal Chaudhuri[4]

[1]Lecturer, Dept. of IT, Kalyani Government Engineering College, West Bengal, India.
[2]Lecturer, Dept. of IT, Kalyani Government Engineering College, West Bengal, India.
[3]4th Year, Dept. of CSE, Kalyani Government Engineering College, West Bengal, India.
[4]Asst. Professor, Dept. of IT, Kalyani Government Engineering College, West Bengal, India.

Kalyani Govt. Government Engineering College, Kalyani, Nadia, 741235, West Bengal, India
{satyen_kgec@rediffmail.com, iqbalqu@gmail.com, kuntal.kgec.cse@gmail.com jnpc193@yahoo.com}

Abstract: Huffman data compression algorithm is used many data compression application. In this paper, this algorithm has been used on data files of same size made by different languages. The same efforts have been made on different size files. The languages have been taking from the different part of India. A comparison has been made based on compression ratio, compression time and decompression time. Finally, one language has been selected based on performance.

***Keywords*:** Huffman Data Compression Algorithm, Lossless data Compression, Compression Ratio, Compression Time and Decompression Time.

1. Introduction

In last decade has witnessed tremendous growth in the Information Technology innovations and applications. Information Technology has become a vital component for the success of business because most of the organizations require fast information dissemination, information processing, storage and retrieval of data. The growth in this area occurred at such a fast rate due to the fact that Information Technology [1][2] opened new vistas in almost all day-to-day problems related with common man. Information Technology has revolutionized our life and has made a significant impact on all dimensions of our day-to-day life. In banking sector, use of credit, debit card, ATM, Tele-banking, Net banking[6]; in transportation, reservation of air tickets[5], railway tickets, buying & selling items on internet, electronic market, inquiry of department, bank transaction on net, entertainment, education, communication, hotel reservation[3], tourism have become reality. Internet is one of the mediums, which being used to access the pool of information.

Proper transformation of data is main theme in this era. Sometimes information becomes so large that it becomes problematic to transmit or storing information in their proper format. So, the concepts of data compression arise. That means, transformation of information in certain format which will take much small space comparing to original data. Shannon-Fanon algorithm [7], Huffman algorithm [4] & Arithmetic Coding [8] are some process of data compression. The works include "Optimal Huffman Tree-Height Reduction for Instruction Level Parallelism", Dept. of Computer Sciences, the University of Texas at Austin reports on a work of exploiting instruction level parallelism(ILP) is a key component of high performance for modern processor. For this purpose, Huffman Algorithm was taken to (1) tree height reduction rewriting expression trees of commutative and associative operations to make the height of the tree reduced (2) software fan-out generating software to fan out tree even when expression store intermediates of the instruction. [8]

Another work on the concurrent update and generation of the dynamic Huffman Code on is made for the dynamic Huffman Encoding. The concurrent procedure performs the tree update and code generation processes in parallel and therefore reduces over 45% number of steps required by the Knuth's work [13].Work on the Fast Adaptive Huffman Encoding Algorithms state that Huffman code suffers from two problems: the prior knowledge of the probability distribution of the data source to be encoded is necessary, and the encoded data propagate errors. The first problem can be solved by an adaptive coding, while the second problem can be party solved by segmenting data into segments. But the adaptive Huffman code performs badly when segmenting data into relatively small segments because of its relatively slow adaptability [7]

The paper on "Optimal Multiple Bit-Huffman Decoding" proposes a new optimal multi-bit Huffman decoding method that combines the barrel shifter and look-ahead approaches. Specifically, the work is on the development of approval partition of the state diagram corresponding Huffman diagram [14].The development of an efficient compression scheme to process the Unicode format data represents a very difficult task. Our work mainly concentrates on this.

In this paper, the design of an efficient scheme of compression using the Huffman Coding to process multiple languages that supports Unicode formatting. Huffman algorithm coding on data compression with variable length bit coding has been used on files of same size made by 10 different languages, Arabic, Assamese, Bengali, Gujarati, Hindi, Kannada, Malayalam, Oriya, Tamil and Telugu. Same things have been made on files of different size from different languages. A comparison has been made on compression ratio, time taken for compression and time

taken for decompression for files of same size on different languages. Finally, one language has been chosen based on their performance. This type of work has not been made so far. This is the reason for making this paper.

The paper is divided into following parts. The first part of this paper has been described the overall data compression. The Huffman data compression algorithm with example has been described in next section i.e. article number 3. The article number 4 has been given the algorithm for use of the Huffman Data Compression algorithm in different language. Finally, the results, conclusion and references have been described in article number 5, 6 and 7.

2. Overview of Data Compression

Data-compression techniques can be divided into two major families; **lossy** and **lossless**.

2.1 Lossy Data Compression Technique

Lossy data compression concedes a certain loss of accuracy in exchange for greatly increased compression. Lossy compression proves effective when applied to graphics images and digitized voice. Most lossy compression techniques can be adjusted to different quality levels, gaining higher accuracy in exchange for less

2.2 Lossless Data Compression

Lossless compression consists of those techniques guaranteed to generate an exact duplicate of the input data stream after a compress/expand cycle. This is the type of compression used when storing database records, spreadsheets, or word processing files.

2.3 Compression Ratio

Compression Ratio (CR) is defined as,

$$CR = \frac{(\text{Size of Original Data} - \text{Size of Compressed Data})*100}{\text{Size of Original Data}}$$

3. Huffman Algorithm

The Huffman Algorithm for Text Compression is an improvement over Shannon-Fano algorithm.

3.1 Huffman Algorithm Overview

Although similar in approach, the Huffman algorithm differs from Shannon-Fano algorithm in the construction of Binary Tree. The Huffman algorithm generates variable length code in such way that high frequency symbols are represented with a minimum number of bits and low frequency symbols are represented by relatively higher number of bits. The decoding of Huffman codes is done by using Huffman decodes Tree. Major difference in Shannon-Fano and Huffman algorithm, is that in SF algorithm ,the tree is built on the Top-Down approach while Huffman Tree is built using the Bottom-Up approach.

3.2 Construction of Huffman Tree and Generating code

1. Pick up two symbols (a, b) from the last two symbols in the sorted list of symbols.
2. Create two free nodes of the binary tree and assign A and B to these nodes.
3. Create a parent node for both nodes and assign it the frequency equal to the sum of frequencies of the child nodes.
4. Delete these two nodes from the list.
5. Parent node is added to the list of free nodes.
6. Repeat steps 1 to 5 until list of symbols becomes empty. This will generate the Huffman Tree.
7. Assign the bits similar to Shannon-Fano Tree i.e. left child is assigned '0' and right child is assigned '1'. Traverse from the root node of Huffman Tree to the leaf containing a particular symbol. This traversal will generate a code for that symbol.

Let us understand Huffman Algorithm with an example.

Table 1. Frequency table

Symbol	Frequency
'e'	60
'a'	20
'b'	15
'd'	5

The Huffman Tree is constructed in the following way and code is generated.

I. Find the sorted list of symbols in decreasing order of frequency. The list will be (e, a, b, d).
II. Pick up two Symbols with minimum frequency. These symbols are 'b' and'd'. Assign them two free nodes as shown figure1.

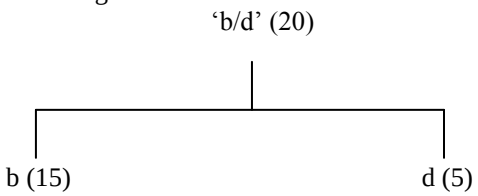


Figure 1. First Human Tree

This sub tree is called 'b/d'. Combined frequency of 'b/d' is 20.

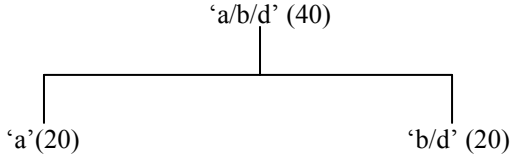
III. Update the sorted list as shown table 2.

Table 2. New Frequency Table

Symbol	Frequency
'e'	60
'a'	20
'b/d'	20

Pick up two symbols with the minimum frequency. These symbols are ('a','b/d').

Assign them free node as under in figure 2(a) and 2(b).



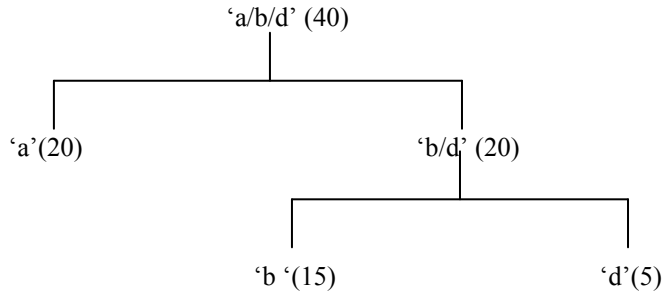


Figure 2(a) & 2(b): Next Huffman Tree

III. Update sorted list as shown in table 3.

Table 3. Modified frequency table

Symbol	Frequency
'e'	60
'a/b/d'	40

Pick up two symbols with the minimum frequency. These symbols are ('e','a/b/d'). Assign them free node as under is shown in figure 3.

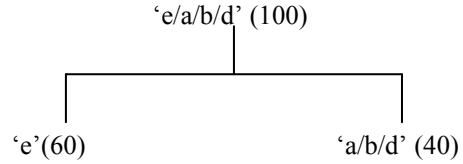


Figure 3. Third Huffman Tree

Finally, Huffman tree has been constructed based previous tree is shown figure 4. The two new symbols are nothing "b" and "d".

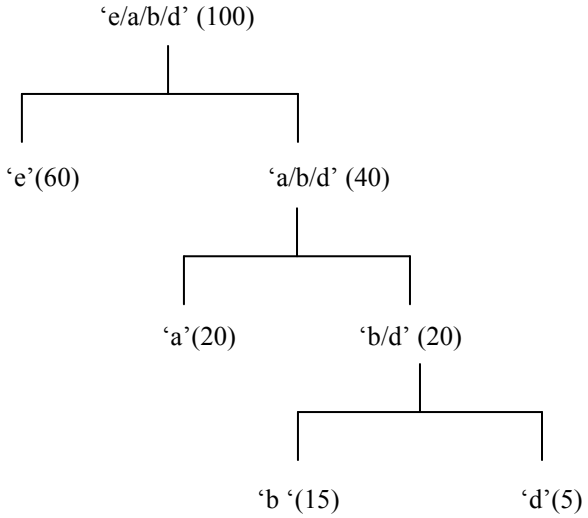
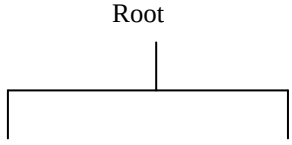


Figure 4. Final Huffman Tree

Update sorted list now it will contain only 'e/a/b/d' and no individual symbol is left. Therefore, the process halts and final Huffman Tree has been constructed. Now, assign the bit '0' and '1' to left and right subtree to obtain Huffman Code is shown in figure 5.



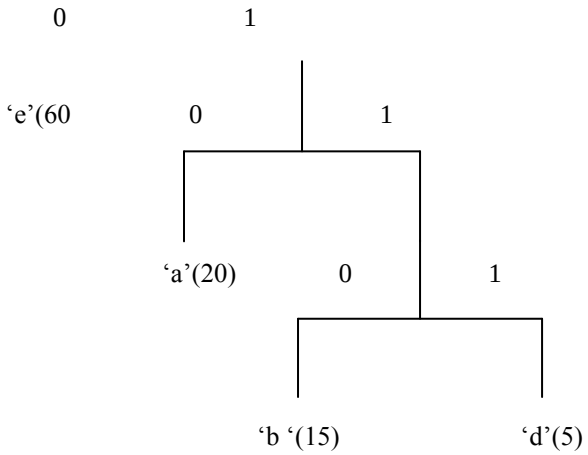


Figure 5. Huffman Tree with assigned code

From the above Huffman tree the code for all symbols are obtained. These codes are shown below in the following table 4.

Table 4. Symbol, Frequency, Huffman code and its size

Symbol	Frequency	Huffman Code	Size of Huffman Code
E	60	0	1
A	20	10	2
B	15	110	3
D	5	111	3

4. Method of selecting the language

This experimental process of selecting the best language has been done by taking ten standard Indian languages is shown figure 6. These files are served as input to the compressing algorithm.

The series of steps that we follow are given below.

Step 1: Initially the Unicode characters are written in Microsoft Word and then copy pasted in Notepad. The data is then saved as text file with the encoding option set to "Unicode" format.

Step 2: Ten files are prepared in the same way by copying and pasting the characters after being writing them in Microsoft word. The sizes of the files are increased gradually from 1Kb (1024 Bytes) to 10 Kb (10240 Bytes).

Step 3: In this way, 100 files are prepared, ten files of incrementing size for each of the ten languages.

Step 4: The Huffman data compression algorithm generates compressed files. The algorithm also generates a log file gives us information about the Original File Size, Compressed File Size, Time Required to Compression.

Step 5: The compressed files are then decompressed by the same algorithm. The decompressed file is compared with the original file to check it losslessness. The decompressed time also gets stored in the log file generated during compressing.

Step 6: A comparative study is made on the compression rate, Compressing and decompressing time from which the best language in selected.

The Figure 6 shows the above process diagrammatically.

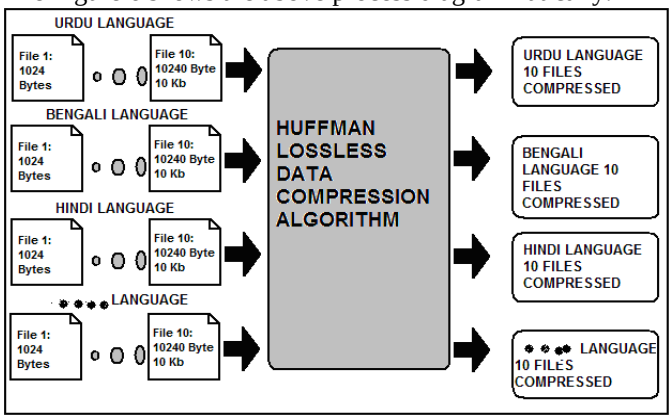


Figure 6: Diagrammatic Representation of the Language selection process

5. Experimental Results

For selecting the suitable language, an effort has been made a comparative study of ten languages already mentioned on compression ratio, compression time and decompression time. The selection of language based on the minimum compression ratio, minimum compression and decompression time are depicted in table 2,3 and 4. The results are also furnished in figure 7, 8 and 9.

6. Conclusion and Future Work

The result describes in tables 5, 6 & 7 shown that most of the times, the compression ratio, compression and decompression time of Arabic language is comparatively better than other nine languages. So, from this result, it can be concluded that Arabic language is suitable for Huffman data compression algorithms. Same study will be made in future on other languages in India.

7. References

[1] Blelloch, E., 2002, ***Introduction to Data Compression***. Computer science Department, Carnegie Mellon University.

[2] Cormark, V. and s. Horspool,, 1987, Data Compression using Dynamic Huffman Coding and Modelling Compute. J., 30: 541-550

[3] Vo Ngoc and M. Alistair , 2006, ***Improved word-aligned Binary compression for text indexing,*** IEEE Trans , Knowledge & Data Engineering, 18:857-861

[4] Kaufman, K. and T. Shumuel, 2005, Semi-Lossless text Compression, Intl. J Foundations of Computer Science.,16:1167-1178

[5] Capocelli, M., R. Giancarlo and J. Taneja, 1986 Bounds on the redundancy of Huffman Codes IEEE Transmission Information Theory, 32:854-857

[6] Gawthrop, J. and W. Liuping, 2005. Data Compression for Estimation of the Physical parameters of estimation of the physical parameters of stable and unstable linear Systems. Automation, 41: 1313-1321

[7] Kesheng, W., J. Otoo and S. Arie, 2006.optimizing Bitmap Image Compression Techniques with specification Index.ACM Databases Systems, 31:1-38

[8] D.A Huffman, "A method for the construction of minimum- redundancy codes,"Proc, . IRE, Vol. 40, pp 1098-1101, Sept. 1952.

[9] R.G Gallager, "Variations on a theme by Huffman," IEEE Trans. Inform Theory ., Vol It-24, pp,668-674, Nov, 1978.

[10] H. Yokoo, "An Improvement of Dynamic Huffman Coding with a simple repetation finder", IEEE Trans., Commun., Vol 39,pp. 8-10,Jan 1991.

[11] B. Landwehr and P. Marwedel, "***A new optimization Technique for improve ment resurce exploitation and critical path minization***," in Symposium on System Synthesis, (Antwerp, Belgium),pp. 65-72,September 1997.

[12] R. Sethi and J. d. Ullman "Using High Performance with an optimization compiler " in Proceedings architectures of high speed Processor pp :185-195,

[13] Sameh Ghwanmeh,Riyad Al-Shalabi "Efficient Data Compression Scheme using Dynamic Huffman Code Applied on Arabic Language" Journal of Computer Science 2(12) 885-888,2006 ISSN 1549-3636.

[14] Mark Nelson and Jean Loup Gailly "The Data Compression Book" 2nd Edition BPB Publisher,Indian Edition 1996.

Table 5: Different file size for different Languages and Compression ratio

Tamil	**Telugu**	**Assamese**	**Oriya**	**Malayalam**	**Gujarati**	**Bengali**	**Hindi**	**Kannada**	**Arabic**
65.917969	59.765625	63.671875	63.867188	**59.082031**	59.179688	64.550781	63.281250	65.234375	65.234375
60.156250	67.578125	65.917969	59.765625	**59.130859**	62.695312	63.330078	62.353516	65.329675	64.306641
62.076823	59.147135	60.774740	66.210938	63.346354	59.505208	61.100260	59.147135	64.290365	**57.584635**
63.696289	60.742188	62.353516	63.916016	64.331055	63.916016	64.599609	66.235352	61.718750	**59.155273**
59.765625	63.691406	66.328125	**57.597656**	62.968750	62.382812	62.714844	63.691406	58.183594	61.738281
60.791016	66.324870	64.664714	**59.163411**	61.735026	64.843750	65.543620	59.505208	58.544922	61.100260
65.652902	**57.338170**	62.039621	67.619978	62.067522	63.909040	57.589286	64.313616	66.629464	62.974330
64.318848	61.437988	65.270996	62.695312	59.509277	68.298340	61.743164	58.789062	62.365723	**57.592773**
58.279080	62.369792	62.369792	61.436632	67.610677	60.134549	65.332031	64.854601	64.854601	65.277778
61.748047	65.957031	66.630859	64.677734	65.332031	65.546875	**56.992188**	65.957031	66.923828	60.156250

Table 6: Different file size for different Languages and Compression Time

Tamil	Telugu	Assamese	Oriya	Malayalam	Gujarati	Bengali	Hindi	Kannada	Arabic
0.059	0.0548	0.0578	0.0592	0.0678	**0.0549**	**0.0549**	0.0568	0.0598	0.0568
0.062	**0.0447**	0.0525	0.0589	0.0598	0.0587	0.0574	0.0589	0.0549	0.0645
0.078	0.0688	0.0714	0.0635	0.0646	**0.0574**	0.0789	0.0587	0.0587	0.0654
0.082	0.0625	0.0845	0.0789	0.0712	0.0789	0.1345	0.1102	**0.0574**	0.0742
0.0102	0.0845	0.0845	0.0848	0.0845	0.0846	0.1289	0.0846	0.0789	0.0845
0.0589	0.0915	0.0987	0.0978	0.0973	0.0942	0.1156	0.0942	0.0897	**0.0847**
0.1648	0.1041	0.1258	0.1096	0.1198	0.1156	0.1345	0.1156	0.1548	**0.0987**
0.1847	0.1185	0.1389	0.1289	0.1385	0.1274	0.1274	0.1274	0.1045	**0.1147**
0.2197	0.1385	0.1149	0.1347	0.1398	0.1345	0.1897	0.0587	0.1356	**0.1137**
0.2198	0.1572	0.1489	0.1597	0.1596	0.1548	0.1945	0.1548	0.1945	**0.1398**

Table 7: Different file size for different Languages and Decompression Time

Tamil	Telugu	Assamese	Oriya	Malayalam	Gujarati	Bengali	Hindi	Kannada	Arabic
0.042	0.0578	0.0547	0.0589	0.0547	0.0548	0.0548	0.0601	0.0573	0.0587
0.045	0.0741	0.0647	0.0612	0.0593	0.0596	0.0658	0.652	0.0548	0.0789
0.069	0.0712	0.0798	0.0698	0.0679	0.0658	0.0758	0.0649	**0.0596**	0.0765
0.072	0.0756	0.0971	0.0756	0.0798	0.0858	0.1456	**0.0654**	0.0658	0.0897
0.081	0.0813	0.0925	0.0849	0.0973	**0.0695**	0.1045	0.0698	0.078	0.0965
0.0145	0.0105	0.1045	0.0845	0.1047	0.0985	0.1025	0.0985	0.0968	0.0879
0.1458	0.01156	0.1354	0.1145	0.1245	0.1025	0.1456	0.1025	0.1647	**0.1012**
0.1305	0.1347	0.1478	0.1374	0.1246	0.1256	0.1256	0.1256	0.1156	**0.1145**
0.15698	0.1498	0.1378	0.1454	0.1496	0.1456	0.1875	0.1596	0.1687	**0.1193**
0.15674	0.1497	0.1689	0.1647	0.1698	0.1647	0.2014	0.1647	0.2014	**0.1354**

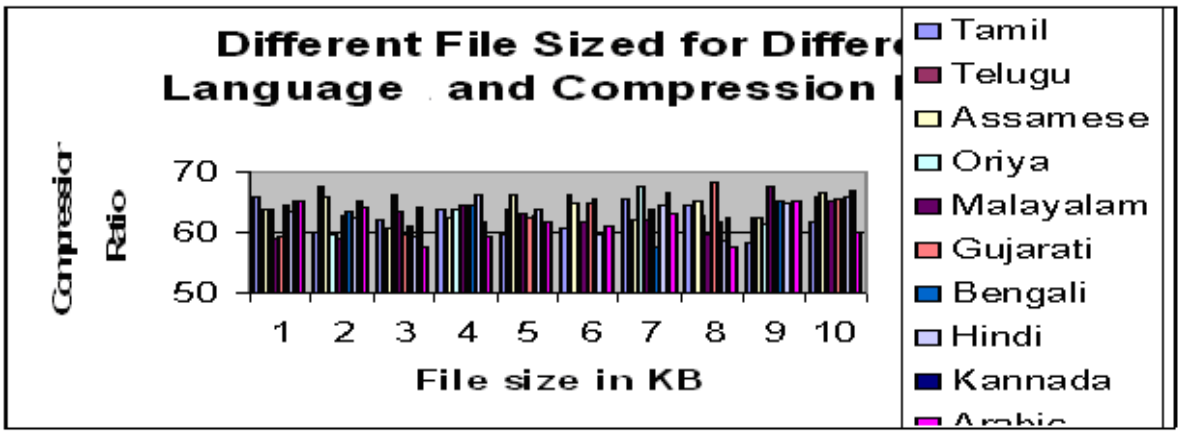


Figure 7. Graph for Different File sized for Different Languages and Compression Ratio

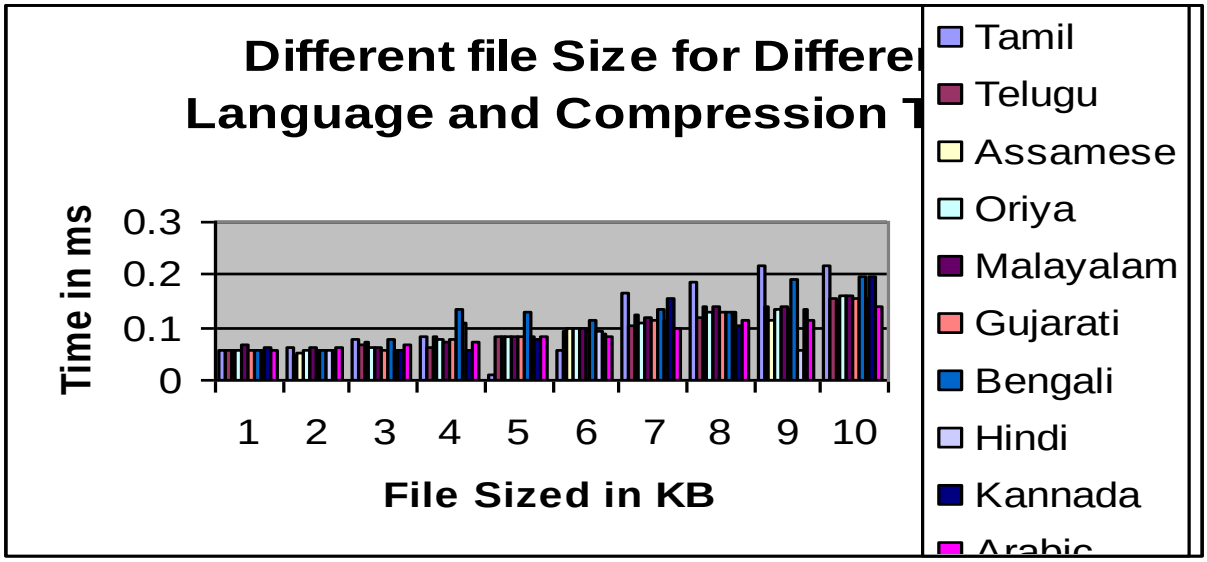


Figure 8: Graph for Different File sized for Different Languages and Compression Time

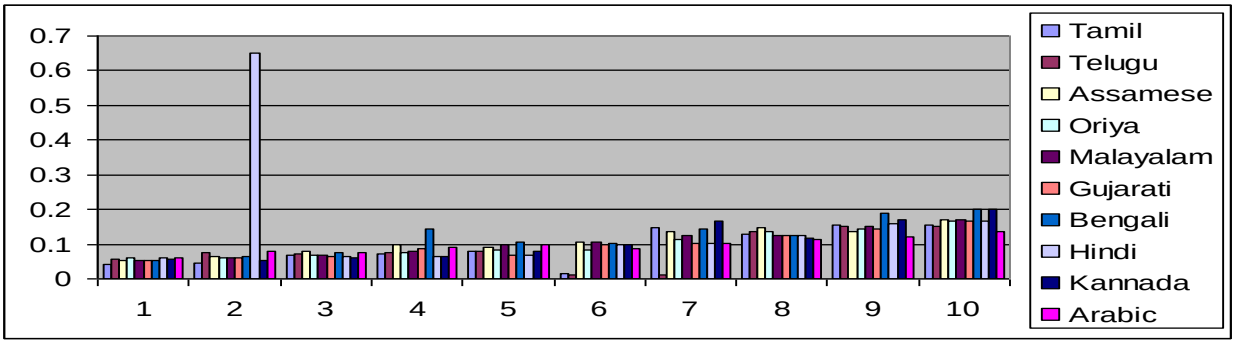


Figure 9: Graph for Different File sized for Different Languages and Decompression Time