**RESULT AND DISCUSSION**

This section presents a detailed evaluation of the compression efficiency of six standard independent algorithms and their hybrid combinations (75 algorithms-dataset combinations) when applied to Hindi text files of varying sizes. The analysis is conducted on three datasets: small, medium, and large files. The results are benchmarked using four parameters: compression ratio (Equation 1), compression speed ( equation 2), decompression speed (Equation 3) and a derived efficiency metric. Efficiency was computed using a composite score: 40% compression ratio, 30% compression speed, and 30% decompression speed (Equation 4 and 5).

**4.1. Compression Ratio**

TABLE 1. Comparison of Independent and Hybrid Algorithm in terms of Compression Ratio

|  |  |  |
| --- | --- | --- |
| **Data set** | **Independent algorithm** | **Hybrid Algorithm** |
| Small File | Bzip2 (6.57) | Bzip2+Brotli/Zstd (6.57) |
| Medium File | Bzip2 (7.56) | Bzip2+Brotli (7.6) |
| Large File | Brotli (117.11) | LZMA + brotli (141.59) |

According to table 1 for small files, both the independent algorithm Bzip2 (6.57) and the hybrids Bzip2 +Brotli / Zstd (6.57) offered the highest compression ratio, showing that some hybrids can match the best-performing standalone compressors .For medium files, the hybrid Bzip2 + Brotli (7.6) marginally outperformed independent Bzip2 (7.56).For large files, the hybrid LZMA + Brotli (141.59) significantly exceeded **Br**otli(117.11)by offering 21% improvement in compression ratios, the best independent algorithm—demonstrating the effectiveness of deep-layered compression for larger datasets

**4.2. Compression Speed**

TABLE 2. Comparison of Independent and Hybrid Algorithm in terms of Compression Speed

|  |  |  |
| --- | --- | --- |
| **Data set** | **Independent algorithm**  **(Speed (MB/s))** | **Hybrid Algorithm**  **(Speed (MB/s))** |
| Small File | Zstd (45.99) | Zstd + LZ4HC (46.31) |
| Medium File | Zstd (76.61) | Zstd+Brotli (70.34) |
| Large File | Zstd (546.22) | Zstd+ Brotli (1071.57) |

Table 2 shows thatZstd led among standalone algorithms for all file sizes.While, the hybrid Zstd + LZ4HC (46.31 MB/s) slightly exceeded Zstd (45.99 MB/s) for small files, while Zstd + Brotli (1071.57 MB/s) more than doubled Zstd's speed for large files, making it ideal for high-throughput compression applications

**4.3. Decompression Speed**

TABLE 3.Comparison of independent and hybrid algorithm in terms of Decompression Speed

|  |  |  |
| --- | --- | --- |
| **Data set** | **Independent algorithm(Speed(MB/s))** | **Hybrid Algorithm(Speed(MB/s))** |
| Small File | LZMA (281.19) | LZ4HC + Zstd (323.83) |
| Medium File | LZMA (456.98) | LZ4HC + Zstd (485.04) |
| Large File | Zstd (593.79) | Zstd + LZ4HC (663.78) |

Table 3 illustrates that for small and medium files, hybrid LZ4HC + Zstd outpaced LZMA, with decompression speeds of 323.83 MB/s and 485.04 MB/s, respectively.In large file decompression, Zstd + LZ4HC (663.78 MB/s) performed best, outperforming even the fastest standard decompressor (Zstd: 593.79 MB/s) by 11.78%.

**4.4. Compression Efficiency**

Efficiency was computed using a normalized composite score: 40% compression ratio, 30% compression speed, and 30% decompression speed (Equation 4 and 5).

TABLE 4.Comparison of independent and hybrid algorithm in terms of Compression Efficiency

|  |  |  |
| --- | --- | --- |
| Data set | Independent algorithm | Hybrid Algorithm |
| Small File | Zstd (0.5406) | Zstd + LZ4HC (0.6764) |
| Medium File | Zstd (0.6293) | Zstd + Brotli (0.582) |
| Large File | Zstd (0.6836) | Zstd + LZ4HC(0.8567) |

From table 4 it is evident that results clearly highlight the dominance of hybrid algorithms, particularly combination like Zstd + LZ4HC in large files which outperform standard Zstd by nearly 25.7% and provides optimal compression without compromising speed.

**5. STATISTICAL AND COMPARATIVE ANALYSIS**

**5.1. Trends in Compression Efficiency Across Dataset Sizes**

Table 5**.** Top 10 Compression Algorithms Based on Weighted and Normalized Efficiency Scores Small, Medium, and Large Hindi Text Datasets

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rank** | **Algorithm / Hybrid**  **(small files)** | **Efficiency**  **(small files)** | **Algorithm / Hybrid**  **(medium files)** | **Efficiency**  **(medium files)** | **Algorithm / Hybrid**  **(Large files)** | **Efficiency**  **(Large files)** |
| 1 | Zstd + LZ4HC | 0.6764 | Zstd(Independent) | 0.6293 | Zstd + LZ4HC | 0.8597 |
| 2 | LZ4HC + Zstd | 0.5708 | Zstd + Brotli | 0.582 | Zstd + Brotli | 0.8057 |
| 3 | Bzip2 + LZ4HC | 0.5591 | Zstd + LZ4HC | 0.5678 | Zstd(Independent) | 0.6836 |
| 4 | Bzip2 + Zstd | 0.5556 | Bzip2 + Zstd | 0.4908 | Zstd + Bzip2 | 0.6779 |
| 5 | Bzip2 + Brotli | 0.5476 | Bzip2 + Brotli | 0.4907 | Brotli + Zstd | 0.6683 |
| 6 | Brotli +Zstd | 0.5430 | Bzip2 + LZ4HC | 0.4899 | Zstd + LZMA | 0.6539 |
| 7 | Zstd (Independent) | 0.5406 | LZ4HC + Zstd | 0.482 | Brotli + LZ4HC | 0.6116 |
| 8 | Zstd + Brotli | 0.4951 | Bzip2(Independent) | 0.4717 | Brotli (Independent) | 0.6073 |
| 9 | LZMA(Independent) | 0.4839 | Bzip2 + LZMA | 0.4584 | Brotli + LZMA | 0.5817 |
| 10 | Bzip2 + LZMA | 0.4788 | Brotli + Zstd | 0.462 | Brotli + Bzip2 | 0.573 |

Efficiency

Figure 1. Normalized and Weighted Efficiency Comparison of Compression Techniques for Hindi Text

Table 5 and Figure 1 jointly illustrate the comparative efficiency of the top ten compression algorithms and hybrid methods across three dataset sizes—small, medium, and large. A consistent trend emerges wherein hybrid combinations significantly outperform most standalone algorithms, particularly for larger datasets. The hybrid method **Zstd + LZ4HC** secures the highest efficiency scores for both small (0.6764) and large (0.8597) files, while the standalone **Zstd** achieves the top rank for medium-sized files (0.6293). The bar chart in Figure 1 visually reinforces this pattern, highlighting the dominance of Zstd-centric hybrid configurations such as **LZ4HC + Zstd**, **Bzip2 + Zstd**, and **Zstd + Brotli** across various ranks and file sizes. Notably, the visual layout also emphasizes the increasing contribution of decompression speed (green bars) in the overall efficiency of top-ranking hybrids, particularly for larger datasets. This integrated analysis suggests that hybrid models leveraging Zstd as a core component consistently yield superior performance due to their ability to balance high compression ratio, fast compression speed, and efficient decompression. Consequently, Zstd + LZ4HC stands out as the most balanced and adaptable compression technique across diverse Hindi text data volumes.

To further validate the efficiency of the proposed hybrid algorithm, a direct performance comparison between **Zstd + LZ4HC** and the **independent standard compression algorithms** was conducted on the large file dataset. This comparison highlights the hybrid’s ability to maintain a high compression ratio while significantly improving speed, resulting in superior overall efficiency.

**TABLE 6.** Performance Comparison of the Proposed Hybrid Algorithm (Zstd + LZ4HC) with Standard Compression Algorithms on Large Hindi Text Files.

| **Algorithm** | **Compression Ratio** | **Compression Speed (MB/s)** | **Decompression Speed (MB/s)** | **Efficiency Score** |
| --- | --- | --- | --- | --- |
| **Zstd + LZ4HC** | **94.82** | **1055.69** | **663.78** | **0.8597** |
| Zstd | 94.49 | 546.22 | 593.79 | 0.6836 |
| Brotli | 117.11 | 312.66 | 426.47 | 0.6073 |
| Bzip2 | 9.77 | 18.75 | 4.88 | 0.021 |
| LZ4HC | 3.79 | 32.51 | 554.34 | 0.258 |
| LZMA | 141.91 | 5.96 | 16.56 | 0.0573 |

Table 6 presents this comparative analysis, showcasing the balanced advantage that Zstd + LZ4HC offers over all standard methods. The hybrid achieves a strong efficiency score of 0.8597, combining a high compression ratio (94.82) with exceptional speed (1055.69 MB/s compression, 663.78 MB/s decompression). Among standard algorithms, Zstd performs best independently (efficiency: 0.6836) but remains outpaced by the hybrid. Others like LZMA and Bzip2, though offering high compression ratios, suffer from slow speeds, resulting in very low efficiency (0.0573 and 0.021, respectively). This concise comparison underscores the effectiveness of hybridization in balancing speed and compression quality for large-scale Hindi text compression.

**5.3. Performance Balance Analysis (Compression Ratio vs. Compression Speed)**

We have analyze compression ratio vs. compression speed for large files, since that's where trade-offs are most pronounced

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TABLE 8. Performance Balance Analysis   |  |  |  | | --- | --- | --- | | **Algorithm** | **Compression Ratio** | **Compression Speed (MB/s)** | | LZMA + Brotli | 141.59 | 6.09 | | LZMA + LZ4HC | 141.10 | 6.01 | | Zstd + Brotli | 94.49 | 1071.57 | | Zstd + LZ4HC | 94.82 | 1055.69 | | Brotli + Zstd | 117.10 | 491.50 | | Brotli + LZ4HC | 117.08 | 426.05 | | Zstd (indep.) | 94.59 | 546.22 | | Brotli + LZMA | 117.5 | 208.44 | | FIGURE 2. Compression Ratio vs. Compression Speed Analysis |

Table 8 and figure 2. analysis reveals that Zstd-based hybrids consistently achieve the best performance balance for large files. In particular, Zstd + Brotli and Zstd + LZ4HC stand out by offering exceptionally high compression speeds (over 1000 MB/s) while maintaining competitive compression ratios (~94.5). These are well-suited for real-time or high-throughput scenarios.

On the other hand, combinations like LZMA + Brotli and LZMA + LZ4HC deliver the highest compression ratios (above 141) but at significantly reduced speeds, making them ideal for storage-focused tasks where time is not critical.

Hybrid algorithms involving Brotli (e.g., Brotli + Zstd, Brotli + LZ4HC) emerge as strong all-rounders, balancing both compression ratio and speed effectively.

Overall, the results affirm that hybrid approaches can be tailored to specific needs, outperforming many standalone algorithms by strategically combining the strengths of two components.

**5.4. Contribution Frequency of Core Algorithms in Hybrid Compression Techniques**

FIGURE 3. Frequency of Top Components in Top-Performing Hybrids. Total combinations analyzed: 30 (10 per size category)

**Contribution Frequency of Core Algorithms in Hybrid Compression Techniques**

Table 9 and figure 3 shows the component impact analysis of the frequency in the top-performing hybrid compression algorithms reveals distinct trends in their relative contribution to overall efficiency. Among the 30 combinations evaluated across small, medium, and large Hindi text files, Zstd emerged as the most frequently occurring component, appearing in 15 of the top hybrids—accounting for 50% of all top-performing combinations. This was followed by Brotli, which appeared 11 times (36.7%), and Bzip2, included in 10 instances (33.3%). LZ4HC showed a solid presence with 8 appearances (26.7%), while LZMA appeared less frequently, in only 3 of the top hybrids (10%). These trends suggest that Zstd, Brotli, and Bzip2 are more likely to contribute to balanced and high-efficiency compression outcomes, making them strong candidates for constructing optimized hybrid algorithms. The dominance of Zstd in particular underscores its adaptability and performance when paired with other compressors.

**CONCLUSION**

This research highlights the remarkable potential of hybrid compression techniques for efficiently compressing Devanagari-encoded Hindi text, a script known for its complexity and richness. By evaluating a range of lossless algorithms, both independent and hybrid, across diverse datasets, this study uncovers key insights into how these methods perform under various compression and decompression conditions.

The results underscore a clear trend: hybrid algorithms consistently outperform their standard counterparts in terms of **compression ratio, speed**,decompression speed and **overall performance**. Among the various combinations tested, the hybrid pairing of **Zstd + LZ4HC(Among top 3 across all the data sets) emerged as the standout performer, offering a 93.27% improvement in compression speed, 11.8% better decompression speed**, and **~25% higher weighted and normalized efficiency** compared to standard algorithms like Zstd in large dataset files. striking the optimal balance between compression ratio and processing time. This demonstrates that leveraging the strengths of multiple algorithms offers a clear advantage, particularly for large-scale text datasets.