Cancer Informatics

Supplementary Issue: Computational Advances in Cancer Informatics (A)

Streamlined Genome Sequence Compression using Distributed Source Coding

Shuang Wang[1], Xiaoqian Jiang[1], Feng Chen[2], Lijuan Cui[2] and Samuel Cheng[2]

[1]Division of Biomedical Informatics, University of California, San Diego, La Jolla, CA, USA. [2]School of Electrical and Computer Engineering, University of Oklahoma, Tulsa, OK, USA.

ABSTRACT: We aim at developing a streamlined genome sequence compression algorithm to support alternative miniaturized sequencing devices, which have limited communication, storage, and computation power. Existing techniques that require heavy client (encoder side) cannot be applied. To tackle this challenge, we carefully examined distributed source coding theory and developed a customized reference-based genome compression protocol to meet the low-complexity need at the client side. Based on the variation between source and reference, our protocol will pick adaptively either syndrome coding or hash coding to compress subsequences of changing code length. Our experimental results showed promising performance of the proposed method when compared with the state-of-the-art algorithm (GRS).

KEYWORDS: genome compression, distributed source coding, graphical model

SUPPLEMENT: Computational Advances in Cancer Informatics (A)

CITATION: Wang et al. Streamlined Genome Sequence Compression using Distributed Source Coding. *Cancer Informatics* 2014:13(S1) 123–131 doi: 10.4137/CIN.S13879.

RECEIVED: June 2, 2014. **RESUBMITTED:** October 5, 2014. **ACCEPTED FOR PUBLICATION:** October 8, 2014.

ACADEMIC EDITOR: JT Efird, Editor in Chief

TYPE: Methodology

FUNDING: This work was funded in part by the NHGRI (K99HG008175), NLM (R00LM011392), iDASH (NIH U54HL108460), and NSF (CCF 1117886). The authors confirm that the funder had no influence over the study design, content of the article, or selection of this journal.

COMPETING INTERESTS: Authors disclose no potential conflicts of interest.

CORRESPONDENCE: shw070@ucsd.edu

Introduction

The vision of miniaturized sequencing devices is turning into reality with the emergence of MinION by Oxford Nanopore.[1] Such devices are promising in a variety of potential applications, ranging from studying of wildlife and clinical capture of sequenced genes to food inspection for identifying pathogens. However, such portable devices are commonly subject to the constraints in processing capabilities, power budget, and storage and communication limitations. With these constraints, the traditional view of genome compression architecture as simple decoder and complex encoder needs to be changed. It is urgent to develop novel techniques to satisfy the emerging reality challenges. Data compression methods (for reducing the storage space with significantly lower computational complexity and memory requirements) become crucial for the efficient management of genomic data in portable devices.

In both situations (with or without reference sequences), traditional genome compression is computationally expensive at the encoder. The complexity is dominated by matching (approximately) repeated patterns of nucleotides – namely adenine (A), cytosine (C), guanine (G), and thymine (T) – between or within the DNA sequences. These patterns are also accompanied by insertions, deletions, and substitutions of single nucleotides.

To date, a number of specialized DNA sequence compression algorithms have been proposed. Following Ziv and Lempel's ideas,[2] Grumbach and Tahi[3] proposed the first DNA sequence compressor, Biocompress, to compress the exact repeating patterns with a specially designed Fibonacci coder. The algorithm was then improved in Biocompress-2[4] by introducing a Markov model for encoding the nonrepeated regions. Chen et al.[5,6] extended the earlier approach to cover approximated repeats by further exploiting the nature of DNA sequences. Meanwhile, the work in Ref. 7 introduced a combined CTW + LZ algorithm for searching approximate repeats and palindrome using hash and dynamic programming. Behzadi and Fessant[8] proposed a dynamic programing approach for the optimal selection of approximate repeats with promising compression efficiency being witnessed. However, such methods are heuristic as the underlying statistics of the sequence patterns are generally ignored. The authors in Refs. 9–11 proposed to combine the matching and substitution of approximate repeats and a specific normalized maximum likelihood model, obtaining a much higher compression ratio. Subsequently, statistical modeling for predicting the generation of symbols and arithmetic coding for such symbols in DNA sequences were proposed for more efficient compression. Cao et al.[12] proposed to estimate the probability distribution of symbols with a panel of experts to tackle the approximate repeat problem. Alternatively, finite context models are proposed to capture different aspects of statistical information along the sequence;[13,14] such reference-free methods are plagued by their low compression rates (not $>6{:}1$) and prohibitive computational consumption for large DNA sets.

Recognizing reference-free architectures do not fully utilize information, a series of algorithms are proposed to compress sequences by matching approximate repeats with a reference sequence. The RLZ algorithm proposed by Kuruppu et al.[15] performed relative Lempel–Ziv compression of DNA sequences with the collection of related sequences. Wang and Zhang[16] proposed the GRS compressor, which is able to compress a sequence using a reference without any additional information. Applying the copy model into the matching of exact repeats in reference sequences, GReEn[17] achieved even larger gains when compared to that in Refs. 15 and 16. Recently, reference-based algorithms[18,19] achieved highly efficient compression performance for the fastq data format, by matching and comparing repeated subsequences in the reference sequences. Although the reference-based architectures can achieve hundreds of folds compression, the requirement of reference sequences makes it impractical for miniaturized devices, which have very limited storage space and communication bandwidth.

In this study, we propose a novel and pioneering architecture for the genome compression application in miniaturized devices with limited processing capabilities, power budget, storage space, and communication bandwidth. The contribution of this paper is threefold.

1. First, to the best of our knowledge, the proposed architecture is the first practical one to meet the demands of miniaturized devices. Motivated by the distributed source coding (DSC) for sensor networks,[20] the proposed scheme includes a simplified encoder without having access to reference sequences or communicating with other encoders, and a complex decoder that detects repeated subsequences in the stored reference sequences and decompress the received encoded bits with the specifically designed graphical model. Hence, the proposed compression system can successfully meet the constraints and requirements of the miniaturized sequencing devices.
2. Second, a flexible encoding and decoding mechanism is proposed. Using feedback from the decoder, the encoder transmits either hashes conducting the detection of variable-size exact repeats in decoder or syndromes obtained with low-complexity Slepian–Wolf (SW) coding[21] of the nonrepeated subsequences. The proposed encoder and decoder perform efficiently by considering both exact repeats and approximately repeated subsequences (eg, insertion, deletion, and substitution).
3. Third,[21] the syndrome and reference sequences at the decoder, we construct a novel factor graph model to tackle the challenge in detecting insertion, deletion, and substitution between the reference and original source. Experimental results show that the proposed architecture can achieve an efficient compression performance with significantly low encoding complexity when compared to the benchmark compressor GRS.

The rest of this study is organized as follows. In the System Architecture section, we introduce the proposed DSC-based genome compression system, which includes the implementation details of the proposed hash-based (exactly) repeated sequence coding with adaptive length and an overview of syndrome-based nonrepeated sequence coding. Then, in the the Syndrome-based Nonrepeated Sequence Coding section, we focus on the design of the coding, which can handle the insertion, deletion, and substitution between sources and reference. The experimental results and concluding remarks can be found in the last two sections.

System Architecture

The block diagram of the proposed genome compression framework is depicted in Figure 1. Suppose that there are two correlated DNA sequences (ie, source and reference sequences) available at the encoder and decoder, respectively, the variations between the two sequences are modeled by insertion, deletion, and substitution. The alphabet of our studied DNA sequence is confined within the set {"A", "C", "G", "T", "N"}, where "N" denotes an unknown base due to a low sequencing quality. Figure 2 shows the logical flow of the proposed framework, which we will discuss in detail.

At the encoder (see the left hand side of Fig. 2), a streaming DNA sequence obtained from the portable sequencer will be first stored in the incoming data buffer for further processing. Second, a sub-sequence x_i^L, which starts with the i-th to be compressed based on the source sequence, is extracted from the incoming data buffer, where its length L and the corresponding coding method are decided by the adaptive code length and types selection module. The compressed sequence can be either low-density parity-check Accumulate (LDPCA) syndromes $S_{x_i^L} = \mathbf{H}x_i^L$ or hash bits $h_{x_i^L}$ depending on whether variations are presented between the source and the reference sequence, based on the decoder feedback, where $\mathbf{H}$ is the parity check matrix in LDPC codes. Third, the encoded sequence will be temporally stored in the forward data buffer and send to the decoder.

At the decoder (see the right hand side of Fig. 2), the received streaming data in the incoming data buffer will be processed by one of the following modules based on the corresponding data compression mode (ie, either hash bits or syndromes).

1. For the received hash data $h_{x_i^L}$, it will be compared with the hashes generated from a bunch of subsequence candidates $y_{j+U}^L, \cdots, y_{j+V}^L$ within the reference sequence for $V-U+1$ total candidates, where j is the current offset compensated start location, and U and V are predefined lower and upper bounds, respectively, of the search region for start locations. Then, the comparison result can be further processed as follows.

   a. If a matched hash $h_{y_k^L}$ for $k = j+U, \cdots, j+V$ is detected (ie, $h_{y_k^L} = h_{x_i^L}$), the next offset compensated start location of the sliding window can be updated as $j = k + L$ (see Fig. 3). Moreover, we claim that y_k^L will be identical to x_i^L, if $h_{y_k^L}$ and $h_{x_i^L}$ are matched with each other, which is the fundamental assumption of our proposed system. Intuitively, the aforementioned assumption can be enforced by choosing a strong hash code with a small search region. The

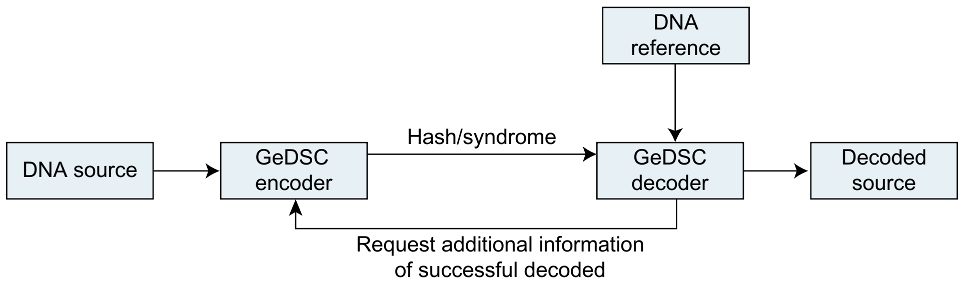

Figure 1. Workflow of genome compression based on DSC.

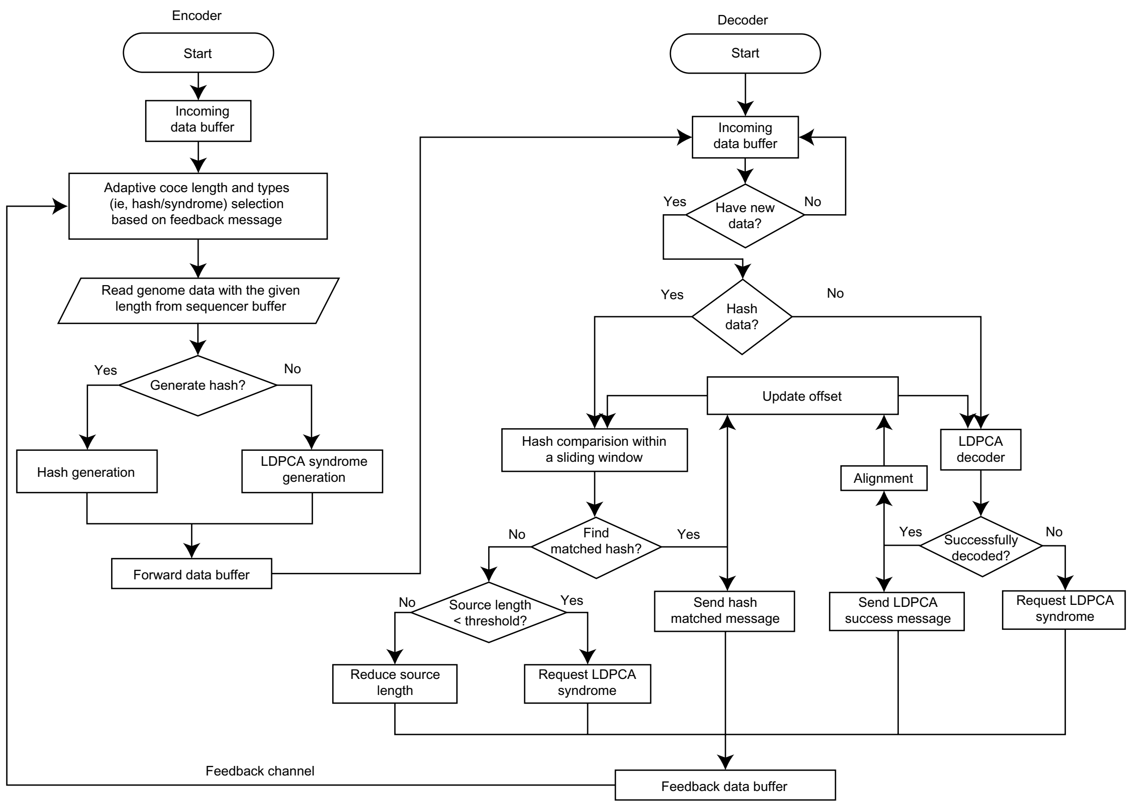

Figure 2. Workflow of genome compression based on DSC.

experimental results based on sequences[22,23] with total more than 238 million bases demonstrate that a 16-bit cyclic redundancy check hash code with a search region $U = -2$ and $V = 10$ provides a strong assertion of such assumption. In addition, the decoder will inform the success to the encoder and request a longer code length based on a predefined protocol as updating $L^{\text{current}} = bL_0$, where L_0 is a predefined initial length and the scaling factor b is updated as $b = b + d_b$, d_b is an incremental constant, and b is initialized as 0. For example, at the beginning, $L^{\text{current}} = L_0$, if a matched hash is detected, the adaptive length L^{current} will be updated as $L^{\text{current}} = d_b L_0$, as the scaling factor $b = 0 + d_b$. Similarly, if n_b number of successively matched hashes are detected, the adaptive length and its corresponding scale factor will be $L^{\text{current}} = n_b d_b L_0$ and $b = n_b d_b$, respectively.

b. If no matched hash can be detected, the following two conditions will be checked.
   i. if $L^{\text{current}} = L_0$, the decoder will inform the hash matching failure to the encoder and request syndromes from the encoder for further action.
   ii. Otherwise, the decoder also informs the hash matching failure to the encoder, but requests a shorter code length by setting $L^{\text{current}} = L_0$.

2. For the received syndromes, the decoder will pass the syndrome to the proposed factor graph-based LDPCA decoder with the capability of handling deletion, insertion, and substitution between the source and the reference. (See the Syndrome-based Nonrepeated Sequence Coding section for more implementation details). The following two conditions will be checked.
   a. If the decoded source $\hat{\mathrm{x}}_i^L$ satisfies both the parity check constraint (ie, $\mathrm{s}_{\mathrm{x}_i} = H\hat{\mathrm{x}}_i^L$) and the hash constraint (ie, $\mathrm{h}_{\mathrm{x}_i^L} = h_{\hat{\mathrm{x}}_i^L}$), the decoder will send an LDPCA success message back to the encoder and update the offset compensated start location j through the Smith–Waterman local alignment between the reference and the decoded source. Moreover, the encoder will send hash codes to the decoder for the next subsequence.
   b. Otherwise, the decoder will request additional LDPCA syndromes from the encoder.

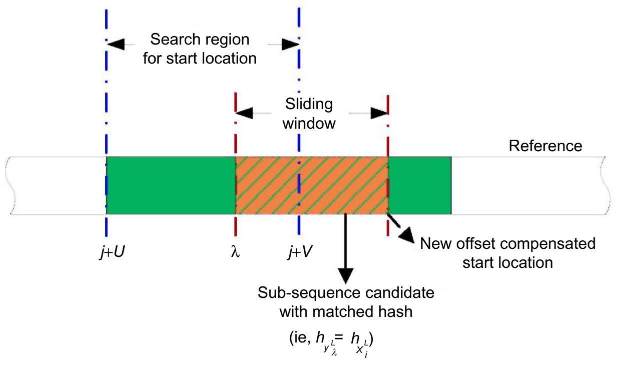

Figure 3. The diagram of hash-based coding.

Syndrome-Based Nonrepeated Sequence Coding

As previously mentioned in our system architecture, if an exact repeat cannot be identified by hash coding, the decoder will request syndromes from the encoder through a feedback channel. In this section, we introduce the codec design of the proposed syndrome-based nonrepeated sequence coding.

Syndrome-based nonrepeated sequence encoding. The first step of the proposed syndrome-based nonrepeat encoder is to convert DNA data into a binary source, such that they can be compressed under a binary LDPCA encoder. Suppose the following mapping rule for the letters within the alphabet, ie, "A" $\rightarrow$ 000, "C" $\rightarrow$ 001, "G" $\rightarrow$ 010, "T" $\rightarrow$ 011, "N" $\rightarrow$ 100, a DNA subsequence x can be represented by the corresponding binary vector x_b. For instance, given a DNA subsequence x = ["A" "T" "G" "C" "T" "N"]^T with length $N = 6$, its corresponding binary vector will be x_b = [000 011 010 001 011 100]^T with length $3N$. Thus, for LDPC-based SW coding (ie, lossless DSC), the compressed syndromes will be generated through $\mathrm{S}_\mathrm{x} = \mathbf{H}\mathbf{x}_\mathrm{b}$, where $\mathbf{H}$ is a sparse parity check matrix with size $M \times 3N$ and $M < 3N$. Thus, the resulting code rate can be expressed as $R = M/N$ bits per base. It is worth mentioning that the computational complexity of the aforementioned encoder is ultra-low, since the only operation is the bit-wise multiplication between the sparse matrix $\mathbf{H}$ and the original source. Moreover, we employ LDPCA codes to implement rate adaptive decoding, where the decoder can incrementally request additional LDPCA syndromes from the encoder through a feedback channel, when facing decoding errors.

Syndrome-based non-repeated sequence decoding. To perform syndrome-based decoding for non-repeat DNA subsequence x with the reference sequence as side information y, the key factor is to be able to explore the variations between the source subsequence x and the reference sequence y, where the variations are modeled by the insertion, deletion, and substitution between the source and reference. Moreover, a substitution can be expressed as an insertion in the source sequence followed by a deletion in the corresponding location in the reference sequence. In this section, we demonstrate that such variations can be effectively estimated through Bayesian inference on graphical models. The graphical model of our proposed syndrome-based decoding with variation is depicted in Figure 4. In Figure 4, the variable nodes (usually depicted by a circle) denote variables such as source symbol, binary source bits, local offset introduced by variation, and syndromes. Besides, factor nodes (depicted by squares) represent the relationship among the connected variable nodes. In the rest of this section, we will describe how to construct the proposed factor graph for the DNA sequence decoding with variations. We first study the parity check constraint imposed by the received syndromes, where $s_1, \ldots, s_M$, the realization of variable node S_l, $l = 1, \ldots, M$, denotes the received syndromes in Figure 4. Similar to the standard LDPC codes, the factor nodes c_l, $l = 1, \ldots, M$, take into account the parity check constraints, where the corresponding factor function can be expressed as

$$c_l(\mathrm{x}_{c_l}, s_l) = \begin{cases} 1, & \text{if } s_l \oplus \bigoplus \mathbf{x}_{c_l} = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where $\mathbf{x}_{c_l}$ denotes the set of neighbors of the factor node c_l, and $\bigoplus \mathbf{x}_{c_l}$ denotes the binary sum of all elements of the set $\mathbf{x}_{c_l}$.

Moreover, x_i^1, x_i^2, x_i^3, and the realization of variable node X_i^r with $i = 1, \ldots, N$, $r = 1,2,3$, are the binary representation for the i-th base x_i in the DNA sequence according to the mapping rule introduced in the Syndrome-based Nonrepeated Sequence Encoding subsection above, where the mapping rule is captured by the factor node f_i, $i = 1, \ldots, N$ with corresponding factor function as follows

$$f_i(x_i, x_i^1, x_i^2, x_i^3) = \begin{cases} 1, & \text{if } \mathrm{map}(x_i^1, x_i^2, x_i^3) = x_i, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

where map(•) denotes the mapping from the binary bits "•" to a letter in the alphabet, eg, the output of map(0, 1, 1) corresponds to the letter "T".

Moreover, since the alphabet is not uniformly distributed in an arbitrary DNA sequence, the prior distribution for the alphabet is captured by the factor node h_{x_i}, where learning prior through training DNA sequences will be discussed shortly in the Results section.

Now, we introduce an additional erasure variable node M_i to capture the variation between reference y_i and source x_i,

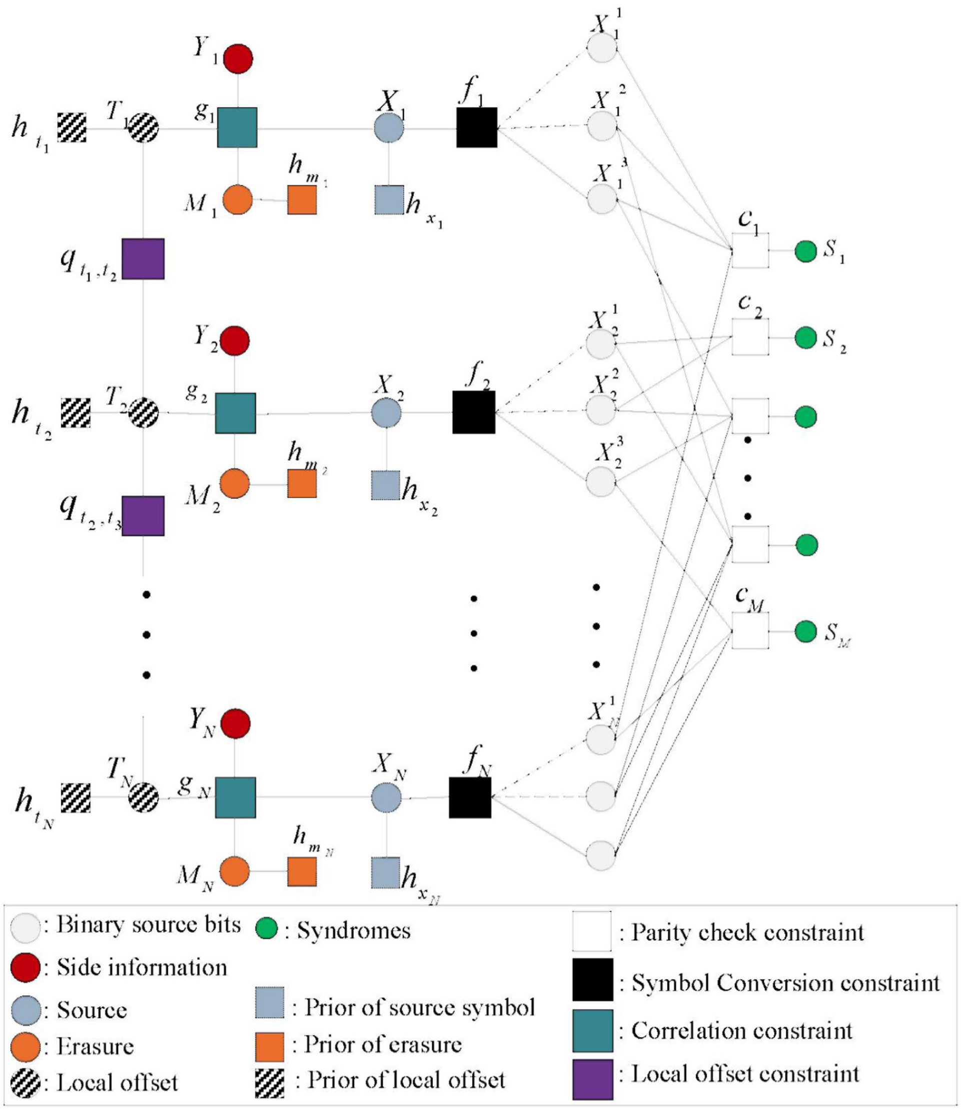

Figure 4. Factor graph of genome compression based on DSC.

where the variable $m_i = 1$ indicates the presence of variations, $m_i = 0$ means the existence of matches $y_{i+t_i} = x_i$ and $t_i = -T, \cdots, T$ are all possible local offsets within the search region $[-T, T]$. Moreover, the corresponding prior distribution of variable m_i is captured by the factor node h_{m_i} with factor function defined as

$$h_{m_i}(m_i) = \begin{cases} 1 - p_e & \text{if exist matches } y_{i+t_i} = x_i \\ p_e, & \text{otherwise,} \end{cases} \quad (3)$$

where p_e can be learnt through training DNA sequence.

For the existence of matches $y_{i+t_i} = x_i$, the local offset t_i is captured by the variable node T_i and its corresponding prior is represented by the factor node h_{t_i} with $h_{t_i}(t_i) = p_{t_i}$, where p_{t_i} can be learnt through training DNA sequences. Furthermore, as the local offsets between adjacent DNA bases do not vary significantly in our assumption, it is expected that adjacent variables t_i will not differ much in value. Such characteristic is captured by the additional factor node $q_{t_i,t_{i+1}}$, where the corresponding factor function is defined as

$$q_{t_i,t_{i+1}}(t_i, t_{i+1}, \alpha) = \frac{\alpha}{2} e^{-\alpha|t_i - t_{i+1}|}, \quad (4)$$

where α is the scale parameter of the Laplace distribution.

The factor node g_i and its corresponding factor function $g_i(m_i, y_i, x_i, t_i)$ are introduced to combine the impact imposed by the side information y_i, erasure m_i, and local offset t_i. For $m_i = 0$, the factor function can be expressed as,

$$g_i(m_i = 0, y_i, x_i, t_i) = \begin{cases} 1 & \text{if } y_{i+t_i} = x_i, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

For $m_i = 1$, the variable nodes T_i and Y_i will be disconnected from the factor node g_i. Therefore, the simplified factor function $g_i(m_i = 1, x_i) = 1$ can be used to take the impact of erasure into account.

Finally, in the context of Bayesian inference, estimating unknown variables corresponds to the evaluations of their marginal distribution, which can be efficiently achieved by performing message passing on a factor graph. With the factor graph defined in Figure 4, the original DNA sequence can be recovered by the estimated posterior distribution of each source x_i.

Results

Two genome sequence data, The Arabidopsis Information Resource (TAIR)[24] and The Institute for Genomic Research (TIGR),[25] are adopted in our experiments. These two databases are collected by professional groups or institutes, and have been widely used by research communities.

The TAIR maintains a database of genetic and molecular biology data for the model higher plant Arabidopsis thaliana.[24] In this experiment, we test TAIR8[22] dataset and TAIR9[23] dataset, where each dataset contains five chromosomes with over 238 million bases in total. Moreover, the genome of TAIR9 is used for testing our compression performance with TAIR8 as reference only available at the decoder. For this experiment, all the hyper-parameters are initialized as follows, the initial code length $L_o = 528$, the incremental constant $b_d = 3$, the scale parameter of Laplace distribution $\alpha = 1$, the maximum local offset search region $T = 4$, and the erasure probability $p_e = 0.01$. The proposed codec is implemented in MATLAB and evaluated on an Intel 3.0GHz CPU.

First, the empirical marginal statistics of the DNA bases {"A", "T", "G", "C", "N"} and those of the local offsets t_i within the range from –4 to 4 are shown in Figure 5A and 5B, respectively, which will be used as the priors in the syndrome-based nonrepeated sequence decoding. In Figure 5A, we verify the assumption that the alphabets of DNA sequences are usually non-uniformly distributed. Moreover, Figure 5B depicts that the maximum local offset with $T = 4$ is sufficiently large for capturing shifts between the reference and the source.

Figure 6A illustrates the relationship between the average code rates and the different maximum local offsets in

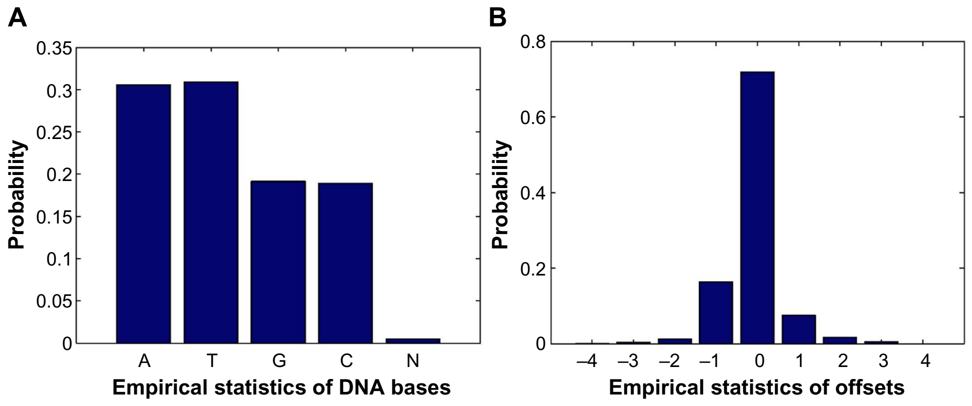

Figure 5. The empirical statistics of (**A**) the DNA bases {"A", "T", "G", "C", "N"} and these of (**B**) the local offsets with the range from –4 to 4.

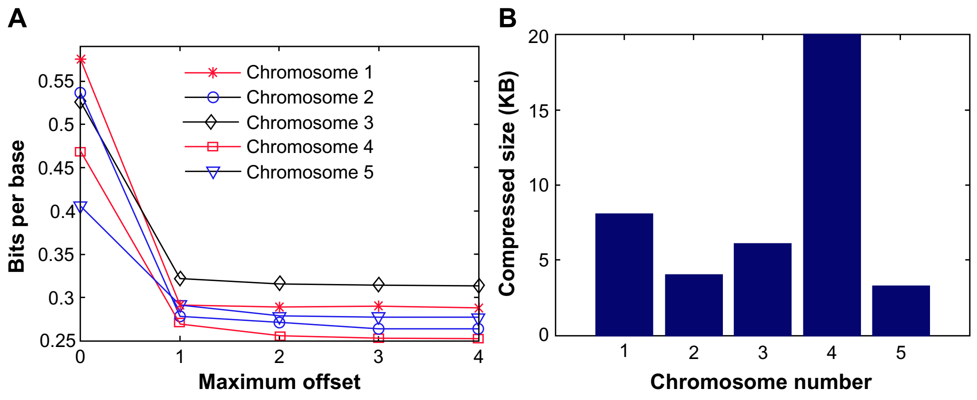

Figure 6. Compression performance of the proposed codec on TAIR dataset, (**A**) the average code rates vs. the different maximum local offsets in syndrome coding; (**B**) the overall compression performance (ie, hash bits + syndromes) for all 5 chromosomes.

syndrome coding based on all five chromosomes. In Figure 6A, we can see that the code rates decrease as the maximum local offsets increase, due to the fact that a larger maximum local offset offers a wider search region for exploring the reference. However, a larger maximum local offset may also result in a higher decoding complexity. Figure 6B shows the overall compression performance (ie, hash bits + syndromes) for all five chromosomes in terms of compressed file size. Moreover, Figure 7 shows a side-by-side comparison of the compression rate and compression time. We can see that both the proposed

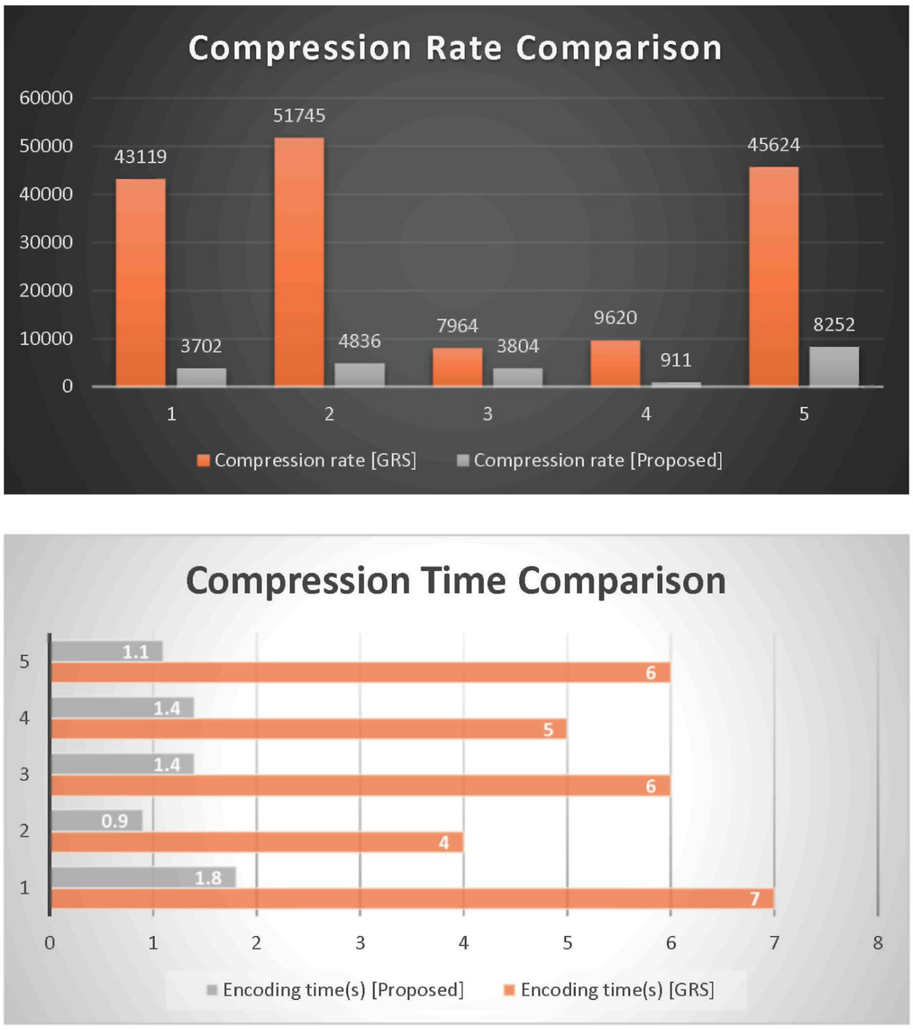

Figure 7. Performance comparison between GRS and our proposed codec on TAIR dataset.

method and GRS algorithm achieve significant file size reductions (ie, up to 8252 × file size reduction).

For the TIGR data, we tested the chromosome 4 (35.8MB) of the TIGR5 dataset using the chromosome 4 of the TIGR6 as the reference by varying the LDPC code length (ie, 528, 1056, 1584, 2112, and 2640) as shown in Table 1. Moreover, we also implemented GRS method on this dataset, and the result was also listed in Table 2 as reference. We can see that the compression performance decreases as the LDPC code length increases. In this case, the proposed algorithm achieved better compression performance when compared with GRS. It is because that the reference chromosome 4 in TIGR6 includes a significant amount of insertions when compared with the same chromosome in TIGR5, where the insertion information in the reference chromosome has no contribution to the size of DSC compressed data.

Our proposed encoder shows a significantly lower encoding complexity. It is worth mentioning that the proposed codec is implemented by MATLAB, where a potential performance boost is highly expected by using more efficient programing languages eg, C/C++. To the best of our knowledge, this is the first study of DSC-based genome compression. There is no doubt that it opens many possibilities for the portable miniaturized applications in which energy consumption and bandwidth usage are of paramount importance.

Conclusion

In this study, we present a DSC-based genome compression architecture. To the best of our knowledge, the proposed framework is the first study of its kind: specially targeted at the low-complexity genome encoding for miniaturized devices, which have limited processing capabilities, power budget, storage space, and communication bandwidth. Compared with traditional reference-based DNA compression algorithm (eg, GRS), the proposed framework offers ultra-low encoding complexity (nonrepeated subsequences are encoded using low-complexity DSC encoding), while (exactly) repeated subsequences are compressed through adaptive length hash coding based on the decoder feedback. The customized factor graph-based decoder tackles the challenges of detecting insertion, deletion, and substitution between the reference and the original source, and it recovers the nonrepeated subsequences based on received syndromes. Last but not least, our proposed genome compression framework incorporates LDPCA codes for rate adaptive decoding. Experimental results show that the proposed architecture could achieve an efficient compression performance with significantly lower encoding complexity when compared to the benchmark compressor GRS.

Table 1. Performances of our proposed method on chromosome 4 of TIGR5 (35.8 MB).

LDPC CODE LENGTH	COMPRESSION SIZE (KB)	ENCODING TIME (SECONDS)	DECODING TIME (SECONDS)
528	3.68	0.04	298
1056	4.58	0.009	596
1584	5.67	0.01	787
2112	6.97	0.01	1102
2640	6.72	0.08	1374

Table 2. Performance of GRS on chromosome 4 of TIGR5 (35.8 MB).

COMPRESSION SIZE (KB)	ENCODING TIME (SECONDS)	DECODING TIME (SECONDS)
26.34	12	6

Author Contributions

Conceived and designed the experiments: SW, XJ, FC. Analyzed the data: FC. Wrote the first draft of the manuscript: SW, XJ, LC. Contributed to the writing of the manuscript: XJ, FC, SC. Agree with manuscript results and conclusions: XJ, FC, SC. Jointly developed the structure and arguments for the paper: SW, FC, LC, SC. Made critical revisions and approved final version: FC, XJ, SC. All authors reviewed and approved of the final manuscript.

REFERENCES

1. MinION: a miniaturised sensing instrument. Oxford Nanopore Tech. 2012. [Online]. Available from: http://www.nanoporetech.com/technology/minion-a-miniaturised-sensing-instrument.
2. Ziv J, Lempel A. Compression of individual sequences via variable-rate coding. *IEEE Trans Inform Theory*. 1978;24(5):530–6.
3. Grumbach S, Tahi F. Compression of DNA sequences. In: Proceedings of Data Compression Conference; 1993; Snowbird, UT, USA; 340–50.
4. Grumbach S, Tahi F. A new challenge for compression algorithms: genetic sequences. *J Inf Process Manage*. 1994;30(6):876–87.
5. Chen X, Kwong S, Li M. A compression algorithm for DNA sequences. *IEEE Eng Med Biol Mag*. 2001;20(4):61–6.
6. Chen X, Li M, Ma B, Tromp J. DNACompress: fast and effective DNA sequence compression. *Bioinformatics*. 2002;18(12):1696–8.
7. Matsumoto T, Sadakane K, Imai H. Biological sequence compression algorithms. *Genome Inform*. 2000;11:43–52.
8. Behzadi B, Fessant FL. DNA compression challenge revisited: a dynamic programming approach. *CPM*. 2005;3537:85–96.
9. Tabus I, Korodi G, Rissanen J. DNA sequence compression using the normalized maximum likelihood model for discrete regression. In: Proceedings of Data Compression Conference; 1993; Snowbird, Utah, USA; 253–62.
10. Korodi G, Tabus I, Rissane J, Astola J. DNA sequence compression – Based on the normalized maximum likelihood model. *IEEE Signal Processing Mag*. 2007;24(1):47–53.
11. Korodi G, Tabus I. Normalized maximum likelihood model of order-1 for the compression of DNA sequences. In: Proceedings of Data Compression Conference; 2007; Snowbird, Utah, USA; 33–42.
12. Cao MD, Dix TI, Allison L, Mears C. A simple statistical algorithm for biological sequence compression. In: Proceedings of Data Compression Conference; 2007; Snowbird, Utah, USA; 43–52.
13. Pinho AJ, Neves A, Bastos C, Ferreira P. DNA coding using finite-context models and arithmetic coding. In: Proceedings of Data Compression Conference Acoustics, Speech and Signal Process; 2009; Taipei; 1693–6.
14. Pratas D, Pinho A. Compressing the human genome using exclusively Markov models. In: 5th International Conference on PACBB (PACBB 2011); 2011; 213–20.

15. Kuruppu S, Puglisi S, Zobel J. Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval. *String Process Inf Retrieval*. 2010;6393:201–6.
16. Wang C, Zhang D. A novel compression tool for efficient storage of genome resequencing data. *Nucleic Acids Res*. 2011;39(7):e45.
17. Pinho A, Pratas D, Garcia S. GReEn: a tool for efficient compression of genome resequencing data. *Nucleic Acids Res*. 2012;40(4):e27.
18. Kozanitis C, Saunders C, Kruglyak S, Bafna V, Varghese G. Compressing genomic sequence fragments using SlimGene. *J Comput Biol*. 2011;18(3):401–13.
19. Fritz MHY, Leinonen R, Cochrane G, Birney E. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Res*. 2011;21:734–40.
20. Xiong Z, Liveris AD, Cheng S. Distributed source coding for sensor networks. *IEEE Signal Processing Mag*. 2004;21(5):80–94.
21. Pradhan SS, Ramchandran K. Distributed source coding using syndromes (DISCUS): design and construction. *IEEE Trans Inform Theory*. 2003;49(3):626–43.
22. Huala E, Dickerman A, Garcia-Hernandez M, et al. The arabidopsis information resource (TAIR): a comprehensive database and web-based information retrieval, analysis, and visualization system for a model plant. *Nucleic Acids Res*. 2001;29(1):102–5.
23. Rhee S, Beavis W, Berardini T, et al. The arabidopsis information resource (TAIR): a model organism database providing a centralized, curated gateway to arabidopsis biology, research materials and community. *Nucleic Acids Res*. 2003;31(1):224–8.
24. TAIR. [Online]. Available from: http://www.arabidopsis.org/. 2014.
25. TIGR rice genome annotation. [Online]. Available from: http://www.arabidopsis.org/. 2014.