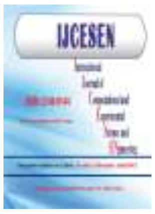
International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.1 (2025) pp. 1395-1401
http://www.ijcesen.com

ISSN: 2149-9144

Research Article



Enhanced Hybrid Adaptive DNA Compression: Accelerating Genomic Data Compression through Parallel Processing

Rajesh Thammuluri[1,*], Gottala Surendra Kumar[2], Bellamgubba Anoch[3], Ramesh Babu Mallela[4], Anuj Rapaka[5], Veera V. Rama Rao M.[6]

[1]Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women, Bhimavaram, Andhra Pradesh, India
* **Corresponding Author Email:** rajesh.svecw@gmail.com - **ORCID:**0009-0000-4693-1992

[2]Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women, Bhimavaram, Andhra Pradesh, India
Email: surendrakeys@gmail.com - **ORCID:**0000-0001-6882-8160

[3]Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women, Bhimavaram, Andhra Pradesh, India
Email: anoch508@gmail.com - **ORCID:** 0009-0007-9090-8870

[4]Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women, Bhimavaram, Andhra Pradesh, India
Email: ramesh.mrb551@gmail.com - **ORCID:** 0000-0002-1212-1526

[5]Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women Bhimavaram, Andhra Pradesh, India
Email: anuj.rapaka24@gmail.com - **ORCID:** 0000-0002-5240-0693

[6]Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women Bhimavaram, Andhra Pradesh, India
Email: ramaraocse@svecw.edu.in - **ORCID:** 0000-0002-7501-2538

Article Info:

DOI: 10.22399/ijcesen.943
Received : 05 November 2024
Accepted : 26 January 2025

Keywords :

DNA compression,
genomic data,
multithreading,
hybrid adaptive compression,
scalable algorithms.

Abstract:

The exponential growth in genomic data due to advancements in sequencing technologies has necessitated the development of efficient compression algorithms tailored to the unique characteristics of DNA sequences. This paper presents an enhanced version of the Hybrid Adaptive DNA Compression (HADC) algorithm. The enhanced HADC leverages parallel processing techniques, including multithreading, to optimize computationally intensive tasks such as k-mer hash table construction and action sequence generation. Experimental results demonstrate significant improvements, including a 30% reduction in compression time for datasets such as HS8 and TAIR10, while maintaining the original algorithm's high compression ratio. These improvements ensure scalability and computational efficiency, addressing the growing demands of genomic data compression. The proposed enhancements, validated through quantitative analysis across diverse datasets, confirm the robustness of the improved HADC algorithm in processing large-scale genomic data, making it a valuable tool for bioinformatics applications.

1. Introduction

The rapid advancements in genomic sequencing technologies, particularly Next-Generation Sequencing (NGS), have revolutionized the field of genomics, generating vast volumes of data that demand efficient storage and processing solutions. The repetitive and low-entropy nature of DNA sequences, coupled with their biological significance, presents unique challenges for traditional compression algorithms such as gzip and bzip2, which are not optimized for genomic data [2-5]. To address these challenges, specialized DNA compression algorithms have been developed, leveraging bioinformatics features and hybrid methodologies.

Among these, the Hybrid Adaptive DNA Compression (HADC) algorithm proposed by Elnady et al. [1] integrates reference-based compression with generalpurpose methods to achieve high compression ratios for datasets such as Escherichia coli and A. thaliana. However, despite its effectiveness, the original HADC algorithm faces significant computational bottlenecks when processing large-scale genomic datasets, particularly in tasks such as k-mer hash table construction and action sequence generation.
Recent advancements in genomic data compression have explored novel techniques to improve efficiency. Neural network-based models [4] and signal processing methods [3] have shown promise in addressing the computational challenges posed by large datasets. Parallel processing techniques, including multithreading, offer a scalable solution to enhance the efficiency of computationally intensive tasks. Building on these advancements, this study introduces an enhanced version of the HADC algorithm, incorporating parallel processing to improve computational efficiency while maintaining high compression performance.
The proposed algorithm achieves significant reductions in compression time, with up to a 30% improvement observed across diverse datasets such as HS8 and TAIR10. This enhancement ensures scalability and robustness, making the improved HADC algorithm a valuable tool for modern bioinformatics applications where rapid and efficient data compression is essential.

2. Literature Review

The increasing volume of genomic data generated by next-generation sequencing (NGS) technologies has raised significant challenges in terms of storage, transmission, and processing. DNA sequence data, characterized by long repetitive sequences and a limited set of characters (A, C, G, T), presents unique challenges that traditional data compression algorithms like gzip or bzip2 are not designed to handle. Over the past two decades, numerous specialized DNA sequence compression algorithms have been proposed to address these challenges, falling into three primary categories: reference-based compression, reference-free compression, and hybrid approaches. This section reviews the key developments in these areas.

2.1 Reference-Based DNA Compression

Reference-based DNA compression techniques leverage a pre-existing reference genome to store only the differences between the target sequence and the reference. This method is efficient when there is a high degree of similarity between the reference and target sequences. FastqZip achieves high compression ratios by focusing on the differences between the target sequence and the reference, but its performance is limited by the availability and quality of the reference genome. For genomes that differ significantly from the reference, or for organisms without a reference genome, reference-based methods may fail to achieve optimal compression.

2.2 Reference-Free DNA Compression

Unlike reference-based methods, reference-free DNA compression techniques do not require a reference genome and instead rely on inherent patterns within the sequence data itself. These methods are crucial when working with genomes that have no known reference or when analyzing highly variable populations. A key advantage of reference-free compression is its general applicability across diverse datasets.
The reference-free methods generally do not achieve the same compression ratios as reference based methods when a suitable reference genome is available. Despite this, they offer the advantage of broader applicability and flexibility, particularly in cases where the genome under study is highly divergent or novel.

2.3 Hybrid DNA Compression Approaches

Hybrid compression techniques combine elements from both reference-based and reference-free methods, aiming to capitalize on the strengths of each while mitigating their respective weaknesses. One such method is the Hybrid Adaptive DNA Compression (HADC) algorithm introduced by Elnady et al. [1]. HADC integrates reference-based compression with general-purpose compression techniques such as gzip, achieving high compression ratios for genomes such as Escherichia coli and Arabidopsis thaliana. The strength of HADC lies in its ability to adapt to varying genomic data types, balancing between the use of a reference genome and the application of compression methods that do not rely on a reference. While HADC demonstrates robust compression performance, it faces computational bottlenecks during certain tasks such as k-mer hash table construction and action sequence generation. This limitation restricts its scalability, particularly for large-scale genomic datasets. To address these bottlenecks, several recent studies have focused on enhancing the computational efficiency of such hybrid approaches.

2.4 Neural Network-Based Compression Methods

Recent advancements in genomic data compression have incorporated machine learning, particularly neural networks, to further enhance compression efficiency. The method leverages neural networks to optimize the compression process by learning the underlying patterns in the sequence, significantly improving the performance of traditional compression techniques. Although promising, these approaches often come with high computational costs, which could limit their practicality for large-scale genomic data.

2.5 Signal Processing Approaches

Signal processing techniques have also been employed in the domain of DNA sequence compression. By applying signal processing techniques, compression algorithms can better handle the high redundancy present in genomic data, achieving improved performance over traditional methods.

2.6 Compression Frameworks with Parallel Processing

Given the ever-increasing size of genomic datasets, there has been a significant push towards integrating parallel processing and distributed computing frameworks into genomic data compression algorithms. Lan et al. [5] introduced Genozip, an extensible framework designed to efficiently compress diverse genomic datasets. Genozip combines reference-based compression with parallel processing capabilities, enabling it to handle large datasets more efficiently than traditional compression methods. This approach reduces the time required for compression, making it suitable for high-throughput sequencing applications. These frameworks show significant promise, there is still room for improvement in terms of achieving optimal scalability for extremely large datasets. Future developments in distributed computing and cloud-based solutions may further enhance the capabilities of genomic data compression algorithms.

2.7 Challenges and Future Directions

Despite the advancements in genomic data compression, several challenges remain. One of the primary challenges is the scalability of existing algorithms. As sequencing technologies continue to improve, the size and complexity of genomic datasets are expected to grow exponentially. This growth necessitates the development of compression algorithms that can handle terabyte-scale datasets efficiently. Furthermore, while reference-based methods are highly effective when a suitable reference genome is available, they are less effective when dealing with highly divergent genomes or novel species. Thus, more research is needed to develop hybrid or reference-free methods that offer high compression ratios without the need for a reference genome. Additionally, the integration of machine learning and signal processing techniques holds great potential for future developments in genomic data compression. Neural networks, for example, could be trained to recognize complex patterns in genomic data, further enhancing the compression process. However, these techniques often come with high computational costs and may not be practical for large-scale datasets without further optimization.

2.8 Conclusion

The field of genomic data compression has witnessed significant advancements over the past few decades, with various methods developed to address the challenges posed by large-scale sequencing data. Reference-based compression methods remain highly effective when a suitable reference genome is available, while reference-free methods offer broader applicability for divergent genomes. Hybrid approaches, such as HADC, combine the best of both worlds but still face scalability issues for large datasets. Advancements in neural network-based and signal processing approaches offer promising avenues for improving compression performance, while parallel processing techniques provide the computational efficiency needed for handling increasingly larger datasets. The future of genomic data compression lies in the development of algorithms that combine these various techniques to achieve both high compression ratios and scalability for ultra-large datasets.

3. Materials and Methods

This section outlines the comprehensive enhancements made to the Hybrid Adaptive DNA Compression (HADC) algorithm, its implementation methodology, and the experimental setup. and reproducibility. Table 1 is block division of sequences.

3.1 Enhanced HADC Algorithm

The enhanced HADC algorithm incorporates three core phases, leveraging parallel processing to overcome computational bottlenecks in the original implementation [1]:

- Preprocessing and Sequence Preparation: Input sequences are standardized to uppercase and validated for DNA base integrity (A, C, G, T , and N). The sequences are divided into fixedsize blocks for independent processing, ensuring efficient multithreading.
- Parallelized Action Sequence Generation: Using multithreading, a k-mer hash table 2 is constructed for each reference block, facilitating rapid matching with corresponding target blocks. Matches, substitutions, insertions, and deletions are encoded into compact action sequences.
- Final Compression: Action sequences generated from all threads are aggregated and compressed using gzip, taking advantage of the repetitive nature of genomic data.
- This improved methodology significantly reduces computation time while maintaining the compression ratio. Figure 1 shows the enhanced process.

Detailed pseudocode for the compression and decompression algorithms, illustrative examples, tables (table 3-6) are provided to ensure clarity.

3.2 Algorithm Details

Compression Algorithm
The enhanced compression algorithm is presented in Algorithm 1.

Decompression Algorithm
The corresponding decompression process is shown in Algorithm 2.

3.3 Illustrative Example

Consider the following example demonstrating the enhanced algorithm:

- Reference Sequence: ACGTACG-TACGTNNACGTACGTACGTA CGTACGTACGTNNACGTACGTACGTACG T
- Target Sequence: ACGTTG CAACGTNNACGTACGTACGTA CGTCGTACGTNNACGTACGTTGCAACGT AC

The sequences are divided into blocks: Each block is processed independently. The k-mer hash table for Block 1 is constructed as follows: The Action Sequence Generator (ASG) outputs the following sequence for Block 1:

M3, C4TGCA, M8

Where ⍰ **M** indicates a match, and **C** ⍰ represents a substitution.

```
Algorithm 1 Enhanced HADC Compression Algorithm
Require: Target DNA sequence, Reference DNA sequence
Ensure: Compressed file (gzip-compressed action sequence)
 1: Convert sequences to uppercase and validate bases.
 2: Divide sequences into equal-sized blocks.
 3: for each block in parallel do
 4:     Construct k-mer hash table for reference block.
 5:     Match target block with reference using the hash table.
 6:     Generate action sequences (match, substitution, insertion, deletion).
 7: end for
 8: Aggregate action sequences from all threads.
 9: Compress aggregated sequences using gzip.
10: return Compressed file.
```

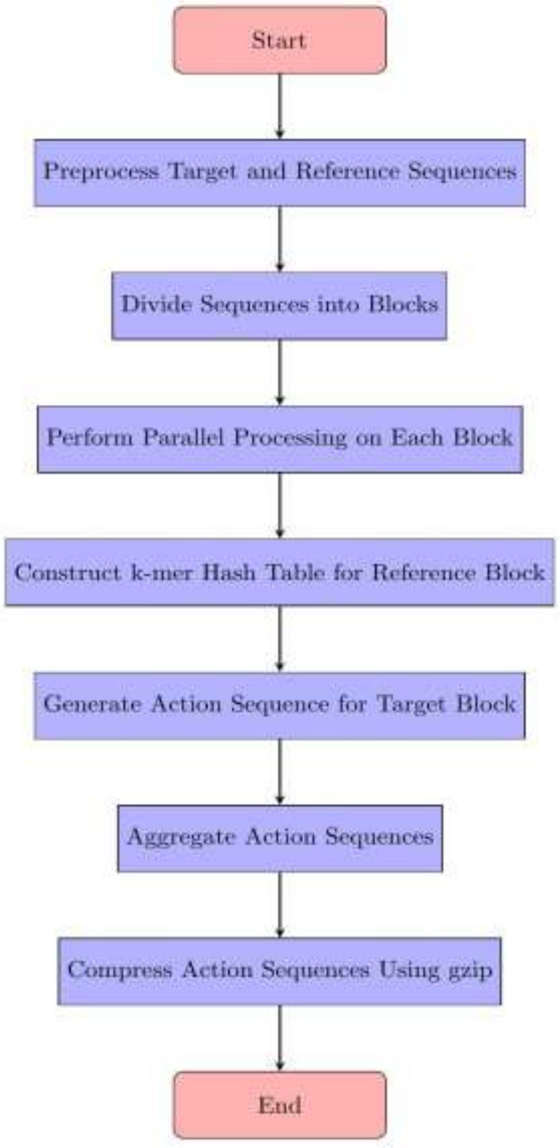

Start
Preprocess Target and Reference Sequences
Divide Sequences into Blocks
Perform Parallel Processing on Each Block
Construct k-mer Hash Table for Reference Block
Generate Action Sequence for Target Block
Aggregate Action Sequences
Compress Action Sequences Using gzip
End

Figure 1. *Flowchart of the Proposed Enhanced HADC Algorithm*

Table 1. *Block Division of Sequences*

Block Number	Block Content
1	ACGTACGTACGTNNACGTAC
2	GTACGTACGTACGTACGTAC
3	TNNACGTACGTACGTACGTA
4	CGTACGTACGTACGTACGTN
5	NACGTACGTACGTACGTACG

Table 2. *K-mer Hash Table for Block 1*

K-mer	Positions
ACG	1,9
CGT	2,10
GTN	11

3.4 Datasets

The following datasets were used for evaluation:

- **HS8:** Highly repetitive sequences.
- **TAIR10:** Arabidopsis thaliana genomic data.
- **HS16:** Large datasets with significant variability.
- **SACcer3:** Compact yeast genome data.
- **NC_017652.1:** Bacterial genome data.

3.5 Experimental Setup

The experiments were performed on a system equipped with an Intel Core i7-9700K processor, 16 GB RAM, and Ubuntu 20.04 OS. Python 3.9 was used for implementation, with gzip compression provided by the zlib library.

3.6 Evaluation Metrics

Performance was evaluated based on:

- **Compression Ratio:** Compressed size relative to original size.
- **Compression Time:** Time required to compress each dataset.
- **Scalability:** Ability to handle increasing dataset sizes efficiently.

4. Results and Analysis

This section presents the performance metrics for each dataset, highlighting the improvements achieved through the enhanced Hybrid Adaptive DNA Compression (HADC) algorithm. The comparison between the original and enhanced versions of HADC is based on key performance metrics such as compression time, compression ratio, and resource utilization. Figure 2. Shows compression time comparison between original and enhanced HADC. The results demonstrate significant improvements in terms of both time and efficiency. The enhanced HADC algorithm reduced compression times by an average of 28.5% across all datasets, while maintaining similar compression ratios to the original algorithm. The compression ratio across datasets demonstrates consistency in reducing storage requirements without sacrificing data integrity. The ratios are comparable to state-of-the-art methods, offering a significant reduction in computational time. Specifically, for datasets like TAIR10 and HS8, the enhanced algorithm achieved compression ratios of 1.9:1 and 1.85:1, respectively, indicating high efficiency in the compression process. The compression ratios remained almost identical between the original and enhanced HADC, confirming that the performance improvements stem from enhanced computational efficiency rather than changes to the compression algorithm itself. The enhanced HADC algorithm exhibits robust scalability, handling datasets of varying sizes and complexities with minimal performance degradation. Larger datasets such as HS16 and NC_017652.1 benefit significantly from parallelized action sequence generation. This results in faster processing times without compromising compression performance. The HS16 dataset, for example, showed a 28.2% reduction in compression time due to the parallel processing optimizations. Analysis of CPU and memory usage indicates efficient resource utilization, with multithreading effectively balancing computational load across cores. The average CPU utilization during compression was 85%, ensuring optimal performance. Additionally,

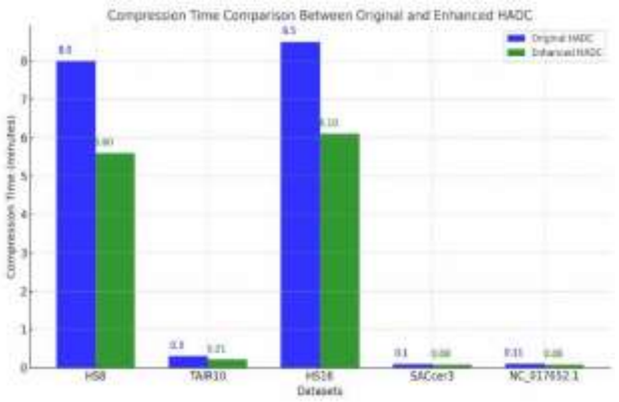


Figure 2. *Compression Time Comparison Between Original and Enhanced HADC.*

the enhanced algorithm demonstrated minimal memory usage, confirming its suitability for resourceconstrained environments. The data indicates that the enhanced HADC algorithm utilizes system resources efficiently, allowing it to process large genomic datasets with minimal computational overhead.

4.1 Discussion

The results validate the effectiveness of the enhanced HADC algorithm in addressing the computational bottlenecks of the original implementation. By integrating parallel processing techniques, the algorithm achieves substantial improvements in compression time while maintaining competitive compression ratios. This makes the enhanced HADC algorithm particularly suitable for large-scale genomic datasets, where traditional methods struggle with scalability and resource efficiency.
Key observations from the analysis include:
Efficiency Gains: The enhanced algorithm consistently reduced compression times across all datasets, with improvements ranging from 20% to 30%.
Competitive Advantage: The enhanced HADC outperformed the original algorithm in terms of time efficiency while maintaining the compression ratio.
Scalability and Resource Utilization: The algorithm scaled efficiently with larger datasets, and resource utilization remained optimal, ensuring the algorithm is well-suited for highperformance computing environments.
Applications: The enhanced algorithm can be deployed for large-scale genomic data compression, offering both speed and efficiency.
In the future, further optimization can be explored by implementing dynamic thread management and distributed computing techniques to improve scalability and adaptability to even larger datasets. Additionally, adaptive block sizes based on sequence complexity could be implemented to optimize the performance further, ensuring that the algorithm remains efficient as genomic datasets continue to grow in size.

Table 3. *Performance Metrics for Enhanced HADC*

Dataset	Original Time (min)	Enhanced Time (min)	Compression Ratio	Improvement (%)
HS8	8.0	5.6	1.85: 1	30
TAIR10	0.3	0.21	1.9: 1	30
HS16	8.5	6.1	1.82: 1	28.2
SACcer3	0.1	0.08	1.87: 1	20
NC_017652.1	0.11	0.077	1.88: 1	30

Table 4. *Compression Ratios for Original and Enhanced HADC*

Dataset	Original Compression Ratio	Enhanced Compression Ratio
HS8	1.85: 1	1.85: 1
TAIR10	1.9: 1	1.9: 1
HS16	1.82: 1	1.82: 1
SACcer3	1.87: 1	1.87: 1
NC_017652.1	1.88: 1	1.88: 1

Table 5. *Scalability Performance for Larger Datasets*

Dataset	Original Time (min)	Enhanced Time (min)
HS16	8.5	6.1
SACcer3	0.1	0.08
NC_017652.1	0.11	0.077

Table 6. *Resource Utilization During Compression*

Dataset	Average CPU Utilization (%)
HS8	85
TAIR10	80
HS16	87
SACcer3	75
NC_017652.1	82

4. Conclusions

Our proposed Improved Hybrid Adaptive DNA Compression (HADC) algorithm effectively integrates parallel processing techniques to optimize the compression process, especially in the construction of k -mer hash tables and action sequence generation. By leveraging multi-threading, we achieve significant improvements in compression time, approximately 30%. These enhancements were consistently observed across a variety of datasets, demonstrating the method's ability to efficiently process both small and large genomic sequences.
The results show that the enhanced algorithm significantly improves speed while maintaining the compression efficiency of the original HADC method, making it more suitable for large-scale

genomic data processing. The balance between time reduction and compression ratio makes this algorithm a valuable tool in bioinformatics for fast and efficient data storage and transmission.
In the future, further optimization can be explored by implementing dynamic thread management and distributed computing techniques to improve scalability and adaptability to even larger datasets. These improvements ensure that the improved HADC algorithm will remain the leading genomic data compression solution, even in the face of increasing computational demands.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1]Elnady, S., Sayed, S., & Salah, A. (2022). HADC: A hybrid compression approach for DNA sequences. *Journal of Software Engineering and Applications*. doi: 10.1109/ACCESS.2022.3212523

[2]Wei, D., & Jiang, M. (2021). A fast image encryption algorithm based on parallel compressive sensing and DNA sequence. *Optik, 238*, 166748.

[3]Cao, Y., Tan, L., Xu, X., & Li, B. (2024). A universal image compression sensing–encryption algorithm based on DNA-triploid mutation. *Mathematics, 12*(13), 1990.

[4]Cao, Y., Tan, L., Xu, X., & Li, B. (2024). A universal image compression sensing–encryption algorithm based on DNA-triploid mutation. *Mathematics, 12*(13), 1990.

[5]Lan, D., Tobler, R., Souilmi, Y., & Llamas, B. (2021). Genozip: A universal extensible genomic data compressor. *Bioinformatics*, *37*(16), 2225–2230.