FCBC hybrid encoding method based on fountain code and Base64 code

Yu Wan
Monash University Joint Graduate School Southeast University - Suzhou
Suzhou, China
220214929@seu.edu.cn

Kun Bi[a*]
State Key Laboratory of Digital Medical Engineering, School of Biological Science and Medical Engineering, Southeast University
Nanjing, China
[a*]Corresponding author: bik@seu.edu.cn

Xiangwei Zhao[b*]
State Key Laboratory of Digital Medical Engineering, School of Biological Science and Medical Engineering, Southeast University,
Nanjing, China
[b*]Corresponding author: xwzhao@seu.edu.cn

***Abstract*-Fountain code is a commonly used and excellent DNA storage encoding method, but the encoding process will judge and screen the generated bases, resulting in duplicate encoding, long encoding time, and high computational resource consumption. This article proposes an FCBC hybrid coding method, which is based on fountain code coding. Base64 code is added during coding screening, and RS code is added after generating bases to control the GC content of bases at around 50% and reduce the probability of homopolymer generation. This improves the pass rate of fountain code base screening and reduces coding time. When the code length is 168, the FCBC hybrid encoding method can reduce the encoding time by 11% for a 2KB txt file compared to the encoding method without Base64 code, and by 20% for a 384KB jpg file. The FCBC hybrid encoding method can effectively reduce the encoding time using only fountain codes, and the introduction of RS codes can avoid errors such as base addition, replacement, and deletion, improving coding accuracy. In the future era of big data storage, the FCBC hybrid encoding method will help accelerate the data encoding process, reduce encoding time, save computing resources, reduce costs, and promote the development of DNA storage.**

Keywords-DNA storage; fountain code; Base64 code; RS code; FCBC hybrid encoding method

I. Introduction

Fountain code is an excellent DNA storage encoding method [1-4], which is an erasure code that can divide the source symbol into different encoding sequence symbols, send the symbols, and under ideal conditions, as long as the number of sequence symbols is sufficient, the source symbol can be restored [5-8]. The fountain code encoding and decoding method is simple and suitable for big data storage needs [9-10]. Fountain codes have a small decoding overhead and low compilation method complexity [11-12]. Initially designed for erasing channels in the communication field [13-14], but after improvement, the Erlich [15] team was the first to use fountain codes for DNA storage. The team first divides the binary data into fixed length non overlapping segments, generates a degree distribution function and encodes it according to the fountain code encoding rules, adds a seed as the address code in front of the data, converts it into bases according to custom rules, and determines whether the homopolymer and GC content of the base sequence meet the requirements. If the requirements are met, the base is retained, otherwise the base is deleted. Professor Zhang Shufang's team from Tianjin University [16-17] conducted experiments using Raptor codes with better performance in fountain codes. Firstly, pre coding was added to ensure good decoding performance of Raptor codes, and then RS (Reed Solomon Code) error correction mechanism was added to ensure the storage quality of information. Peng Yuan [18] chose to combine chaotic encoding with the LT code in the fountain code in the encoding section, grouping several small data packets into chaotic groups and performing chaos and diffusion, so that a single small data packet has a stable GC content and reduces the length of the homopolymer. And in order to prevent errors such as base replacement and loss during the generation, replication, and sequencing of DNA strands, RS error correction codes are also added to ensure the accuracy of information storage.

During the encoding process of fountain codes, the generated bases will be judged and screened. If there are homopolymers in the sequence or the GC content of the bases is unqualified, it will affect the subsequent DNA synthesis and sequencing. This base should be discarded and randomly selected for re encoding under the same degree until the required bases are generated. Therefore, fountain code encoding involves repetitive encoding, filtering, and judgment processes, which consume a lot of computing resources and have low encoding efficiency. Especially when facing big data encoding, the consumption time increases exponentially, which is not conducive to practical applications. To address the above issues, it is necessary to choose an appropriate encoding method to improve the pass rate of fountain code base screening and reduce encoding time.

Base64 code [19] is one of the common encoding methods used for 8-bit bytecode, which represents binary data based on 64 printable characters and is an information hiding technique. Zhang [20] used Base64 code for DNA storage, converting binary data into bases through 64 printable characters and custom balance code encoding, and controlling the base GC content to around 50%, reducing the probability of homopolymer generation, which meets the requirements of biochemical experiments. However, Base64 code cannot completely avoid homopolymers and can still cause errors in DNA synthesis and sequencing.

This article proposes a hybrid encoding method based on Fountain Code+Base64+RS code, called FCBC (Fountain Code and Base64 Code), to address the issues of long encoding time, repeated encoding, and high computational resource consumption in fountain code coding. The FCBC hybrid coding method improves the coding method by not only using fountain codes, but also introducing Base64 codes and RS codes to reduce the probability of base homopolymer production and control the base GC content at around 50%, improve the success rate of fountain code base screening, reduce the number of repeated coding times, reduce coding time, and save computational resources. And the FCBC hybrid coding method also uses RS error correction codes [21-22], which can ensure that there are no errors such as base addition, replacement, and loss, and ensure coding accuracy.

This article first converts the file to be encoded into binary data, uses fountain code to package and combine encoding, and then uses Base64 code to convert the binary data into bases, and adds an address code to determine whether the bases meet the requirements. If the requirements are met, the bases are retained, and if not, they are re encoded until the requirements are met. Finally, RS error correction code is added to prevent errors in sequencing and synthesis of bases. The fountain code coding process diagram is shown in Figure 1.

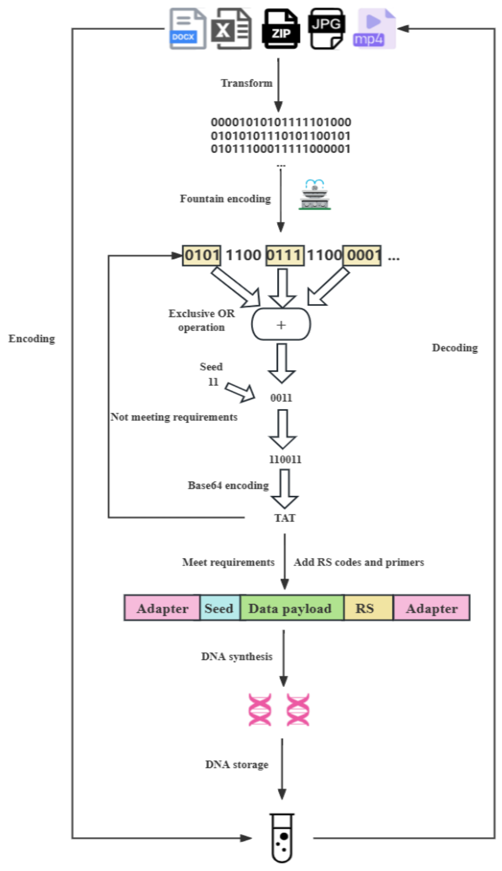

Figure 1 Fountain code coding flow chart

II. Method Design

This method will construct a hybrid encoding method for FCBC based on fountain code, Base64 code, and RS code. Firstly, fountain code will be used to encode binary data, and Base64 code will be introduced. Then, according to the four base conversion rule, the base GC content and homopolymer will be determined to be qualified. If qualified, the base will be retained and RS code will be added. If unqualified, this base will be discarded and re encoded until all data coding is completed. This method will first verify the impact of RS code and Base64 code on the FCBC hybrid coding method, including only fountain code encoding. The time taken from data conversion to encoding completion will be counted and analyzed. Based on this data, the impact of RS code and Base64 code on the FCBC hybrid coding method will be explored.

A. FCBC hybrid encoding method

1) Design of FCBC hybrid encoding method

This article will construct a mixed encoding method for fountain code, Base64 code, and RS code using FCBC. Firstly, select files of different sizes such as txt, docx, xlsx, zip, and jpg, and convert them into binary data. Afterwards, the binary data is divided into multiple encoding packets according to a certain code length, and the XOR operation is randomly selected based on the degree distribution function. Afterwards, the binary data obtained from XOR operation is grouped into 6 bits, encoded according to Base64 code, and converted into bases to determine whether the bases meet the requirements of biochemical experiments. If the requirements are met, the bases are retained, and RS code and primers are added. If not, the same degree encoding is used, and XOR operation is randomly selected until the requirements are met.

2) Fountain code coding

Fountain code encoding divides the source data into several equally sized packets, with the length of each binary data packet considered as the code length, and generates a degree distribution function based on the number of packets. When encoding, a degree will be selected from the degree distribution function, and packets will be randomly selected according to the degree for XOR operation. After generating binary data, it will be converted to bases according to the four base conversion rule. If the base does not meet the requirements of the biochemical experiment, the same degree is used to randomly select the package for XOR operation until the generated base meets the requirements; If the base meets the requirements, the base sequence is retained. Fountain code encoding process:

Step 1: Divide the data to be encoded into k small packets of a certain length, and randomly select an integer d in the range of 1 to k according to the illuminance distribution function ρ(d). Use d as the degree of encoding, which is the number of encoded packets.

$$\rho(i)=\begin{cases}\dfrac{1}{k}, & i=1\\[2mm] \dfrac{1}{i(i-1)}, & 2\le i\le k\end{cases} \tag{1}$$

Step 2: Evenly and randomly select d different packets from k small data packets according to the degree distribution function, perform XOR operation on the data in these d packets, and send the obtained result to the receiver. The degree distribution flowchart is shown in Figure 2.

Step 3: Continue to loop according to Step 2 until the data received by the receiving end can fully restore the original data, and then stop receiving the data.

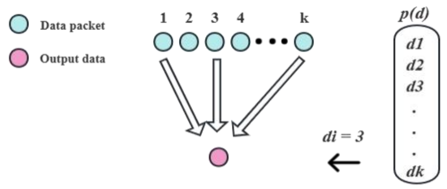

Fig. 2 Flow chart of degree distribution

3) Base64 encoding

On the basis of fountain code encoding, Base64 code is introduced to accelerate encoding efficiency and reduce encoding time. Base64 encoding first groups binary data into 6-bit groups, converts each group into decimal data, and converts decimal data into Base64 characters according to the Base64 encoding table. Then, characters are grouped into 7-bit groups. The balance code consists of 8-bit binary data, including 4 1s and 4 0s. After grouping the characters, the 1st, 3rd, 5th, and 7th characters are converted into binary data, while the 2nd, 4th, and 6th characters are converted into binary data according to the balance code. The two parts of the data are combined and converted into bases according to the four base conversion method. "00" is converted into "A", "01" is converted into "C", "10" is converted into "T", and "11" is converted into "G".

During the Base64 code encoding process, binary data can only be converted to "C" or "G" when "1" appears in the balanced code encoding; Binary data can only be converted to "A" or "T" when "0" appears in the balanced code encoding. In an 8-bit binary balanced code, there are 4 bits 1 and 4 bits 0, so the probability of 1 and 0 occurring is 50%. Therefore, the probability of binary conversion to base "C" or "G" is 50%, and the probability of conversion to "A" or "T" is also 50%. Therefore, Base64 encoding controls the base GC content to be around 50%. And in the balanced code, the distribution of "0" and "1" is uniform, and there are almost no long sequences of "0" and "1". Therefore, coding can effectively control the appearance of homopolymers in the sequence, and the base sequence meets the requirements of DNA synthesis. The Base64 encoding flowchart is shown in Figure 3. The balance code encoding flowchart is shown in Figure 4. Base64 encoding process:

Step 1: Encode the binary data of the fountain code into a group of 6-bit binary data.

Step 2: Convert the grouped data into Base64 characters according to the Base64 encoding table, as shown in Table 1.

Table 1. Base64 Encoding table

Decimal data	Base64 cod	Decimal data	Base64 codeword	Decimal data	Base64 codeword	Decimal data	Base64 codeword
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Step 3: Encode Base64 into a group of 7 codewords, and convert the 1st, 3rd, 5th, and 7th codewords into binary according to the Base64 encoding table. The length of the binary data corresponding to the 4 codewords is 24 bits; Convert the second, fourth, and sixth code words into binary according to the balanced code encoding table. The balanced code encoding table is shown in Table 2, and the total length of the binary data corresponding to the three code words is 24 bits. Afterwards, the converted 24 bit binary data will be corresponded one-to-one, and the data will be converted into bases according to the rules of quaternary conversion.

Table 2. Balance Code Encoding table

Base64 codeword	Decimal data	Binary data	Base64 codeword	Decimal data	Binary data
A	232	11101000	g	27	00011011
B	216	11011000	h	43	00101011
C	184	10111000	i	75	01001011
D	177	10110001	j	139	10001011
E	209	11010001	k	147	10010011
F	113	01110001	l	163	10100011
G	201	11001001	m	51	00110011
H	153	10011001	n	83	01010011
I	169	10101001	o	99	01100011
J	57	00111001	p	195	11000011
K	89	01011001	q	29	00011101
L	105	01101001	r	45	00101101
M	178	10110010	s	77	01001101
N	210	11010010	t	141	10001101
O	114	01110010	u	149	10010101
P	226	11100010	v	165	10100101
Q	202	11001010	w	53	00110101
R	154	10011010	x	85	01010101
S	170	10101010	y	101	01100101
T	58	00111010	z	197	11000101
U	90	01011010	0	46	00101110
V	106	01101010	1	78	01001110
W	180	10110100	2	142	10001110
X	212	11010100	3	198	11000110
Y	116	01110100	4	150	10010110
Z	228	11100100	5	166	10100110
a	204	11001100	6	54	00110110
b	156	10011100	7	86	01010110
c	172	10101100	8	102	01100110
d	92	01011100	9	23	00010111
e	108	01101100	+	39	00100111
f	60	00111100	/	71	01000111

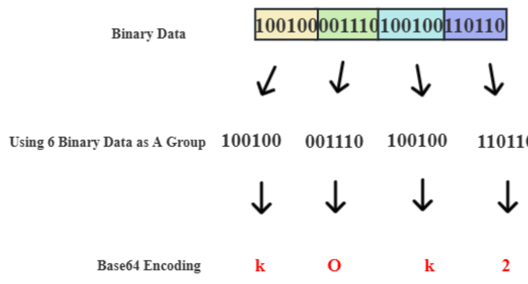

Figure 3 Base64 encoding flowchart

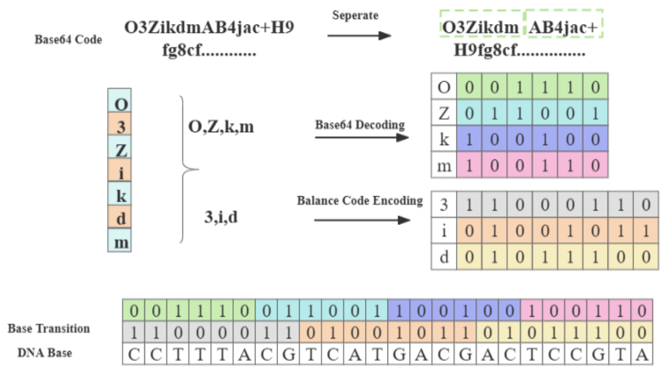

Figure 4: Balance code encoding flowchart. Divide the Base64 codewords into groups of 7 bits, restore the 6th bit binary data according to the Base64 encoding table for the 1st, 3rd, 5th, and 7th bits, convert the 2nd, 4th, and 6th bits into 8-bit binary data according to the balanced code encoding table, and concatenate the converted binary data to convert it into bases according to the quaternary rule

B. Encoding method design

This article explores the factors affecting the encoding time of FCBC hybrid encoding methods from multiple aspects such as code length, RS encoding, and Base64 encoding. Files such as txt, docx, xlsx, zip, and jpg were selected, and their corresponding file sizes were selected to calculate the time from encoding to completion, as shown in Table 3.

Table 3. Test Data File Types and Their Corresponding File Sizes

File Type	File Size
txt	2KB, 3KB, 4KB, 5KB, 6KB
docx	10KB, 11KB, 12KB, 13KB, 14KB
xlsx	10KB, 11KB, 12KB, 13KB, 14KB
zip	18KB, 28KB, 47KB, 93KB, 151KB
jpg	58KB, 156KB, 194KB, 262KB, 384KB

Fountain code encoding divides binary data into several data packets according to a certain length, and the length of the binary data in the packet is called the code length. When the amount of data is constant, the length of the code indicates a higher number of encoded packets, while the length of the code indicates a lower number of encoded packets. In fountain code coding, the code length is short, and the synthesized single sequence has smaller bases, making it easier to pass base screening. However, the large number of coding packets leads to a higher degree distribution function, which will affect the total coding time; If the code length is longer and the degree distribution function is smaller, the number of encoding packets will be smaller. However, if a single sequence is synthesized with longer bases, the pass rate of base screening will be lower, which will affect the total encoding time. On the basis of fountain code coding, Base64 code is added. According to the characteristics of Base64 coding and the rules of base conversion, 42 binary data will be converted into 24 amino acids. To ensure the effectiveness of coding, the code length needs to be set to an integer multiple of 42. Considering the nature of synthetic DNA not exceeding 200 bases [23-25], the code length is set to 126, 168, and 210 respectively. After conversion, 8 base address codes and 8 base RS codes are added 40 base primers were used, and the final number of synthesized DNA bases was 128, 152, and 176, respectively.

This article will be divided into four parts, using five types of test data files and their corresponding sizes to calculate the time from binary data conversion to encoding completion, and discussing the impact of RS code and Base64 code on fountain code encoding methods, as well as the impact of RS code and Base64 code on FCBC hybrid encoding methods.

1) Comparison between RS code and RS code+fountain code

Compare the fountain code with the fountain code+RS code and explore the impact of RS code on the encoding time of the fountain code. Fountain code divides data into several groups according to code length, generates a random degree distribution function based on the number of groups, selects a degree distribution function during encoding, and selects the same number of encoding packets for XOR operation to convert to bases. If the GC content of bases meets the requirements of biochemical experiments, the screening is passed, the bases are retained, and if not passed, they are discarded and re encoded until all data are encoded. The encoding time is counted.

The fountain code+RS code is based on the fountain code coding. After the base screening is passed, the RS code is encoded, and the generated RS code is converted into a base and added to the base sequence. After all data is encoded, count the encoding time.

2) Comparison between Fountain Code and Fountain Code+Base64 Code

Compare the fountain code with the fountain code+Base64 code and explore the impact of Base64 code on the encoding time of the fountain code. The fountain code encoding is the same as before. The fountain code+Base64 code encoding method uses Base64 encoding on the basis of the fountain code encoding, and groups the Base64 code words according to 7 bits. The first, third, fifth, and seventh code words are converted into 6-bit binary data, and the second, fourth, and sixth code words are converted into 8-bit binary data according to the balanced code encoding method. After combination, they are converted into bases, and then the bases are screened and judged. By filtering, the data is retained, otherwise it is re encoded until all codes are encoded, and the total coding time is counted.

3) Comparison of Fountain Code+Base64 Code and FCBC Hybrid Encoding Methods

Compare the fountain code+Base64 code and FCBC mixed encoding methods, and calculate the encoding time separately. The fountain code+Base64 coding method is the same as before, while the FCBC hybrid coding method performs RS coding on the basis of fountain code+Base64 coding, and converts the RS code into a base and adds it to the base sequence. After all data encoding is completed, count the total encoding time.

4) Comparison of Fountain Code+RS Code and FCBC Hybrid Encoding Methods

Compare the fountain code+RS code and FCBC mixed coding methods, and calculate the coding time separately. The fountain code+RS code encoding method is the same as before, while the FCBC hybrid encoding method is based on this. After the fountain code generates binary data, Base64 encoding is performed. Base64 codewords are grouped according to 7 bits, and the 1st, 3rd, 5th, and 7th codewords are converted into 6-bit binary data. The 2nd, 4th, and 6th codewords are converted into 8-bit binary data according to the balanced code encoding method, combined and converted into bases, and then the bases are screened and judged. If filtering is passed, the data will be retained and RS encoding will be performed. If filtering is not passed, the data will be discarded and re encoded, and the total encoding time will be counted.

III. Experimental Results and Analysis

A. Comparison of encoding time between RS code and fountain code+RS code

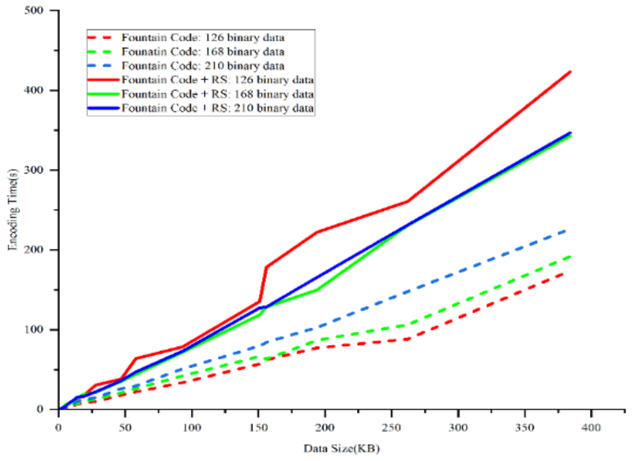

Figure 5 Comparison chart of fountain code and fountain code+RS code encoding time

From Figure 5, it can be seen that under the same code length and encoded data conditions, the fountain code+RS code encoding consumes longer time than the fountain code, and RS code occupies most of the time. As the file size gradually increases from 2KB-384KB, the RS code encoding time also gradually increases.

From the perspective of code length, for a txt file with encoded data of 2KB, the code length is 126, and RS code accounts for 60% of the total time consumption, which consumes 147% more time; The code length is 168, and the RS code accounts for 50% of the total time, which consumes 99% more time; When the code length is 210, the RS code takes up 38% of the total time and consumes an additional 60% of the encoding time.

For a JPG file with a data size of 384KB and a code length of 126, the RS code takes up 60% of the total time and consumes an additional 144% of the time; The code length is 168, and the RS code accounts for 45% of the total time, which consumes an additional 80% of the time; The code length is 210, and RS code encoding accounts for 35% of the total time, which consumes an additional 53% of the encoding time.

Comparing the encoding times of different code lengths, when encoding a 384KB jpg big data file, the total encoding time for code length 126 is 423.16s, the total encoding time for code length 168 is 342.76s, and the total encoding time for code length 210 is 346.72s. Code length 168 is relatively good, and when facing big data files, the advantage of code length 168 is more obvious..

B. Comparison of encoding time between fountain code and fountain code+Base64 code

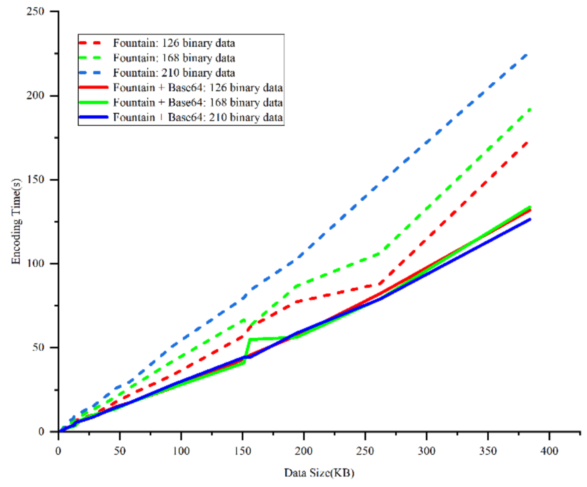

Figure 6 Comparison of encoding time between fountain code and fountain code+Base64 code

From Figure 6, it can be seen that under the same code length and encoded data conditions, the encoding time of fountain code+Base64 code is shorter than that of fountain code. As the file size gradually increases from 2KB-384KB, Base64 code can reduce more encoding time and achieve more significant performance.

From the perspective of code length, when the data is a 2KB txt file and the code length is 126, Base64 encoding can reduce encoding time by 40%; A code length of 168, Base64 encoding can reduce encoding time by 45%; With a code length of 210, Base64 encoding can reduce encoding time by 32%.

When the data is a 384KB jpg file with a code length of 126, Base64 can reduce encoding time by 25%; A code length of 168, Base64 encoding can reduce encoding time by 30%; With a code length of 210, Base64 encoding can reduce encoding time by 44%.

Comparing the encoding times of different code lengths, when encoding a 384KB jpg big data file, the total encoding time of code length 126 is 131.87s, code length 168 is 133.80s, and code length 210 is 126.34s. All three code lengths can reduce encoding time, and the encoding time is roughly the same, with excellent performance.

C. Comparison of encoding time between fountain code+Base64 code and FCBC hybrid encoding method

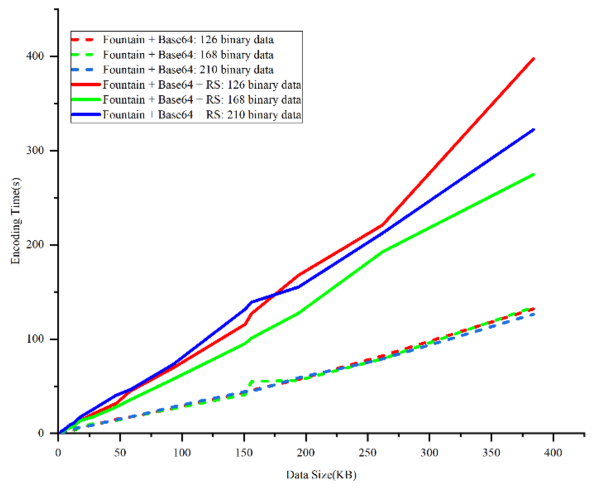

Figure 7 Comparison of fountain code+Base64 code and FCBC mixed encoding method

From Figure 7, it can be seen that under the same code length and encoded data conditions, the fountain code+Base64 code consumes less time than FCBC, and the difference becomes more and more significant with the increase of data volume. As the file size gradually increases from 2KB-384KB, the RS code encoding time gradually increases.

From the perspective of code length, when the encoded data is a 2KB txt file with a code length of 126, the RS code takes up 66% of the total time, which consumes an additional 197% of the time; When the code length is 168, the RS code accounts for 71% of the total time consumption and consumes 249% more time; When the code length is 210, the RS code accounts for 63% of the total time consumption and consumes an additional 167% of the encoding time.

When the data is a 384KB jpg file, the code length is 126, and the RS code takes up 67% of the total time, which consumes an additional 202% of the time; When the code length is 168, the RS code accounts for 51% of the total time consumption and consumes 105% more time; When the code length is 210, the RS code takes up 61% of the total time and consumes an additional 155% of the encoding time.

Comparing the encoding times of different code lengths, when encoding a 384KB jpg big data file, the total encoding time of code length 126 is 397.65s, code length 168 is 274.64s, and code length 210 is 322.16s. A code length of 168 is a relatively good encoding method, and its advantages are more obvious when facing big data files.

D. Comparison of encoding time between fountain code+RS code and FCBC hybrid encoding method

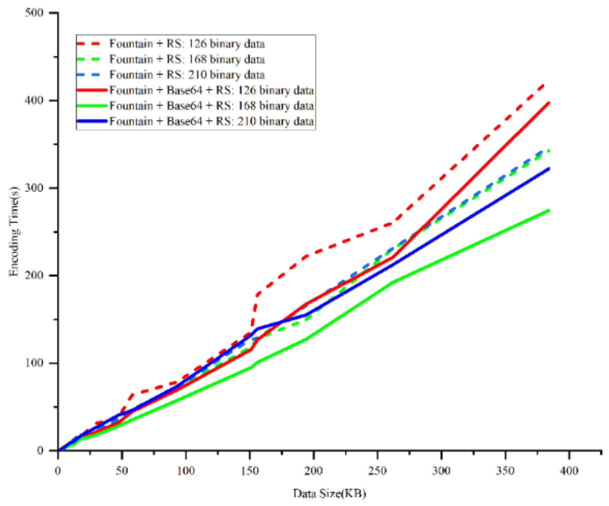

Figure 8 Comparison of fountain code+RS code and FCBC mixed coding method

From Figure 8, it can be seen that under the same code length and encoded data conditions, the FCBC hybrid encoding method model consumes less time than fountain codes and RS codes. As the file size gradually increases from 2KB-384KB, Base64 code can reduce more encoding time, and the difference becomes more significant as the data volume increases.

From the perspective of code length, when encoding a 2KB txt file with a code length of 126, Base64 encoding can reduce encoding time by 28%; When the code length is 168, Base64 encoding can reduce encoding time by 11%; With a code length of 210, Base64 encoding can reduce encoding time by 6%.

When the data is a 384KB jpg file with a code length of 126, Base64 encoding can reduce encoding time by 5%; The code length is 168, and Base64 encoding can reduce encoding time by 20%. With a code length of 210, Base64 encoding can reduce encoding time by 7%.

Comparing the encoding times of different code lengths, when encoding a 384KB jpg big data file, the total encoding time of code length 126 is 397.65s, code length 168 is 274.64s, and code length 210 is 322.16s. Various code lengths can reduce encoding time, and when the code length is 168, the encoding time is lower, making it a better code length choice.

IV. Discussion

A code length of 126 is a better encoding method for using only fountain codes. The fountain code will screen the encoded bases, and if it meets the requirements of biochemical experiments, it will be retained. If it does not meet the requirements, it will be discarded. Using a lower code length helps the bases pass the screening. Although there are more coding packets, the overall consumption time is reduced.

Adding RS code to the fountain code will increase the encoding time regardless of the code length used, and as the amount of data increases, time consumption increases. On the basis of generating bases, the RS code encodes the bases and concatenates the generated codewords after the bases to ensure the correctness of the base sequence. If a shorter code length is used, although a single base RS encoding consumes less time, an increase in the number of encoding packets will lead to an overall increase in time consumption; If a longer code length is used, the number of encoding packets will be smaller, but the encoding time for a single base RS will increase, resulting in an overall increase in time consumption. Comparing code lengths 126, 168, and 210, when the code length is 168, the overall encoding time is relatively good. Therefore, for the fountain code+RS code encoding method, a medium code length can ensure that the number of single base sequences is not too many and the total number of encoding packets is appropriate, thus achieving better performance.

Adding Base64 code to the fountain code will reduce encoding time regardless of the code length used. Base64 code will generate binary data encoding for fountain codes, resulting in a GC content of around 50% for the generated bases and reducing the probability of encoding homopolymers, thereby improving the pass rate of fountain code base screening and reducing the total encoding time. The use of different types of code lengths can effectively control encoding time, and the difference is not significant, indicating that the encoding methods of Base64 code and fountain code are not affected by code length. Choosing lower code length encoding results in less unit encoding time but more encoding packets; Choosing a longer code length encoding results in a larger unit of encoding time but a smaller number of encoding packets, with a smaller difference in total time consumption. Under different code length conditions, it meets good performance, indicating that Base64 code is widely applicable and suitable for practical use.

For the FCBC hybrid encoding method, regardless of the code length, the RS code will increase the encoding time of the hybrid encoding method, and the consumption time will rapidly increase with the increase of data volume. Although Base64 codes can reduce encoding time, adding codewords to RS codes after generating bases will increase time consumption on top of the initial encoding time. When choosing a longer code length, the number of encoding packets is lower, but a longer single base sequence increases the RS code encoding time, which will lead to an increase in the total encoding time; When choosing a lower code length, the single base sequence is shorter, and the RS code encoding time decreases. However, the number of encoding packets increases, resulting in an increase in total time. For the FCBC hybrid coding method, the encoding time of various types of codes is similar, but after adding RS codes, a code length of 168 is a better encoding choice, indicating that RS codes will affect the encoding model time. If the code length is too long or too short, it will lead to an overall increase in encoding time. Therefore, it is necessary to choose an appropriate code length encoding based on the actual situation.

For the FCBC hybrid encoding method, regardless of the code length, Base64 code will reduce the time of the hybrid encoding model, and the difference becomes more and more significant with the increase of data volume. The use of Base64 code can effectively control the GC content of bases, reduce the probability of homopolymer generation, and make the generated bases easier to pass screening, reducing coding time. Therefore, Base64 code can effectively reduce mixed coding time and improve computational efficiency. Although Base64 codes can reduce encoding time, when the code length is shorter to 126, the base number of a single sequence decreases, and the RS encoding time decreases. However, the number of encoding packets increases, resulting in an overall higher encoding time; When the code length is 210, although the number of encoding packets decreases, there are more individual bases, resulting in a higher overall time. Therefore, a code length of 168 is a relatively good encoding code length and can be used for big data encoding.

The FCBC hybrid encoding method is suitable for DNA storage, which can convert different types of files into binary data, encode them into bases in a short time and ensure accuracy, but there are still certain limitations. The FCBC hybrid encoding method adopts the four base conversion rule, and the theoretical maximum information storage density is 2 bits/character. For big data storage, more bases are needed to store big data information, which will consume more resources. Therefore, it is necessary to increase the information storage density and increase the amount of binary data stored per unit base. On the basis of the FCBC hybrid coding method, introducing degenerate base coding can improve information storage density. Degenerate base coding is the addition of additional coding bases on the basis of traditional DNA bases, representing combinations of ordinary bases and storing more data information. For example, in the sequence "GMA", "M" represents a combination of "G" and "T", thus there are two types of base variants: "GGA" and "GTA". In theory, the storage density of degenerate base encoded information in base 10 is 3.2 bits/character, while in base 15 it is 3.9 bits/character, both of which are higher than the theoretical information storage density in base four. Combining FCBC hybrid encoding method with degenerate base encoding helps to improve information storage density, reduce computational resource consumption and DNA synthesis costs, and is suitable for future big data storage needs.

V. Conclusion

Faced with rapidly growing data, DNA has stable properties and high reliability, making it an ideal storage medium. However, there still exist problems such as low storage density of DNA information, high resource consumption for encoding, and high requirements for synthetic biochemical experiments. The FCBC hybrid encoding method can ensure that the bases comply with biochemical experimental limitations, accelerate the fountain code base screening process, save computational resources, and effectively improve DNA information storage density, breaking through traditional information storage density limitations. The use of more efficient, convenient, and high-density DNA storage encoding methods will help promote the development of DNA storage, enabling faster application of DNA storage in practical applications and making it the next generation of storage media.

Funding

This work was supported by National Natural Science Foundation of China（62201141）and the National key research and development project（2020YFA0712104）.

References

[1] ZHANG Da-lu, GE Qi, FENG Yi-bo, et al. (2022) Comparison and Analysis on Scientific Research Programs on DNA Data Storage. 42(6): 116-129.

[2] PANDA D, MOLLA K A, BAIG M J, et al. (2018) DNA as a digital information storage device: hope or hype? Biotech , 8(5):239-247.

[3] YANG Yang, FAN Chun-hai. (2021) The current advance in artificial chromosome based DNA information storage. Synthetic Biology Journal, 2(3): 305-308.

[4] EXTANCE A. (2016) How DNA could store all the world's data. Nature, 537: 22-24.

[5] Zan Xiang-zhen, YAO Xiang-yu, XU Peng, et al. (2021) A survey on error correcting algorithms in DNA storage. Journal of Guangzhou University(Natural Science Edition), 20(2): 13-22.

[6] ZHOU Ting-yao, LUO Yuan, JIANG Xing-yu. (2021) DNA data storage: preservation approach and data encryption. Synthetic Biology Journal, 2(3): 371-383.

[7] CHEN Da-ming, ZHANG Xue-bo, LIU Xiao, et al. (2021) A global patent analysis: trends in DNA synthesis and information storage. Synthetic Biology Journal, 2(3): 399-411.

[8] Jeong J, Park S. J, Kim J. W, et al. (2021) Cooperative sequence clustering and decoding for DNA storage system with fountaincodes. Bioinformatics, 37(19): 3136-3143.

[9] Schwarz P. M and Freisleben B. (2021) NOREC4DNA: using near-optimal rateless erasure codes for DNA storage. BMC Bioinformatics, 22:406.

[10] Anavy L, Vaknin I, Atar O, et al. (2019) Data storage in DNA with fewer synthesis cycles using composite DNA letters. nature biotechnology, 10(37): 1229-1236.

[11] He X and Cai K. (2021) Basis-Finding Algorithm for Decoding Fountain Codes for DNA-Based Data Storage. Computer Science-Information Theory, 1-13.

[12] Andreas F. M, Neelesh B. M, Jonathan S. Y. (2007) Performance of Fountain Codes in Collaborative Relay Networks. IEEE TRANSACTIONS ON WIRELESS COMMUNICA TIONS, 6(11) : 4108-4119.

[13] WANG Shi-wei. (2014) DNA Storage with Error Correction Mechanism. Chang Sha: National University of Defense Technology, Chang Sha.

[14] CAIRE G, SHAMAI S, SHOKROLLAHI A, et al. (2016) Universal variable-length data compression of binary sources using fountain codes. Information Theory Workshop. IEEE, 123-128.

[15] ERLICH S L, ZIELINSKI D. (2017) DNA Fountain enables a robust and efficient storage architecture. Science, 355(6328) : 950-954.

[16] ZHANG Shu-fang, PENG Kang, SONG Xiang-ming, et al. (2019) Research Progress on DNA Data Storage Technology. COMPUTER SCIENCE, 46(6): 21-28.

[17] ZHANG Shufang, PENG Kang. (2020) DNA Information Storage Technology Based on Raptor Code. Laser and Optoelectronics Progress, 57(15): 1-7.

[18] PENG Yuan, SHI Xiao-long. (2020) Research on DNA storage technology based on chaotic block coding. Journal of Guangzhou University (Natural Science Edition), 19(5): 25-28.

[19] SHI Chun-hong. (2020) Research on Information Steganography Based on Base64 Coding. China Computer and Communication, 1: 118-119.

[20] Yi Zhang, Linlin Kong, Fei Wang, et al. (2020) Information stored in nanoscale: Encoding data in a single DNA strand with Base64. Nano Today 33: 100871.

[21] GRASS R N, HECKEL R, PUDDU M, et al. (2015) Robust chemical preservation of digital information on DNA in silica with error-correcting codes. Angewandte Chemie International Ed in English, 54(8): 2552-2555.

[22] BLAWAT M, GAEDKE K. (2016) Forward Error Correction for DNA Data Storage. Procedia Computer Science, 80(5): 1011-1022.

[23] WATSON J D, CRICK F H. (1953) Molecular structure of nucleic ac-ids: a structure for deoxyribose nucleic acid. Nature, 248(4): 623-624.

[24] Wu J. Q, Zhang Y. F, Wang B, et al. (2022) Enhancing Physical and Thermodynamic Properties of DNA Storage Sets With End-Constraint. IEEE Transactions on Nanobioscience, 21(2): 184-193.

[25] Hannah F, Marius W, Georges H, et al. (2022) Fractal construction of constrained code words for DNA storage systems. Nucleic Acids Research, 50(5): e30.