ScBlkCom: An Integrated Compression Algorithm for Single-Cell RNA Sequencing Data

Yuanxin Zhang[1, 2], Yang Lu[2], Xiao Sun[1*], Jue Fan[2*]

[1]School of Biological Science and Medical Engineering, Southeast University, Nanjing 211189, China
[2]Xinge Yuan Biotechnology Co., Ltd., Nanjing 211189, China

Abstract: High-throughput sequencing advancements have shifted genomic project bottlenecks from data generation to computational storage and analysis. Single-cell RNA-seq (scRNA-seq) data exhibits unique structural features, including extensive labeled sequence identifiers, which conventional compression tools fail to optimize. This study proposes ScBlkCom, a specialized compression scheme for scRNA-seq data. The method partitions sequencing data into distinct blocks and applies tailored compression strategies: differential encoding for numerical attributes, Huffman coding for categorical labels, and context-adaptive encoding for sequence identifiers. Experiments demonstrate ScBlkCom achieves 84.29% higher compression gain compared to single-module approaches and outperforms generic tools (e.g., GZIP, BZIP2) by 6.44% in compression ratio, while maintaining stable processing speeds. This block-wise adaptive framework effectively addresses scRNA-seq data redundancy, offering enhanced storage efficiency for large-scale single-cell studies.

1 Introduction

With the rapid development of high-throughput sequencing technologies, the cost of genomic sequencing has drastically decreased, while the speed of data generation has grown exponentially [1]. This technological innovation has driven the rapid development of genomics, transcriptomics, and other multi-omics studies, making high-throughput sequencing data a crucial component of today's big data field. The volume of these data has rapidly shifted from the PB (petabyte) scale to the EB (exabyte) scale [2]. Single-cell sequencing (scRNA-seq), as an important application of high-throughput sequencing technology, captures gene expression information at single-cell resolution, providing an essential tool for understanding cell heterogeneity and dynamic changes. In recent years, with the development of cell barcodes and microfluidic technologies, the throughput of scRNA-seq has significantly increased, enabling the generation of hundreds of GB to TB-scale data in a single experiment [3]. Typical scRNA-seq studies, such as those using the 10x Genomics platform, can detect tens of thousands of cells in multiple samples at once, increasing the data volume by several orders of magnitude. This data explosion has posed significant challenges in terms of storage, transmission, and processing, becoming a critical bottleneck in current life sciences research.

Single-cell transcriptome sequencing technology has wide application prospects in various scientific fields, especially in clinical medicine and basic biological research. ScRNA-seq represents an advanced approach for measuring gene expression at the single-cell level. It helps uncover inter-cell heterogeneity [4], explore cell-specific changes in-depth, identify rare cell types, and analyze stem cell or progenitor cell differentiation. However, as data volumes soar, the storage and transmission of these data face significant challenges. There is an urgent need for high-performance compression algorithms to optimize data storage space, enhance transmission efficiency, and, in turn, support the continuous advancement of single-cell research and its applications.

Currently, there are several types of compression methods for high-throughput sequencing data, primarily categorized into file-based general compression schemes (such as Gzip and 7-Zip), reference-based compression algorithms, and non-reference-based compression algorithms. While general compression schemes can be directly applied to genomic data, they exhibit significant limitations when processing scRNA-seq data. Existing general compression schemes are unable to optimize the high-dimensional characteristics and complex structure of scRNA-seq data, resulting in suboptimal compression ratios. Additionally, the compression and decompression speeds are insufficient to meet the demands of rapidly growing data scales and processing needs. To address these challenges, various researchers have proposed multiple compression algorithms aimed at improving high-throughput sequencing data compression efficiency. These algorithms are mainly divided into reference-based and non-reference-based categories. Reference-based compression algorithms achieve effective compression by aligning the sequence to a

* Corresponding author: Fan Jue, fanjue@singleronbio.com; Xiao Sun, xsun@seu.edu.cn

reference sequence and utilizing the similarity between matching segments [5]. These methods perform well when the reference sequence is sufficiently complete and the match rate is high. Notable reference-based algorithms include HRCM [6] and AMGC [7]. The HRCM algorithm extracts information from both the reference and the sequence to be compressed, constructs an index based on the reference sequence, matches the sequence to find similar parts, and then encodes matching and other related information using various encoding methods (such as run-length encoding) for lossless compression. The AMGC algorithm matches genomic sequences with a reference sequence, producing three blocks: fully matched, partially matched, and unmatched sections. For the fully matched block, it uses bit-plane differential encoding; for the partially matched block, it applies adaptive binary mismatch position encoding; and for the unmatched portion, it utilizes recursive segmentation for matching compression.

Non-reference-based compression algorithms, on the other hand, do not rely on an external reference sequence. They primarily exploit the redundancy and contextual features inherent in the data to achieve compression. These algorithms demonstrate better generality in diverse data environments. Notable non-reference-based compression algorithms include Spring [8], PgRc [9], and ColoRd [10]. The SPRING algorithm focuses on shorter sequences in the genome, employing hash-based reordering to cluster similar sequences, followed by run-length and entropy encoding for compression. For longer sequences, it utilizes the BSC compression library. The PgRC algorithm classifies genomic data sequences by quality, constructs a pseudo-genome from high-quality sequences, and encodes sequence position information based on the relationship between the sequence and the pseudo-genome. It also employs encoding algorithms to record pairing information for paired-end data, facilitating compression. The ColoRd algorithm constructs a string graph, where repetitive sequence fragments serve as nodes, and reduces storage by sharing references to repeated sequence fragments, thereby efficiently compressing long sequences.

Reference-based algorithms depend on the completeness and high match rate of the reference sequence, while non-reference algorithms are limited in their data compression capabilities. They also suffer from slower overall compression and decompression speeds, making them inadequate for large-scale datasets generated by new technologies. Current compression schemes for scRNA-seq data still have shortcomings, particularly in optimizing sequences that contain abundant identity tag information. Therefore, it is crucial to design an efficient compression algorithm tailored to the internal characteristics and structural patterns of scRNA-seq data.

To address these issues, this paper proposes an efficient compression algorithm, ScBlkCom, which employs different compression methods based on the distinct features of the data. First, it divides single-cell transcriptome sequencing data into three types of blocks based on their characteristics: cell identity blocks, molecular identity blocks, and capture fragment blocks. The cell identity blocks exhibit high redundancy and fixed structural features, the molecular identity blocks contain short and sparse data fragments, and the capture fragment blocks are complex and contextually dependent. For these three different block types, differential encoding, Huffman coding, and context-adaptive binary arithmetic coding are applied. By integrating these three compression techniques and adjusting parameters based on the block characteristics, the algorithm fully exploits data features and redundancy, improving overall compression performance. Experimental results demonstrate that, compared to existing mainstream compression tools, ScBlkCom offers a significant advantage in compression ratio and also maintains stable compression times. This algorithm achieves lossless compression and does not affect downstream tasks such as gene expression clustering and pathway analysis. It provides a new and efficient solution for the compression and storage of large-scale single-cell sequencing data, particularly for single-cell RNA sequencing (scRNA-seq).

2 Methods

2.1 Single-Cell Transcriptome Block Partitioning

Single-cell transcriptome sequencing(scRNA-seq) data [11] is typically stored in FASTQ format, where each data record includes three components: the sequence identifier (Name) line, the nucleotide sequence (Seq) line, and the quality score (Quality) line. The Name line contains additional information related to the sequencing sample, such as the sequencing platform type and primer information, which describe the background and configuration of the sequencing experiment and ensure the ability to distinguish between different sample sources and experimental setups. The Seq line contains not only the nucleotide sequence obtained by sequencing but also a series of identity tags, including cell barcode, unique molecular identifiers (UMI), linkers, and polyT sequences [12-13]. These identity tags are defined as a set, denoted as ID = {Cell Barcode, UMI, Linker, PolyT}. The complete Seq data is composed of the tag set ID and the mRNA nucleotide sequence obtained from sequencing, expressed as: Seq = ID + mRNA, where mRNA is part of the nucleotide sequence set {read sequence}. The FASTQ file stores paired-end reads, with each file containing information from both ends of the DNA strand [14-15]. In paired-end sequencing, one end of the DNA strand is sequenced first, producing read1, which is stored in the Sequencing read1 (R1) file. The opposite end is then sequenced, producing read2, which is stored in the Sequencing read2 (R2) file. R2 contains the captured mRNA sequence. The two files, R1 and R2, form the paired-end reads, and the sizes of these files are usually similar. Each line in R2 corresponds to the same mRNA sequence's information as the corresponding line in R1. In addition to the mRNA sequence in R1, each line also contains identity tag information, including the cell barcode, unique molecular identifier (UMI), linker,

and polyT sequences. It is important to note that some single-cell sequencing technologies may not include the linker in the R1 file. The barcode is used to mark the source of individual cells, and the UMI ensures the uniqueness and traceability of each molecule. The linker connects the barcode with the UMI, ensuring the integrity of the data structure. PolyT is complementary to the PolyA tail of the mRNA, ensuring the capture of mature mRNA, and also serves as a fixed sequence pattern to help distinguish cell identity blocks from other blocks. The Quality line reflects the sequencing machine's confidence in each nucleotide base, corresponding directly to the nucleotide sequence. These three components together provide the complete data foundation for single-cell gene expression research.

The R1 and R2 files are core components of single-cell transcriptome sequencing data, storing different information about the mRNA sequence. The nucleotide sequence in the R1 file has a fixed structural pattern, such as the strict length and arrangement of the barcode and linker [16], exhibiting high regularity and redundancy, which provides an excellent condition for compression optimization. Although R1 and R2 files are similar in size, the barcode and UMI information in the R1 file makes it crucial in single-cell analysis. Optimizing the compression of the R1 file, especially by fully utilizing its fixed structure, is critical for improving the compression ratio and analysis efficiency of single-cell sequencing data.

To efficiently compress single-cell transcriptome sequencing data, the algorithm in this paper designs a pattern recognition and data partitioning process before compression, which parses the data in the R1 and R2 files and divides it into three primary blocks (see Figure 1): cell identity block, molecular identity block, and capture fragment block. During preprocessing, the algorithm extracts barcodes, linker, UMI, PolyT, and one-end mRNA sequences from the Seq line of each R1 file, and extracts the other-end mRNA transcript sequence from the Seq line of each R2 file. The cell identity block consists of the barcode, linker, and PolyT, and is mainly used to mark the single-cell source, maintain data structure integrity, and assist in parsing the starting position of the mRNA sequence. The molecular identity block contains the UMI, which is used to support molecular tracking and deduplication analysis. The capture fragment block consists of one-end mRNA sequences from the R1 file and mRNA capture sequences from the R2 file, providing the complete transcript sequence information for gene expression and serving as the core data source for gene expression quantification and differential gene analysis.

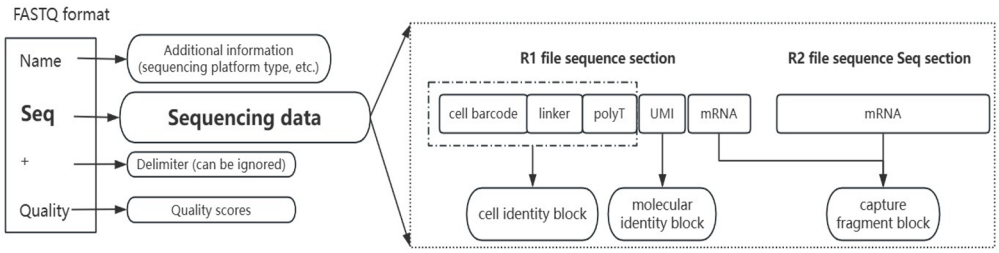

Figure 1. scRNA-seq Data Parsing and Block Partitioning.

In the proposed compression scheme, the Name and Quality fields have simple structures and contain frequently occurring characters, making them suitable for compression using Huffman coding. In contrast, the Seq line data is more complex. For the Seq data, this paper performs data partitioning during the preprocessing stage, dividing it into three blocks. Since the blocks have different characteristics, three compression techniques are integrated to optimize compression efficiency, with each technique tailored to the specific features of each block. Specifically, for the cell identity block, its fixed length and patterned structure result in high regularity, which is easily recognized and utilized by encoding algorithms, enabling more efficient processing. The molecular identity block contains unique molecular identifiers (UMI), and its uniqueness supports molecular tracking and deduplication analysis. The capture fragment block is the core data source for gene expression quantification and differential gene research. Due to the high complexity and diversity of its data, a strategy adapted to its characteristics is required during compression. This block-specific optimization method significantly reduces storage space while effectively preserving data integrity, achieving lossless compression, and providing a reliable solution for the efficient storage and transmission of single-cell transcriptome sequencing data.

2.2 Data Compression Method

The compression method ScBlkCom proposed in this paper employs various compression techniques tailored to the specific characteristics of single-cell transcriptome data. First, for the Name and Quality fields, which have high repetition and contain frequently occurring characters, and given their simple structure, Huffman coding is used for compression. This method leverages the distribution characteristics of symbol frequencies to achieve efficient compression. Then, for the Seq part, which is more complex, ScBlkCom partitions the data during the preprocessing stage into three blocks: cell identity block, molecular identity block, and transcript block. For these three blocks, this paper applies differential encoding, Huffman coding, and context-adaptive coding techniques to efficiently compress the data.

2.2.1 Differential Encoding Compression for the Cell Identity Block

The cell identity block consists of cell barcodes, linkers, and PolyT sequences. It has a fixed length and a patterned structure, serving as an important foundation for the analysis and interpretation of single-cell transcriptome data. The cell barcode is arranged in a specific pattern to mark the source of the single cell; the linker is a fixed-length short sequence connecting the barcode, UMI, and PolyT, and typically contains repetitive and regular sequences; the PolyT sequence is complementary to the PolyA tail of mRNA and is highly repetitive. These similarities and regularities result in significant redundancy in the cell identity block, making it suitable for differential encoding compression.

Differential encoding calculates the differences between adjacent data segments in the identity block, representing similar segments as relative differences and storing only the changing parts while ignoring the repeating parts. This process greatly reduces storage space. To support this, the algorithm first generates a reference sequence file. In this process, the algorithm extracts all information from the cell barcode and linker files, and according to predefined sequence patterns (e.g., a fixed arrangement of barcodes and linkers), it parses their arrangement rules and splitting positions. Taking the Singleron GEXSCOPE® technology as an example, the algorithm systematically arranges each barcode from the barcode database with each linker from the linker database through full permutations and multi-sequence combinations, generating a complete set of reference sequences. This reference sequence library contains approximately three million types of sequences. It is important to note that different single-cell sequencing technologies use different barcode and linker coding libraries, so the reference sequence library is optimized and designed for the specific technology. These reference sequences strictly adhere to the format and pattern rules, ensuring their integrity and consistency when used as reference benchmarks. The generated reference sequence file is then applied to the subsequent differential encoding step, providing precise mapping support for sequence alignment and matching.

2.2.2 Huffman Encoding Compression for the Molecular Identity Block

Huffman coding is primarily used for compressing the molecular identity block (UMI) in scRNA-seq data. The UMI in the molecular identity block is a short sequence used to uniquely mark molecules. In practice, some UMIs occur much more frequently than others, resulting in a significant difference in symbol distribution between high- and low-frequency UMIs.

Huffman coding first scans and counts the frequency of elements in the block, building a Huffman tree. In the tree construction process, high-frequency elements are merged to form upper nodes, and low-frequency elements sink to the lower branches of the tree. Then, based on the structure of the tree, variable-length codes are assigned to each element: shorter codes for high-frequency elements and longer codes for low-frequency elements. For the molecular identity block, the distribution of UMI fragments is concentrated and highly redundant. By counting the frequency of UMI sequences, a Huffman tree is constructed, and variable-length codes are generated. The data block is then converted based on these codes to achieve compression.

2.2.3 Context-Adaptive Encoding Compression for the Capture Fragment Block

The capture fragment block includes one end of the mRNA sequence from the R1 file Seq line and the other end of the mRNA capture sequence from the R2 file Seq line. These nucleotide sequences exhibit significant contextual correlation, meaning the probability of occurrence of the current base is influenced by the combination of surrounding bases. For example, in mRNA sequences, a "G" following a "C" has a higher probability in the Cytosine-phosphate-Guanine (CpG) dinucleotide region, while a "T" following an "A" is more common in some non-coding regions [17]. The nucleotide sequences in the capture fragment block are typically long and large in scale, with complex distributions. The contextual characteristics of the symbols require dynamic modeling during compression to precisely capture the relationships between bases, effectively reducing storage space.

In practice, the algorithm first scans the nucleotide sequences in the capture fragment block, counting the frequency of each base in different contextual environments. Based on these statistics, the algorithm dynamically constructs a context model. The encoder assigns shorter codes to symbols in high-probability contexts and longer codes to symbols in low-probability contexts, thus minimizing the overall encoding length. During the scanning process, the encoder continuously updates the context information and adjusts the encoding scheme based on real-time changes, allowing the compression process to accurately adapt to the contextual characteristics of the nucleotide sequences. This dynamic modeling and encoding strategy effectively captures the contextual dependencies of the sequences, improving the compression efficiency of the capture fragment block.

3 Results and Discussion

This paper presents a thorough evaluation of the compression algorithm ScBlkCom for single-cell transcriptome sequencing data and compares it with leading compression algorithms in the field. Specifically, the performance of ScBlkCom is analyzed by comparing it with three existing, high-performing compression algorithms: Spring, PgRc, and ColoRd. These algorithms are mature and widely used methods in the compression of high-throughput sequencing data, each with different compression performance and applicable scenarios. To comprehensively assess the compression performance of the proposed algorithm, this paper uses single-cell RNA-seq data from public databases (e.g., NCBI) and from

Singleron Biotechnologies Co., Ltd.. The comparison is made based on two dimensions: compression ratio and compression time, to demonstrate the advantages and potential improvements of the ScBlkCom algorithm in practical applications. All experiments were conducted on a virtual machine with the following environment: Linux Mint 21.3 (Virginia), based on Ubuntu 22.04.1, with kernel version 6.8.0-50-generic. The hardware configuration includes an Intel Xeon E5-2630 v4 processor (16 cores, hyper-threading support, 2.20 GHz), 62 GB of RAM, and a 2TB main storage partition.

Table 1. Testing Dataset

Dataset	Data size (MB)	Sequencing technology	Sequencing platform	Read length	Whether single-cell data
SRR4017489	18897.53	WGS	Illumina HiSeq2000	101	No
SRR689233	4719.91	RNA-Seq	Illumina HiSeq 2000	150	No
SRR31700402	4245.38	10xGenomics Single Cell Sequencing	Illumina NovaSeq 6000	150	Yes
R220714041	3935.18	Single Cell Immunobank Sequencing	Illumina NovaSeq 6000	150	Yes
Old_brain_treated	31613.60	Single-cell dynamic RNA sequencing	Illumina NovaSeq 6000	150	Yes
R211112045	2571.65	Single-cell targeted sequencing	Illumina NovaSeq 6000	150	Yes
R220218069	15547.62	scRNA-seq	Illumina NovaSeq 6000	150	Yes

3.1 Establishment of a Comprehensive Test Dataset

The detailed information of the test datasets used in this evaluation is shown in Table 1. These datasets cover various sequencing technologies and experimental conditions, with samples from humans (Homo sapiens) and mice (Mus musculus). The datasets used in this study involve several sequencing methods, including traditional whole genome sequencing (WGS), RNA sequencing (RNA-seq), and five single-cell sequencing methods. Specifically, WGS is a classical technique that uses high-throughput sequencing technology to analyze the complete genome sequence. RNA-seq, as a classic transcriptome sequencing method, generates bulk RNA-seq data. Additionally, the 10x Genomics single-cell sequencing technology, based on the 10x Genomics platform, uses the Single Cell 5' v2 version to achieve high-throughput sequencing after isolating single cells using microdroplets. Furthermore, Singleron Biotechnologies Co., Ltd. provided single-cell sequencing datasets generated by four different sequencing technologies. These four unique single-cell sequencing methods include single-cell immune repertoire sequencing for capturing single-cell-level T-cell receptor (TCR) and B-cell receptor (BCR) rearrangement information; single-cell dynamic RNA sequencing, focusing on capturing RNA dynamic changes; single-cell targeted sequencing, targeting specific RNA or DNA fragments for precise sequencing; and single-cell transcriptome sequencing (scRNA-seq), used to study gene expression heterogeneity and dynamic changes. By selecting a diverse range of sequencing datasets, this study provides a comprehensive evaluation of the proposed compression algorithm's adaptability and performance under different biological contexts and sequencing conditions, thereby validating the algorithm's broad applicability.

3.2 ScBlkCom and Internal Encoding Modules Compression Ratio Performance Analysis

To evaluate the compression performance of ScBlkCom, this study compares it with methods using only Huffman coding and context-adaptive coding. In this research, the compression ratio is defined as the ratio of the original file size to the compressed file size. A higher compression ratio indicates a smaller compressed file size, which reflects better compression performance. For Huffman coding, the traditional static Huffman coding method is directly used as the baseline for comparison. In the context-adaptive coding comparison, the Prediction by Partial Matching (PPMd) algorithm is selected. PPMd [18-19] is a classic context-adaptive coding method, which dynamically predicts the occurrence probabilities of symbols in different contexts and uses arithmetic coding techniques to perform data compression.

When Huffman coding is applied alone to compress the entire file, the resulting compressed file size is generally larger than that obtained using the PPMd algorithm. This difference highlights the advantage of PPMd, which uses context information to dynamically adjust the length of symbol encoding. PPMd shows better compression efficiency in gene sequences with contextual correlations. However, PPMd falls short in handling fragments with different frequencies, such as the molecular identity block, as it does not fully leverage the characteristics of the block, resulting in slightly lower compression efficiency.

This paper compares the compressed file sizes obtained by the ScBlkCom algorithm across several public datasets with the compression results from methods based on single encoding modules (Huffman coding and context-adaptive coding), as shown in Figure 2. To more accurately quantify the performance of the compression tools, this paper defines "compression gain" using the following formula:

$$compression\ gain = (\%) = \begin{cases} (A - B) / B \times 100\%, & \text{when the tool achieves optimal compression ratio on the test dataset} \\ (A - C) / C \times 100\%, & \text{When the tool does not achieve the optimal compression ratio on the test dataset} \end{cases} \quad (1)$$

Here, A, B, and C represent the compression ratios of the current tool, the suboptimal tool, and the optimal tool, respectively. When processing single-cell datasets, the ScBlkCom algorithm achieves an average compression gain of 84.29% compared to a single encoding module.

Table 2. Comparison of ScBlkCom with Other High-throughput Sequencing Data Compression Algorithms.

Dataset	Original size(MB)	Spring	PgRc	ColoRd	ScBlkCom	compression gain (%)	Whether single-cell data
SRR4017489	18897.53	8.54	13.42	**15.28**	14.03	-8.21	No
SRR689233	4719.91	8.25	7.62	**9.12**	8.59	-5.77	No
SRR31700402	4245.38	11.38	10.58	10.73	**12.14**	6.26	Yes
R220714041	3935.18	12.84	20.92	13.99	**21.19**	1.26	Yes
Old_brain_treated	31613.60	10.89	14.86	9.12	**15.11**	1.63	Yes
R211112045	2571.65	7.54	11.01	12.87	**14.01**	8.11	Yes
R220218069	15547.62	12.08	13.44	10.21	**15.80**	14.92	Yes

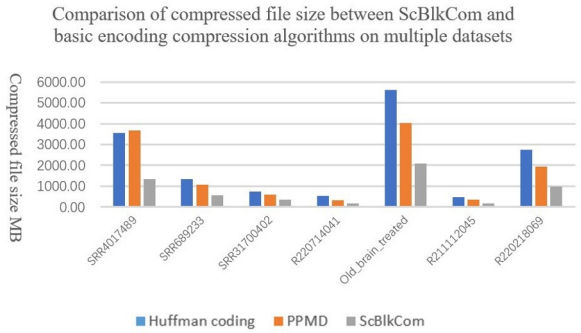

Figure 2. Comparison of Compressed File Sizes between ScBlkCom and Basic Encoding Compression Algorithms.

Compared to single encoding modules, ScBlkCom demonstrates higher compression ratios, smaller compressed file sizes, and better overall compression performance across all datasets. This improvement is attributed to the unique modular design of ScBlkCom. Specifically, for the identity block, the differential encoding technique only records the changes between data, taking advantage of its repetitiveness and regularity. The molecular identity block is compressed using Huffman coding, which assigns variable-length codes based on symbol frequency. The capture fragment block employs context-adaptive encoding, dynamically analyzing the contextual characteristics of base sequences to generate efficient encoding schemes. With this flexible combination strategy, ScBlkCom can precisely select the most suitable encoding method based on the structure of different blocks, thus outperforming single encoding algorithms in terms of compression ratio.

3.3 Comparison with Existing Compression Algorithms

Further comparison of the ScBlkCom algorithm with existing single-cell data compression algorithms was performed. Three mainstream algorithms were selected for evaluation: Spring, PgRc, and ColoRd. The compression ratios of these algorithms were assessed across multiple public datasets and laboratory-generated datasets.

For the ScBlkCom algorithm, if it achieved the optimal compression ratio on a given dataset, the compression gain relative to the second-best algorithm was calculated. If it did not achieve the optimal compression ratio, the compression loss relative to the best algorithm was calculated. The detailed results are shown in Table 2, which provides a comprehensive quantitative comparison of the compression performance of each algorithm across different datasets.

As shown in the results of Table 2, for single-cell transcriptomic datasets (such as R220714041, Old_brain_treated, and R220218069), ScBlkCom demonstrates the best compression performance, indicating that the algorithm is capable of fully exploiting the structural characteristics of single-cell data to achieve higher compression efficiency. In non-single-cell datasets (such as SRR4017489 and SRR689233), ScBlkCom shows compression performance second only to the best algorithm, still exhibiting good compression effectiveness. Based on the processing results of single-cell datasets, ScBlkCom achieves an average compression gain of 6.44% compared to the second-best compression tool. Overall, ScBlkCom outperforms Spring, PgRc, and ColoRd in terms of compression

efficiency on large-scale single-cell datasets, demonstrating a clear competitive advantage.

Compared to non-single-cell datasets, single-cell datasets contain more identity information, such as cell identity blocks and molecular identity blocks, which exhibit high regularity and redundancy. For the cell identity block, ScBlkCom optimizes compression efficiency by using differential encoding combined with reference sequences to identify fixed patterns. For the molecular identity block, the algorithm performs dynamic compression using Huffman coding based on the frequency differences of UMI symbols, effectively reducing storage space. For the capture fragment block, the algorithm leverages the contextual correlation of its base sequences and performs fine-grained compression through context-adaptive encoding. These optimization strategies make the proposed algorithm significantly outperform other compression methods on single-cell datasets.

3.4 Comparison of Compression Time Across Different Algorithms

Further comparison of the running efficiency of the ScBlkCom algorithm with existing single-cell data compression algorithms was conducted by analyzing the compression time of each algorithm across multiple datasets. The compression time for each algorithm on different datasets is visualized through a box plot (see Figure 3).

From the box plot, it can be observed that the compression time varies across different algorithms when processing different datasets. Among them, the ScBlkCom algorithm (yellow box) demonstrates significant stability. The smaller box indicates a lower degree of data dispersion, with compression times being more concentrated across multiple datasets and showing minimal fluctuation.

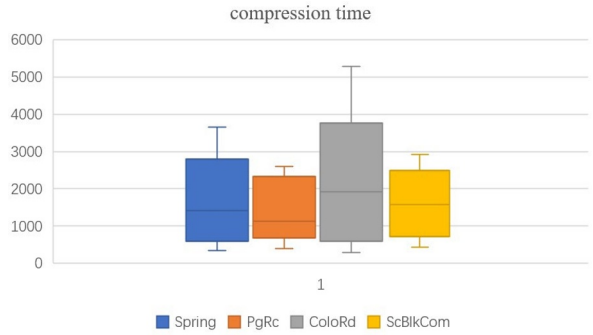

Figure 3. Comparison of Compressed File Sizes between ScBlkCom and Basic Encoding Compression Algorithms.

4 Conclusion

This paper proposes a novel single-cell RNA sequencing data compression algorithm, ScBlkCom, designed to fully exploit the structural characteristics of single-cell data and improve data compression efficiency, especially for the complex and diverse nature of single-cell transcriptomics data. The method integrates various compression techniques such as differential encoding, Huffman coding, and context-adaptive encoding, and tailors compression strategies based on the distinct features of the data. Specifically, for cell identification blocks, the algorithm leverages the high redundancy of fixed patterns, combining differential encoding and reference sequence files to achieve efficient compression; for molecular identifier blocks, Huffman coding is employed, dynamically assigning codes based on the frequency distribution of UMIs; for capture fragment blocks, the algorithm exploits the contextual relationships in the base sequences, applying context-adaptive encoding for fine-grained processing. The proposed algorithm has demonstrated good compression performance across multiple datasets. Experimental results show that although compression time may be longer for certain datasets, the method has clear advantages over existing compression algorithms in terms of compression ratio and processing stability, especially when handling large-scale datasets, where it exhibits higher adaptability.

The ScBlkCom algorithm still has potential for optimization in terms of compression time and efficiency. Future research could explore the introduction of parallel computing techniques to fully leverage the advantages of multi-core processors, enabling parallel data processing and significantly reducing compression time. Additionally, optimizing data matching strategies through more precise pattern recognition and data association could further enhance operational efficiency. With these optimizations, the compression algorithm is expected to have greater potential in genomic data storage.

References

1. J. Jovic, X. Liang, H. Zeng, L. Lin, F. Xu, Y. Luo, *Clin. Transl. Med.*, **12**:e694, (2022).
2. E.L. Van Dijk, H. Auger, Y. Jaszczyszyn, C. Thermes, *Trends Genet*., **30**(9):418-426, (2014).
3. D. Grün, A. van Oudenaarden, *Cell*, **163**(4):799-810, (2015).
4. B. Adamson, T.M. Norman, M. Jost, M.Y. Cho, J.K. Nuñez, Y. Chen, et al., *Cell*, **167**(7):1867-1882.e21, (2016).
5. C. Cheng, W. Chen, H. Jin, X. Chen, *Cells*, **12**(15):1970, (2023).
6. H. Yao, Y. Ji, K. Li, S. Liu, J. He, R. Wang, Biomed. Res. Int., 2019:3108950, (2019).
7. J. Wang, Y. Niu, T. Xu, M. Ma, D. Gao, G. Shi, arXiv preprint arXiv:2304.01031, (2023).
8. S. Chandak, K. Tatwawadi, I. Ochoa, M. Hernaez, T. Weissman, *Bioinformatics*, **35**(15):2674-2676, (2019).
9. T.M. Kowalski, S. Grabowski, *Bioinformatics,* **36**(7):2082-2089, (2020).
10. M. Kokot, A. Gudyś, H. Li, S. Deorowicz, *Nature Methods*, **19**(4):441-444, (2022).
11. T. Nakazato, T. Ohta, H. Bono, PLoS One, **8**(10): e77910, (2013).

12. P. Melsted, V. Ntranos, L. Pachter, *Bioinformatics*, **35**(21):4472-4473, (2019).
13. K. Lebrigand, V. Magnone, P. Barbry, R. Waldmann, Nat. Commun., **11**(1), (2020).
14. A. Lafzi, C. Moutinho, S. Picelli, H. Heyn, Nat. Protoc., **13**(12):2742-2757, (2018).
15. D. Jovic, X. Liang, H. Zeng, L. Lin, F. Xu, Y. Luo, Clin. Transl. Med., **12**(3), (2022).
16. P.V. Kharchenko, Nat. Methods, **18**(7):723-732, (2021).
17. G. Chen, B. Ning, T. Shi, Front. Genet., **10**:317, (2019).
18. Y. Liu, H. Peng, L. Wong, J. Li, *Bioinformatics*, **33**(21):3364-3372, (2017).
19. B.N. Law, *IET Signal Process.*, **13**(6):569-580, (2019).