# Robust data storage in DNA by de Bruijn graph-based decoding

Li-Fu Song[1,2], Feng Geng[3], Zi-Yi Gong[1,2], Bing-Zhi Li[1,2*], Ying-Jin Yuan[1,2*]

[1] Frontier Science Center for Synthetic Biology and Key Laboratory of Systems Bioengineering (Ministry of Education), Tianjin University, Tianjin 300072, China

[2] School of Chemical Engineering and Technology, Tianjin University, Tianjin 300072, China

[3] College of Pharmacy, Binzhou Medical University, Yantai 264003, China

*Corresponding authors: bzli@tju.edu.cn,yjyuan@tju.edu.cn

## Abstract

DNA data storage is a rapidly developing technology with great potential due to its high density, long-term stability, and low maintenance cost. The major technical challenges include various errors such as the indels, rearrangements, and strand breaks that frequently arise during DNA synthesis, amplification, sequencing, and preservation. Here, we report the development of a de Bruijn graph-based, greedy path search algorithm (DBG-GPS) aiming to handle such issues. We demonstrate that accurate data recovery could be achieved from

deep error-prone PCR products. The robustness of this method was further verified with one hundred parallel data retrievals from PCR products containing numerous rearrangements introduced by intended unspecific amplifications. With DBG-GPS, we successfully decoded the original information even after treating the DNA at 70°C for 65 days. DBG-GPS shows linear decoding complexity and is faster than the multiple alignment-based methods by two orders of magnitude, suited for large-scale data storage.

## Keywords

Synthetic Biology, DNA data retrieval, DNA codec system, de Bruijn graph, Fountain codes

## Introduction

DNA is the natural solution for the preservation of genetic information of all life forms on earth. Recently, million years old genomic DNA of mammoths was successfully decoded revealing its high stability as an information carrier under frozen conditions[1]. Owing to its stability, high density, and low maintenance cost revealed by recent studies, DNA has been considered an ideal storage medium to meet the challenge of data explosion in the future[2–9]. Frequently occurring errors in DNA synthesis, amplification, sequencing, and preservation, however, challenge the reliability of data storage in DNA. Efforts to solve this problem led to the implementation of a codec system which requires two layers of error correction (EC) codes, the outer layer codes handling the strand dropouts and the inner layer codes dealing with intramolecular errors, ensuring

accurate information readouts[10–13]. Strand dropouts are erasures and therefore they can be well solved by erasure codes, *e.g.* Fountain, Reed-Solomon (RS) codes[10–12]. By contrast, the performance of existing inner layer codec systems, *e.g.* multiple alignment-based methods, is less satisfactory as they are not efficient in correcting indel errors, and in particular, not capable of dealing with rearrangements and strand breaks. Highly robust inner codec systems remain to be explored.

Base-shift arises from indel is difficult to handle with traditional EC codes, which detect indels but cannot position them. A two-dimensional coding strategy with traditional EC codes can transform the indel errors into substitutions. In this strategy, a single indel in one dimension will lead to a series of 'substitutions' in another dimension (**Fig. S1**). Thus, this strategy only works with low indel rates. Increasing indels will lead to an overwhelming number of "substitutions" exhausting the column EC codes[8,10]. Recently developed hash encoded, decoded by greedy exhaustive search (HEDGES) codes can be used to correct higher frequency of indels[14]. Based on the simulation, HEDGES codes require 40% redundancy codes for correction of 3% indel errors, which would significantly increase the cost of data writing. Strand rearrangements caused by unspecific amplifications are frequently encountered and could be a potential threat to data integrity especially when massive multiple retrievals from a large oligo pool are required. Currently, people optimize PCR conditions to avoid rearrangements to occur and use gel purification to physically exclude rearranged strands from the data pool. None of the existing methods can be used to correct rearrangements. DNAs are subjected to hydrolysis and degradation[12,13] under certain conditions and in

long-term storage, highlighting the importance of data stability studies on influences of harsh conditions, such as high temperature or UV exposure, and methods of DNA protection[8,14]. New coding methods that can tolerate fragmented DNA strands are nevertheless critical for enhancing the robustness of DNA data storage.

As the synthesis of long DNA is currently impractical, data storage in DNA relies on a large number of fragments. Consequently, data decoding involves the assembly of information pieces carried by individual DNA fragments, a process similar to genome assembly from shotgun sequencing. De Bruijn graph theory uses *k*-mer components to represent sequences. The analogy of genome assembly to data decoding and the success of the application of De Bruijn graph theory to genome assembly[20,21] prompted us to investigate its potentials in DNA data storage.

In this study, we report a strand reconstruction algorithm, de Bruijn Graph-based Greedy Path Searching (DBG-GPS) that allows robust DNA data storage. This algorithm handles indels, rearrangements, and strand breaks by strand reconstruction from multiple, error-rich strand copies, resulting in accurate and rapid data retrievals. Using this method, an up to 10% high indel rate, intended rearrangements and forced DNA strand breaks were shown to be efficiently corrected. DBG-GPS is faster than the clustering and multiple alignment (CL-MA) based methods in decoding by two orders of magnitudes. The linear decoding complexity revealed by simulation suggests that DBG-GPS could be adapted to large-scale data storage.

# Results

## The great potentials of de Bruijn graph-based decoding

To inspect the potentials of de Bruijn graph theory in DNA data storage, we first analyzed effects of representative substitutions, indels, and strand breaks in de Bruijn graph as shown in **Fig. 1a**. The simple de Bruijn graph on the right was constructed from the four sequence copies of “AACGGCGGCTGT” on the left with a *k*-mer size of four (**Fig. 1a**-**i**). The sequence of “AACGGCGGCTGT” is the path of “*a*→*b*→*c*→*d*→*e*→*c*→*f*→*g*→*h*” in the constructed graph. A substitution of “G” to “A” in sequence #2 will lead to a branch path, as shown in red color, and decreased coverages of the correct nodes (**Fig. 1a-ii**). With traditional EC codes, indels are much more difficult to handle than substitutions due to the base shift issue. The representative base indels in de Bruijn graph, however, are highly similar to substitutions (**Fig. 1a-iii**). Strand breaks also cause serious problems for traditional EC codes. With de Bruijn graph, the strand breaks only reduce the coverages of the correct nodes, without changes the graph structure. A greedy path search process, in combination with a path recognition mechanism, could decode the correct strand. In this way, the errors are transformed into a path search and selection problem by de Bruijn graph theory. With this mechanism, as long as the correct path, e.*g.* “*a*→*b*→*c*→*d*→*e*→*c*→*f*→*g*→*h*” in **Fig. 1a-i**, is preserved, the corresponding DNA strand can be decoded theoretically. Thus, the robustness of the correct path determines the potentials of the graph theory-based decoding.

Clearly, the sequence error rates and copy number together determine the robustness of a strand path in de Bruijn graph. To further investigate how the sequence copies and error rates affect path robustness, we simulated DNA sequences with different error rates. Variant numbers of the simulated error-containing sequences were utilized for de Bruijn graph construction and path integrality analysis (**Fig. 1b, c**). The integrality of the correct path was tested by checking the existence of all *k*-mers in this path in the constructed graph. The rate of the sequences with a complete path in the graph was defined as 'maximal strand decoding rate' ($\boldsymbol{S_m}$), which stands for the maximal strand decoding rate achievable by an actual de Bruijn graph-based decoder. Sequences with different types of errors were separately simulated. Twenty copies of sequences in a length of 320 bp containing different types and rates of errors were estimated for $\boldsymbol{S_m}$ values. As shown in **Fig. 1b**, the $\boldsymbol{S_m}$ values as expected are inversely related to the error rates but are similar regardless of the error types at the same error rates. Importantly, with merely twenty sequence copies, the sequence path in de Bruijn graph shows high robustness with an error rate of 5%, a common error level in reported DNA data storage studies[11,12,15]. In practice, emerged errors are mixtures of different types. Therefore, we decided to investigate the sequence path robustness by simulations with sequences containing mixed errors. The simulations were performed with variable total error rates from 1% to 10%, among which the errors are consisted of substitutions (50%), insertions, and deletions (25% each), and variable strand copies from 5 to 100. As shown in **Fig. 1c**, even with a high error rate of 10%, $\boldsymbol{S_m} > 0.99$ can still be obtained with sequence copies of 60 only (red dashed line), revealing the great potentials of de Bruijn graph-based decoding.

## Development of DBG-GPS for efficient strand reconstruction

The decoding process requires three basic steps, de Bruijn Graph construction, greedy path search, and path recognition. The de Bruijn graph size is dependent on the total number of correct *k*-mers plus noise *k*-mers. The number of correct *k*-mers is fixed for the storage of any given volume of information. The number of noise *k*-mers becomes a determinant with a larger number increasing de Bruijn graph size and possible paths, leading to longer path search time. Hence, we propose to include an additional step to exclude noise *k*-mers to reduce search time, illustrated in **Fig. 2a**.

### Elimination of noise *k*-mers decreases the greedy path search complexity

With simulations, about thirty times enlargement of the de Bruijn graph size was observed when 4% random errors were introduced into 10,000 DNA strands with 100 sequence copies each (**Fig. S2a**). To quantify the noise *k*-mers in a de Bruijn graph, we introduce an indicator $K_n$, which is the ratio of noise *k*-mers and correct *k*-mers in a de Bruijn graph. Clearly, more strand copies and errors will introduce more noise *k*-mers to the de Bruijn graph, resulting in a larger $K_n$ value. Monte Carlo simulations show a quick increase of $K_n$ by increasing either the error rate or the sequence copies (**Fig. S2a**). The number of branch paths and decoding time increase exponentially with error rates or strand copy numbers (**Fig. S2b**).

DNA data operations introduce a large number of errors. The occurrence of each error is however a low probability event. Therefore, the coverage of noise

*k*-mers is generally much lower than that of the correct ones. Elimination of the *k*-mers with only one occurrence significantly reduced branch paths not affecting the integrity of correct paths (**Fig. S2b**). We next attempted to mathematically describe the validity of using coverage as an indicator to recognize noise *k*-mers. Monte Carlo simulations were performed with 10,000 DNA strands, in which a Poisson model was employed for the simulation of the copy number distribution of each strand. The expectation value (λ) of strand copies was pre-set to be 50 and the error rate being 4%. The resulting coverage distribution of noise *k*-mers and correct *k*-mers were further analyzed. A coverage cut-off of 8 was found effective in the removal of most noise *k*-mers (**Fig. S2c**).

**Path recognition by CRC codes and both-way search strategy**

Noise *k*-mers cannot be completely removed solely by the elimination of low coverage *k*-mers. Branch paths still exist due to the remaining noise *k*-mers. A method to recognize the correct path is thus required. Traditional EC codes including cyclic redundancy check (CRC), RS, and hash codes are choices for this purpose. We failed to find a proper hash function that can generate short keywords. RS codes, optimized for correcting erasures, are not suited for path recognition, which does not involve erasures. CRC codes were chosen because of the efficiency of these codes in error detection instead of error correction. The effectiveness of CRC codes in path recognition was well demonstrated by intensive (one million) Monte Carlo simulations.

To further reduce the greedy path search time, we proposed a both-way search strategy that allows simultaneous search from both ends of DNA

strands. In this strategy, an additional ‘anchor’ element was designed for each DNA strand. The anchor corresponds to the index located on the other end of the DNA strand (**Fig. 2b)**, and is generated from the index by a specific function. A given index processed by the function generates a specific number, represented by the anchor sequence. For proof-of-concept, the pseudorandom function of Python is used for the generation of the anchors from the indexes as seeds. Pseudorandom function generates an identical ‘random’ value with the same seed. Thus, informed with indexes, the greedy path search engine calculates corresponding anchors, and carries out the both-way search strategy. The both-way search strategy significantly reduced the decoding time, with effects more obviously observed when sequence error rates are high. Up to 60% reduction was evident with an error rate of 10% and a copy number of 100, while decoding accuracy was well maintained (**Fig. S3**).

## Integration of DBG-GPS with outer erasure codes and performance test

DBG-GPS is designed as an inner decoding mechanism for the error-free reconstruction of DNA strands. It can be easily combined into the decoding process of outer erasure codes, *e.g.* fountain codes or RS codes, as shown by **Fig. 2c**. Python implementation of DBG-GPS using fountain codes as outer codes is available at https://switch-codes.coding.net/public/switch-codes/DNA-De-Bruijn-Decoding/git/files.

To systematically analyze the decoding accuracy of our Python implementation of DBG-GPS, Monte Carlo simulations of 10,000 DNA strands (each strand

with 60 copies) containing various error rates ranging from 1% to 10% were performed and the results were subjected to decoding using the Python implementation and even at 10% of error, decoding accuracy was still remarkably satisfactory with the overall $S_r$ higher than 96.7% (**Tab. S2**). We next tested the decoding speed of DBG-GPS in comparison with CL-MA based methods. To the best of our knowledge, the CL-MA based methods are the only methods that have been used to calculate the consensus sequence from multiple error-rich strand copies[12,17]. To compare the decoding speed of DBG-GPS with CL-MA based methods, DNA fountain codes were used to generate 10,000 DNA droplets and fifty sequence copies for each droplet. 4% errors (2% substitutions, 1% insertions and 1% deletions) were randomly introduced. A total of 500,000 error-rich sequences were then subjected to strand decoding by DBG-GPS and CL-MA respectively. Considering the complexity of the clustering process for the CL-MA-based decoding, we fed the Muscle 3.8.31[16] program with pre-clustered reads. For decoding speed comparison, estimation based on the previously reported clustering time in literature was used[14]. Since the multiple alignment process was performed by a standalone program (Muscle 3.8.31[16]), there will be 10,000 times of IO (input and output) with the file system during strand decoding by CL-MA. To estimate the IO time, one sequence of the DNA droplet was repetitively submitted to Muscle for multiple alignment analysis for 10,000 times. This process is iterated three times and the average time is subtracted from CL-MA based decoding time. The *k*-mer counting step of DBG-GPS was performed with JellyFish 2.3. As shown in **Tab. 1**, DBG-GPS shows 103 times faster than the CL-MA based methods. Importantly, in the range of data scale from 10 to 10,000 DNA droplets, DBG-GPS exhibited linear decoding complexity (**Fig.**

**S5**), suggesting that this method is suited for large-scale data storage.

Large-scale simulations were then used to assess the capacity of DBG-GPS with big data sets. An Ubuntu server 20.10 installation disk image (~ 1 GB) was used as the input file to generate DNA droplets. Each DNA droplet was set to carry 60 bytes of information with a total DNA length of 316bp, around the technical limits of current DNA synthesis technologies. Ten million DNA droplets, representing a data scale of 600 megabytes (MB) were simulated. With an expected copy number of fifty, error rate of 4% (approximately 2% substitutions, 1% insertions, 1% deletions), a 155GB sequence file was generated. Loading the *k*-mer counting results of Jellyfish on a computation workstation equipped with 640GB memories, the Python script encountered memory limitations. We had to reduce the DNA droplet number to five million, representing a data size around 300MB. Preliminary decoding simulations on this data scale revealed a problem of long decoding time. Detailed analysis showed that this is due to repeated presentations of same *k*-mers located in different strands. We then solved the problem using a combination of two strategies, with the first employing a filtering process during the production of DNA droplets to exclude seriously entangled DNA strands, and the second setting an upper boundary for path number in greedy path search. The greedy path search engine aborts search if the number of found paths exceeding this boundary. Five million DNA droplets were simulated three times independently. On average, 4,971,880 DNA droplets ($S_r$ = 99.44%) were recovered, proving the great capacity of DBG-GPS.

## Experimental verification of the robustness of DBG-GPS

To verify the robustness of DBG-GPS, a 314 KB zipped file, including two text files and four jpeg pictures, was encoded into 12,000 DNA strands, each in a length of 200 bp using fountain as outer codes. Each strand encodes thirty bytes of data, three bytes of CRC codes, four bytes of index codes, and four bytes of anchor codes (**Fig. 2b**). The overall strand redundancy was designed as 10.5% based on the expectation of more than 92% of the strands could be correctly decoded. A final concentration of 2 ng/μl designed DNA strands were synthesized by Twist Bioscience (**Fig. 3a**), and used as templates for PCR amplification and data retrievals. Different types of errors, including indels, substitutions, rearrangements, and strand breaks were intentionally introduced by three harsh experiments (**Fig. 3b**-**d**), followed by decoding to examine the robustness of DBG-GPS.

### Data retrievals with deep error-prone PCR products

A serial of six error-prone PCR (ePCR), with each reaction containing 30 thermal cycles, was performed using an error-prone PCR Kit to amplify the ssDNA library introducing large numbers of substitution errors, estimated 8 errors per 1,000 bp per amplification cycle base on the manufacturer's instruction. All six ePCR products, ePCR#1-6, were sequenced by Illumina technology. Due to unspecific amplifications, in addition to substitution errors, ePCR#2-6 products contain DNAs of different sizes, evident by a smear from 100 to 4,000 bp on agarose gel (**Fig. 4a**). As some of these DNA sequences were too long to be sequenced by Illumina platform, these PCR products were digested with DNase I to a target length of 200 bp. After digestion, all

fragments were collected for sequencing. The sequencing raw reads were subjected to DBG-GPS based decoding.

Regardless of rough amplification conditions expected to introduce a large number of errors, strikingly the $S_m$ values of all six samples were higher than 99% (**Fig. 4b**), with the $S_m$ values of ePCR#1-3 being 100%, indicating perfect reservation of all strand paths even after 90 cycles of ePCR. $S_r$ values in a range of 93.2-99.8% were obtained for the six samples, ensuring reliable data decoding by the outer fountain codes. Decrease of $S_r$ was observed with increased ePCR cycles. More PCR errors were expected to increase branch paths, raising the necessary cutoff for noise *k*-mers elimination. More correct *k*-mers were consequently discarded by mistake, resulting in a lower $S_r$ value.

To further test the robustness of DBG-GPS, sequencing data with low coverage ranging from 1 to 256 was obtained by random sub-sampling on the sequencing results of ePCR#1 and ePCR#2. The *k*-mer dropout rates ($K_d$), noise *k*-mer indicator ($K_n$) and strand recovery rate ($S_r$) were calculated and plotted against coverage values (**Fig. 4c**). As expected, with the same coverage, both $K_n$ and $K_d$ values of ePCR#2 are always higher than ePCR#1, consistent with accumulating errors in each additional round of ePCR. Upon increasing coverage, the $K_n$ values of both samples reflecting the numbers of noise *k*-mers increased quickly, while the $K_d$ values reflecting the dropouts of correct *k*-mers decreased quickly. As shown in **Fig. 4c**, the two samples can be reliably decoded at coverage as low as 16 and 24.

## Multiple data retrieval and accelerated aging test

Multiple data retrievals from storage media are required in practice. Thus, data robustness sustainable during multiple retrievals needs to be tested for our algorithm. Till now, only a limited number (<10) of multiple retrievals from high-quality PCR products have been reported for DNA data storage. To mimic the required multiple retrievals in practice and further demonstrate the robustness of our method, 100 times parallel data retrievals from PCR-amplified ssDNA library were carried out (**Fig. 3**), in which massive unspecific amplifications indicated by smears from 100 to 8,000 bp on agarose gel were observed (**Fig. 5a**). The $S_m$ values of all retrievals were still remarkably high in the range from 99.44% to 99.78% (**Fig. 5b**). The $S_r$ values ranged from 98.45% to 99.02%, ensuring accuracy in these data retrievals.

The CL-MA based codec systems highly rely on intact DNA strands for decoding, whereas DBG-GPS relies on short *k*-mers. Thus DBG-GPS likely tolerates DNA degradation. To test this hypothesis, accelerated aging experiments were performed as shown in **Fig. 3c**. The master pool was first PCR-amplified followed by incubation at 70°C for a prolonged period of time up to 65 days. Agilent 2100 Bioanalyzer was used to analyze the samples. A 14 days of incubation caused the complete disappearance of the 200bp peak prominent in the original sample, indicating that the integrity of the DNA strand has been severely damaged (**Fig. S6**). Such degree of degradation would have made it impossible to recover the encoded data using the previously reported codec systems. By contrast, DBG-GPS based decoding enabled high strand recovery rates, ensuring accurate data retrievals from all samples that

have been heat-treated for 7, 14, 30 and even 65 days. A high $S_r$ value of 98.18% was obtained with the sample treated at 70°C for 65 days (**Fig. 5c**). This $S_r$ value is only slightly lower than that of the control stored at -20°C (99.40%). Data retrievals from accelerated aged DNAs have been reported[10,18,19]. In these studies, the authors tested the effects of different methods for the preservation of data encoded DNA, among which embedment in silicon beads was shown to be the best allowing recovery of information after treatment at 70°C for 7 days. According to the authors, this was thermally equivalent to storing information on DNA at 9.4°C for 2000 years[10]. Based on this estimation, our DBG-GPS algorithm can be used to retrieve data from a DNA solution preserved at 9.4°C for 18,571 years.

## Discussion

Data robustness, density and low-cost are important parameters, determining the technical advancements of a storage medium. In general, higher data robustness requires more redundancy codes, which decrease information density and cost effectivity. Based on information theory, redundancy codes in the form of “data repetition” are unfavorable since production of additional data copies on traditional media is cost-ineffective and time-consuming. By contrast, data repetition in the form of DNA strand copies can be obtained inexpensively and speedily. The traditional definition of “coding efficiency” needs to be adapted to DNA data channel. With current technologies, producing a single copy of DNA data is not even doable. It has been reported that serial dilutions to reduce the copy numbers of information-bearing DNA strands till the average copy number below ten resulted in massive strand dropouts, making

reliable data retrieval impractical[20]. Even if a single copy of DNA data is obtainable, it may be damaged due to DNA degradation under certain conditions or during long-term storage. Taken together, the design for DNA storage should be based on the reality and necessity, the presence of more than one copies of DNA strands, and not too many copies (below 100) to maintain high information density. In other words, “multicopy” is crucial for the robustness of DNA data storage and should ideally be maintained in a reasonable range. A new theory to design EC codes taking this unique “multicopy” feature into consideration will help solving the main technical issue in DNA storage, the error-free decoding of DNA sequences in a completely different fashion.

To the best of our knowledge, CL-MA based methods are the only reported methods used for DNA decoding from error-containing multicopies. These methods are computation-intensive, and are not robust when dealing with strand rearrangements and breaks. In this study, we developed a strand reconstruction algorithm based on the de Bruijn graph theory and tested its robustness. We were able to accurately and rapidly decode DNA data from error-rich strand copies using this algorithm. The errors include indels, rearrangements and strand breaks. It is worth noting that the encoded information from a DNA solution that has been incubated at 70°C for 65 days can be accurately recovered with our algorithm. The degree of damage to DNA under this heating condition was estimated to be equivalent to that under 9.4°C for more than 18,000 years[10]. Such unprecedented robustness highlights the importance of our method and suggests the great potential of de Bruijn graph theory in DNA data channel. Our algorithm is presumably compatible with a

variety of DNA preservation, *e.g.* silicon beads[10], nanoparticles[18], and alkaline salts[19], to further enhance data stability.

The Python codes implemented in this study use a simple hash dictionary for the retrieval of *k*-mers. C++ implementation with a more advanced *k*-mer data structure[21,22] is expected to significantly reduce the memory cost, as well as decoding time. In data decoding from 50 sequence copies containing 4% errors, the performance of DBG-GPS is superior to that of the CL-MA based methods in speed by one hundred fold. A higher ratio of noise *k*-mers is expected to decrease decoding speed and accuracy with the DBG-GPS method. Future incorporation of EC codes to help identify and exclude noise *k*-mers prior to the greedy path search is expected to further improve the decoding speed and accuracy, which is particularly important in the extreme case of high error rate.

# Methods

### Library design, PCR and Sequencing

A DNA library of 12,000 oligonucleotides in length of 200 nt was synthesized by Twist Biosciences. Error-prone PCR was performed with Controlled Error-prone PCR Kit, TIANDZ, CAT#:160903-100. Thermo cycle parameters were 94°C for 3 mins, 94°C for 1min, 45°C for 1min, 72°C for 1min, 30 cycles. Six serial error-prone PCR were performed to introduce high rates of errors. The first round ePCR utilized 0.6 µl samples of the master pool as templates. The five following ePCR use 1 µl products of the previous round ePCR as templates. All PCR experiments were performed with the following primers: P1

5'-CCTGCAGAGTAGCATGTC-3', P2 5'-CGGATGCATCAGTGTCAG-3'. The PCR products were sequenced by Tianjin Novogene Sequencing Center & Clinical Lab. Sequencing libraries were generated with purified PCR products using Illumina TruSeq DNA PCR-Free Library Preparation Kit (Illumina, USA) following manufacturer's recommendations and index codes were added. For the products of the second-round error-prone PCR with very large DNA strands, DNase digestion is applied to break the long strands into small fragments around 200~300 bp. After digestion, all fragments are collected for further library construction using the protocol as just described. The library quality was assessed on the Qubit@ 2.0 Fluorometer (Thermo Scientific) and Agilent Bioanalyzer 2100 system. All the libraries were sequenced on Illumina HiSeq platform and 150 bp paired-end reads were generated. For preparation of the 100 PCR samples for parallel retrievals, we first performed one round of PCR amplification with Taq DNA polymerase with a total volume of 100 µl. Then, 100 parallel PCR amplification was performed using 1 µl of the first round PCR reaction mixtures without purification. The purification step is removed in purpose of enhancing unspecific amplification. The 100 PCR samples were then sequenced by Illumina sequencing platform. All the Taq DNA Polymerase based PCR amplifications, except the error-prone PCRs, used Vazyme 2X Rapid Taq Master Mix (CAT#: P222-03).

### ***k*-mers counting and elimination of low coverage *k*-mers**

The python implementation provided in this study used a simple hash dictionary structure for counting of k-mers, which is not memory efficient. Thus, although the scripts implemented with functions to read the common

nucleotide sequence files, *i.e.* FastQ or FastA, and count the *k*-mers accordingly, for large data sets, we strongly suggest to use a professional *k*-mer counting system, *e.g.* Jellyfish[21], which is much more memory efficient and supports multi-cores. To count *k*-mers by the python scripts, please call the following functions of DeBruijnGraph() object: for reading and counting of FastQ file, use DeBruijnGraph-openFastQ('file name'); for reading and counting of Fasta file, use DeBruijnGraph-openFasta('file name').

For elimination of low coverage *k*-mers, call function DeBruijnGraph-remove_low_cov_kmers(*n*), then all *k*-mers with coverages lower than *n* will be removed. For usage of Jellyfish for counting of *k*-mers and dumping of *k*-mers satisfying the criteria, please refer to the Jellyfish manual, which is available at (http://www.cbcb.umd.edu/software/jellyfish/). The dumped *k*-mers can be introduced to the DeBruijnGraph() object using DeBruijnGraph-openDump('file name') function for further decoding test.

**Setting of *k*-mer size**

Obviously, with a specific *k*-mer size, the data scale can be decoded is limited. For example, it is clear that only small amount of data can be decoded with an extremely short *k*-mer size of three. However, the mathematical answer to this question is yet unsolved. In de Bruijn Graph theory, for each *k*-mer, the front *k*-1 bases were used for positioning (*i.e.* indexing), and the one base at the terminal was used for extension of path, *i.e.* for encoding of data. Thus, we estimate the decoding capacity ($p$) of *k*-mers with a specific size of $k$ as follows:

$$p = 4^{k-1} \times 2bits$$

Where $4^{k-1}$ is the combination number of possible indexes with a length of $k-1$, 2 bits stands for the encoding capacity of one base at the tail. Based on this formula, the decoding capacities of different *k*-mer sizes were calculated and listed in **Tab. S3**. As shown in the table, the decoding capacity is estimated to be around 57.6 PB with a *k*-mer size of 30, and $6\times10^5$ EB for a *k*-mer size of 40, indicating a capable method for large-scale data decoding. Additionally, it should be noticed that, due to the repeated presentations of identical *k*-mers in different DNA strands, the actual data scale can be higher than this estimation theoretically.

**Details of greedy path search**

The greedy search process was illustrated in **Fig. S4**. The details are described as follows: ***Step 1,*** Select an index, *e.g.* 562451, and calculate the anchor value by the Python pseudorandom function using the index value as seed in case of both-way search mode. An anchor value of 47736 is obtained in the case of 562451. ***Step 2,*** Encode the index into DNA string based on the corresponding binary string, *i.e.* 562451 = 10001001010100010011= ATATATGGGATCAGAA. Similarly, the anchor of 47736 is encoded into DNA string of ATATATATGCGTTTGG in case of both way search. The greedy path search starts with ATATATGGGATCAGAA as an initial path. Similarly, the tail search start with ATATATATGCGTTTGG as an initial path in the case of both-way search. ***Step 3.*** Judgement step. This step is for the selection of an end with few path numbers for extension of paths. ***Step 4,*** Extension of paths on the end with fewer path numbers. ***Step 5,*** Judgment step. This step is to

judge if the path search has finished. ***Step 6***, This step is specifically required for both-way search mode. The head paths are fused with tail paths if the overlap sequences are identical. For one-way search, this step is not required. ***Step 7***, Encode the complete paths into binary strings. Perform parity checks and drop the wrong routes using the embedded CRC codes. For other EC codes applied, use the checking mechanism correspondingly. ***Step 8***, Judgement step. Check if there is only one path passed the CRC checking. If it is true, choose it as decoded strands, otherwise, mark a missing strand for this index. If multiple routes or no routes pass the CRC codes, the searching-based decoding fails. Only in the case of one route passed the CRC checking, the decoding succeeds. ***Step 9***, Judgement step. Judge if the search is finished. If not, iterate *step 1* to *step 9* until all indexes are searched for possible strands. If yes, finish the inner decoding process. This algorithm is implemented in the deBruijnGraph.py object by Python. The deBruijnGraph() object was implemented to search in both-way mode by default. Switch the value of DeBruijnGraph() ->dual_search to 'False' to active the one-way search mode.

**Error analysis in the concept of de Bruijn graph theory**

Although distribution profiles of error rate and copy number among different strands shape the DNA data quality, they cannot reflect the data quality straightforwardly. Additionally, the sequence error rate is complicated for accurate accounting. For better estimation of the qualities of DNA data and simplification of the error analysis process in the concept of de Bruijn graph theory, here we introduce two indicators, *i.e.* $K_d$ (in range of 0 to 1) and $K_n$

(in range of one to $4^k$), which together can well define the data quality. The $K_d$ stands for dropout rate of all correct *k*-mers and $K_n$ is the ratio of noise *k*-mers and correct *k*-mers. DNA data with a $K_d$ close to 1 and a $K_n$ close to 0 stands for high-quality data with a high data coverage rate and low noises. Other than error rate which is complicated for accounting, $K_d$ and $K_n$ can be easily calculated by the following formulas:

$$K_d = K_{lost}/K_{ori} \qquad 1$$

$$K_n = K_{all}/K_{corr} \qquad 2$$

In formula 1, $K_{ori}$ stands for the number of *k*-mers in the original encoded sequences and $K_{lost}$ stands for the number of *k*-mers that presented in the original encoded sequences but not presented in the sequencing results. In formula 2, $K_{all}$ stands for the number of *k*-mers in the sequencing results and $K_{corr}$ stands for the number of correct *k*-mers, *i.e. k*-mers that presented in the original encoded sequences. DNA data with a $K_d$ close to 1 and a $K_n$ close to 0 stands for a high-quality data with high data integrity and low noises. The maximal strand decoding rate $S_m$ is determined by $K_d$. With specific $K_d$, the actual strand decoding rate $S_r$ is affected by the value of $K_n$. The greedy path search becomes extremely slow when decoding DNA data with higher $K_n$ values, due to too many possible paths for each strand. The DBG-GPS will stop decoding if the path number exceeds a specific threshold. In this proof-of-concept study, we use 100 for fast decoding, 1,000 for high-quality decoding.

## The fountain codes

The outer fountain codes used in this study is modified from https://github.com/dbieber/fountaincode. DNAFountain and DNADroplet objects were implemented for the handling of DNA strings. The assignment mechanism of the random degrees and chunk indexes of the Droplet object was modified to support DBG-GPS based decoding. Luby transform is applied for the generation of random degrees. Specific index encoded in the DNA Droplet is used for the selection of a degree from a pre-generated degree table. This index is also used as seed for the generation of the random strand combinations. The decoding process of the original Glass object has been modified to improve its performance.

## Details of the simulation experiments

Simulations for investigation of the potentials of de Bruijn graph were performed with 10,000 DNA strands generated by the fountain codes with a strand length of 320 bp. For de Bruijn graph construction, a *k*-mer size of 21 was applied. The simulation studies on the both-way search strategy are performed with 10,000 DNA droplets. 100 sequence copies with random base errors in rates of 10%. For the large-scale simulations, an Ubuntu Server 20.10 live install disk image (Link provided in **Supplementary materials**) was used as input file for the generation of random DNA droplets. Each DNA droplet was set to encode 60 bytes of data and two bytes of CRC codes. The length of index and anchor are both set to four bytes. Primers of P1 5'-CCTGCAGAGTAGCATGTC-3' and P2 5'-CGGATGCATCAGTGTCAG-3' were added to the ends. This will result with a DNA droplet in length of 300 bp,

around the theoretical limits of current DNA synthesis technologies. To avoid entangled DNA strands, a filter process that always discards the DNA droplets contains a *k-1* mer that already presents more than four times was applied to the droplet generating stage. For counting of *k*-mers by Jellyfish 2.3, a *k*-mer size of 24, and *k*-mer coverage cut-off of 5.

**Data Availability**

All data files and raw sequencing data that support the findings of this study have been deposited in public cloud server. The links are provided in supplementary materials.

**Code availability**

All the original algorithms proposed in this study were implemented with Python and the source codes are available at https://switch-codes.coding.net/public/switch-codes/DNA-De-Bruijn-Decoding/git/files.

## Acknowledgements

This work was supported by National Natural Science Foundation of China under the grants 21621004 to Y.Y. , Seed Foundation of Tianjin University to L. S. and Natural Science Foundation of Shandong Province under the grants ZR2017LC006 to F.G.. The authors would like to thank Dr. Yan Zhang for his great efforts and constructive comments which have helped to improve this manuscript significantly. The authors also would like to thank Dr. Sheng Ye, Dr.

Hao Qi and Dr. Weigang Chen for very helpful suggestions.

## Competing interests

Li-Fu Song, Ying-Jin Yuan and Feng Geng have filed a patent covering the encoding structure and the de Bruijn graph based decoding process (CN 110190858 A).

## Author contributions

LFS and Y.Y. conceived the study. LFS proposed the idea, wrote the Python codes, designed the experiments, performed the simulations, and analyzed the data. FG helped with experiment design and data analysis. ZYG performed the error-prone PCR experiments. BZL and YJY edited the manuscript and supervised the whole work.

# References

1. van der Valk, T. *et al.* Million-year-old DNA sheds light on the genomic history of mammoths. *Nature* **591,** 265–269; 10.1038/s41586-021-03224-9 (2021).

2. Anavy, L., Vaknin, I., Atar, O., Amit, R. & Yakhini, Z. Data storage in DNA with fewer synthesis cycles using composite DNA letters. *Nature biotechnology*; 10.1038/s41587-019-0240-x (2019).

3. Koch, J. *et al.* A DNA-of-things storage architecture to create materials with

embedded memory. *Nature biotechnology* **38,** 39–43; 10.1038/s41587-019-0356-z (2020).

4. Zhirnov, V., Zadegan, R. M., Sandhu, G. S., Church, G. M. & Hughes, W. L. Nucleic acid memory. *Nature materials* **15,** 366–370; 10.1038/nmat4594 (2016).

5. Church, G. M., Gao, Y. & Kosuri, S. Next-generation digital information storage in DNA. *Science (New York, N.Y.)* **337,** 1628; 10.1126/science.1226355 (2012).

6. Ceze, L., Nivala, J. & Strauss, K. Molecular digital data storage using DNA. *Nature reviews. Genetics* **20,** 456–466; 10.1038/s41576-019-0125-3 (2019).

7. Ping, Z. *et al.* Carbon-based archiving: current progress and future prospects of DNA-based data storage. *GigaScience* **8**; 10.1093/gigascience/giz075 (2019).

8. Chen, W. *et al.* An artificial chromosome for data storage. *National Science Review*; 10.1093/nsr/nwab028 (2021).

9. Tabatabaei, S. K. *et al.* DNA punch cards for storing data on native DNA sequences via enzymatic nicking. *Nature communications* **11,** 1742; 10.1038/s41467-020-15588-z (2020).

10. Grass, R. N., Heckel, R., Puddu, M., Paunescu, D. & Stark, W. J. Robust chemical preservation of digital information on DNA in silica with

error-correcting codes. *Angewandte Chemie (International ed. in English)* **54,** 2552–2555; 10.1002/anie.201411378 (2015).

11. Erlich, Y. & Zielinski, D. DNA Fountain enables a robust and efficient storage architecture. *Science (New York, N.Y.)* **355,** 950–954; 10.1126/science.aaj2038 (2017).

12. Organick, L. *et al.* Random access in large-scale DNA data storage. *Nature biotechnology* **36,** 242–248; 10.1038/nbt.4079 (2018).

13. Gao, Y. *et al. Low-Bias Amplification for Robust DNA Data Readout* (2020).

14. Press, W. H., Hawkins, J. A., Jones, S. K., Schaub, J. M. & Finkelstein, I. J. HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints. *Proceedings of the National Academy of Sciences of the United States of America* **117,** 18489–18496; 10.1073/pnas.2004821117 (2020).

15. Heckel, R., Mikutis, G. & Grass, R. N. A Characterization of the DNA Data Storage Channel. *Scientific reports* **9,** 9663; 10.1038/s41598-019-45832-6 (2019).

16. Edgar, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research* **32,** 1792–1797; 10.1093/nar/gkh340 (2004).

17. Antkowiak, P. L. *et al.* Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction.

*Nature communications* **11,** 5345; 10.1038/s41467-020-19148-3 (2020).

18. Chen, W. D. *et al.* Combining Data Longevity with High Storage Capacity—Layer□by□Layer DNA Encapsulated in Magnetic Nanoparticles. *Adv. Funct. Mater.* **29,** 1901672; 10.1002/adfm.201901672 (2019).

19. Kohll, A. X. *et al.* Stabilizing synthetic DNA for long-term data storage with earth alkaline salts. *Chemical communications (Cambridge, England)* **56,** 3613–3616; 10.1039/d0cc00222d (2020).

20. Chen, Y.-J. *et al.* Quantifying molecular bias in DNA data storage. *Nature communications* **11,** 3264; 10.1038/s41467-020-16958-3 (2020).

21. Marçais, G. & Kingsford, C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics (Oxford, England)* **27,** 764–770; 10.1093/bioinformatics/btr011 (2011).

22. Pandey, P., Bender, M. A., Johnson, R., Patro, R. & Berger, B. Squeakr: an exact and approximate k-mer counting system. *Bioinformatics (Oxford, England)* **34,** 568–575; 10.1093/bioinformatics/btx636 (2018).

23. Cyrus Rashtchian *et al.* Clustering Billions of Reads for DNA Data Storage.

**Fig. 1 | The potentials of de Bruijn graph-based decoding. a,** Representations of base substitutions and indels in de Bruijn graph. ***i***, Simple de Bruijn graph constructed from four copies of "AACGGCGGCTGT" with a *k*-mer size of four. The circles stand for the *k*-mers and the numbers inside the circles are coverages of corresponding *k*-mers in all graphs. ***ii***, De Bruijn graph as consequence of one substitution occurring in sequence #2; ***iii***, De Bruijn graph as consequence of one base insertion on sequence #2 and one base deletion on sequence #3; ***iv***, De Bruijn graph involves four strand breaks. **b,** Estimated maximal strand decoding rates (***$S_m$***) of different types and rates of errors with 20 sequence copies; For clarity, the standard derivations are detailed in **Tab. S1**. **c**, ***$S_m$*** values estimated with different sequence copies and rates of mixture errors (50% substitutions, 25% insertions and 25% deletions).

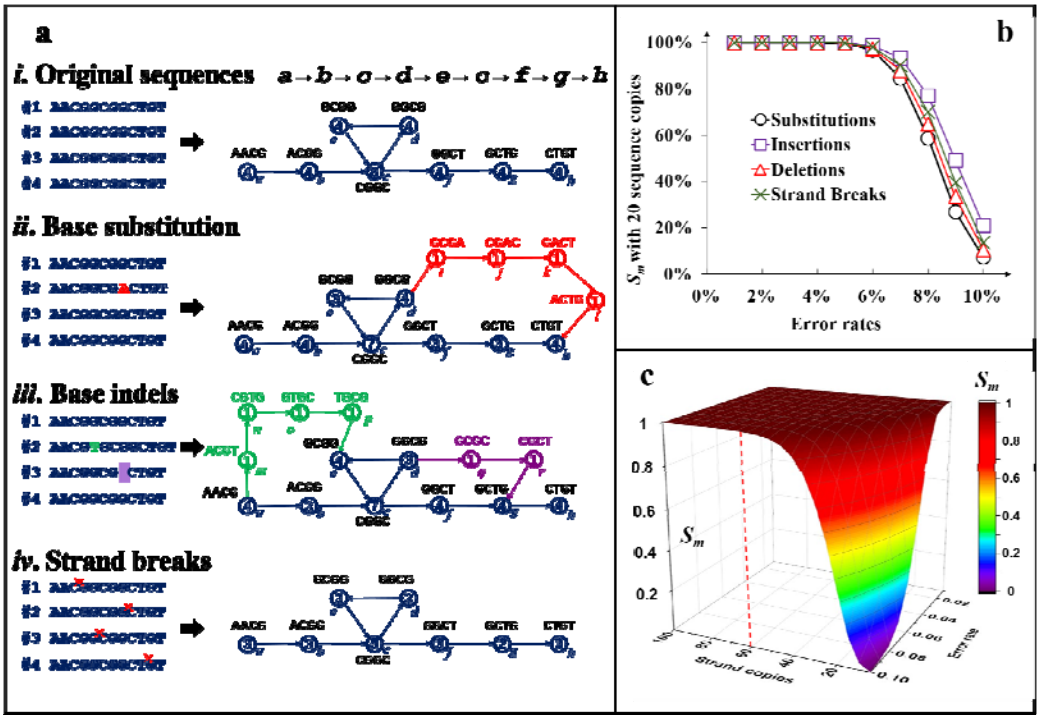


**Fig. 2 | Proposed four-step framework of DBG-GPS and its integration with outer erasure codes. a,** Proposed four-step framework of DBG-GPS. **b,** Proposed structure of DNA strands for DBG-GPS. There are four major parts including: A1 – The index/seed of the DNA strand; A2 – Data payload for encoding of data; A3 – Cyclic Redundancy Check codes (CRC) for selection of the correct path; A4 – Anchor cods which can be calculated by specific function based on the value of A1. A4, Anchor codes (optional) for both-way search; **c,** Integration of DBG-GPS with outer erasure codes. DBG-GPS is a full resolution for inner code decoding.

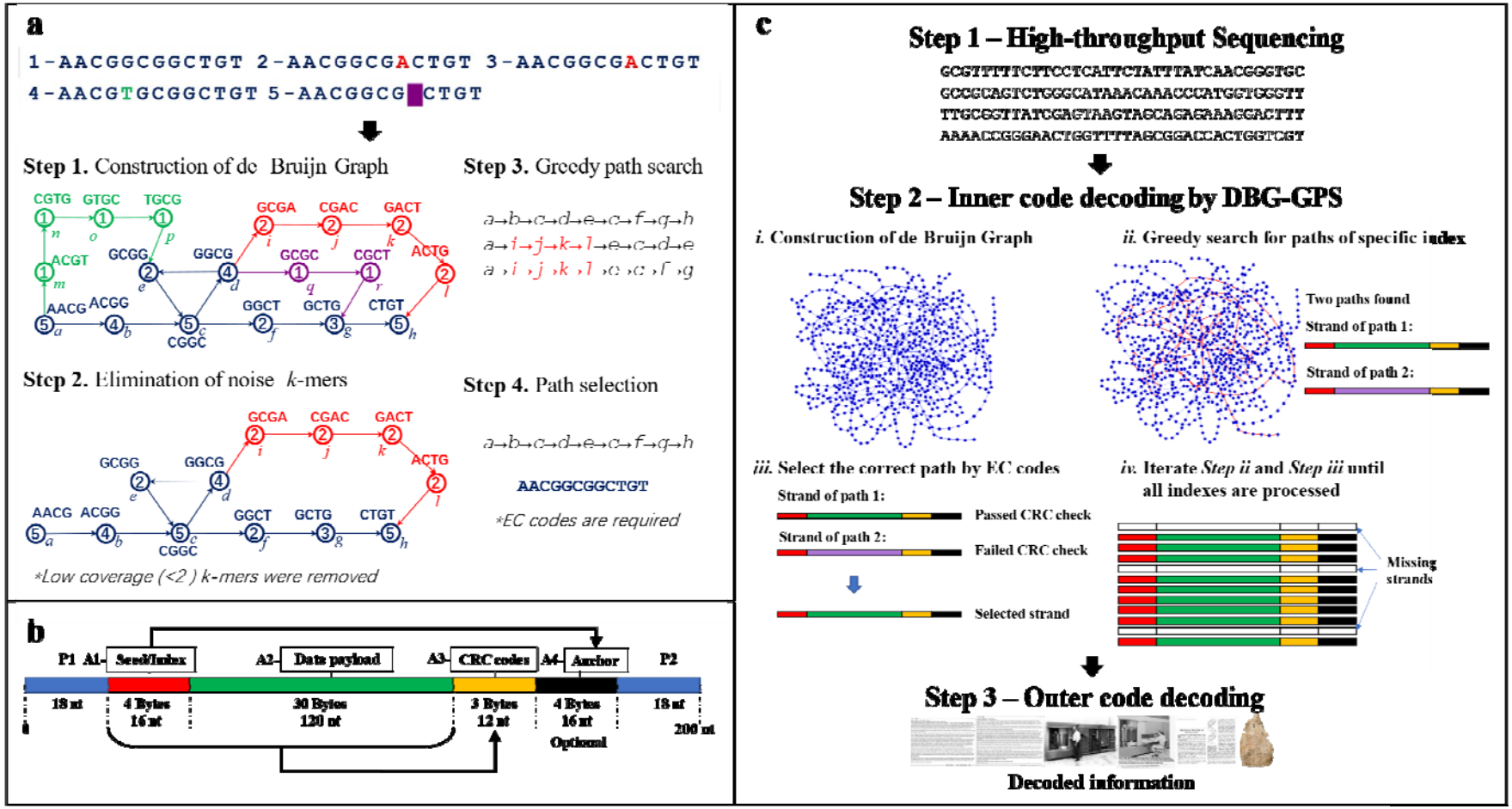

a
1-AACGGCGGCTGT 2-AACGGCGACTGT 3-AACGGCGACTGT
4-AACGTGCGGCTGT 5-AACGGCG CTGT
Step 1. Construction of de Bruijn Graph
Step 2. Elimination of noise k-mers
*Low coverage (<2) k-mers were removed
Step 3. Greedy path search
a→b→c→d→e→c→f→g→h
a→i→j→k→l→e→c→d→e
Step 4. Path selection
a→b→c→d→e→c→f→g→h
AACGGCGGCTGT
*EC codes are required
b
P1 A1- Seed/Index A2- Data payload A3- CRC codes A4- Anchor P2
18 nt
4 Bytes 16 nt
30 Bytes 120 nt
3 Bytes 12 nt
4 Bytes 16 nt Optional
18 nt
200 nt
c
Step 1 – High-throughput Sequencing
GCGTTTTTCTTCCTCATTCTATTTATCAACGGGTGC
GCCGCAGTCTGGGCATAAACAAACCCATGGTGGGTT
TTGCGGTTATCGAGTAAGTAGCAGAGAAAGGACTTT
AAAACCGGGAACTGGTTTTAGCGGACCACTGGTCGT
Step 2 – Inner code decoding by DBG-GPS
i. Construction of de Bruijn Graph
ii. Greedy search for paths of specific index
Two paths found
Strand of path 1:
Strand of path 2:
iii. Select the correct path by EC codes
Strand of path 1:
Passed CRC check
Strand of path 2:
Failed CRC check
Selected strand
iv. Iterate Step ii and Step iii until all indexes are processed
Missing strands
Step 3 – Outer code decoding
Decoded information

**Fig. 3 | Experimental verification of the robustness of DBG-GPS**. **a**, A 314kb file is wrote on DNAs to generate a "Master Pool". **b,** Serial deep error-prone PCR to introduce base errors. **c**, Multiple data retrievals with intended unspecific amplifications to introduce strand rearrangements. **d**, Accelerated aging experiments to introduce strand breaks.

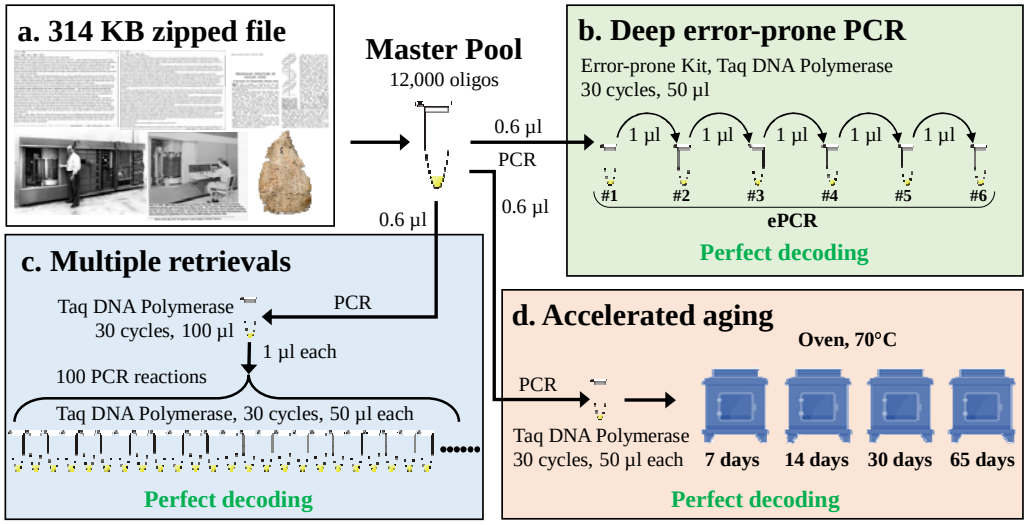
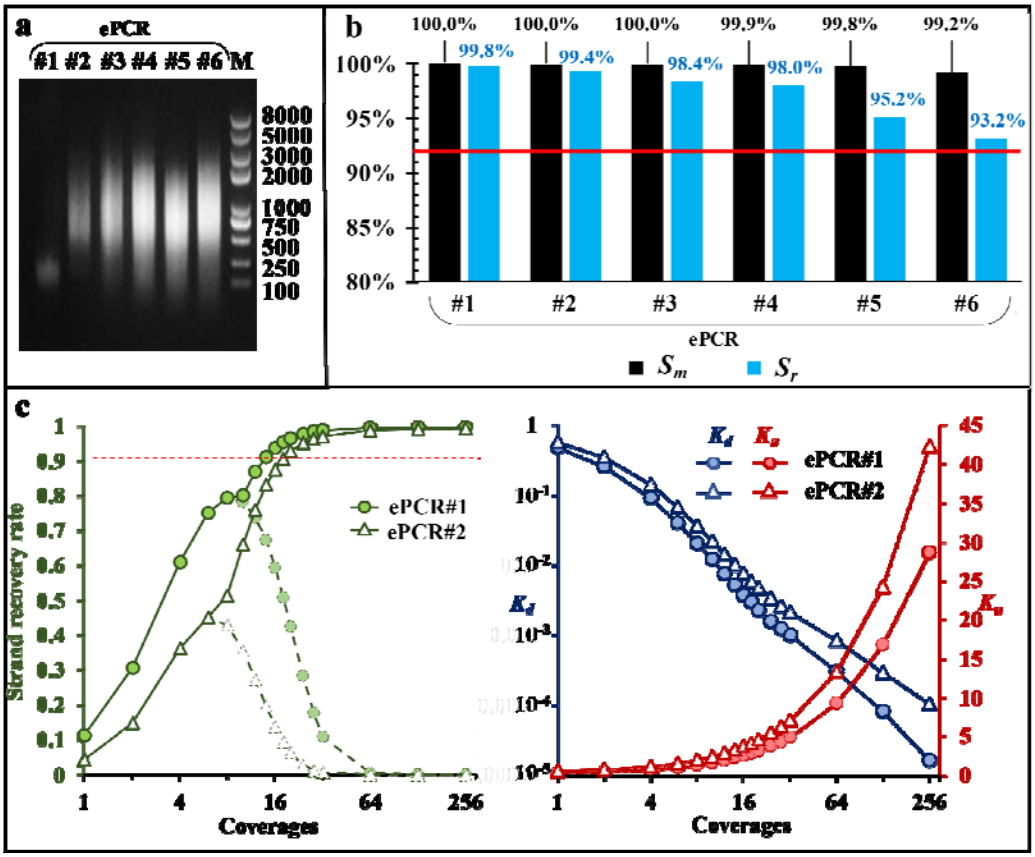
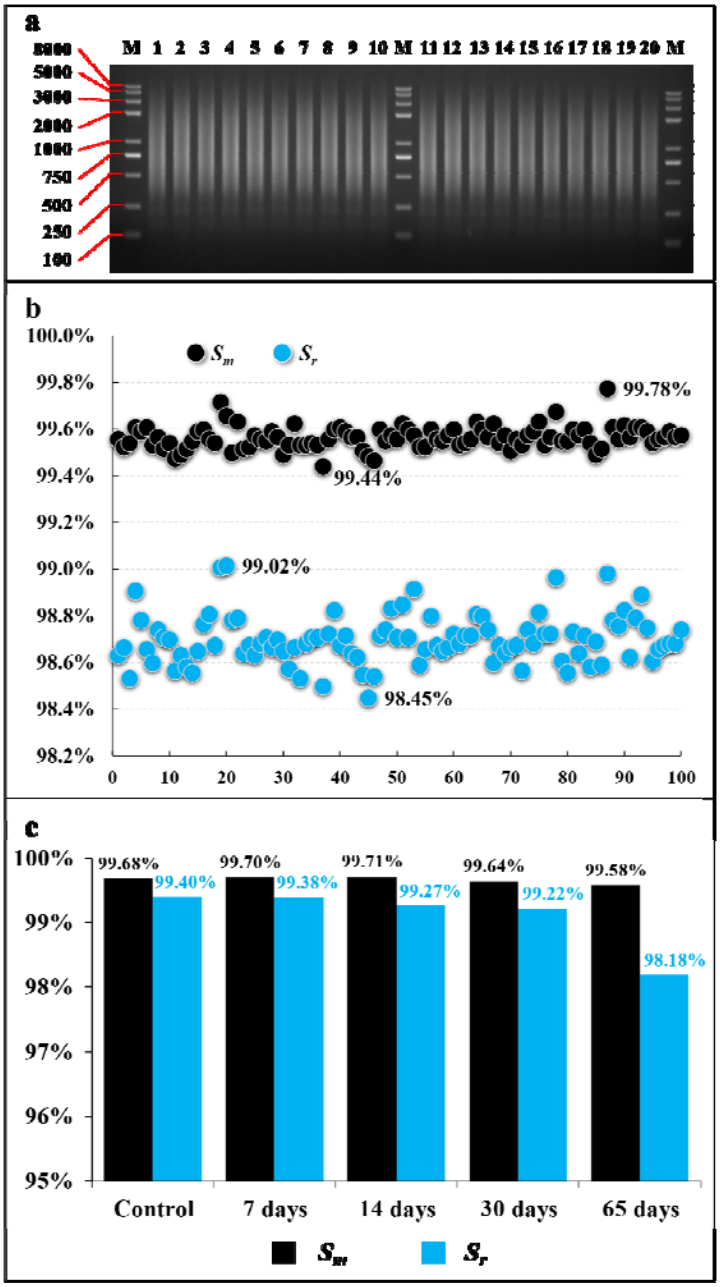


**Fig. 4 | Reliable data recovery with deep error-prone PCR products. a,** Agarose electrophoresis analysis of the serial error-prone PCR (ePCR) products. **b**, Reliable strand reconstruction with low quality ePCR products. The dashed red line stands for the decoding lower boundary of the outer fountain codes in this case study. **c**, Low coverage decoding by random sub-sampling. Sequencing reads of ePCR#1 and ePCR#2 are used for random sub-sampling individually. For each coverage tested, 12 independent sub-sampling simulations were performed and the averages values were presented. The standard deviations are presented in **Tab. S5** for clarity. Coverages in range of 1 to 256 were considered. The average strand recovery rates ($S_r$), *k*-mer dropout rates ($K_d$) and $K_n$ values were presented. The dashed curves are the strand recovery results without elimination of low coverage *k*-mers. The horizontal dashed red line stands for the decoding limits by the outer Fountain codes.



684

**Fig. 5 | Multiple retrieval and accelerated aging tests. a,** Agarose electrophoresis analysis of the parallel PCR products for multiple data retrievals. Twenty representatives out of 100 samples are shown. M stands for the DNA Marker. **b**, Strand decoding results of the 100 multiple retrievals with DBG-GPS. **c**, Strand decoding rates by DBG-GPS from accelerated aged DNA samples.


a
M 1 2 3 4 5 6 7 8 9 10 M 11 12 13 14 15 16 17 18 19 20 M
5000
3000
2000
1000
750
500
250
100
b
100.0%
99.8%
99.6%
99.4%
99.2%
99.0%
98.8%
98.6%
98.4%
98.2%
$S_m$
$S_r$
99.78%
99.44%
99.02%
98.45%
0 10 20 30 40 50 60 70 80 90 100
c
100%
99%
98%
97%
96%
95%
99.68%
99.40%
99.70%
99.38%
99.71%
99.27%
99.64%
99.22%
99.58%
98.18%
Control
7 days
14 days
30 days
65 days
$S_m$
$S_r$

692

**Tab. 1 | Comparison of the DBG-GPS and CL-MA based methods in decoding speed.**

| DBG-GPS | | CL-MA | |
|---|---|---|---|
| **Stages** | **Time (s)** | **Stages** | **Time (s)** |
| ***k*-mer counting** ***(Jellyfish 32 threats)*** | 0.63 ± 0.04 | **Clustering**[23] ***(32 threats)*** | 0.36 |
| **Strand decoding** ***(Python Script)*** | 36.84 ± 0.91 | **Multi-alignment** ***(Muscle[16], IO time subtracted)*** | 3874.48 ± 30.82 |
| | | **Consensus calling** ***(Majority voting[13])*** | 15.42 ± 0.40 |
| **Total** | 37.47 ± 0.95 | **Total** | 3890.26 ± 31.22 |

# Supplementary materials

**Fig. S1 | A two-dimensional coding strategy of traditional EC codes handles the base shift problem of indel inefficiently.**

**Fig. S2 | Path search complexity caused by base errors can be remarkably released by elimination of low coverage *k*-mers.**

**Fig. S3 | A both-way search strategy reduces decoding time.**

**Fig. S4 | Details of DBG-GPS algorithm.**

**Fig. S5 | Linear decoding complexity of DBG-GPS.**

**Fig. S6 | Agilent 2100 analysis of the accelerated aged samples.**

**Tab. S1 | Computational estimation of the maximal strand decoding rates ($S_m$) of different types and rates of errors with 20 strand copies.**

**Tab. S2 | Strand decoding performance of four-step DBG-GPS algorithms estimated by Monte Carlo simulations.**

**Tab. S3 | Theoretical decoding capacities of DBG-GPS with different *k*-mers sizes.**

**Tab. S4 | Details of strand recovery results of deep ePCR experiments.**

**Tab. S5 | Details of the sub-sampling simulation results with ePCR#1 and ePCR#2.**