Contact Matrix Compressor

Yeremia Gunawan Adhisantoso, Jörn Ostermann

Institut für Informationsverarbeitung
Leibniz University Hannover
Germany
{adhisant,office}@tnt.uni-hannover.de

Abstract

The study of three-dimensional folding of chromosomes is important to understand genomics processes. This is done through techniques, such as Hi-C, that analyze the spatial organization of chromosomes in a cell. The data coming from the study is a 2-dimensional quantitative maps with genomic coordinate systems. We present a novel approach called Contact Matrix Compressor(CMC) for the efficient compression of Hi-C data. By exploiting the properties of the data, such as diagonally dominant and symmetrical, CMC achieves a much higher compression. CMC outperforms the existing method Cooler, and also the generic compression methods LZMA as well as BZip2.

Introduction

Genomic processes can be better understood by studying the relationship between chromosome organization and genome activity. This is done through techniques that analyze the spatial organization or interactions of chromosomes in a cell. These techniques, referred to as chromosome conformation, are categorized based on the scope of the experiment. Chromosome Conformation Capture (3C) [1] quantifies the interaction between two specific loci (genomic region), while Hi-C [2] [3] quantifies all interactions between all possible pairs of loci of all chromosomes simultaneously. The scale of a Hi-C experiment allows us the identification of long range interactions. However, it is generating a huge quantity of data at around 30 GB per sample. In this paper, we focus on the data coming from Hi-C experiments. Throughout this paper, we refer to the data coming from a Hi-C study as the contact matrix.

As the name name suggests, the contact matrix is a two-dimensional array. Each row and column of the contact matrix corresponds to a region in a certain chromosome. A region is described by the chromosome where it belongs to, the start position of the region and the end position of a region. The size of a region is constant, meaning that the difference between end and start position is the same for all regions. Each value in the contact matrix represents the number of contacts between a pair of regions or loci. Many of the values are zero, therefore it is more efficient to store the contact matrix in a sparse representation or in a coordinate-value form. An example of this representation can be seen in Table 1. All columns with suffix "1" refer to the row coordinate, and columns with the suffix "2" refer to the column coordinate in the contact matrix. Columns with the prefix "start" or "end" describe the start position and end position of a region, respectively. The second row in Table 1 describes that there are 4 contacts between two different regions. The first region

2022 Data Compression Conference (DCC) | 978-1-6654-7893-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/DCC52660.2022.00048

2375-0359/22/$31.00 ©2022 IEEE
DOI 10.1109/DCC52660.2022.00048

Table 1: An example of how Hi-C data is represented in a coordinate-value form.

chrom1	start1	end1	chrom2	start2	end2	value
chr1	10,000	15,000	chr1	10,000	15,000	5
chr1	10,000	15,000	chr1	15,000	20,000	4
chr1	10,000	15,000	chr1	790,000	795,000	1

Table 2: Compact data representation by splitting table 1 into 2 different tables.

bins			elements		
chrom	start	end	bin1_id	bin2_id	value
chr1	10,000	15,000	1	1	5
chr1	15,000	20,000	1	2	4
chr1	790,000	795,000	1	3	1

is located on chromosome 1 (written as chr1), with chromosomal position between 10,000 and 15,000. The second region is also located on chromosome 1, with chromosomal position between 15,000 and 20,000. The contact matrix is symmetrical because we quantify the interactions between all possibles pairs of regions (see Figure 1). Moreover, Lieberman-Aiden et al. [2] described that the probability of contact decreases as a function of distance for contacts within a chromosome. This probability is almost constant for contacts between two different chromosomes. Therefore, we expect a contact matrix to be a diagonally dominant matrix.

A format called Cooler was proposed by Abdennur et al [4] to store the contact matrix efficiently. The Cooler format is based on the container format HDF5 [5]. HDF5 allows flexible data organizations of collections of multi-dimensional arrays and supports random access. Moreover, it supports data compression for efficient storage, based on ZLIB [6] and SZIP [7]. To increase storage efficiency, the coordinate-value form table is split into two different tables (see Table 2). The first table, called bins, stores all of the possible regions. The regions stored in the bins table are sorted by chromosome, start and end positions. Then, an identifier is assigned implicitly, starting from 1, to each region. The second table, called elements, stores the identifier of the first region bin1_id, the identifier of the second region bin2_id and the number of contacts value.

However, the performance of the ZLIB compressor is worse compared to some generic compressors, such as LZMA [8] and BZip2 [9]. Additionally, Cooler does not exploit the properties of the contact matrix to achieve better compression. In this paper, we present a novel approach for the compression of contact matrices, Contact Matrix Compressor(CMC). To achieve better compression, we exploit some properties of the contact matrix, such as diagonally dominant and also symmetrical. The data is split into smaller matrices such as sub-contact matrices and further down into tiles. These structures allow random access and parallel decompression process.

Methods

Our approach is a combinations of transformations and methods to efficiently store the matrix coming from Hi-C studies [2][3]. For a given set of chromosomes, the contact matrix contains all of the possible interactions between chromosomes. As an example, suppose that there are 3 chromosomes in the data. The contact matrix is structured as depicted in Figure 1. In our proposed approach, the contact matrix is split into sub-contact matrices. Each sub-contact matrix quantifies all interactions between two chromosomes, enabling chromosome-pair level random access. The sub-contact matrix is classified as intra-chromosomal for interaction between a chromosome and itself and it is classified as inter-chromosomal for an interaction between two different chromosomes. Lieberman-Aiden et al. [2] describe that the probability of contact decreases as a function of distance for intra-chromosomal sub-contact matrices and constant for inter-chromosomal sub-contact matrices. Hence, a contact matrix is expected to be a symmetrical and diagonally dominant. These properties are exploited to reduce the overall storage costs with a combination of transformations, such as row-column masking, diagonal transform, and row binarization. Moreover, it is sufficient to store only the sub-contact matrices in the upper triangle part of the contact matrix. For encoding of the transformed data, we use a specialized codec for binary matrices (or bi-level images) called JBIG [10].

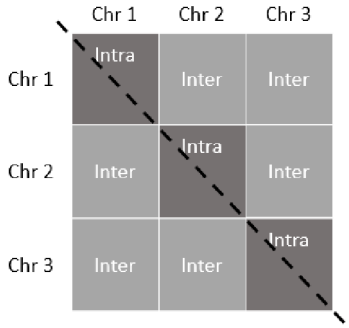


Figure 1: The sub-contact matrices can be classified into intra-chromosomal or inter-chromosomal based on the chromosomes pair.

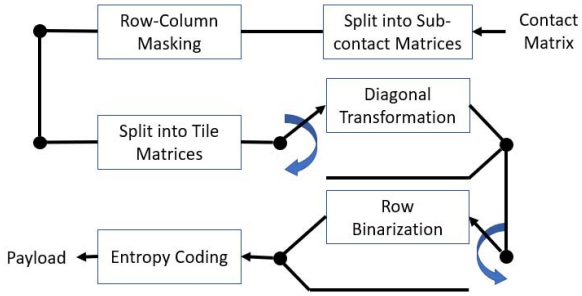


Figure 2: The compression pipeline of the contact matrix compressor comprises transformations and entropy coding. For a faster encoding and decoding speed, some of the transformations can be switched off.

The coding of a sub-contact matrix comprises the following steps, and is depicted in Figure 2: First, by using row-column masking, the sub-contact matrix is reduced by marking the rows and columns that contains only zeros. The flags are stored in the arrays called mask. The marked rows and columns are removed, yielding the reduced sub-contact matrix. Then, each of the sub-contact matrices is split into smaller matrices called tile matrices. By splitting into smaller matrices, the user can balance the trade-off between compression efficiency and random access speed. Optionally, the tile matrix is diagonally transformed. This allows values with similar magnitudes

to be placed in the same rows. After that, optionally, the row binarization decomposes the values of the tile matrix into their binary forms. In combination with the previous transformation, this step significantly increases the compression efficiency. At the end of the pipeline, the tile matrix is entropy coded. In the case that row binarization is enabled, the entropy codec selected is based on JBIG standard. Otherwise, methods such as LZMA or BZip2 can be used as entropy codec.

Row-Column Masking

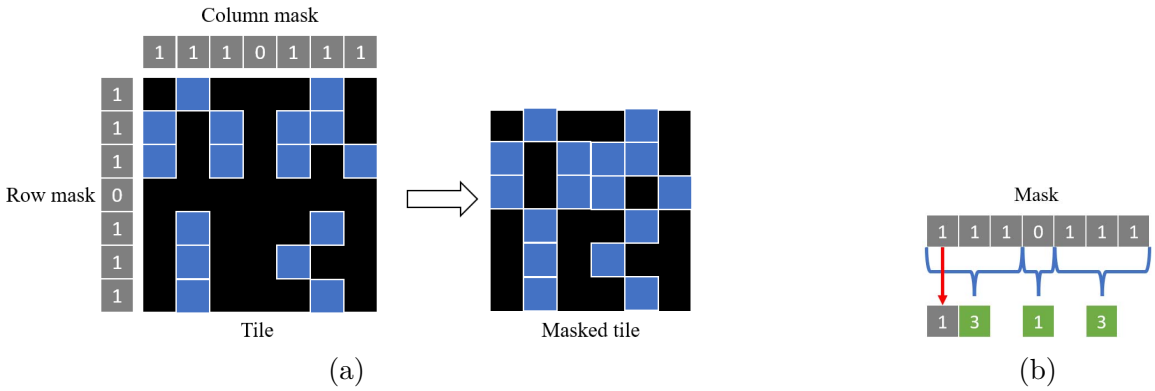


Figure 3: (a) The blue boxes denote values greater than zero and the black boxes denote zeros. Both the row and column mask mark the rows and the columns which contain only zeros. Only the masked tile matrix is transformed and compressed. (b) The binary run-length encoding requires only the first value and all of the run-lengths to represent the original data.

The sub-contact matrix contains values of zero and non-zero. Some of the rows or columns contains only zeros, depicted in the example of Figure 3a. To reduce the redundancy in the data, those rows and columns are marked and then removed from the sub-contact matrix. For rows or columns with only zeros, the flag is set to 0. Otherwise, it is set to 1. This process is called row-column masking. For the decoding process, rows or columns containing zeros are concatenated in the places where the value of the mask is zero.

An array can be transformed by the run-length encoding (RLE), yielding the value/run-length pairs. For a binary array, the value of an array element is either zero or one, simplifying the encoding. If the value of the current run-length is one, the value of the next run-length must be zero and vice versa. Only the first value and all of the run-lengths of the value/run-length pairs are required to reconstruct the original binary array. At the end of the transformation, both masks are encoded using the binary run-length encoding (see Figure 3b).

Splitting into Tiles

To decrease the random access time and to allow parallel decoding, the sub-contact matrix is split into smaller structures called tile matrices as shown in Figure 4. The tile matrix is parameterized by the maximum number of rows and columns $tile\_size$,

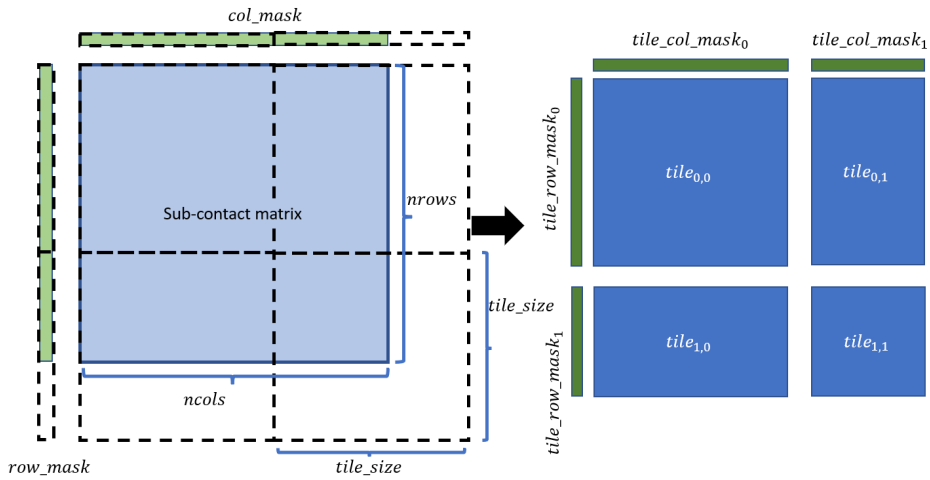


Figure 4: The sub-contact matrix is divided into tile matrices. The tile matrix is parametrized with the number of columns and the number of rows *tile_size*.

allowing a simpler computation of the partition. Typically, tiles are squares. The tile matrices at the right or bottom boundary of a sub-contact matrix can be rectangular. Each tile $tile_{i,j}$ is assigned with index i and j for row coordinate and column coordinate, respectively.

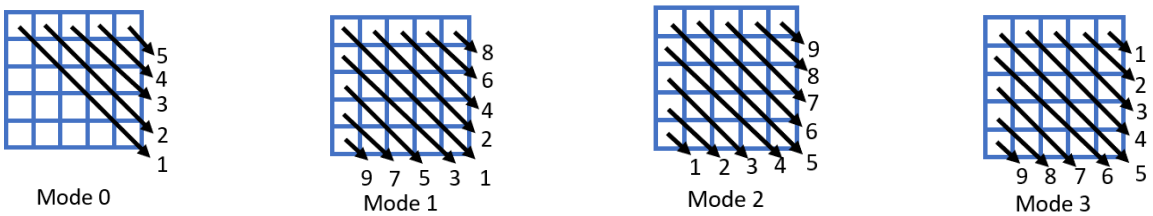
Diagonal Transformation

An integer number can store a value exactly. The number of bits required depends on the range of the allowed values. Based on the finding described by Lieberman-Aiden et al [2], most of the values with high magnitude are located in the main diagonal of the matrix. Normally, an (unsigned) integer value follows a certain bit-length and it applies to the whole matrix. Using a variable-length integer, the bit-length of the values in the upper and lower triangle can be reduced, resulting in a smaller compressed data. To set a specific bit-length for each value is too complex and will generate a huge overhead in terms of storage and processing. Hence, we propose a transformation called diagonal transformation. The core idea of this transformation is to place values with similar magnitude in the same row. First, the values in the diagonal direction are taken and placed in the first row of the new matrix. Then, we take the values from the next diagonal and place it next to the last position of the value placed. When the last column of the new matrix is reached, we continue to the next row of the new matrix. Later, the bit-length of the integer is specified for each row by the next transformation called Row Binarization. For a symmetrical matrix, the values in the lower triangle of the matrix are redundant. After the diagonal transformation, all of the redundant values are placed in the lower half of the matrix (see Figure 5). Hence, those rows can be removed, given the property that the number of rows and columns at the beginning are equal.

As the result of splitting of the sub-contact matrix into tile matrices, depending on the relative position of the tile matrix to the sub-contact matrix, the values with higher magnitude are not located in the main diagonal anymore. The problem is

Figure 5: Using diagonal transformation, all of the redundant values in the lower triangle, depicted in black, are placed to the lower half of the matrix. Then, those rows can be removed.

mitigated by using the other forms of diagonal transformation. In total we propose four diagonal transformations which can be applied to the tile matrix depending on its relative position to the sub-contact matrix.



(a) Mode 0 (b) Mode 1 (c) Mode 2 (d) Mode 3

Figure 6: The proposed diagonal transformations modes. The mode is chosen based on the class of chromosomal pairs and the relative position of the tile matrix.

We apply mode 0 for tile matrices with index $i == j$ of intra-chromosomal contact matrix and mode 2 for tile matrices with index $i < j$. Tiles located in the lower triangle of the intra-chromosomal sub-contact matrix contain only zeros. Therefore, those are not stored. For an inter-chromosomal matrix, the mode of the tiles depend on the relative position of the sub-contact matrix to the contact matrix. All tiles of the inter-chromosomal sub-contact matrix located in the upper triangle of the contact matrix are assigned to mode 2 and mode 3 for the lower triangle part.

Row Binarization

To assign variable bit-length integer number, each row is decomposed into its binary form, yielding binary rows. The number of binary rows (or the bit-length) depends on the maximum value of each row and can be computed as follows:

$$q_i = \left\lceil \log_2 \left(\max_{\forall i,j} \{a_{i,j}\} + 1 \right) \right\rceil . \tag{1}$$

where q_i is the number of binary rows decomposed from a row i. $a_{i,j}$ is the value of the matrix at row i and column j. The value q_i is not stored in the compressed data. For each binary row that corresponds to a certain bit position of the original row, a flag called sentinel flag is added at the beginning. This flag signals that the current binary row represents the last bit of the original row. Therefore, the number of binary rows equals to the bit-length. The binary rows are concatenated in row direction, starting from the binary row that represents the first bit with the next bit

and so on. This process forms a binary matrix from binary rows As an example, consider the following matrix $\mathcal{A}$:

$$\mathcal{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}. \tag{2}$$

The bit-length of each row q_i is computed based on the maximum value, where the maximum values are 3 and 6 for the first and the second row, respectively. Based on the maximum values, the first row requires 2 bits (q_1) and the second row requires 3 bits (q_2). The rows are decomposed into binary rows, depicted with an arrow in equation 3, and a sentinel flag is added to the beginning of each row:

$$q_1 = 2, q_2 = 3, \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{3}$$

The values depicted in red are the sentinel flags. At the end of the process, all of the binary matrices from all rows are concatenated in row direction, forming a single binary matrix.

Entropy Coding

In the case where the row binary split transformation is disabled, a generic codec such as LZMA [8] or BZip2 [9] can be used to compress the data. Otherwise, the input of this process is a binary matrix and equivalent to a bi-level image. Thus, we code this binary image using an encoder compliant to the JBIG standard (ISO/IEC 11544 [10]), that specifies the lossless compression of bi-level images. It takes advantage of spatial correlation of bi-level pixels or contexts (in this case values of the binary matrices). The length of the context varies between in total of 10 to 12 neighboring values in both row and column directions, depending on context mode. We decided to use JBIG and not JBIG2 (ISO/IEC 14492 [11]) as both technologies offer comparable results for lossless compression and we do not need additional features offered by the JBIG2 standard such as lossy compression.

Results

For the evaluation, we used the test data compiled by Rao et. al [12] and described in Table 3. The experiment data are stored in Cooler format and then the content are extracted. We refer the extracted content [1] of Cooler as the raw data and the file size is based on the raw data. Thus, we does not include the encoding and decoding time of Cooler.

All experiments were carried out on a Linux machine with 8-cores Core i9-9900K at 5.0 GHz, 64 GB of RAM and solid state drive. Our software is written in Python and the code for transformations are compiled using Cython. For the entropy coding, we use the software Jbig-kit [13]. We activated all of the optional transformations to maximize the compression performance of CMC. The tile size was set to 10,000. The

[1] `https://cooler.readthedocs.io/en/latest/schema.html`

Table 3: Test items used for the evaluation. The raw data are extracted from the Cooler format.

Test Item	File name	File size [MB]
1.5.1	GSE63525_CH12-LX_combined	22,030
1.5.3	GSE63525_GM12878_dilution_combined	12,788
1.5.4	GSE63525_GM12878_diploid_maternal	15,603
1.5.5	GSE63525_GM12878_insitu_DpnII_combined	12,224
1.5.9	GSE63525_HMEC_combined	16,493
1.5.10	GSE63525_HUVEC_combined	22,288
1.5.11	GSE63525_IMR90_combined	32,386
1.5.12	GSE63525_K562_combined	30,863
1.5.13	GSE63525_KBM7_combined	37,956
1.5.14	GSE63525_NHEK_combined_30	20,021

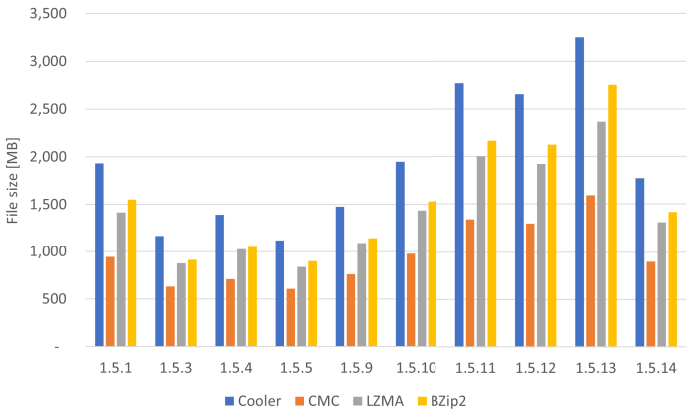


Figure 7: File size of each test item compressed using Cooler, CMC, LZMA and BZip2.

diagonal transformation mode was selected as described in Section Diagonal Transformation and the row binarization was turned on whenever the diagonal transformation was applied. For both LZMA and BZip2, we compressed all of the information extracted from a Cooler data into a single file. Hence, random access functionality is not supported for both LZMA and BZip2.

As shown in Figure 7, the CMC outperforms all other methods in terms of compression, and that both LZMA and BZip2 perform better than Cooler. To have a better view on the relative performance of CMC compared to BZip2 and LZMA, we computed the compression ratio of CMC w.r.t. the other methods. It is computed by dividing the compressed file size of the other approaches by the file size of the CMC. The relative compression ratios are shown in Figure 9. Compared to CMC, Cooler is about half as efficient and CMC is at least 1.4 times better than LZMA.

In terms of encoding time (see Figure 8), CMC is comparable to LZMA and faster than BZip2. However, CMC's decoding speed is about ten times as slow as LZMA and slower compared to BZip2, as shown in Figure 10. CMC has a symmetrical complexity.

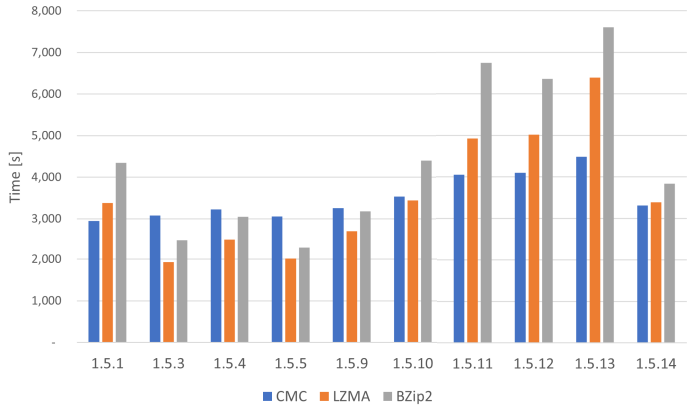


Figure 8: The encoding time of CMC, LZMA and BZip2. The encoding performance of CMC is comparable to LZMA2 while BZip2 is the slowest.

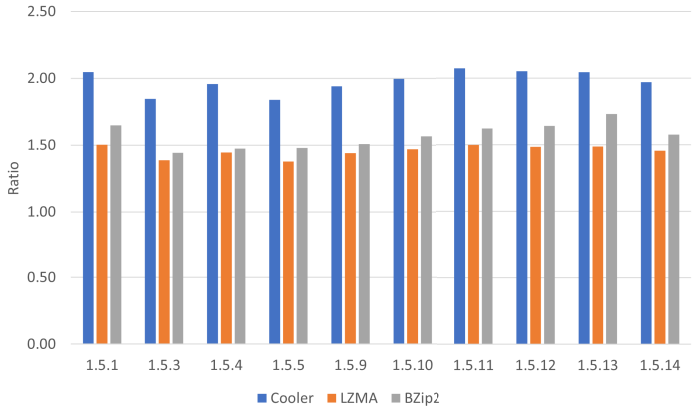


Figure 9: Relative compression ratio of CMC compared to the other approaches.

Therefore, the decoding and the encoding speed is about the same. Compared to CMC, LZMA has an asymmetrical complexity. The encoding process requires the searching of a matching pattern, resulting in a higher encoding complexity. Finally, the decoding speed can be scaled up by utilizing parallel processing. The tiles and sub-contact matrices are independent from each other. This simplifies the decoding process and allows for possible parallel decoding process.

Summary

We have presented Contact Matrix Compressor(CMC), a compressor specialized for the coding of contact matrices coming from Hi-C experiments. It outperforms Cooler, and generic compressors LZMA as well as BZip2. CMC is about 2 times and 1.4 times as efficient as Cooler and LZMA, respectively, while offering random access capability at the chromosome-pairs and tiles levels. Better compression is achieved by exploiting the properties of the contact matrix, such as symmetrical and diagonally dominant. To exploit such properties, the diagonal transformation and the row binarization are utilized. By splitting into smaller structures such as sub-contact matrices and tiles, CMC enables random access at a more granular level.

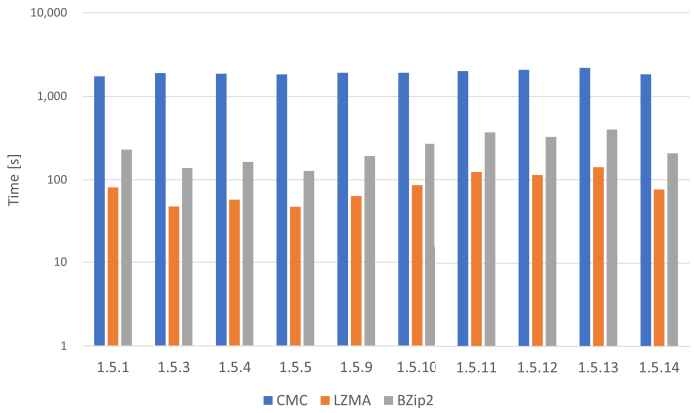


Figure 10: The decoding time of CMC, LZMA and BZip2.

References

[1] Job Dekker, Karsten Rippe, Martijn Dekker, and Nancy Kleckner, "Capturing chromosome conformation," *Science*, vol. 295, no. 5558, pp. 1306–1311, 2002.

[2] Erez Lieberman-Aiden, Nynke L Van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R Lajoie, Peter J Sabo, Michael O Dorschner, et al., "Comprehensive mapping of long-range interactions reveals folding principles of the human genome," *Science*, vol. 326, no. 5950, pp. 289–293, 2009.

[3] Nynke L Van Berkum, Erez Lieberman-Aiden, Louise Williams, Maxim Imakaev, Andreas Gnirke, Leonid A Mirny, Job Dekker, and Eric S Lander, "Hi-C: a method to study the three-dimensional architecture of genomes.," *Journal of Visualized Experiments*, , no. 39, pp. e1869, 2010.

[4] Nezar Abdennur and Leonid A Mirny, "Cooler: scalable storage for Hi-C data and other genomically labeled arrays," *Bioinformatics*, vol. 36, no. 1, pp. 311–316, 2020.

[5] Quincey Koziol, Dana Robinson, et al., "HDF5," Tech. Rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2018.

[6] Peter Deutsch and Jean-Loup Gailly, "Zlib compressed data format specification version 3.3," Tech. Rep., RFC 1950, May, 1996.

[7] Wenhua Yu, Ruifang Li, Bin Gui, and Yongfeng Shang, "sZIP, an alternative splice variant of ZIP, antagonizes transcription repression and growth inhibition by ZIP," *Journal of Biological Chemistry*, vol. 285, no. 19, pp. 14301–14307, 2010.

[8] Igor Pavlov, "LZMA SDK (software development kit)," 2007.

[9] Julian Seward, "Bzip2 and libbzip2," *available at http://www. bzip. org*, 1996.

[10] "Information technology — Coded representation of picture and audio information — Progressive bi-level image compression," Standard, International Organization for Standardization, Geneva, CH, 1993.

[11] "Information technology — Lossy/lossless coding of bi-level images," Standard, International Organization for Standardization, Geneva, CH, 2001.

[12] Suhas SP Rao, Miriam H Huntley, Neva C Durand, Elena K Stamenova, Ivan D Bochkov, James T Robinson, Adrian L Sanborn, Ido Machol, Arina D Omer, Eric S Lander, et al., "A 3d map of the human genome at kilobase resolution reveals principles of chromatin looping," *Cell*, vol. 159, no. 7, pp. 1665–1680, 2014.

[13] Markus Kuhn, "JBIG-KIT," *http://www. cl. cam. ac. uk/˜ mgk25/jbigkit*, 1995.