IMAGE COMPRESSION-BASED APPROACH TO MEASURING THE SIMILARITY OF PROTEIN STRUCTURES

MORIHIRO HAYASHIDA AND TATSUYA AKUTSU

Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611-0011, Japan
E-mail: {morihiro, takutsu}@kuicr.kyoto-u.ac.jp

This paper proposes series of methods for measuring the similarity of protein structures. In the proposed methods, an original protein structure is transformed into a distance matrix, which is regarded as a two-dimensional image. Then, the similarity of two protein structures is measured by a kind of compression ratio of the concatenated image. We employed several image compression algorithms: JPEG, GIF, PNG, IFS, and SPC, and audio compression algorithms: MP3 and FLAC. We applied the proposed method to clustering of protein structures. The results of computational experiments suggest that SPC has the best performance.

Keywords: image compression; audio compression; protein structure similarity

1. Introduction

Analysis of protein structures is an important topic in bioinformatics and computational biology. In particular, classification of protein structures is important and thus many studies have been done and several databases have been developed such as SCOP[1] and CATH.[2] Classification of protein structures is usually done based on some measure of the similarity between protein structures.

However, an agreement on which is the best similarity measure is not yet obtained and a variety of structure comparison methods have been proposed. Most of existing methods are based on *protein structure alignment*. Various methodologies have been employed for protein structure alignment, which include double dynamic programming,[3] iterative improvement,[4] combinatorial extension,[5] comparisons of distance matrices,[6] use of partial order graphs,[7] and contact map overlap.[8] In most of structure alignment methods, some scoring function is defined for measuring the quality of the obtained alignment. Then, the structure alignment problem is defined as finding a structure alignment with the optimal or near optimal score. However, score functions are defined in more or less ad-hoc manners and there is no consensus or theoretical justification. Furthermore, many of existing structure alignment methods are not very efficient.

Krasnogor and Pelta recently proposed a novel approach to measuring the similarity of protein structures.[9] Their method is similar to the contact map overlap

(CMO) approach.[8] In their method, each protein structure is transformed into a 0-1 matrix, which is further regarded as a 0-1 sequence. Then, two protein structures are compared based on the compression ratio of the sequence obtained by concatenating two 0-1 sequences. Their method is quite simple to implement and very fast. They demonstrated the usefulness of the method by means of application to clustering of protein structures. It is worthy to mention that several works have been done on measuring the similarity of biological sequences based on data compression approach.[10,11]

Though the approach by Krasnogor and Pelta is novel and useful, the distances between residues are truncated into 0 or 1. As a result, the similarity measure depends on the threshold, which should be determined by try and error. The same drawback applies to CMO,[8] In this paper, we try to overcome this drawback using a very simple idea. We employ *image compression* in place of sequence compression. Each distance matrix (not 0-1 matrix) is directly compressed by using an image compression algorithm. In this paper, we examine the following image compression algorithms: JPEG, GIF, PNG, IFS, and SPC, and audio compression algorithms: MP3 and FLAC. We apply the proposed methods to clustering of protein structures as in.[9]

The organization of the paper is as follows. We begin with a brief review the method by Krasnogor and Pelta. Next, we present our proposed methods. Then, we describe details and results of computational experiments. Finally, we conclude with future work.

2. Structure Comparison Using Sequence Compression

Krasnogor and Pelta employed sequence compression to measure the similarity of two proteins. Their method is based on the universal similarity metric (USM), which was originally proposed by Li *et al.*[11] USM is based on Kolmogorov complexity. The Kolmogorov complexity $K(o)$ of an object o is defined to be the length of the shortest program P for a Universal Turing Machine U that is required to output o.[12] That is, $K(o)$ is defined by

$$K(o) = \min\{|P| \mid P \text{ is a program such that } U(P) = o\}.$$

$K(o)$ is considered to be a measure of the amount of information contained in o. Besides, the conditional Kolmogorov complexity of o_1 given o_2 is defined by

$$K(o_1|o_2) = \min\{|P| \mid P \text{ is a program such that } U(P, o_2) = o_1\},$$

where $U(P, o_2) = o_1$ means that program P outputs o_1 when o_2 is given. Based on these, information distance between two objects o_1 and o_2 can be defined as

$$InfDist(o_1, o_2) = \max(\ K(o_1|o_2),\ K(o_2|o_1)\ ).$$

Since this distance is not normalized, USM was proposed as a normalized measure:[11]

$$USM(o_1, o_2) = \frac{\max(\ K(o_1|o_2^*),\ K(o_2|o_1^*)\ )}{\max(\ K(o_1),\ K(o_2)\ )},$$

where o_i^* denotes the shortest program for o_i.

It is well-known that Kolmogorov complexity of a given object is not computable. Thus, Krasnogor and Pelta employed a sequence compression algorithm ('compress' command in UNIX). Let $C(s)$ be the size of the compressed sequence of s. They used $C(o_1)$ and $C(o_1 \cdot o_2) - C(o_2)$ in place of $K(o_1)$ and $K(o_1|o_2^*)$ respectively, where $o_1 \cdot o_2$ denotes the concatenation of two sequences o_1 and o_2. It should be noted that o_k is a 0-1 sequence obtained from a contact map M_k of protein P_k, where $M_k[i,j] = 1$ if the distance between ith residue and jth residue is less than threshold Θ, otherwise $M_k[i,j] = 0$. o_k is obtained by simple raster scanning of matrix M_k.

3. Similarity Metric Based on Image and Audio Compression

We define a contact map M_k of protein P_k as the distance matrix between residues as $M_k[i,j] = \sqrt{(r_k[i] - r_k[j])^2}$, where $r_k[i]$ denotes the three-dimensional coordinate of ith C-alpha atom of P_k.

We transform the contact map M_k to a raw image format, PPM (Portable Pixel Map), and a raw audio format, WAV. PPM can represent $(2^8)^3 = 16777216$ colors using 3 bytes memory for a pixel, where each byte is used for red, green, and blue, respectively, zero means black color, and $16777215 (= (2^8)^3 - 1)$ means white color. We transform $M_k[i,j]$ to the corresponding pixel with the color of the integer part of $cM_k[i,j]$, where c is a constant, and we set $c = 4 \cdot (2^8)^2 = 262144$ in the experiment section. If $cM_k[i,j]$ is greater than or equal to $(2^8)^3 = 16777216$, we set the color white. Fig. 1c and 1d show examples of such images for proteins 1ash and 1aa9, respectively. In order to concatenate two images horizontally, the two images must have the same height. Therefore, we fill the smaller image with black color to the height of the other (See Fig. 1c).

On the other hand, WAV can represent $(2^8)^2 = 65536$ sounds using an integer of $[-32768, 32767]$. We transform $M_k[i,j]$ to the sound with the integer part, $A_k[i,j]$, of $c'M_k[i,j] - 32768$, where c' is a constant, and we set $c' = 500$ in the experiment section. If the integer value of a sound is greater than 32767, we set it 32767. We concatenate two audios as follows: $o_1 \cdot o_2 = S(A_1,2) \cdot S(A_2,2) \cdot S(A_1,3) \cdot S(A_2,3) \cdots$, where $S(A_k,b) = A_k[1, 1+b] \cdots A_k[n_k - b, n_k]$, and n_k denotes the number of residues of protein P_k. That is, we concatenated diagonals of A_1 and A_2.

Krasnogor and Pelta approximated $K(o_1|o_2^*)$ of USM by $C(o_1 \cdot o_2) - C(o_2)$. However, $C(o_1 \cdot o_2)$ is not always equal to $C(o_2 \cdot o_1)$. Therefore, we approximate $K(o_1|o_2^*)$ by $\max(C(o_1 \cdot o_2) - C(o_2), C(o_2 \cdot o_1) - C(o_2))$. Then, the approximated USM for image and audio compression, AUSM, is given as follows:

$$AUSM(o_1, o_2) = \frac{\max(\ C(o_1 \cdot o_2),\ C(o_2 \cdot o_1)\ ) - \min(\ C(o_1),\ C(o_2)\ )}{\max(\ C(o_1),\ C(o_2)\ )}.$$

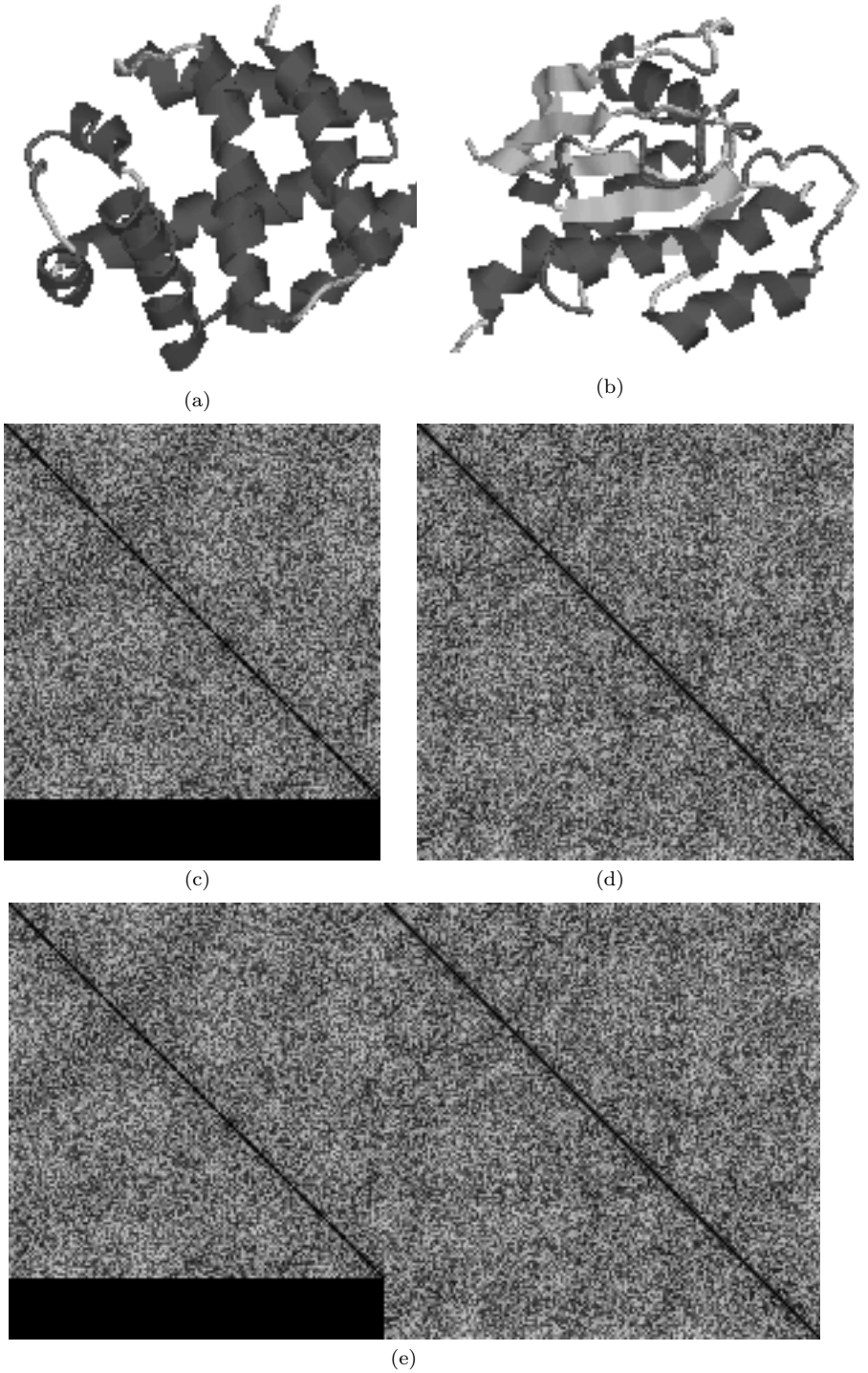


Fig. 1. An example of compressed images. (a) 1ash. (b) 1aa9. (c) The image of 1ash, which is filled with black color to the height of 1aa9. (d) The image of 1aa9. (e) The concatenated image of 1ash and 1aa9.

4. Computational Experiments

4.1. *Image Compression Algorithms*

We employed the following image compression algorithms. In this subsection, we briefly review their algorithms.

4.1.1. *JPEG*

JPEG is usually lossy compression. An image is split into blocks of eight by eight pixels. A two-dimensional forward discrete cosine transform (DCT) is done for every blocks. After quantization, the image is compressed using Huffman coding.[13]

4.1.2. *GIF*

GIF is based on the Lempel-Ziv algorithm,[14] which is a dictionary coder. It reads an input sequence, constructs a dictionary dynamically, and replaces the sequence with words of the dictionary.

4.1.3. *PNG*

PNG is also based on the Lempel-Ziv algorithm,[14] and uses Huffman coding,[13] where PNG has been developed to replace GIF. The compression rate of PNG is often higher than that of GIF.

4.1.4. *IFS*

IFS stands for Iterated Function Systems, is a quadtree-based fractal image coder/decoder, and was implemented by Polvere.[15] The software called Mars is available from `http://inls.ucsd.edu/~fisher/Fractals/Mars-1.0.tar.gz`. Note that the software can accept only grayscale images using one byte memory for a pixel as raw image files.

4.1.5. *SPC*

SPC is a lossless image compression, and was developed by Said and Pearlman.[16] It uses a simple pyramid multiresolution scheme enhanced with predictive coding, and contains S (Sequential) transform and P (Prediction) transform.

In S transform, a sequence $c[i]$ is transformed to two sequences $l[i]$ and $h[i]$ with half the length so that the average variance of the two sequences is smaller than the variance of the original sequence if the correlation coefficient of $c[2i]$ and $c[2i+1]$ is larger than $\frac{1}{3}$. Since $h[i]$ often has small variance in image compression, we can reduce the errors of linearly predicted values for $h[i]$ using $l[i]$s and $h[i+1]$.

These transformations are done sequentially to the rows and columns of an image. Finally, these sequences are encoded using Huffman or arithmetic coding.

The software is available from `http://www.cipr.rpi.edu/research/SPIHT/EW_Code/lossless.tar.gz`. Note that the software can accept only grayscale images as raw image files.

4.2. *Audio Compression Algorithms*

In addition, we employed the following audio compression algorithms.

4.2.1. *MP3*

MP3 uses MDCT (Modified Discrete Cosine Transform), and is a lossy compression format.

4.2.2. *FLAC*

FLAC (Free Lossless Audio Codec) is a lossless compression format (`http://flac.sourceforge.net/`). FLAC is similar to SPC, and uses simple polynomial fitting and general linear predictive coding. Special Huffman coding called Rice coding can be applied for residual errors. In order to obtain the best compression, we used "–best" option.

4.3. *Data*

We used a dataset in Krasnogor and Pelta,[9] which was first used in Chew and Kedem.[17] The dataset contains proteins identified by their PDB codes. We obtained their PDB-style files (version 1.71) with coordinates from the Astral database[18] as follows (See Table 1): 16 globins (1ash, 1eca, 1hlb, 1hlm, 1ithA, 1mba, 1myt, 2hbg, 2lhb, 3sdhA, 1babA, 1babB, 1flp, 1lh2, 2vhbA, 5mbn), 2 all alpha proteins except globins (1cnpB, 1jhgA), 7 all beta proteins (1qa9B, 1cd8, 1cdb, 1ci5A, 1hnf, 1neu, 1qfoA), 4 TIM barrels (4enl, 2mnr, 1chrA1, 6xia), and 6 alpha and beta proteins except TIM barrels (1ct9A1, 1aa9, 1gnp, 1qraA, 5p21, 6q21A).

4.4. *Experiments*

For each pair of proteins included in the Chew-Kedem dataset, we generated two raw image files, o_1 and o_2, and the two concatenated image files, $o_1 \cdot o_2$ and $o_2 \cdot o_1$, from the two three-dimensional structures, and also raw audio files. We applied the above compression algorithms, JPEG, GIF, PNG, IFS, SPC, MP3, and FLAC, respectively, to the raw files, and calculated $AUSM(o_1, o_2)$. We obtained hierarchical clustering results using the nearest neighbor (single linkage) method.

5. Results

Fig. 2 and 3 show the clustering results on the Chew-Kedem dataset for the compression algorithms, JPEG, GIF, PNG, IFS, SPC, MP3, and FLAC. We can see

Table 1. The Chew-Kedem dataset.

Class of SCOP	Family	Proteins
All alpha	a.1.1.2[a]	1ash, 1eca, 1hlb, 1hlm, 1ithA, 1mba, 1myt, 2hbg, 2lhb, 3sdhA, 1babA, 1babB, 1flp, 1lh2, 2vhbA, 5mbn
	a.4.12.1	1jhgA
	a.39.1.2	1cnpB
All beta	b.1.1.1	1qa9B, 1cd8, 1cdb, 1ci5A1, 1hnf, 1neu, 1qfoA
Alpha and beta	c.1.11.1[b]	4enl
	c.1.11.2[b]	2mnr, 1chrA1
	c.1.15.3[b]	6xia
	c.26.2.1	1ct9A1
	c.37.1.8[c]	1aa9, 1gnp, 1qraA, 5p21, 6q21A

Note: [a] Globins, [b] TIM beta/alpha-barrel, and [c] G proteins.

from these figures that SPC classified the dataset best. Although two all alpha proteins (1jhg and 1cnp) and three all beta proteins (1qa9, 1cdb and 1ci5) were mixed in the clustering result by Krasnogor and Pelta,[9] and an all beta protein (1hnf) was classified in globins, our SPC result classified all beta proteins correctly. The two all alpha proteins (1jhg and 1cnp) and all beta proteins were mixed in PNG similarly to the result by Krasnogor and Pelta. However, 1hnf was correctly classified in all beta proteins in PNG. Although GIF and PNG use similar compression algorithms, the result of PNG was better.

6. Conclusions

We proposed image and audio compression-based approach to measuring the similarity of protein structures, and applied them to the Chew-Kedem dataset. The clustering result by SPC image compression algorithm was the best in several image and audio compression algorithms, and was comparable to or better than that by Krasnogor and Pelta. Almost all image compression algorithms have been developed based on the property that neighbor pixels often have similar colors in images. However, similar substructures located at distant locations should also be compressed. Therefore, it is considered that the best performance was obtained with the SPC algorithm. It is expected that better similarity measure is obtained by improving the SPC algorithm. In addition, values handled by image compression algorithms are restricted to integers of a few bytes. In this paper, we transformed distances between residues of a protein to integers. In future work, we would like to develop compression algorithms for distances with real values.

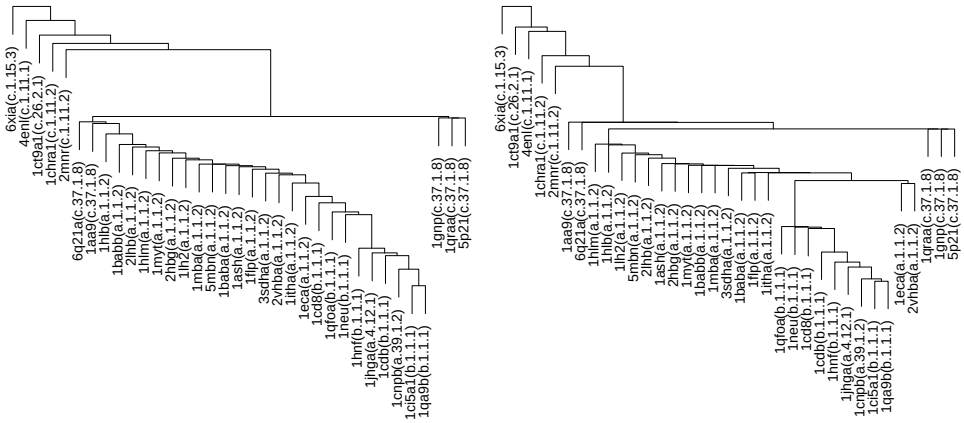
Acknowledgments

This work was partially supported by Grants-in-Aid "Systems Genomics" and #19650053 from MEXT, Japan.

References

1. A. Andreeva, D. Howorth, S. E. Brenner, T. J. P. Hubbard, C. Chothia and A. G. Murzin. SCOP database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Research*, 32:D226–D229, 2004.
2. F. M. Pearl, C. F. Bennett, J. E. Bray, A. P. Harrison, N. Martin, A. Shepherd, I. Sillitoe, J. Thornton and C. A. Orengo. The CATH database: an extended protein family resource for structural and functional genomics. *Nucleic Acids Research*, 31:452–455, 2003.
3. W. R. Taylor and C. A. Orengo. Protein structure alignment. *Journal of Molecular Biology*, 208:1–22, 1989.
4. T. Akutsu. Protein structure alignment using dynamic programming and iterative improvement. *IEICE Trans. on Information and Systems*, E79-D:1629–1636, 1996.
5. I. N. Shindyalov and P. E. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 11:739–747, 1998.
6. L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal Molecular Biology*, 233:123–138, 1993.
7. Y. Ye and A. Godzik. Multiple flexible structure alignment using partial order graphs. *Bioinformatics*, 21:2362–2369, 2005.
8. A. Caprara, R. Carr, S. Istrail, G. Lancia and B. Walenz. 1001 Optimal PDB structure alignments: Integer programming methods for finding the maximum contact map overlap. *Journal of Computational Biology*, 11:27–52, 2004.
9. N. Krasnogor and D. A. Pelta. Measuring the similarity of protein structures by means of the universal similarity metric. *Bioinformatics*, 20:1015-1021, 2004.
10. A. Kocsor, A. Kertész-Farkas, L. Kaján and S. Pongor. Application of compression-based distance measures to protein sequence classification: a methodological study. *Bioinformatics*, 22:407–412, 2006.
11. M. Li, J. H. Badger, X. Chen, S. Kwong, P. E. Kearney and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17:149–154, 2001.
12. M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, 1997.
13. D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40:1098–1101, 1952.
14. J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, IT-24:530–536, 1978.
15. M. Polvere and M. Nappi. A feature vector technique for fast fractal image coding. *Technical report University of Salerno*, 1998.
16. A. Said and W. A. Pearlman. Reversible image compression via multiresolution representation and predictive coding. *Proc. SPIE*, 2094:664–674, 1993.
17. L. P. Chew and K. Kedem. Finding consensus shape for a protein family. *Algorithmica*, 38:115–129, 2003.
18. J. M. Chandonia, G. Hon, N. S. Walker, L. Lo Conte, P. Koehl, M. Levitt and S. E. Brenner. The ASTRAL compendium in 2004. *Nucleic Acids Research*, 32:D189–D192, 2004.

Fig. 2. The clustering results on the Chew-Kedem dataset for image compressions, (a) JPEG, (b) GIF, (c) PNG, (d) IFS, and (e) SPC.



(f) MP3 (g) FLAC

Fig. 3. The clustering results on the Chew-Kedem dataset for audio compressions, (f) MP3 and (g) FLAC.