Chamaeleo: a robust library for DNA storage coding schemes

Zhi Ping [1], Haoling Zhang[1], Shihong Chen[1,2], Qianlong Zhuang[1,3], Sha Joe Zhu[4,5], Yue Shen[1,2]

[1]Guangdong Provincial Key Laboratory of Genome Read and Write, Shenzhen Institute of Synthetic Biology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, George Church Institute of Regenesis, BGI-Shenzhen, Guangdong, China,

[2]China National GeneBank, BGI-Shenzhen, Jinsha Road,Shenzhen, 518120, China,

[3]Qingdao University, 308 Ningxia Road, Shinan District, Qingdao,Shandong, China

[4]Big data Institute, University of Oxford, Old Road Campus, Oxford, OX3 7LF, United Kingdom

[5]Sensyne Health Plc., Science Park, Schrodinger Building, Heatley Rd, Oxford OX4 4GE, United Kingdom

Abstract

Summary: *Chamaeleo* is currently the only collection library that focuses on adapting multiple well-accepted coding schemes for DNA storage. It provides a tool for researchers to study various coding schemes and apply them in practice. *Chamaeleo* adheres to the concept of high aggregation and low coupling for software design which will enhance the performance efficiency. Here, we describe the working pipeline of *Chamaeleo*, and demonstrate its advantages over the implementation of existing single coding schemes.

Availability: The source code is openly available at https://github.com/ntpz870817/Chamaeleo, it can be also installed by the command of pip.exe, "pip install chamaeleo". Alternatively, the wheel file can be downloaded at https://pypi.org/project/Chamaeleo/. Detailed documentation is available at https://chamaeleo.readthedocs.io/en/latest/.

Contact: shenyue@genomics.cn

Supplementary information: Supplementary data are available online.

1 Introduction

Compared to the orthodox information storage media, DNA has extremely high information capacity and durability. From the beginning of the 21st Century, many scientists devoted to developing efficient strategies to store digit informaiton in DNA. As a vital part of research in DNA storage, coding scheme is used for transcoding digit to DNA following particular algorithm. It is the first and last steps in storing digit information as DNA.

Several well-accepted coding schemes had been proposed in recent years. These schemes focus on optimizing the coding efficiency under certain constrains. Simple code, Goldman's code, and Grass' code use different algorithms to increase coding efficiency as well as eliminate homopolymer. Erlich's DNA Fountain is based on LT code, it improves the coding efficiency and applies screening steps to restrict the occurrence of homopolymer and limit GC content. Yin-Yang code provides 1,536 derived rules by the incorporation of Yin and Yang rule, and can be used to find the best rule for particular file based on its byte frequency.

Although these schemes are implemented as a program, these individual programs do not have excellent code architecture from the perspective of software engineering. A few program-based validation and improved work will be hindered. Meanwhile, these programs exist independently and are implemented in different programming languages. Therefore, it is difficult to compare these performances in practice or customize them in specific requirements.

Through this work, we introduce Chamaeleo, an open-source library focuses on different coding schemes for DNA storage. This library provides a useful platform for researchers to study different coding schemes and its efficient implementation is suitable for supporting real-world DNA storage applications.

2 Library overview

Currently, Chamaeleo provides four classical coding schemes, including Simple code, Goldman's code, Grass' code, and Yin-Yang code. In addition, Chamaeleo also provides functionalities for data processing, customized message output, and error correction.

Chamaeleo has several features:

- *Readability*: To improve the readability for developers afterwards, this library provides a naming convention in which the identifiers can denote their types and functions.
- *Efficiency*: We use some native acceleration strategies of Python, which enables transcoding process faster with accuracy guaranteed.
- *Maintainability and Expandability*: The library contains five modules, including transcoding module, data handle module, process monitoring module, optional error-correction module, and customized message output module. The modularization of workflow, compared to individual implementations of particular coding schemes, can coordinate arbitrarily and help users to customize the encoding and decoding process flexibly.

3 Materials and methods

Chamaeleo can be used for the encoding and decoding process for DNA storage in the terminal and visual interface. In Figure 1, we demonstrate how Chamaeleo can be used to

complete the transcoding process. Specifically, it is divided into two main processes: encoding process and decoding process.

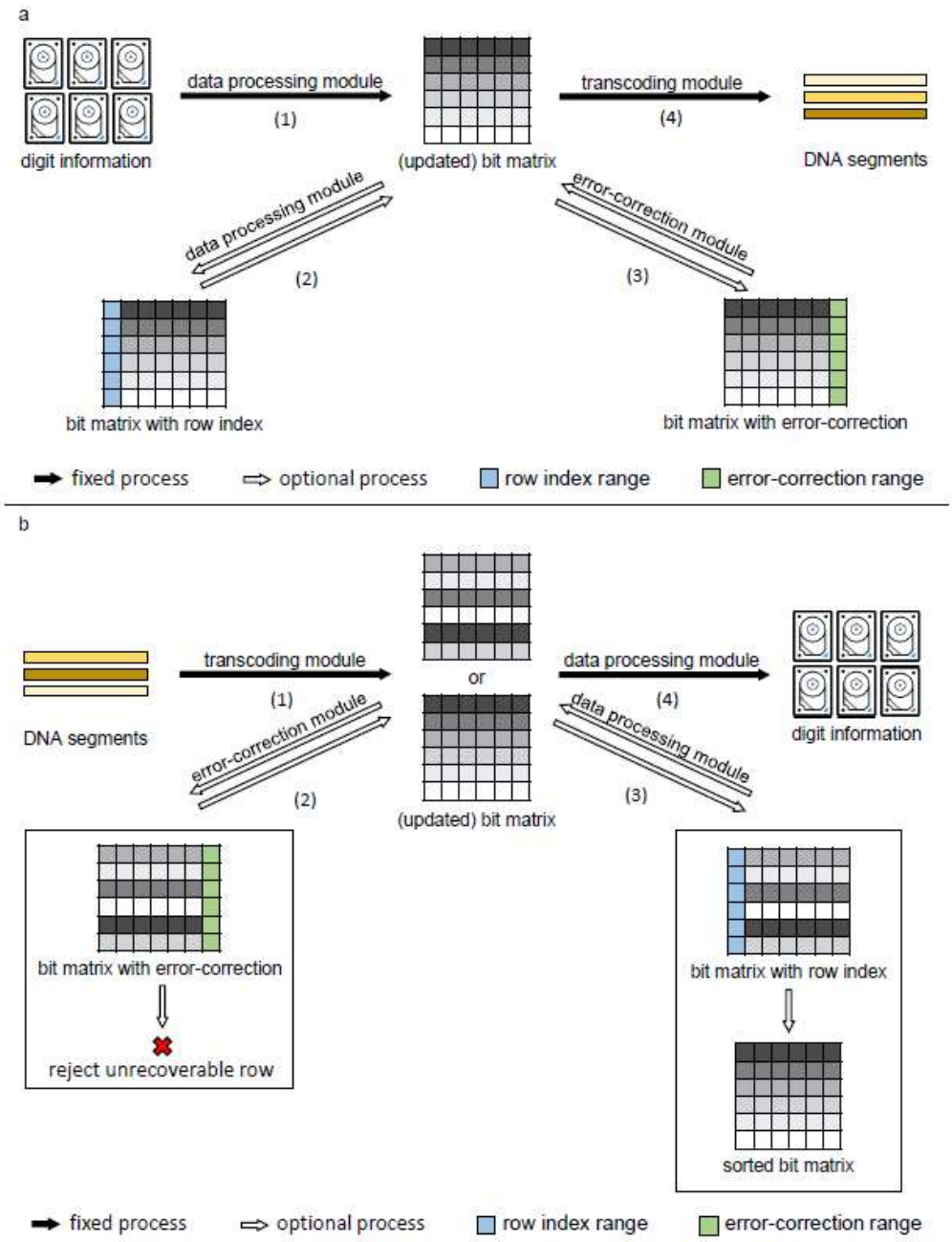

Fig.1 Detailed illustration of encoding and decoding processing. Step 1 and 4 in (a) and (b) are required, and step 2 and 3 are optional. The complete execution process considering all options is: (1) -> (2) -> (3) -> (4).

3.1 Encoding process

The illustration of encoding process is shown in Figure 1a. A digit file specified by users is first converted to bit matrix by data processing module. The process of adding row indexes (bit format) is optional depending on codes. Error-correction can then be added in the above binary matrix optionally by error-correction module. Currently, Hamming code and Reed-Solomon code were included. This library converts the matrix into DNA segments by using the code initialized by users in the transcoding module, and the file consist of DNA segments is finally outputted through data processing module.

3.2 Decoding process

This process and the encoding process are complementary to each other, which is shown in Figure 1b. When the error-correction module was added in their customized process, the incorrect and unrepairable data will be discarded and user will be notified. Later, the matrix will be sorted by its row indexes in this process when the user added the row index in the encoding process.

4 Discussion and future works

Files with different attributes were transcoded into DNA sequences using incorporated coding schemes. The runtime of encoding and decoding using different coding schemes were benchmarked (see Table 1). For more examples, please refer to supplementary information. The results were proved to be consistent with the comparison between these schemes using previous packages. Furthermore, the unique data handle module introduced in this library, handling bit arrays and DNA sequences, also showed a fast performance for reading and writing of both binary and DNA sequences. We anticipate that this library will facilitate the development of future tools of DNA storage and help to create more a comparative model for different coding schemes.

Table 1. Partial examples of encoding runtime.

File name	Simple code	Goldman's code	Grass' code	Yin-Yang code
Microsoft Winmine	1.81s	0.61s	0.36s	1.35s
The Wandering Earth	9.67s	2.18s	1.26s	4.13s
United Nations Flag	0.12s	2.55s	1.48s	5.93s

This library would be an open source tool for developers and users. In the future, we expect more developers to incorporate their unique coding algorithms or strategies for DNA digital storage into this library in the future. Two issues can be tackled in the next version: (1) incorporate more coding schemes as well as their unique module; (2) establish an evaluation system and criteria which can compare different coding schemes.

Funding

This work was supported by National Key Research and Development Program of China (No. 2018YFA0900100), the Guangdong Provincial Key Laboratory of Genome Read and Write (No.2017B030301011) and China National GeneBank, Shenzhen.
Conflict of Interest: SJZ is a currently an employee at the Sensyn Health Plc. This work was completed when SJZ was working at the University of Oxford, and consulting for the BGI.

Reference

Church G. *et al.* Next-generation digital information storage in dna. *Science*, 337 (6102): 1628–1628, 2012. doi: 10.1126/science.1226355.

Erlich Y. and Zielinski D.. Dna fountain enables a robust and efficient storage architecture. *Science*, 355 (6328): 950–954, 2017. doi: 10.1126/science.aaj2038.

Garlan D. and Shaw M.. An introduction to software architecture. In *Advances in software engineering and knowledge engineering*, pages 1–39. World Scientific, 1993. doi: 10.1142/9789812798039_0001.

Goldman N. *et al.* Towards practical, high-capacity, low-maintenance information storage in synthesized dna. *Nature*, 494 (7435): 77, 2013. doi: 10.1038/nature11875.

Grass R. *et al.* Robust chemical preservation of digital information on dna in silica with error-correcting codes. *Angewandte Chemie International Edition*, 54 (8): 2552–2555, 2015. doi: 10.1002/anie.201411378.

Hamming R.. Error detecting and error correcting codes. *The Bell system technical journal*, 29 (2): 147–160, 1950. doi: 10.1002/j.1538-7305.1950.tb00463.x.

Li W.J. *et al.* Fileprints: Identifying file types by n-gram analysis. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pages 64–71. IEEE, 2005. doi: 10.1109/IAW.2005.1495935.

Ping *et al.* Towards practical and robust dna-based data archiving by codec system named'yin-yang'. *bioRxiv*, page 829721, 2019a. doi: 10.1101/829721.

Ping *et al.* Carbon-based archiving: current progress and future prospects of dna-based data storage. *GigaScience*, 8 (6): giz075, 2019b. doi: 10.1093/gigascience/giz075.

Reed I.S and Solomon G.. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8 (2): 300–304, 1960. doi: 10.1137/0108018.