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] SynBio Research Platform, Collaborative Innovation Center of Chemical Science and Engineering, 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

Data storage in DNA, which store information in polymers, is a potential technology with high density and long-term features. However, the indels, strand rearrangements, and strand breaks that emerged during synthesis, amplification, sequencing, and storage of DNA molecules need to be handled. Here, we report a de Bruijn graph-based, greedy path search algorithm

(DBG-GPS), which can efficiently handle all these issues by efficient reconstruction of the DNA strands. DBG-GPS achieves accurate data recovery with low-quality, deep error-prone PCR products, and accelerated aged DNA samples (solution, 70℃ for two weeks). The robustness of DBG-GPS was verified with 100 times of multiple retrievals using PCR products with massive unspecific amplifications. Moreover, DBG-GPS shows linear decoding complexity and more than 100 times faster than the multiple alignment-based methods, indicating a suitable solution for large-scale data storage.

Keywords

DNA data storage, de Bruijn graph, Indel, DNA rearrangements, DNA break, Unspecific amplification, Erasure codes, Fountain codes, Reed-Solomon codes

Introduction

With properties of high density, low maintenance, and long persistent period, DNA as storage media is believed to be a potential candidate for solving the world challenge of data explosion[1–6]. However, the massive errors that emerged during DNA synthesis, amplification, sequencing, and storing make reliable data storage in DNA a challenge. Previous studies have shown that the implementation of a reliable codec system for DNA data storage requires two layers of error correction (EC) codes. The outer layer codes handle the strand dropouts and the inner layer codes respond for the error-free decoding of DNA strands as many as possible[7–10]. Strand dropouts are erasures, which

can be well solved by erasure codes, *e.g.* Fountain, Reed-Solomon (RS) codes[8–10]. However, for the design of robust inner codes, there exist three challenges yet to be solved.

The first challenge is the efficient handling of indels. Although elegant EC codes have been developed in the field of information theory, they are not optimal for indels with base shift problem. Previously, base deletions have been occasionally treated as erasures[8–10]. However, other than unreadable bases, *i.e.* erasures, base deletions also cause shift forwards of downstream nucleotide bases. This base shift issue, which applies to base insertions as well, causes serious problems for traditional erasure codes as illustrated in **Fig. S1**. Despite indels can be detected by traditional erasure codes, the positions of the missed and inserted bases cannot be inferred. The successful data decoding in previous studies using RS erasure codes as inner codes owing to a two-dimensional coding strategy, accurate DNA synthesis and sequencing technologies[8,10]. However, this mechanism is low efficient in handling the indels. A single indel in the row dimension (DNA strand) will lead to a series of 'substitution' errors in the column dimension. Thus, this strategy only works with low indel rate. When slightly more indels are introduced, the 'substitutions' become overwhelming and quickly exhaust the column EC codes. Recently, *Press et. al.* implemented a hash encoded, decoded by greedy exhaustive search (HEDGES) codes that can correct indels with a single copy of DNA strand[11]. Despite outstanding technical achievements in indel correction with single copy data, HEDGES requires 40% redundancy codes for correction of merely 3% indel errors. This would significantly increase the cost of DNA data storage. A cost-efficient mechanism to handle the indels is highly desirable.

The second challenge is the correction of strand rearrangements. Copying and retrieval of DNA data requires polymerase chain reaction (PCR) for amplification of DNA strands. However, as a common sense in molecular biology, PCR experiments result in unspecific amplification occasionally even with optimized conditions. Large data storage in DNA requires enormous DNA strands of various sequence contents. PCR amplification with various DNA strands can result in unspecific amplifications even more frequently. Unspecific amplification will cause massive strand re-arrangements. This is a serious problem to the data integrity, which has not been solved yet. Although previous studies have proved that perfect decoding of DNA data can be achieved with high quality PCR products, limited number of data retrievals have been performed, which is far from the industrial standards of storage media. The robustness of DNA data storage under massive multiple retrievals is still not verified yet. To achieve robust data retrieval, the unspecific amplification problem emerged occasionally needs to be handled. The third challenge is the strand break problem. As typical biological polymers, DNA molecules are highly degradable[12,13]. Previous studies developed methods for protection of DNA molecules by silicon beads or earth alkaline salts[8,14], which can reduce the speed of DNA degradation significantly[8]. However, degradation of DNA molecules is inevitable fundamentally, especially for long-term storage. Thus, a new mechanism that can reconstruct the original DNA sequences from small fragmented DNA strands is highly desirable for enhancing the robustness of DNA data storage.

DNA data storage achieves data operations by DNA synthesis, PCR, and sequencing processes. These processes always generate multiple error-rich

copies of DNA strands fast and cheaply. This unique ‘multiple copy’ feature, also termed as “natural redundancy” in previous studies, or ‘data repetition’ by definition of information theory[15], makes DNA data storage a unique channel[15,16]. Data repetition is regarded as a low efficient way of redundancy coding in traditional media and channels since additional data copies are super time- and cost-expensive. However, data repetition in the forms of DNA strand copies is cheap, fast, and basically unavoidable. Indeed, it may increase the cost significantly to make a single copy DNA data since additional efforts are required. Efforts on reducing the “multiple copy” feature by serial dilutions observed massive strand dropouts when reducing the average copy number under ten, making reliable data retrieval impractical[17]. Additionally, DNA molecules, as highly degradable polymers, break easily. Thus, even DNA data with single copy of each strand is obtained, the data stability is still questionable because strand breaks, which cannot be handled by any reported codes, are fatal to DNA data integrity. Taken together, a small number of strand copies, *e.g.* 10 to 100, which do not hamper the data density, is required for practical DNA data storage.

In this study, we report the robust data storage in DNA by an efficient strand reconstruction algorithm based on de Bruijn Graph-based Greedy Path Searching (DBG-GPS). DBG-GPS can handle all the three challenges by taking advantage of the multi-copy feature for high-efficient DNA strand reconstruction. With the optimized procedure, DBG-GPS shows high resistance to indels as well as substitutions. While correction of indels with single copy data is still challenging[18,19], errors (indels and/or substitutions) in a high rate of 10% can be efficiently corrected by DBG-GPS. Importantly,

DBG-GPS achieves robust data retrieval with low-quality PCR products with massive strand rearrangements caused by unspecific amplifications, and highly degraded DNA samples with massive DNA strand breaks. Furthermore, although python scripts are utilized, DBG-GPS shows more than 100 times faster than the clustering and multiple alignment-based methods. The linear decoding complexity revealed by simulations makes DBG-GPS a suitable solution for large-scale data storage.

Results

Principles and potentials of de Bruijn graph-based decoding

Due to the difficulties of long DNA synthesis, large data storage in DNA has to distribute the information in massive DNA fragments. Thus, decoding of DNA data is assembly of the original data from small pieces of information carried by massive DNA fragments. This process is similar to the process of genome assembly, in which the complete genome sequences are assembled from small pieces of DNA sequences obtained from shotgun genome sequencing. However, as shown by a large number of genome sequencing projects, it is a challenge to obtain the complete genome sequences by shotgun genome sequencing solely[20]. Imperfect genome assembly is acceptable since following up experiments can be performed to improve the assembly[20]. Obviously, imperfect assembly is unacceptable for data storage applications. To decode the DNA data accurately, the extra options of indexes and error correction codes bring opportunities. De Bruijn graph theory, which is an efficient way to represent a sequence in terms of its *k*-mer components, has been successfully

applied to genome assembly with second-generation sequencing data[20,21]. However, the potentials of de Bruijn graph theory in DNA data storage haven't been explored thus far.

To investigate the potentials of de Bruijn graph theory in DNA data storage, we first analyzed the representations of 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" in the left with a *k*-mer size of four (**Fig. 1a-*i***). The DNA sequence of "AACGGCGGCTGT" is a path, *i.e.* "$a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow c \rightarrow f \rightarrow g \rightarrow h$", 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 reduced weights (coverages) of the correct path (**Fig. 1a-*ii***). Indels are much more difficult to handle than substitutions by traditional EC codes. However, the representations of base indels in de Bruijn graph are very similar to substitutions: both creating interfering path(s) and reducing the weights of the correct path (**Fig. 1a-*iii***). Although strand breaks can cause serious problems for traditional EC codes, they only cause subtle influence on the de Bruijn graph, by reducing weights of the correct path without changing the graph structure. A greedy path search algorithm and a path selection mechanism for recognition of the correct path, could recover the DNA strand theoretically. In this way, the errors are transformed into a path search and selection problem by de Bruijn graph theory with multiple sequence copies. With this mechanism, as long as the correct path, e.*g.* "$a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow c \rightarrow f \rightarrow g \rightarrow h$" in **Fig. 1a**, is preserved, the corresponding DNA strand can be decoded theoretically. Thus, the path robustness indicates the potentials of de Bruijn graph theory-based decoding.

Obviously, the sequence error rates and copy number together determine the robustness of a de Bruijn graph path. To further investigate how the strand copies and error rates affect path robustness, we simulated DNA strands with different rates of errors. Variant numbers of the simulated containing sequences were utilized for de Bruijn graph construction and path integrality analysis. 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 strands with a complete path in the graph was defined as 'maximal strand decoding rate ($\boldsymbol{S_m}$), which stands for the theoretical maximal strand decoding rate which can be achieved by an actual de Bruijn graph-based decoder. Firstly, we run the simulations with different types of errors separately. With twenty strand copies, the path robustness with different types of errors in the range of 1% to 10% with a step length of 1% were estimated. As shown in **Fig. 1b**, although substitutions, indels, and strand breaks are quite different errors based on traditional information theory, they show very close $\boldsymbol{S_m}$ values with the same error rates. Importantly, with merely 20 sequence copies, the de Bruijn graph path shows high robustness with an error rate of 5%, which is a common error level of DNA data storage reported[9,10,22]. In realistic situations, the errors that emerged in DNA data channel are mixtures of all types of errors. Thus, we then perform thorough simulations with an error mixture ratio of 50% substitutions, 25% insertions and 25% deletions with different rates of errors and copies of strands. Total error rates in the range of 1% to 10% with a step length of 0.5%, and strand copies in the range of 5 to 100 with a step length of 5 were considered. As shown in **Fig. 1c**, $\boldsymbol{S_m} > 0.99$ can be obtained with 60 sequence copies (red dashed line) with high rate errors of 10%, revealing the great potentials of de Bruijn graph-based decoding.

Development of a general framework for de Bruijn graph-based decoding

Based on the analysis of previous section, two major steps are required for decoding of DNA strands from multiple error-rich sequence copies utilizing de Bruijn graph theory: 1) greedy path search for possible path candidates and 2) selection of the correct path. In this section, we describe the issues emerged during the implementation of a de Bruijn graph-based decoding algorithm and strategies to solve them. A four-step framework is then proposed for effective strand reconstruction practice with de Bruijn graph. Eventually, the DBG-GPS algorithm was implemented following this framework with Python.

Elimination of low coverage *k*-mers decreases the greedy path search complexity

A serious problem emerged is the explosion of de Bruijn graph size caused by base errors. 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 copies each (**Fig. S2a**). Greedy path search with this enlarged de Bruijn graph resulted in overwhelming number of possible paths, leading to unbearably long path search time. The enlarged de Bruijn graph contains massive noise *k*-mers, *i.e.* *k*-mers introduced by base errors. To quantify the noise *k*-mers, we introduce an indicator K_n, which is the ratio of noise *k*-mers and correct *k*-mers in the de Bruijn graph. Clearly, more errors and strand copies will introduce more noise *k*-mers to the de Bruijn graph, which will result in a larger K_n. 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 the increasement of error rates (**Fig. S2b**) or strand copies (**Fig. S2b**). Thus, a pre-processing step that can remove the noise *k*-mers is desirable.

Although DNA data operations can introduce massive errors, these errors are low probability events individually. Thus, the occurrences, *i.e.* the coverages of noise *k*-mers, should be significantly lower than the correct ones. In other words, the low coverage *k*-mers are more likely to be noise *k*-mers. Preliminary testing with the elimination of *k*-mers with coverage of one significantly reduced the decoding burden of branch paths without significant affections to the path integrity (**Fig. S2b**). To further investigate the feasibility of coverage as an indicator for distinguishing noise *k*-mers, we used a Poisson distribution model to simulate the copy number distribution of DNA strands. The coverage distribution of noise *k*-mers and correct *k*-mers are then investigated based on Monte Carlo simulations. The simulation is performed with 10,000 DNA strands, a λ value (*i.e.* the expectation value) of fifty for strand copies and an error rate of 4% (a common level of error rates reported in previous studies**)**. The vast majority of the noise *k*-mers show coverages lower than eight, while the correct ones show coverages higher than eight (**Fig. S2c**). Thus, a coverage cut-off of eight can effectively remove most of the noise *k*-mers suggesting an effective mechanism for the elimination of noise *k*-mers.

Cyclic redundancy check (CRC) codes effectively recognize the correct path

Elimination of low coverage *k*-mers can effectively remove most of the noise *k*-mers based on Monte Carlo simulation, the remaining noise *k*-mers can still have great affections on the accuracy of strand decoding. Thus, a mechanism to recognize the correct path and verify its integrality is required. Kinds of traditional EC codes can achieve this goal theoretically, *e.g.* CRC codes, RS code, or hash functions. Although hash functions are suitable for such tasks in principle, we failed to find a proper hash function that can generate short key words of few bytes. Since erasure is not presented in the path selection process, RS codes, which is capable of handling erasures, is not an optimal solution hence. Here, we choose CRC codes which is powerful in error detection without unnecessary error-correction mechanism. Intensive Monte Carlo experiments show that CRC codes can recognize the correct path in all one million times of independent simulations, proving its effectiveness in the selection of correct paths, *i.e.* correct strands.

A both-way search strategy increases decoding speed

Although elimination of noise *k*-mers based on coverage can remarkably reduce the greedy path search burden, the remained noise *k*-mers still can increase the decoding time significantly, especially with high error rates. To further reduce the greedy path search time, we proposed a both-way search strategy which can search for possible paths simultaneously from both ends of DNA strands. To enable this both-way search strategy, we designed a DNA strand structure with an additional ‘anchor’ element as shown in **Fig. 2c**. The

anchor is an integer generated from the index by a hash-like function. This function generates one specific number with a given index. For proof-of-concept, the pseudorandom function of Python is used for the generation of the anchors by using the indexes as seeds. Due to the feature of pseudorandom function, it always gets the same 'random' value(s) with the same seed. Thus, the greedy path search engine can calculate the anchor based on the index without additional information, enabling the both-way search strategy. Simulation shows that this both-way search strategy can reduce the decoding time significantly, especially when sequence error rates are high. With an error rate of 10% and a copy number of 100, the both-way search strategy shows about 60% reduction in decoding time compared with one-way search (**Fig. S3**). No significant difference in decoding accuracy is observed with the two search strategies.

Four-step framework for effective strand reconstruction practice with DBG-GPS algorithm

For effective strand reconstruction with DBG-GPS, we proposed a four-step framework as illustrated in **Fig. 2a**: ***Step 1***, Construction of de Bruijn Graph. Here, the construction of de Bruijn Graph refers to the process of counting of *k*-mers. In the current python implementation, a simple hash dictionary data structure was applied. For further studies with a more advanced data structure of the de Bruijn Graph, we suggest creating the advanced data structure after the elimination of noise *k*-mers (Step 2). ***Step 2***, Elimination of noise *k*-mers. In the present study, we use coverage as an indicator for the elimination of noise *k*-mers. Additional codes for the identification of noise *k*-mers can also be

introduced to make the elimination more efficient in future studies. ***Step 3***, Greedy path search. If the error rate is still high, a both-way search strategy can be applied to reduce the decoding time. ***Step 4***, Selection of the correct path. The paths found are translated into bit strings and the embed EC codes are then used for the selection of the correct path. Only in case of one path passes the EC code checking, this path will be selected and strand decoding succeed. Following this framework, we implemented the DBG-GPS algorithm with Python. Reliable strand reconstruction (S_r > 96.7%) was achieved with various error rates range from 1% to 10% with sixty strand copies as proved by Monte-Carlo simulations (**Tab. S2**).

Integration of DBG-GPS with outer erasure codes

DBG-GPS is designed as an inner decoding mechanism for the error-free reconstruction of DNA strands. It can be easily combined with erasure codes, *e.g.* fountain codes or RS codes, for large-scale data storage. Integration of DBG-GPS with erasure codes is quite simple. Simply add CRC codes for strand integrity verification during strand encoding, and use the DBG-GPS as an inner decoding process for the decoding of DNA strands. To use other kinds of codes for strand verification with DBG-GPS, just replace the CRC codes with specific codes and switch to the corresponding checking method during strand selection. Since the encoding process is highly similar to the previous studies, we only illustrated the overall decoding process using DBG-GPS as the inner decoding method as shown in **Fig. 2c**. There are three major steps in the decoding process. ***Step 1***, High-throughput sequencing. The DNA strands are sequenced by high-throughput sequencing technologies. ***Step 2***, Inner

code decoding. The aim of this step is to decode DNA strands as many as possible. DBG-GPS is a complete solution for ***Step 2***. ***Step 3***, Decode the original information by outer erasure codes. A python implementation using fountain codes as outer codes and DBG-GPS as inner codes is available at https://switch-codes.coding.net/public/switch-codes/DNA-Fountain-De-Bruijn-Decoding/git/files.

Decoding speed of DBG-GPS

To the best of our knowledge, the clustering and multiple alignment-based (CL-MA) methods are the only methods that can rebuild the DNA strands from multiple error-rich strand copies[10,23,24]. To compare the decoding speed of DBG-GPS with CL-MA methods, we produced 10,000 DNA droplets with DNA fountain codes using CRC codes as redundancy codes for strand selection. For each DNA droplet, we generated fifty sequence copies with 4% random errors introduced (2% substitutions, 1% insertions and 1% deletions). The 500,000 error-rich reads generated were then utilized for strand decoding by DBG-GPS and CL-MA respectively. Since the clustering process is quite complicated in reported studies, for the CL-MA-based decoding, we feed the program with grouped reads. For decoding speed comparison, a clustering time was estimated based on the recent study by Cyrus *et al*.[23]. Since the multiple alignment process was performed by a standalone program (Muscle[25]), 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 10,000 times. This process is iterated three times and the

average time is subtracted in the CL-MA based decoding speed test. The *k*-mer counting step of DBG-GPS was performed with JellyFish. As shown in **Tab. 1**, although a Python implementation of DBG-GPS was utilized in the decoding speed test, DBG-GPS still shows 103 times faster than the CL-MA based methods without significant differences in decoding accuracy. A C++ implementation of DBG-GPS is expected to further enhance its performance remarkably. Importantly, DBG-GPS shows linear decoding complexity while increasing the total strand scale from 10 to 10,000 (**Fig. S5**), making it a suitable solution for large-scale data storage in DNA.

Experimental verification of the robustness of DBG-GPS

A 314 KB file compressed from two text files and four jpeg pictures, was encoded into 12,000 DNA strands in the length of 200 bp using the DNA fountain codes implemented in this study. Each oligo encodes thirty bytes of data, three bytes of CRC codes, four bytes of index, and four bytes of anchor codes. The overall strand redundancy is 10.5% enabling reliable data retrieval if more than 92% of the strands could be decoded. The designed DNA strands were synthesized by Twist Bioscience. The single strand DNAs (ssDNA) obtained from Twist Bioscience were dissolved in ddH_2O. 0.6μl of the dissolved ssDNA was used as template for PCR amplification and data retrieval by sequencing. To verify the robustness of DBG-GPS with different types of errors, *i.e.* the base errors (indels and substitutions), strand rearrangements, and strand breaks, we performed three harsh experimental tests, as shown in **Fig. 3a**, **b**, and **c** correspondingly.

Reliable data recovery with deep error-prone PCR products

To verify the robustness of DBG-GPS with high rates of base errors including indels and substitutions, we utilized error-prone PCR (ePCR) for amplification of the ssDNA library obtained from Twist Bioscience. Six series of ePCR were performed using an error-prone PCR Kit from TIANDZ. All ePCR experiments were performed for 30 thermal cycles with the highest error-rate settings instructed by the Kit manual. Accordingly, these settings could introduce eight errors per 1,000 bp per amplification cycle. The first round ePCR utilized 0.6 µl samples of the ssDNA master pool as templates. The five following ePCR experiments use 1 µl products of the previous round ePCR as templates. The six ePCR samples obtained, *i.e.* ePCR#1, ePCR#2, ePCR#3, ePCR#4, ePCR#5, and ePCR#6, were sequenced on Illumina HiSeq platform. Due to unspecific amplification, PCR products of ePCR#2, ePCR#3, ePCR#4, ePCR#5 and ePCR#6 show severe smear band in the range of 200 bp to 3,000 bp which cannot be sequenced directly by Illumina platform. Thus, the PCR products were digested with a target length of ~200 bp using DNase I. After digestion, all digested fragments are collected for sequencing and DBG-GPS based decoding.

As shown in **Fig. 4a**, the estimated S_m values of the six samples are all very high (>99%). It was surprising to notice that the S_m values of ePCR#1, ePCR#2, and ePCR#3 were estimated to be 100%, indicating perfect reservation of all strand paths even after 90 cycles of ePCR. We then apply DBG-GPS to recover the encoded DNA strands with the six ePCR samples. As expected, with high S_m values, high strand recovery rates (S_r), between 93.2%

to 99.8%, were obtained, especially for ePCR#1, ePCR#2, and ePCR#3. However, significant decerase trend of S_r was observed with more error-prone PCR cyles were introduced. Considering the S_m values of ePCR#5 and ePCR#6 are still very high, the decrease of S_r is mainly caused by massive base errors introduced. Although the S_r value of ePCR#6 (93.2%) is relatively lower than the other samples, with the 10.5% strand redundancy codes, we can recover the encoded data correctly by the outer fountain codes. Interestingly, the maximal decoding rate (**S_m**) of ePCR#6 was estimated to be up to 99.2%, which was just slightly decreased. In future studies, incorporation of advanced codes designed for de Bruijn graph-based decoding could further enhance the strand recovery rate up to 99.2%.

To further prove the robustness of DBG-GPS, we perform random sub-sampling experiments on the sequencing results of ePCR#1 and ePCR#2 with different coverages, *i.e.* reading copies, range from 1 to 256. The *k-mer* dropout rates, K_d, noise indicator K_n and S_r in different coverages were calculated corresponding. For each coverage tested, 12 independent random sub-sampling experiments were performed and the average values of estimated K_d, K_n and S_r are presented. As shown in **Fig. 5a**, as expected, the K_n values of ePCR#2 are higher than ePCR#1, indicating high rates of errors were introduced to ePCR#2. The K_n differences of the two samples become more and more significant when increasing the coverages. The K_d value of ePCR#2 is higher than the K_d value of ePCR#1. The K_d values of the two samples dropped quickly while increasing the coverages from 1 to 256. As shown in **Fig. 5b**, the two samples can be reliably decoded at low coverages (coverage 16 for ePCR#1, coverage 24 for ePCR#2).

Massive parallel retrieval test with unspecific amplification PCR products

For practical data storage, massive multiple retrievals to test the data robustness is required. However, only a limited number of retrievals is tested for DNA data storage yet. To achieve reliable data storage with multiple retrievals, the occasionally emerged unspecific amplification problem of PCR needs to be handled. As shown by the deep error-prone PCR experiments, DBG-GPS based decoding can work with PCR with products with unspecific amplifications. To further prove its robustness in handling unspecific amplification, as well as the robustness of DNA data storage under multiple reading, we performed 100 times of parallel data retrievals with PCR products with serious intended unspecific amplifications. As shown in **Fig. 3**, 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 and the sequencing results are committed for data decoding with DBG-GPS. Although all PCR products show serious unspecific amplifications as expected (**Fig. 6a**), DBG-GPS achieves high-quality strand recoveries on all of them as shown by **Fig. 6b**. The S_m values are estimated to be in the range of 99.44% to 99.78%. The strand recovery rates S_r obtained by DBG-GPS are in range of 98.45% to 99.02%, suggesting a robust mechanism in handling unspecific amplification problem.

Reliable data recovery with seriously degraded DNA samples

Different from the previously reported codec systems, which require complete DNA strands for decoding, the decoding process of DBG-GPS utilizes very short *k*-mer fragments. Thus, DBG-GPS can be utilized for data decoding with degraded DNA samples, of which massive truncated DNA strands caused by strand breaks. To verify the capability of DBG-GPS with degraded DNA samples, we performed accelerated aging experiments as shown in **Fig. 3c**. We first amplified the master pool with Taq DNA polymerase. Two liquid samples of 50 μl of the amplified Taq PCR products were accelerated aged at 70℃ for one and two weeks respectively. The accelerated aged samples were sequenced with Illumina sequencing platform. As shown by the analysis results of Agilent 2100 Bioanalyzer (**Fig. 7b** and **c**), the two DNA samples show serious degradation. Such degree of degradation would totally destroy the encoded data with previous reported codec systems. Dramatically, with DBG-GPS based decoding, accurate data decoding was achieved on both samples with high quality (>99% strand recovery, **Fig. 7d**). It worth mention that the DNA sample that accelerated aged at 70℃ for two weeks shows no obvious peak at the 200bp, indicating most of DNA strands are corrupted. The high-quality decoding with the two samples well proved the capacity of DBG-GPS in handling DNA degradation.

Exploration of DBG-GPS's capacity by large-scale simulations

To further test DBG-GPS's performance with large-scale data sets, large-scale simulations were also performed. An Ubuntu server 20.10 installation disk

image (~ 1 GB) was used as the input file for the generation of DNA droplets. Each DNA droplet was set to carry 60 bytes of information with a total length of 316 bp, around the technical limits of current DNA synthesis technologies. We first tried with ten million DNA droplets, representing a data scale of 600 megabytes (MB). With an expected copy number of 50, error rate of 4% (approximately 2% substitutions, 1% insertions, 1% deletions), a 155 GB sequence file was generated. However, although a computation workstation equipped with 640 GB memories is employed, the *k*-mer counting step by Jellyfish encountered memory limitations. We then reduced the DNA droplet number to five million, representing a data size of 300 MB. Preliminary decoding simulations on this data scale reveal a problem of long decoding time. Detailed analysis shows that this is due to the entangled DNA strands caused by repeated presentation of *k*-mers in different strands. To solve this problem, we propose two strategies. The first strategy is applying a filtering process during the production of DNA droplets to filter out the seriously entangled DNA strands. The second strategy is setting a path number boundary for greedy path search. If the greedy path search engine finds more paths than this value, it aborts search. This will reduce the strand decoding rate theoretically. We run the simulations with 5 million DNA droplets three times independently. On average 4,971,880 DNA droplets (recovery rate = 99.44%) were successfully recovered.

Conclusions

DNA data channel is unique with massive ‘data repetition’ in the forms of multiple error-rich strands. Efficient mechanism that can utilize this data

‘repetition’ feature for robust data storage is still missing due to the challenge of handling the massive errors, especially indels. Furthermore, the PCR-based strand amplification and storage of DNA molecules can lead to critical issues of strand rearrangement and strand break. In this study, we developed an effective mechanism, named ‘DBG-GPS’, that can well handle these problems by efficient strand reconstruction from multiple, error-rich sequences based on de Bruijn graph. DBG-GPS is applicable for all kinds of molecular data storage using any kind of polymers, *e.g.* DNA data storage with extended alphabets of non-natural nucleotides. Unlike previous studies, DBG-GPS can directly construct the DNA strands without computation-intensive clustering and multi-alignment process which cannot scale up well. Strategies of low coverage *k*-mers elimination, CRC codes, and both-way search were introduced and solved the decoding challenges caused by base errors. While traditional EC show difficulties in handling indels, DBG-GPS can efficiently correct high rates (10%) errors of indels, as well as substitutions. More errors can be corrected by incorporating more sequences for decoding. Importantly, by the strand reconstruction mechanism with short *k*-mers, DBG-GPS achieves reliable data recovery with unspecific amplification PCR products and highly degraded DNA samples. These achievements greatly enhanced the robustness of DNA data storage. This new mechanism, in combination with the previous reported methods of protecting DNAs by silicon beads[8], nanoparticles[26], and alkaline salts[14], could easily enable robust data storage in DNA for over ten thousands of years even in room temperature. Furthermore, despite Python scripts were utilized in this study, DBG-GPS shows more than 100 times faster than the CL-MA based methods. The linear decoding complexity revealed by simulations makes DBG-GPS a potential solution for

large-scale DNA data storage. Although DBG-GPS based decoding shows great potentials in DNA data storage, the pure python implementation, is far from optimal. The simple hash dictionary-based data structure for *k*-mer retrieval is memory intensive. Although no theoretical limitation on the data scale is shown with DBG-GPS, we failed to simulate with 500MB data on a computational workstation with 640 GB memories installed due to memory limitation. *C*++ implementation with a more advanced *k*-mer data structure is expected to significantly reduce the memory cost, as well as decoding time[27,28]. Further studies on the design of error correction codes that can incorporate with the noise *k*-mers elimination can further boost the capacity of DBG-GPS.

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□ for 3 mins, 94□ for 1min, 45□ for 1min, 72□ for 1min, 30 cycles. Six series 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-'CCTGCAGAGTAGCATGTC', P2- 'CGGATGCATCAGTGTCAG'. 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.

_k_-mers counting and elimination of low coverage _k_-mers

The python implementation provided in this study used simple hash dictionary structure for counting of k-mers. De Bruijn graph is quite memory expensive. 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[27], which is much more memory efficient and multi-core supporting. 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.

Choice of *k*-mer size

Obviously, with specific *k*-mer size, the data scale can be decoded is limited. For example, it is clearly that only very 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 6×10^5 EB for *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. However, to avoid decoding troubles with entangled DNA strands by repeated

k-mers, we strongly recommend to use a *k*-mer size with a decoding capacity at least ten times larger than the actually encoded data.

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 start with ATATATGGGATCAGAA as initial path. Similarly, the tail search start with ATATATATGCGTTTGG as initial path in case of both-way search. ***Step 3.*** Judgement step. This step is for selection of an end with few path numbers for extension of paths. ***Step 4***, Extension of paths on the end with fewer path number. ***Step 5***, Judgement step. This step is for 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 string. 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 case of one route passed the CRC checking, the decoding succeeds. ***Step 9***, Judgement step. Judge if the search finished. If not, iterate *step 1* to *step 9* until all index are searched for possible strands. If yes, finish the inner decoding process. The DeBruijnGraph() was implemented to search in both-way mode by default. To active the one-way search mode, switch the value of DeBruijnGraph() ->dual_search to 'False'.

Simplified error analysis in concept of de Bruijn graph theory

Although distribution profiles of error rate and copy number among different strands shapes 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 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 a high-quality data with 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 exceed specific threshold. In this proof-of-concept study, we use 100 for fast decoding, 1,000 for high-quality decoding.

Details of the implementation of fountain codes

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

Details of 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 18 was applied supporting a data scale of ~4 GB theoretically.

For the large-scale simulations, an Ubuntu Server 20.10 live install disk image (MD5: f51594d36008c456cf0360aeadd130d6, size: 1 GB) was used as input file for 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-'CCTGCAGAGTAGCATGTC' and P2- 'CGGATGCATCAGTGTCAG' were added to the ends. This will result with a DNA droplet length of 300 bp, around the theoretical limits of current DNA synthesis technologies. To avoid of entangled DNA strands, a filter process that always discard 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, a *k*-mer size of 24, and *k*-mer coverage cut-off of five, *i.e.* all *k*-mers with a coverage less than five is discarded, was applied.

Data Availability

All the algorithms proposed in this study were implemented with Python and the source code is available at https://switch-codes.coding.net/p/switch-codes/d/DNA-Fountain-De-Bruijn-Decoding/git/tree/ForPublication.

The compressed file in size of 314 KB is available at:

https://switch-codes.coding.net/public/switch-codes/DNA-Fountain-De-Bruijn-Decoding/git/files/master/input_files/314kb.rar.

The encoded files include:

Two famous talks, "I Have a Dream" from Martin Luther King and William Faulkner's speech of accepting the Nobel Prize in literature, as well as four pictures (two pictures of IBM RAMAC 350, one picture of the front page of "Molecular Structure of Nucleic Acids", and one picture of Oracle) were compressed into one binary file of 314kb with free version of WinRAR (http://www.winrar.com/).

Sequencing reads of deep error-prone PCR experiments:

https://pan.baidu.com/s/1xi-sY6erk-jfZ81ygJJmzA (ePCR#1, fetch codes: ATGC, MD5: 8690080a5081a963150e2f29fe3beafb)

https://pan.baidu.com/s/1Go0o6_gwrLnsGX1ZJicqTw (ePCR#2, fetch codes: ATGC, MD5: b5526124551f449775aa7f0ad5a239b7)

https://pan.baidu.com/s/10edj1eeT1VPAm5wHjoNgyA (ePCR#3, fetch codes: ATGC, MD5: 03a6c119dca076dd8db374ec7970d269)

https://pan.baidu.com/s/1DX8bODrUVSf9ceM2tyL5Vw (ePCR#4, fetch codes: ATGC, MD5: 61bdfcb77c1735483776d072da69f5b9)

https://pan.baidu.com/s/1otVqB60PnEK_lCTaB9N8AQ (ePCR#5, fetch codes:

ATGC, MD5: c63cbd86ca7441382a62bb5b648b012f)

https://pan.baidu.com/s/1h9CapXHfORi6wNEnj3LYLg (ePCR#6, fetch codes: ATGC, MD5: d7bffa155719aec7aea6f5162c719a32)

Sequencing reads of accelerated aging experiments:

https://pan.baidu.com/s/1fi4hEOiNser1bKHmqSS9mg (Control, fetch codes: ATGC, MD5: eae7a080e5bbcb8c57a9f8359b032622)

https://pan.baidu.com/s/1XAGfCL0fzENED6mb7pV3zg (70℃ for one week, fetch codes: ATGC, MD5: 5274eb78f640aef7b45c0382217901d4)

https://pan.baidu.com/s/1X3u9TBnLlTg39AXuHbks-Q (70℃ for two weeks, fetch codes: ATGC, MD5: 9fc954c37a57dfea67a95b44193acaa7)

Sequencing reads of 100 parallel retrievals:

https://pan.baidu.com/s/1BmEh6RZvSR7sQyTOB8pA-w (1-20, fetch codes: ATGC, MD5: fddf5f1811332da8ba0f5cd6f2ec7921)

https://pan.baidu.com/s/1vOoI_6V_iW8lO0J-CW84Gw (21-40, fetch codes: ATGC, MD5: f31fa5cdb6d8d00c55804bbe3f3e2214)

https://pan.baidu.com/s/1Yhzk9__WdFnV2NRup6ynDA (41-60, fetch codes: ATGC, MD5: 414b082eee36f1ce1996bd23014dd567)

https://pan.baidu.com/s/193erX4g9S5-mVYeNiV06YQ (61-80, fetch codes: ATGC, MD5: a4bffa7e8595c0afab78f2c5f5d9de1c)

https://pan.baidu.com/s/1fPYyIJ8kmuQpWh2sON-RwQ (81-100, fetch codes: ATGC, MD5: 789eecd1c452163391ffbe30fc30efc9)

Acknowledgements

This work was supported by Seed Foundation of Tianjin University and Natural Science Foundation of Shandong Province (ZR2017LC006). The authors would like to thank Dr. Sheng Ye and Dr. Weigang Chen for their constructive comments in improving the manuscript.

Competing interests

A patent has been filed for the DBG-GPS based decoding method presented in this study.

Author contributions

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

References

1. 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).

2. 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).

3. 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).

4. 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).

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

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

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

8. 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).

9. 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).

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

11. 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).

12. An, R. *et al.* Non-enzymatic depurination of nucleic acids: factors and mechanisms. *PloS one* **9,** e115950; 10.1371/journal.pone.0115950 (2014).

13. Zupanič Pajnič, I. *et al.* On the long term storage of forensic DNA in water. *Forensic science international* **305,** 110031; 10.1016/j.forsciint.2019.110031 (2019).

14. 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).

15. Song, L. & Zeng, A.-P. Orthogonal Information Encoding in Living Cells with High Error-Tolerance, Safety, and Fidelity. *ACS synthetic biology* **7,** 866–874; 10.1021/acssynbio.7b00382 (2018).

16. 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).

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

18. Brakensiek, J., Guruswami, V. & Zbarsky, S. Efficient Low-Redundancy Codes for Correcting Multiple Deletions. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms,* edited by R. Krauthgamer (Society for Industrial and Applied Mathematics, Philadelphia, PA, 012016), pp. 1884–1892.

19. Smagloy, I., Welter, L., Wachter-Zeh, A. & Yaakobi, E. Single-Deletion Single-Substitution Correcting Codes. In *2020 IEEE International Symposium on Information Theory (ISIT)* (IEEESunday, June 21, 2020 - Friday, June 26, 2020), pp. 775–780.

20. Zerbino, D. R. & Birney, E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome research* **18,** 821–829; 10.1101/gr.074492.107 (2008).

21. Pevzner, P. A., Tang, H. & Waterman, M. S. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*

of the United States of America **98,** 9748–9753; 10.1073/pnas.171285098 (2001).

22. 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).

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

24. 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).

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

26. 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).

27. 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).

28. 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).

Fig. 1: 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 12bp sequence of "AACGGCGGCTGT" with four copies 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 with one substitution in sequence #2; ***iii,*** De Bruijn graph with one insertion and one deletion on sequence #2 and #3 respectively; ***iv,*** De Bruijn graph with four strand breaks. **b,** Estimated maximal strand decoding rates (S_m) of different types and rates of errors with 20 sequence copies; Random errors were introduced with specific type and rate of errors from 1% to 10% with a step size of 1%. The error-rich sequences generated were then used for construction of the de Bruijn graph. The integrity of the correct paths was checked. This process is iterated for five times and the average S_m values are shown. The standard derivations, which are too small to be shown, 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). Sequence copies in range of 1 to 100 with a step size of 5, mixture error rates in range of 1% to 10% with a step length of 1% were considered. The maximal strand recovery rates (S_m) are the rates of DNA strands with complete paths in the graph during simulations. A S_m value > 99% was obtained with merely sixty sequence copies as shown by the dashed red line.

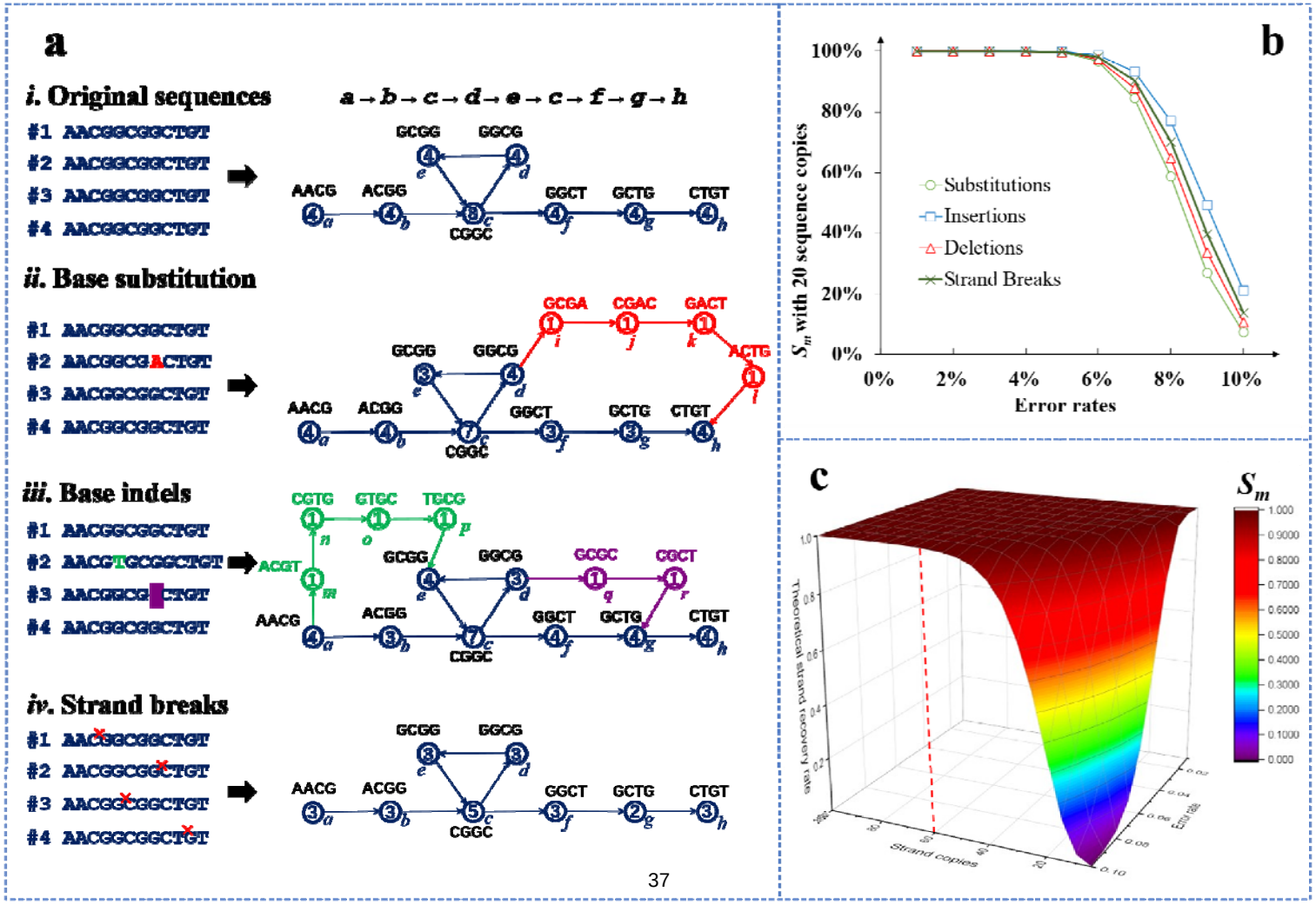

a
i. Original sequences
a → b → c → d → e → c → f → g → h
#1 AACGGCGGCTGT
#2 AACGGCGGCTGT
#3 AACGGCGGCTGT
#4 AACGGCGGCTGT
GCGG
GGCG
AACG
ACGG
GGCT
GCTG
CTGT
CGGC
ii. Base substitution
#1 AACGGCGGCTGT
#2 AACGGCGACTGT
#3 AACGGCGGCTGT
#4 AACGGCGGCTGT
GCGA
CGAC
GACT
ACTG
iii. Base indels
#1 AACGGCGGCTGT
#2 AACGTGCGGCTGT
#3 AACGGCG CTGT
#4 AACGGCGGCTGT
CGTG
GTGC
TGCG
ACGT
GCGC
CGCT
iv. Strand breaks
#1 AACGGCGGCTGT
#2 AACGGCGGCTGT
#3 AACGGCGGCTGT
#4 AACGGCGGCTGT
b
100%
80%
60%
40%
20%
0%
0%
2%
4%
6%
8%
10%
Error rates
S_m with 20 sequence copies
Substitutions
Insertions
Deletions
Strand Breaks
c
S_m
Theoretical strand recovery rate
Strand copies
Error rate
1.000
0.9000
0.8000
0.7000
0.6000
0.5000
0.4000
0.3000
0.2000
0.1000
0.000

Fig. 2: Framework of DBG-GPS and its integration with outer erasure codes.
a, A proposed four-step framework for de Bruijn graph-based strand decoding with multiple error-rich sequences: ***Step 1***. Construction of de Bruijn Graph; ***Step 2***. Elimination of noise k-mers; ***Step 3***. Greedy path search; ***Step 4***. Path selection by EC codes. **b,** Proposed structure of the DNA strand for DBG-GPS. There are four parts in the designed DNA strand: **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 if multiple paths were found; **A4** – Anchor cods which can be calculated by specific function based on the value of **A1**. **A4** enables both-way search mode which is optional. **c,** Integration of DBG-GPS with outer erasure codes. DBG-GPS is a full resolution for inner code decoding which can be easily combined with erasure codes, *e.g.* fountain codes, RS codes.

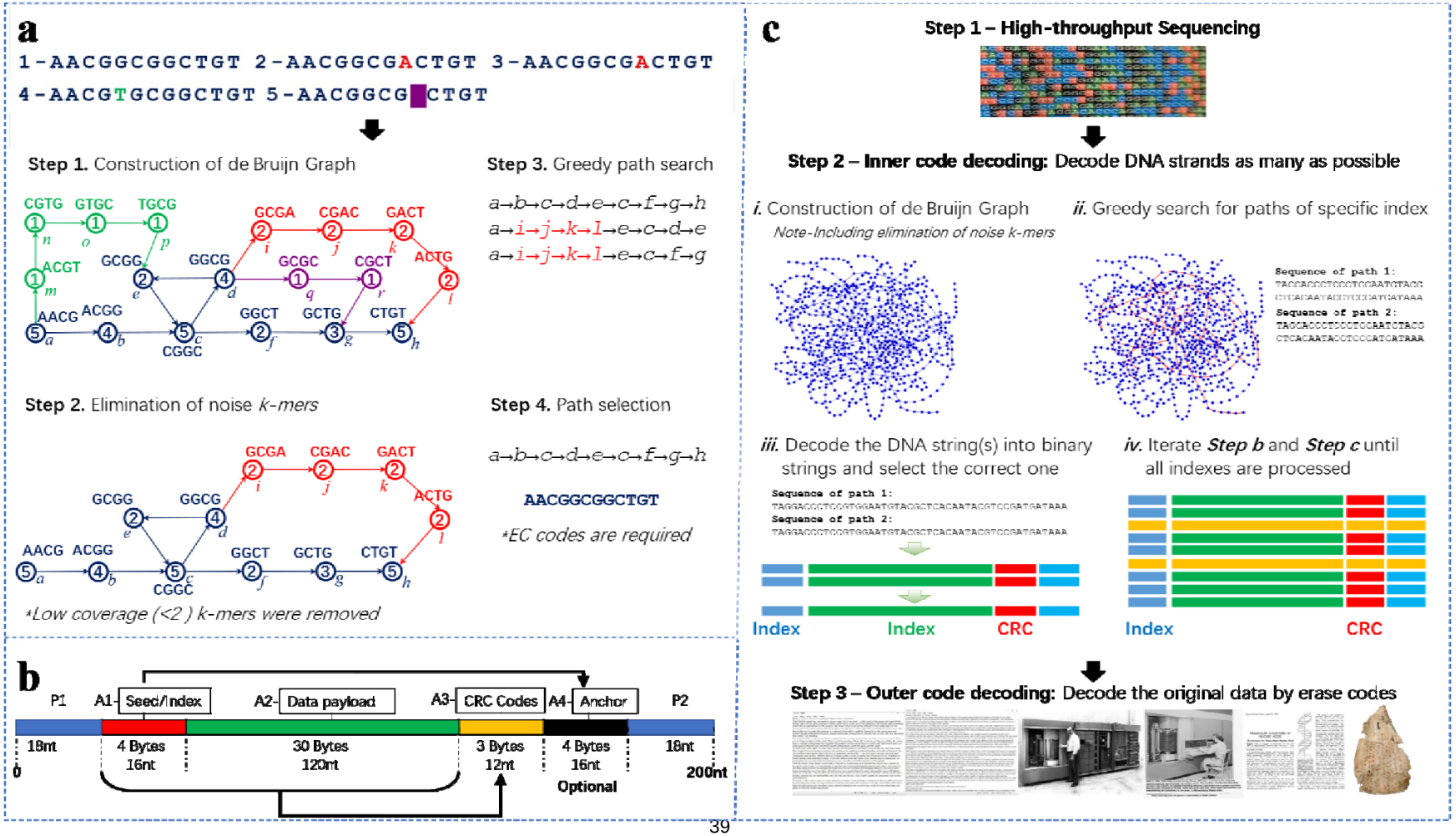

a
1-AACGGCGGCTGT 2-AACGGCGACTGT 3-AACGGCGACTGT
4-AACGTGCGGCTGT 5-AACGGCG CTGT
Step 1. Construction of de Bruijn Graph
Step 2. Elimination of noise k-mers
*Low coverage (<2) k-mers were removed
Step 3. Greedy path search
a→b→c→d→e→c→f→g→h
a→i→j→k→l→e→c→d→e
a→i→j→k→l→e→c→f→g
Step 4. Path selection
a→b→c→d→e→c→f→g→h
AACGGCGGCTGT
*EC codes are required
b
P1
A1- Seed/Index
A2- Data payload
A3- CRC Codes
A4- Anchor
P2
18nt
4 Bytes 16nt
30 Bytes 120nt
3 Bytes 12nt
4 Bytes 16nt
Optional
18nt
0
200nt
c
Step 1 – High-throughput Sequencing
Step 2 – Inner code decoding: Decode DNA strands as many as possible
i. Construction of de Bruijn Graph
Note-Including elimination of noise k-mers
ii. Greedy search for paths of specific index
Sequence of path 1:
Sequence of path 2:
iii. Decode the DNA string(s) into binary strings and select the correct one
iv. Iterate Step b and Step c until all indexes are processed
Index
Index
CRC
Index
CRC
Step 3 – Outer code decoding: Decode the original data by erase codes

Fig. 3 Experimental verification of the robustness of DBG-GPS. **a**, Procedure of the series deep error-prone PCR experiments. **b**, Multiple retrieval test with unspecific amplification PCR products. **c**, Accelerated aging experiments. These three experiments were performed with focus of indels, strand rearrangements and strand breaks respectively. Although super harsh conditions were applied, we could decode the original data correctly in all cases presented.

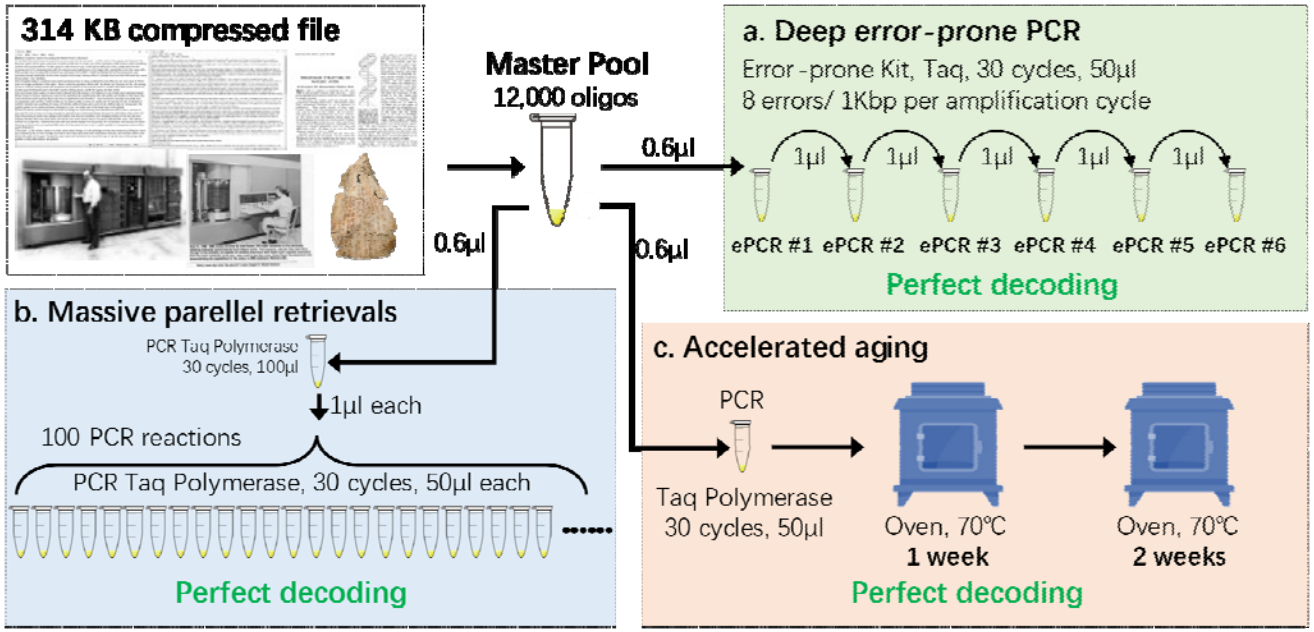


Fig. 4 Reliable data recovery with deep error-prone PCR products. a, Agarose electrophoresis analysis of the series error-prone PCR (ePCR) products. The first-round ePCR (#1) using 0.6 µl sample of master pool as templates. The following series ePCR used 1 µl of previous round PCR products as templates. The six ePCR were performed with highest error rate setting of 8 errors per 1,000 bp for each amplification cycle. *Note, the length of initially synthesized ssDNA is 200bp.* **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. After 180 cycles (30 cycles × 6 = 180 cycles) of error-prone amplifications, the data still can be correctly decoded using DBG-GPS.

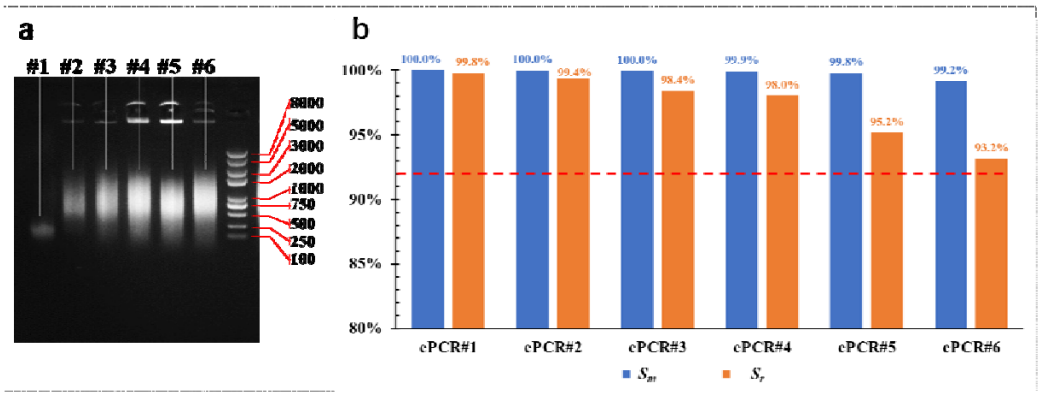


Fig. 5 Low-coverage decoding by random sub-sampling of sequencing reads.

All sequencing reads of each sample are used for random sub-sampling individually. For each coverage tested, we run 12 times of independent sub-sampling and the averages values were presented. The standard deviations are too small to be visualized. 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. **a**, K_d and K_n values. **b**, S_r values. The dashed curves in **b** 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. The blue dashed lines with arrows point out the data points that a *k*-mer elimination process was applied. The numbered red arrows show the coverages required by the two data sets for reliable data recovery by the outer Fountain codes.

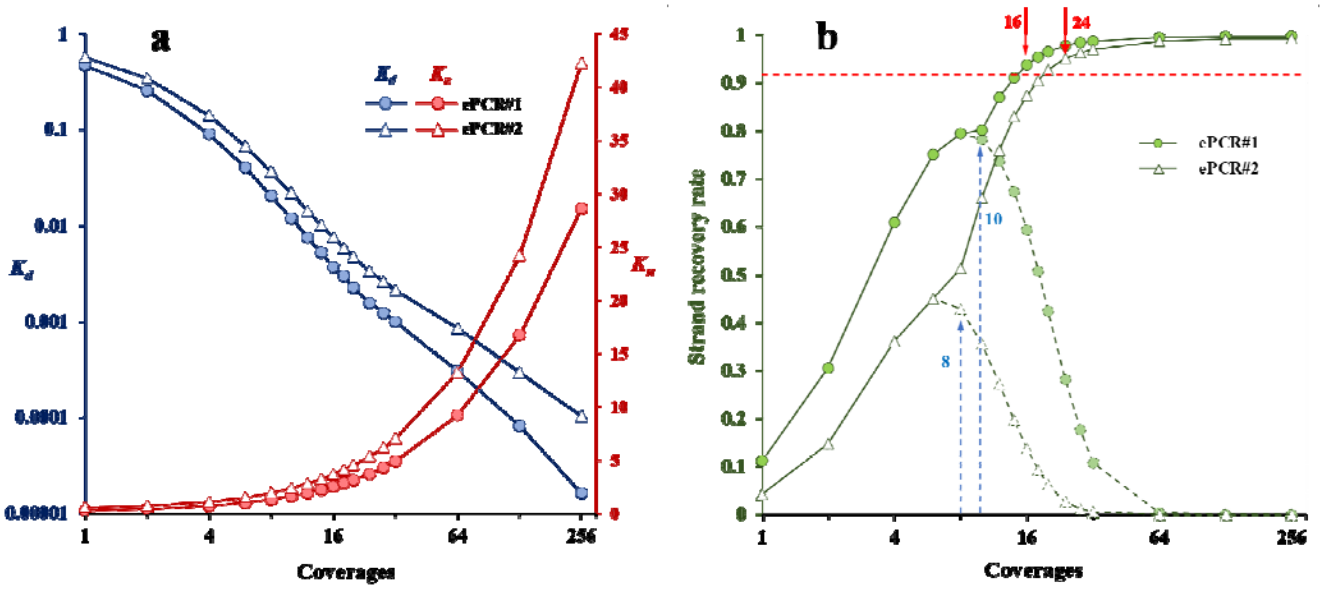
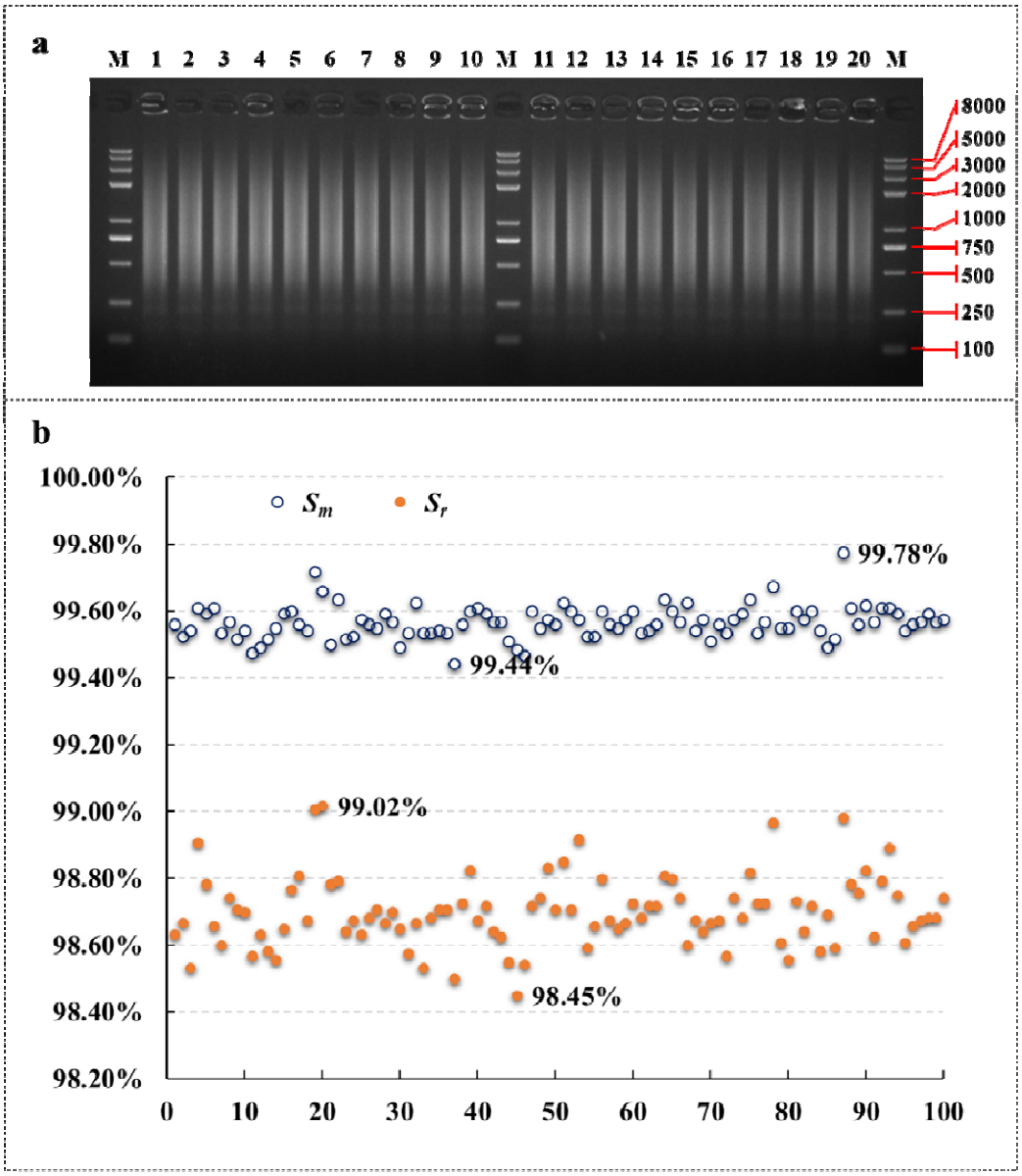
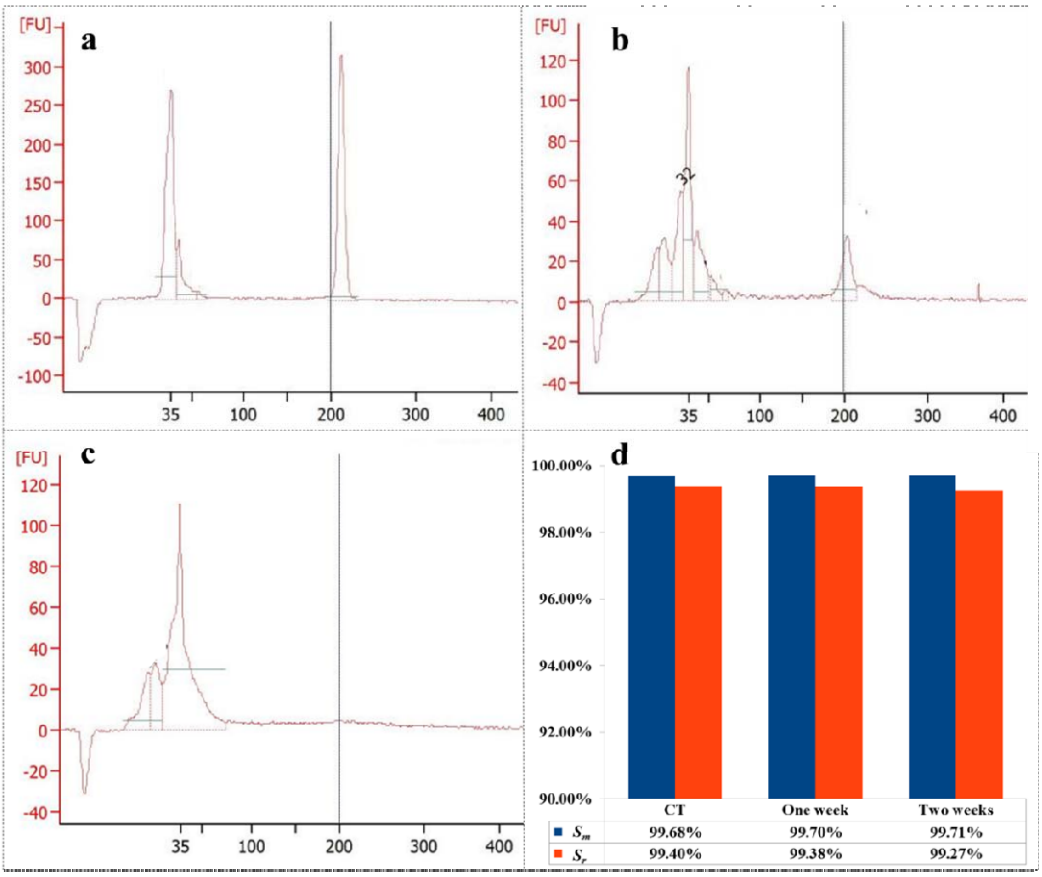


Fig. 6 Massive multiple retrieval test using PCR products with massive unspecific amplifications. a, Agarose electrophoresis analysis of the parallel PCR products. All PCR samples show similar results. Thus, only 20 of the total 100 samples are presented. **b**, Strand decoding results of the 100 retrievals using DBG-GPS. All samples show high quality strand decoding which support accurate data retrieval.



860

Fig. 7 Reliable data retrieval with accelerated aged DNA samples. a, **b**, and **c** are DNA integrity analysis results of DNA samples without accelerated aging, accelerated aged for one week, and accelerated aged for two weeks respectively. **d**, strand recovery results of the three samples. Although the integrity of the DNA strands was damaged seriously after accelerated aged at 70℃ for two weeks as shown in figure **c**, high strand recovery rate (S_r > 99%) was achieved by DBG-GPS based decoding. Indeed, no significant differences could be observed between the three samples, suggesting a robust decoding mechanism against DNA degradation.



Tab. 1 Decoding speed of DBG-GPS in comparison with clustering and multi-alignment (CL-MA) based method. The simulations were performed with 10,000 DNA strands in length of 300 bp. For each strand, fifty strand copies with an error rate of 4% were generated and utilized for decoding. The time of clustering stage in CL-MA based decoding was estimated based on a recent study by Cyrus Rashtchian *et al*.[23] The consensus calling stage used majority voting algorithm same as the recent study by Antkowiak, P.L. *et al.*[24] DBG-GPS shows more than 100 times faster than CL-MA methods (3890.26s/37.47s ≈103.82).

DBG-GPS		**CL-MA**	
Stages	**Time**	**Stages**	**Time**
k*-mer counting** ***(Jellyfish 32 threats)	0.63 ± 0.04s	**Clustering**[23] ***(32 threats)***	~0.36s
Strand decoding ***(Python Script)***	**36.84 ± 0.91s**	**Multi-alignment** ***(Muscle[25], IO time subtracted)***	**3874.48 ± 30.82s**
		Consensus calling ***(Majority voting, Python[24])***	15.42 ± 0.40s
Total	37.47 ± 0.95s	**Total**	~3890.26 ± 31.22s

Supplementary materials

Fig. S1 Traditional error correction codes cannot well handle indels due to the base shift problem. Erasure codes are designed for errors of unreadable data blocks without data shift issues. To correct indels by erasure codes, two-dimensional erasure codes must be applied. However, one indel in one dimension will lead to many substitution errors that have to be corrected by EC codes in another dimension. This leads to low efficiency of such strategy.

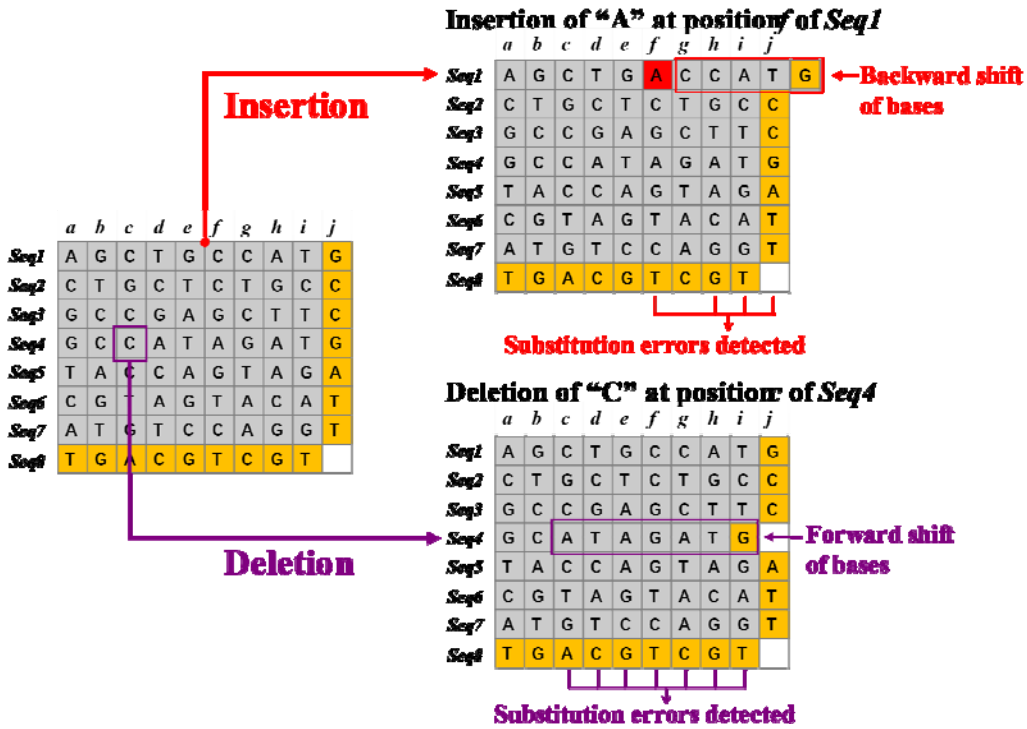
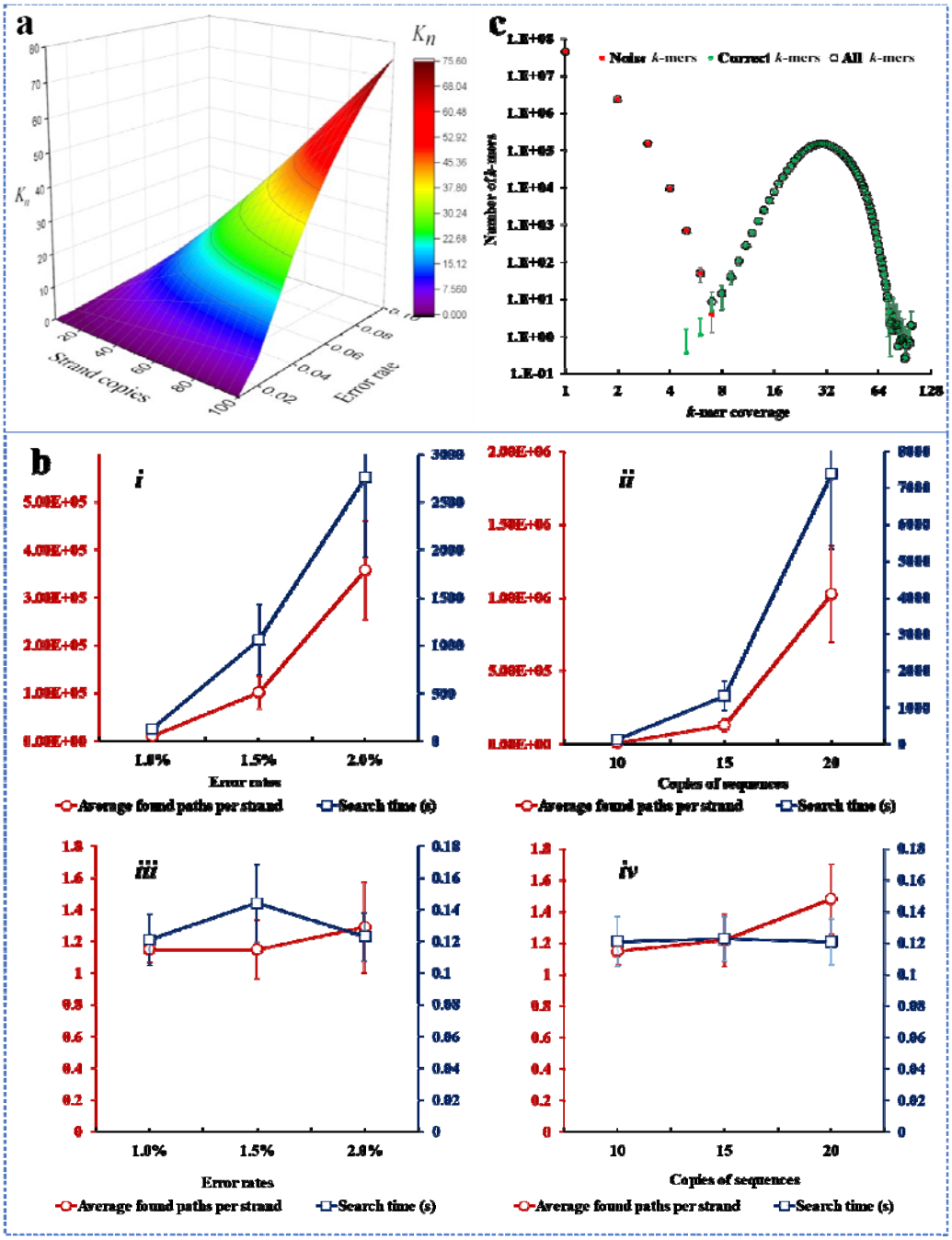


Fig. S2 Path search complexity caused by base errors can be remarkably released by elimination of low coverage *k*-mers. a – Massive noise *k*-mers introduced increase the de Bruijn graph scale remarkably. b - Greedy path search burden caused by noise *k*-mers can be remarkably reduced by elimination of low coverage *k*-mers. The number of paths found and search time increased exponentially as increasing the rates of errors (b-*i*) or the number of sequence copies (b-*ii*); The greedy search time and revealed number of paths decreased remarkably after elimination of low coverage *k*-mers (b-*iii* and b-*iv*). **c**, Coverage distributions of correct *k*-mers and noise *k*-mers revealed by monte carlo simulation. The simulation is performed with 10,000 DNA strands, a λ value (i.e. the expectation value) of 50 for strand copies, a base error rate of 4%.


a
K_n
Strand copies
Error rate
c
Noise k-mers
Correct k-mers
All k-mers
Number of k-mers
k-mer coverage
b
i
ii
iii
iv
Error rates
Copies of sequences
Average found paths per strand
Search time (s)

903

904

Fig. S3 Both-way search strategy shows fast decoding than one-way search. The simulation is performed with 10,000 DNA droplets. 100 sequence copies with random base errors in rates of 10% are generated for decoding test.

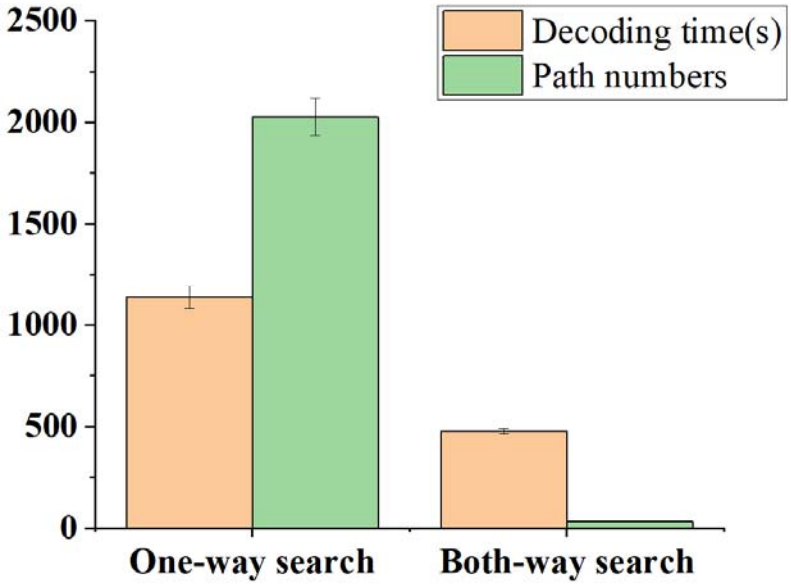


Fig. S4 Details of de Bruijn graph-based greedy searching algorithms. The detailed description is presented in **Methods** section.

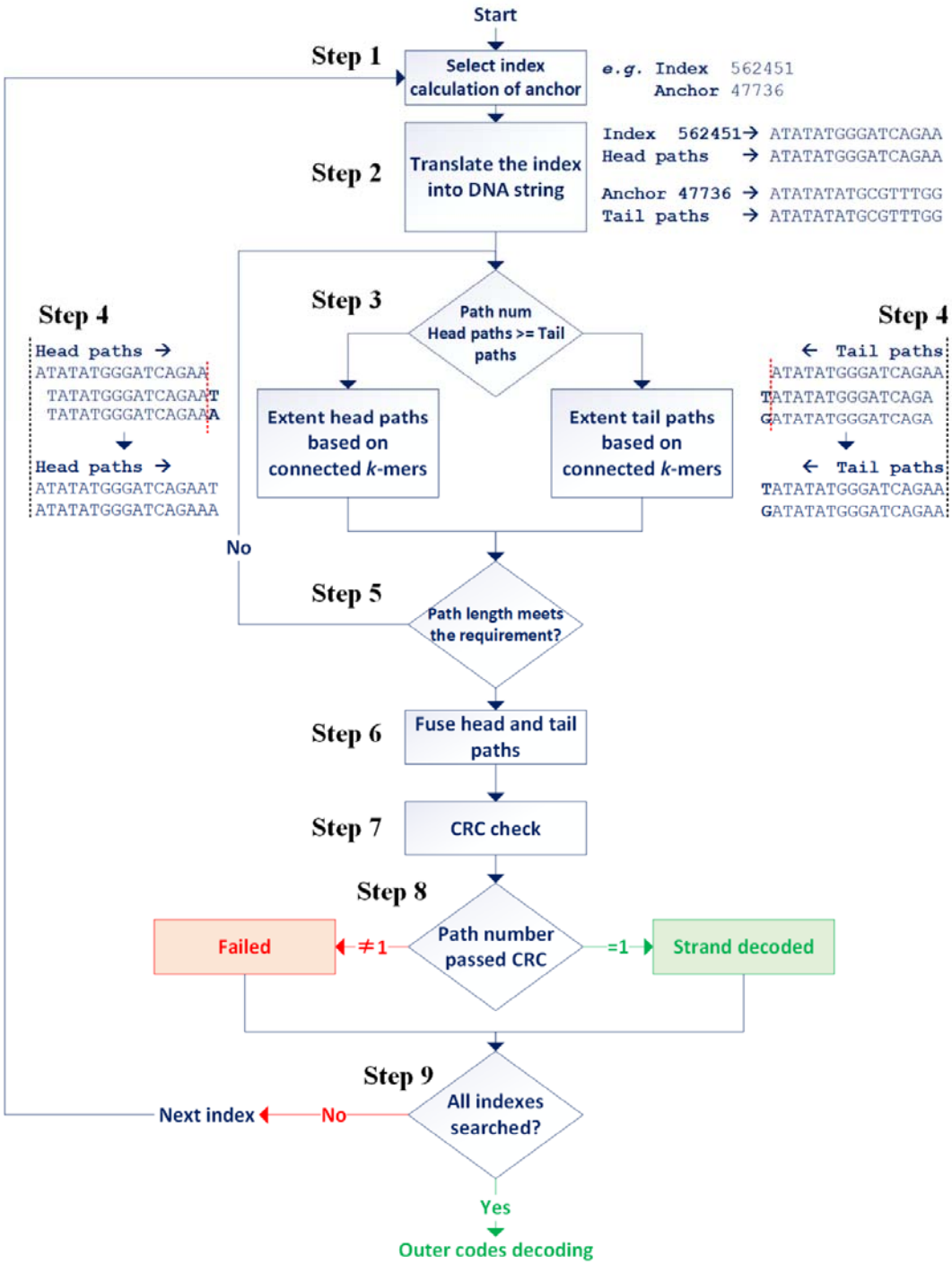
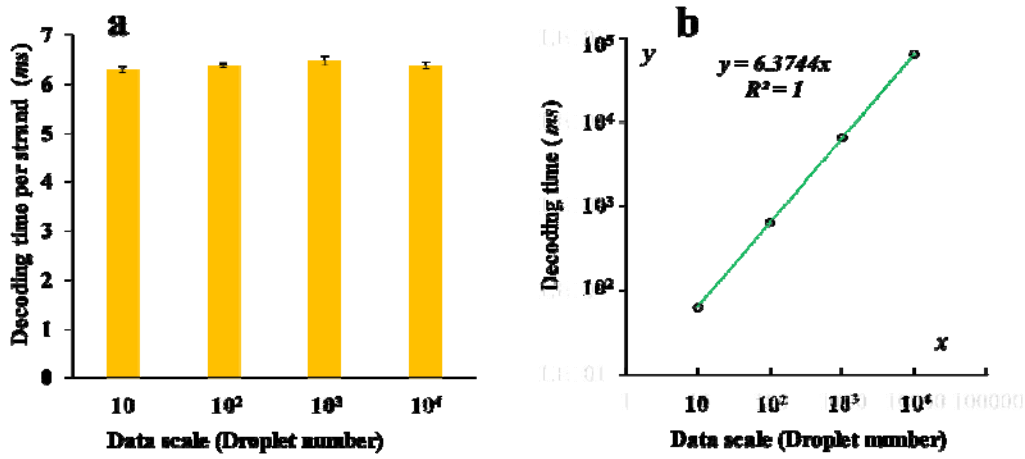


Fig. S5 Linear decoding complexity of DBG-GPS. a, Average decoding time per DNA strand. **b**, Linear fitting of decoding times with data scales.



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

Error Rates	Substitutions		Insertions		Deletions		Strand Breaks	
	S_m	SD	S_m	SD	S_m	SD	S_m	SD
1%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%
2%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%
3%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%
4%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%
5%	99.6%	0.1%	99.9%	0.1%	99.7%	0.1%	99.8%	0.0%
6%	96.7%	0.1%	98.7%	0.1%	97.3%	0.1%	98.1%	0.2%
7%	84.6%	0.3%	93.3%	0.3%	87.9%	0.1%	90.2%	0.3%
8%	58.7%	0.4%	77.0%	0.4%	64.7%	0.2%	70.0%	0.3%
9%	26.9%	0.6%	49.2%	0.8%	33.6%	0.5%	39.6%	0.3%
10%	7.3%	0.2%	21.0%	0.3%	10.6%	0.6%	13.7%	0.3%

Tab. S2 Performance of four-step DBG-GPS algorithms estimated by Monte-Carlo simulations. The simulations were performed with 10,000 DNA droplets produced by DNA fountain codes. For each DNA droplet strand, sixty error-rich copies were generated with variant error rates in range of 1% to 10% with a step length of 1%. For specific error rate considered, all simulated error-rich DNA strand sequences were submitted to the python implementation of DBG-GPS for strand decoding. Ten individual simulations were performed for each error rates. The average maximal strand recovery rates (S_m), and strand recovery rates (S_r) of DBG-GPS are listed.

Error rates	S_m	S_r
1%	100.00±0.00%	99.99±%
2%	100.00±0.00%	99.98±%
3%	100.00±0.00%	99.83±%
4%	100.00±0.00%	99.55±%
5%	100.00±0.00%	99.61±%
6%	100.00±0.00%	99.60±%
7%	100.00±0.00%	99.65±%
8%	99.99±0.01%	99.55±%
9%	99.96±0.02%	99.16±%
10%	99.70±0.07%	96.71±%

Tab. S3 Theoretical decoding limits of DBG-GPS with different *k*-mers sizes. The formula used for decoding capacity estimation was described in **Method** section.

***k*-mer sizes**	**10**	**20**	**30**	**40**
Theoretical decoding limits	0.5 MB	0.55 TB	57.6 PB	6×10^{5} EB

Tab. S4 Details of data retrieval results of deep ePCR experiments.

	Total reads	S_m	S_r
ePCR#1	19,972,663	100%	99.8% (11,978/12,000)
ePCR#2	17,430,377	100%	99.4% (11,928/12,000)
ePCR#3	17,075,696	100%	98.4% (11,808/12,000)
ePCR#4	20,614,695	99.9%	98.0% (11,764/12,000)
ePCR#5	16,651,758	99.8%	95.2% (11,420/12,000)
ePCR#6	18,281,133	99.2%	93.2% (11,179/12,000)