Super-robust data storage in DNA by de Bruijn graph-based decoding

Lifu Song[1,2], Feng Geng[3], Ziyi Gong[1,2], Bingzhi Li[1,2*], Yingjin Yuan[1,2*]

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

[2] SynBio Research Platform, Collaborative Innovation Center of Chemical Science and Engineering, School of Chemical Engineering and Technology, Tianjin University, Tianjin 300072, PR China

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

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

Abstract

High density and long-term features make DNA data storage a potential media. However, DNA data channel is a unique channel with unavoidable 'data reputations' in the forms of multiple error-rich strand copies. This multi-copy feature cannot be well harnessed by available codec systems optimized for single-copy media. Furthermore, lacking an effective mechanism to handle base shift issues, these systems perform poorly with indels. Here, we report the efficient reconstruction of DNA strands from multiple error-rich sequences directly, utilizing a De Bruijn Graph-based Greedy Path Search (DBG-GPS) algorithm. DBG-GPS can take advantage of the multi-copy feature for efficient

correction of indels as well as substitutions. As high as 10% of errors can be accurately corrected with a high coding rate of 96.8%. Accurate data recovery with low quality, deep error-prone PCR products proved the high robustness of DBG-GPS (314Kb, 12K oligos). Furthermore, DBG-GPS shows 50 times faster than the clustering and multiple alignment-based methods reported. The revealed linear decoding complexity makes DBG-GPS a suitable solution for large-scale data storage. DBG-GPS's capacity with large data was verified by large-scale simulations (300 MB). A Python implementation of DBG-GPS is available at https://switch-codes.coding.net/public/switch-codes/DNA-Fountain-De-Bruijn-Decoding/git/files.

Keywords

DNA data storage, de Bruijn graph, Indel correction, Erasure codes, Reed-Solomon, Fountain codes

Introduction

With properties of high density, low maintain cost, and long persistent period, DNA as storage media is a potential candidate for solving the world crisis of data explosion[1–4]. However, massive and complex errors that emerged make reliable data retrieval 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 codes handle the strand dropouts and the inner codes correct the base errors within DNA strands[5–7]. Strand dropouts are erasures based on the definition of information theory, which can be well solved by the sophisticated erasure codes, *e.g.* Fountain,

Reed-Solomon (RS) codes[5–7]. However, to the best of our knowledge, an effective inner code, which can well handle all three types of base errors, *i.e.* substitutions, insertions and deletions (indels), is still missing. The key obstacle is the difficulty in handling the indels. Although the field of information theory has developed elegant EC codes, they are not optimal for indels with base shift problem. Base deletions have been treated as erasures in previous studies[5–7]. However, other than unreadable bases, *i.e.* erasures, base deletions also cause shift forwards of downstream nucleotide bases. This base shift issue, which exists in 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 exact positions of the lost or odd 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[5,7]. However, this mechanism is low efficient in handling the indels. As illustrated in **Fig. S1**, 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 when the error rate is very low. When slightly more indels are introduced, the overwhelming 'substitutions' induced will quickly exhaust the column EC codes.

DNA data storage achieves data manipulations by processes of DNA synthesis, PCR, and sequencing. These processes always generate multiple copies of DNA strands fast and cheaply. This unique 'multiple copy' feature, also termed as "natural redundancy" in the previous studies, or 'data repetition' by definition of information theory[8], makes DNA data storage a unique channel[8,9]. In the view of traditional information theory, 'data repetition' is a low efficient way to

enhance the robustness of data. This deep-rooted idea is based on the fact that additional copies are time- and cost-expensive for traditional media and communication systems. However, as just mentioned, additional copies are cheap, fast, and basically unavoidable for DNA data storage. Indeed, it may increase the cost significantly to make a single copy DNA data. Efforts on reducing the “multiple copy” feature by serial dilutions show that massive strand dropouts are observed when reducing the average copy number blow ten, making reliable data retrieval impractical[10]. Additionally, single DNA molecules, as highly degradable polymers, break easily. Thus, even single copy DNA data is obtained, the data stability is still questionable since strand breaks, which cannot be handle by any reported codes, are fatal to DNA data integrity. Recently, *Press et. al.* implemented a hash encoded, decoded by greedy exhaustive search (HEDGES) codes which can correct indels with a single copy of DNA recently[11]. Despite HEDEGES shows outstanding technical achievements, the coding rate of HEDGES drops below 0.6 for correction of 3% errors. This would increase the cost of DNA data storage by 66.7% theoretically. To make use of the “multiple copy” feature, previous studies applied the algorithms of clustering and multi-alignment[7,12]. However, such strategy is not efficient and cannot scale up well. A new algorithm that can efficiently decode DNA strands from multiple error-rich strand copies is highly desirable.

Here, we describe the fast and accurate reconstruction of DNA strands from multiple error-rich sequences directly, utilizing a de Bruijn Graph-based Greedy Path Searching (DBG-GPS) algorithm. In combination with simple Cyclic Redundancy Check (CRC) codes for path selection, DBG-GPS can take advantage of the multi-copy feature for high-efficient correction of indels as

well as substitutions. While correction of multiple indels with single copy data is still a big challenge[13,14], As high as 10% of errors can be corrected in high coding rate (0.95) by DBG-GPS with merely sixty strand copies. More errors still can be corrected by the incorporation of additional sequence copies. Furthermore, DBG-GPS shows outstanding linear decoding complexity, which is essential for large-scale DNA data storage. DBG-GPS shows much faster decoding speed than the clustering and multiple alignment-based methods reported. We verified the robustness of DBG-GPS by both deep error-prone PCR experiments and large-scale *in silicon* simulations.

Results

Principles and potentials of de Bruijn graph-based decoding with multiple error-rich sequences

Decoding of DNA data is a process of re-assembling the original data from small pieces of information carried by massive short DNA fragments, which is quite similar to the process of genome assembly. However, the accuracy demand of data storage is much higher than genome assembly. Although it is not possible to obtain accurate genome sequences by simple assembly process, it is acceptable for genome sequencing applications since following up gap-filling experiments can be performed[15]. Obviously, gap-filling is not an acceptable option for data storage application. Fortunately, the extra options of indexes for sorting of fragmented information and redundancy codes for error correction bring opportunities to achieve this challenging goal. 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[15,16]. To reveal the potentials of de Bruijn graph theory in DNA data storage, we first analyzed the representations of substitution and indel errors in a de Bruijn graph as illustrated in **Fig. 1**. As shown in Fig. 1**A-I**, a simple de Bruijn graph on the right was derived from the four copies of "AACGGCGGCTGT" in the left with a *k*-mer size setting of four. The DNA sequence of "AACGGCGGCTGT" can be represented by the path of "$a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow c \rightarrow f \rightarrow g \rightarrow h$" in the graph. A base substitution error of "G" to "A" in sequence 2 will lead to a branch path, as shown in red color, in de Bruijn graph and the weights (coverages) of the correct path was reduced (**Fig. 1A-II**). In the case of base indels, which are quite different from base substitutions in the view of traditional information theory, the representations of base indels in de Bruijn graph are similar to base substitutions: both creating interfering path(s) and reducing the weight of the correct path (**Fig. 1A-III** and Fig. 1**A-IV**). A greedy path search algorithm and a path selection mechanism, which in combination can accurately find the correct path, could recover the DNA strand theoretically. In this case, the base 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, *i.e.* "$a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow c \rightarrow f \rightarrow g \rightarrow h$" in **Fig. 1**, is preserved, the corresponding DNA strand can be decoded theoretically.

Clearly, the sequence error rates and copy number determine the robustness of de Bruijn graph path. To further test the path robustness with variant rates and types of random errors, as well as sequence copies, which indicates the potentials of de Bruijn graph theory-based decoding, we utilized a Poisson model to generate DNA sequences with different rates and types of random errors. Variant numbers of the generated sequences with specific rates and

types of error were then committed for de Bruijn graph construction. The integrality of the correct path was then tested by checking the existence of all *k*-mers belong to this path in the constructed graph. To investigate the effluences of different types of errors to the path integrity, we firstly run the simulations with specific type of errors solely. With specific error rate and sequence copy, different DNA strands with a total number of 10,000 was applied for error-rich sequences generation and path integrity analysis. This process was iterated for 3 times independently. The average frequency that the paths maintain their integrality was defined as 'maximal strand decoding rate' ($\boldsymbol{S_m}$). $\boldsymbol{S_m}$ value is the theoretical maximal decoding rate which can be achieved by an actual decoding implementation. Although it has been reported that indels are much more challenging to handle compared with substitutions, no significant differences of $\boldsymbol{S_m}$ values could be observed between the different types of errors. Indeed, the simulated $\boldsymbol{S_m}$ values of substitutions is slightly lower than that of indels, as shown by the case with an error rate of 0.1 and sequence copies of 50 (**Fig. 1B**). In real situation, the emerged errors are mixtures of three types of errors. We then perform series simulations with an error type ratio of 50% substitutions, 25% insertions and 25% deletions. Total error rates in range of 1% to 10% with a step length of 0.5%, and sequence copy numbers in range of 5 to 100 with a step length of 5 were considered. As shown in **Fig. 1**, $\boldsymbol{S_m}$ >0.99 can be obtained with merely 60 sequence copies (red dashed line) with an arbitrary error rate below 10%, well proved the great potentials of the de Bruijn graph theory in DNA data.

Development of a general framework for de Bruijn graph-based decoding with multiple error-rich sequences

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

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

A serious problem emerged is the explosion of de Bruijn graph size caused by errors. About forty 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. S 2**). Greedy path search with this enlarged de Bruijn graph resulted with overwhelming number of possible paths, leading to unbearably long path search time. The enlarged de Bruijn graph contains massive noisy *k*-mers, i.e. *k*-mers introduced by base errors. To quantify the noisy *k*-mers, we introduce an indicator K_n, which is the ratio of noisy *k*-mers and correct *k*-mers in the de Bruijn graph. Clearly, more errors and strand copies will introduce more noisy *k*-mers to the de Bruijn graph. Monte carlo simulations show quick increase of K_n by increasing either the error rate or the sequence copies (**Fig. 2**A). The number of branch paths and decoding time increased exponentially by increasing the error rates (**Fig. 2B-*i***) or the strand copies (**Fig. 2B-*ii***). Thus,

a pre-processing step which can remove the noisy *k*-mers is highly desirable. Although DNA data operations can introduce massive errors to DNA data, these errors are low probability events individually. Thus, the coverages, *i.e.* the occurrences of noisy *k*-mers, should be significantly lower than the correct ones. In other words, the low coverage *k*-mers are more likely to be noisy *k*-mers. Preliminary testing with elimination of *k*-mers with a coverage cut-off of one significantly reduced the decoding burden of branch paths with no significant affections to the correct path integrity (**Fig. 2B-*iii*** and ***iv***). To further investigate the feasibility of coverage as an indicator for distinguishing noisy *k*-mers, we used a Poisson distribution model to simulate the copy number distribution of DNA strands. The coverage distribution of noisy *k*-mers and correct *k*-mers are then investigated based on Monte Carlo simulation. 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**)**. As shown in **Fig. S3**, vast majority of the noisy *k*-mers show coverages lower than ten, while the correct ones show coverages around fifty. Thus, a coverage cut-off of ten can effectively remove most of the noisy *k*-mers (**Fig. S3**), suggesting an effective mechanism for elimination of noisy *k*-mers.

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

Although elimination of low coverage *k*-mers can effectively remove most of the branch paths based on Monte Carlo simulation, a small fraction branch paths are conserved. For example, due to exponential replication feature of DNA data storage, errors emerged in early stage can be enriched massively which cannot be removed by low coverage cut-off. 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 codes, or hash functions. Although hash functions are ideal for such kind of tasks in principle, we failed to find a proper hash function that can generate short key words with merely 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, but do not have a built-in error-correction capability. Intensive Monte Carlo experiments show that CRC codes can recognize the correct path in all one million times of independent simulations, proving its effectiveness on selection of correct paths, *i.e.* correct strands.

A both-way search strategy increases decoding speed

Although elimination of noisy *k*-mers based on coverage can reduce the greedy path search complexity remarkably, significant amount of remained noisy *k*-mers cannot be removed, especially when error rates are high. To further reduce the greedy path search time, we proposed a both-way search strategy which can search for possible paths simultaneously on 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. 3C (A4)**. 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 generation of the anchors using the indexes as seeds. Due to the feature of pseudorandom function, it always gets a same anchor value when supply with the same seed. Thus, the greedy path search engineer 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. As shown in **Fig. S4**. With an error rate of 10% and copy number of 100, the both-way search strategy shows about 60% reduction in decoding time compared with one-way search. No significant difference in decoding accuracy is observed with the two search strategies.

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

For effective strand reconstruction with DBG-GPS, we suggested a four-step framework as illustrated in **Fig. 3A** and detailed as follows: ***Step 1***, construction of de Bruijn Graph. Here, the 'construction of de Bruijn Graph' refer to the process of 'counting of *k*-mers'. In current python implementation, a simple hash dictionary data structure was applied. For further studies with more advanced data structure of the de Bruijn Graph, we suggest to create the advanced data structure after elimination of noisy *k*-mers (Step 2). ***Step 2***, elimination of noise *k*-mers. In present study, we use coverage as an indicator for elimination of noisy *k*-mers. Additional codes for identification of noisy *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 embbed EC codes are then used for selection of the correct path. Only in case of one path passes the EC code checking, this path will be selected and the decoding of corresponding strand succeed.

Integration of DBG-GPS with outer erasure codes

DBG-GPS is designed as inner decoding mechanism for correction of errors within DNA strands. It can be easily combined with an erasure code, *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 inner decoding process for 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 basically the same as previous studies, we only illustrated the overall decoding process using DBG-GPS as inner decoding method as shown in **Fig. 3B**. 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. In previous studies, clustering and multiple-alignment processes, which are time and computation intensive, are utilized for strand reconstruction[7,12,17] . DBG-GPS can directly rebuild the strands from error-rich sequencing reads without clustering and multi-alignments. 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. Except the strand integrity verification codes, no additional codes are required by DBG-GPS. Most previous studies also employed redundancy codes for checking of strand integrity [6,7,12]. Thus, the DBG-GPS

decoding algorithm can be directly applied to these systems theoretically, *e.g.* DNA fountain codes[6], RS-based storage system[7,12] *etc*. However, to check the effectiveness of DBG-GPS on these systems, both the encoded strand sequences and the sequencing reads are required. The only study we found matches this criteria is the recent study by Antkowiak *et al.*[12]. Low-cost photolithographic synthesis in combination with a RS codec system is used for low cost data writing. However, they reported a serious problem of massive strand dropouts due to the error-prone synthesis process of photolithographic synthesis[12]. Interestingly, preliminary test using DBG-GPS's principles on the sequencing results of Antkowiak *et al.*'s study revealed a high theoretical strand recovery rate above 99%, a potential remarkable improvement compared with the reported recovery rate of 83%.

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 ∞), 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 noisy *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:

$$K_d = K_{lost}/K_{ori} \quad (1)$$

$$K_n = K_{all}/K_{corr} \quad (2)$$

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.

Decoding speed of DBG-GPS

DBG-GPS shows an excellent linear decoding complexity while increasing the total strand scale from 1 to 10,000 (**Fig. S6**). To compare the decoding speed of DBG-GPS with clustering and multiple alignment-based methods (CL-MA), 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 indicating perfect clustering. For decoding speed comparison, a clustering time was estimated based on the recent study by Cyrus *et al*.[17]. Since multiple alignment process was performed by a standalone program (Muscle[18]), there will be 10000 times of IO (input and output) with the hard disk system during strand decoding with 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 for three times and the average time is subtracted in the CL-MA based decoding analysis. As shown in **Tab. 1**, although a Python implementation of DBG-GPS was used in decoding test, DBG-GPS shows 53 times faster than the CL-MA based methods without significant differences in decoding accuracy. A C++ implementation of DBG-GPS can further enhance its performance.

Experimental verification of the robustness of DBG-GPS

As shown in **Fig. 4**, to experimentally verify the robustness of DBG-GSP, a 314 KB compressed file derived from 6 different types of files (two documents, four pictures, detailed in **Methods**), was encoded into 12,000 DNA strands in length of 200 bp with an overall strand redundancy of 14.6% by the DNA fountain codes implemented in this study. Each oligo encodes thirty bytes of data, as well as three bytes of CRC codes, four bytes of index, and four bytes of anchor. The designed DNA strands were synthesized by Twist Bioscience. The single strand DNAs (ssDNA) obtained from Twist Bioscience were dissolved in ddH_2O. About 1,000 copies of the dissolved ssDNA were used as

templates for PCR amplification. The PCR products are sequenced by Illumina technology. The sequencing reads were used for decoding of the original information by DBG-GPS.

In details, to better understand the robustness of DBG-GPS, we utilized both accurate PCR and error-prone PCR to amplify the ssDNA library obtained from Twist Bioscience. The accurate amplification of the ssDNA library was performed with high fidelity KAPA HiFi DNA Polymerase with 22 thermo cycles. Two series of error-prone PCR (ePCR) were performed with 30 thermo cycles using an error-prone PCR kit from TIANDZ using the highest error-rate settings. The second round ePCR use 1 µl products of the first round ePCR to introduce more errors. Thus, we obtained three PCR samples, *i.e.* accurate PCR (A-PCR), two series of ePCR (E-PCR-1st, E-PCR-2st). The three samples were sequenced by an Illumina NovaSeq platform and 150 bp paired-end reads were generated. This result with 6,164,085, 19,972,663, and 17,430,377 raw reads respectively. We first tried to recover the original information by using all sequencing reads of each sample independently. The S_m values of the three sequencing samples were estimated to be 99.33% (11950), 100% (12000), and 99.98% (11998) respectively. With optimized parameters for elimination of noisy *k*-mers, the DBG-GPS recovered 11,915 (99.29%), 11,978 (99.8%), and 11,928 (99.4%) strands for the three samples respectively. Although detailed analysis shows the ePCR products do have more base errors than the accurate PCR products, it is surprising that the two ePCR samples show higher data quality than the accurate PCR samples regarding to the strand decoding rates.

To better look into the data qualities and prove the robustness of DBG-GPS, we perform random sub-sampling of all sequencing reads with different

coverages, *i.e.* reading copies, range from 1 to 256. The strand recovery rates, K_d, and K_n in different coverages were calculated corresponding. For each coverage tested, three independent random sub-sampling experiments were performed. As shown in **Fig. 5A**, all three samples can be reliably decoded in low coverages (coverage of 8 for A-PCR, coverage of 12 for E-PCR-1st, coverage of 16 for E-PCR-2st). It is worth to point out that low quality reads are removed before decoding in reported systems. In this study, raw reads are directly supplied to DBG-GPS for decoding without any pre-treatments proving the capacity of DBG-GPS in handling low quality reads. It is also interesting to notice that although the strand recovery rate of A-PCR is higher than the ePCR examples at low coverages, the recovery rates of E-PCR-1st and E-PCR-2st overtake that of A-PCR at coverages of 16 and 24 respectively, suggesting a more uniformed distribution of DNA strands in the ePCR products. As shown by red colour in **Fig. 5B**, the K_n values of ePCR products are higher than A-PCR with E-PCR-2st as the highest one as expected. The K_n differences among the three samples become more and more significant when increasing the coverages. The K_d value of A-PCR is lower than the two ePCR samples and E-PCR-2st shows the highest K_d value at low coverage. All K_d values of the three samples dropped quickly while increasing the coverages. The two ePCR samples shows faster dropping of K_d value than the A-PCR sample. At a coverage of 6, the K_d values of ePCR samples all become lower than the K_d value of A-PCR, which is consistent with the strand recovery rate results. The higher K_d values of ePCR products at low coverages are caused by the base errors introduced by error-prone PCR, which is consistent with the K_n curves.

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

Considering the expensive array-based DNA synthesis, we performed large-scale simulations to test DBG-GPS's performance with large-scale data sets. An ubuntu server 20.10 installation disk image (~ 1 GB) was used as input file for 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 10 million DNA droplets, representing a data scale of 600 mega bytes (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. The usage of swap disk makes the counting extremely slow. We thus gave up the decoding simulation on this data scale. We then reduced the DNA droplet number to 5 million, representing 300 MB of input data. 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 provide two strategies. The first strategy is applying a filtering process during 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 founds 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 for three times independently. On average 4,971,880 DNA droplets (recovery rate = 99.44%) were successfully recovered by DBG-GPS.

Discussion

In the forms of multiple error-rich strands, ‘data reputation’ is widely presented in DNA data storage. However, no effective mechanism can utilize this data ‘reputation’ feature for robust data storage due to the challenge of handling the massive errors. In this study, we developed an effective mechanism, named ‘DBG-GPS’, which can efficiently reconstruct the DNA strands from multiple, error-rich sequences based on de Bruijn Graph theory. 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 or arbitrary mixtures of three types of errors, using merely 40 copies of sequences. More errors still can be corrected by incorporating more sequences for decoding. The great linear decoding complexity revealed makes DBG-GPS an ideal solution for large-scale DNA data storage. Furthermore, DBG-GPS shows more than fifty times faster than the CL-MA based methods. The robustness of DBG-GPS has been verified by both error-prone PCR experiments (314KB) and large-scale (300MB) *in silicon* simulations. It is worth to point out that while previous systems used high-quality reads for decoding due to their incapacity of handling low-quality reads[6,7,12], raw reads were directly used for decoding without any pre-processing step to remove the low quality reads in this study. Furthermore, due to the unique decoding mechanism, DBG-GPS does not rely on the DNA strand integrity for data recovery. Preliminary test using single-end reads for decoding, simulating all

DNA strands with at least one break, shows no differences with pair-end reads in decoding quality. DNA strand breaks are key threads to DNA data integrity. Thus, although further studies are required, we strongly believe that DBG-GPS has great potentials in promoting the data reliability of long-term storage. Finally, DBG-GPS can work with low-quality PCR products with massive DNA fragments with an incorrect length as proved by error-prone PCR, deep copy experiments. 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 expensive. 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 640GB memories installed due to memory limitation. *C*++ implementation with more advanced *k*-mer data structure is expected to significantly reduce the memory cost, as well as decoding time[19,20]. Furthermore, novel error correction codes that can incorporate with the noisy *k*-mers elimination and greedy path search steps can further boost the capacity of DBG-GPS.

Methods

Library design, PCR and Sequencing

A DNA library of 12,000 oligonucleotides in length of 200nt was synthesized by Twist Biosciences. For accurate amplification of DNA library, KAPA HiFi HotStart ReadyMix was used with thermo cycle parameters: 98□ for 30s, 98□ for 15s, 60□ for 15s, 72□ for 15s, 72□ for 15s, 22 cycles. The ssDNA of synthesized DNA library with 1,000 copies was used as templates for replication. 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. Two series error-prone PCR were performed to introduce high rates of errors. The first round PCR used the synthesized ssDNA with an average copy number about 1,000 copies of each strand as templates. The second round PCR used 1μl products of first round PCR 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 200bp. 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 an Illumina NovaSeq platform and 250 bp paired-end reads were generated.

Counting of *k*-mers and elimination of low coverage *k*-mers

De Bruijn graph is quite memory expensive. The python implementation provided in this study used simple hash dictionary structure 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[19], 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 by Jellyfish can be introduced to the DeBruijnGraph() object for further decoding test using DeBruijnGraph-openDump('file name') function.

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. S1**. 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 highly 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 ten times larger than the actually encoded data.

Greedy path search

The greedy search process was illustrated in **Fig. S5**. 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 exactly 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’.

Large-scale simulations

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/public/switch-codes/DNA-Fountain-De-Bruijn-Decoding/git/files.

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/).

All sequencing data can be found in the following links:

Sequencing raw reads of accurate amplification:

https://pan.baidu.com/s/1XxvHwZR_B8UNBZKzyBBE2A (forward reads, fetch codes: ATGC, MD5: b0e6ff9820c090a22c0ed360018c8e46)

https://pan.baidu.com/s/1fr23i-MoliMhjCqWwCBN_w (reverse reads, fetch codes: ATGC, MD5: a97e8066d47052140422f501992fdacf)

Sequencing raw reads of two-rounds of error-prone PCR:

https://pan.baidu.com/s/1nNvWolx85cpbXXnigMuyQw (first-round error-prone PCR, forward reads, fetch codes: ATGC, MD5: 640142956d03f2de1b495e4cb55d37d5)

https://pan.baidu.com/s/14JMO-CEd9udfab7W-hkROA (first-round error-prone PCR, reverse reads, fetch codes: ATGC, MD5: ddb5d12e158316e6cd508b45b12e9469)

https://pan.baidu.com/s/1SZsy0-HP-lWxPq2ZNmdwlg (second-round error-prone PCR, forward reads, fetch codes: ATGC, MD5: 7cd61814025f372aef2de48186f0c364)

https://pan.baidu.com/s/1B8ZbTgrAXs2d04KcbXdyyQ (second-round error-prone PCR, reverse reads, fetch codes: ATGC, MD5: b64cc129d7446623437481474624095a)

Acknowledgements

This work was supported by Seed Foundation of Tianjin University and Natural Science Foundation of Shandong Province (ZR2017LC006).

Competing interests

The authors have declared that no competing interests exist.

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. 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).

2. 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).

3. 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).

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

5. 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).

6. 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).

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

8. 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).

9. 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).

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

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. 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).

13. Brakensiek, J., Guruswami, V. & Zbarsky, S. 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.

14. Smagloy, I., Welter, L., Wachter-Zeh, A. & Yaakobi, E. in *2020 IEEE International Symposium on Information Theory (ISIT)* (IEEESunday, June 21, 2020 - Friday, June 26, 2020), pp. 775–780.

15. 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).

16. 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).

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

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

19. 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).

20. 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. ***a.*** Simple de Bruijn graph constructed from 12bp sequence of “AACGGCGGCTGT” with four identical copies. (*k*-mer size = 4). The circles stand for the *k*-mers and the numbers inside the circles are coverages of corresponding *k*-mers. ***b.*** De Bruijn graph of four sequence copies of “AACGGCGGCTGT” with one substitution in sequence 2; ***c.*** De Bruijn graph of four sequence copies of “AACGGCGGCTGT” with one insertion and one deletion on sequence 2 and 3 respectively; ***d.*** De Bruijn graph of four sequence copies of “AACGGCGGCTGT” with one strand break on each sequence. **B** - Simulated maximal recovery rates with vary copies of DNA strands with 10% different types of errors respectively. The number of sequence copies is in range of 1 to 60 with a step size of 1. High error rate of 10% can be tolerated with merely 40 sequence copies. **C** - Simulated maximal recovery rates using ten copies of DNA strands with different types and rates of errors; During the simulation, artificial random errors are introduced into ten copies of sequences with specific type and rate of errors from 1% to 10% with a step size of 1%. The sequences with artificial errors were then used for construction of the de Bruijn graph. The De Bruijn graph path of the correct sequence was checked. This process is iterated for 10,000 times. The maximal decoding rates are the rates of “successful decoding events” during simulations. $\boldsymbol{S_m}$ stands for “maximal strand decoding rate”, which is calculated by checking the integrity of all strand paths.

721

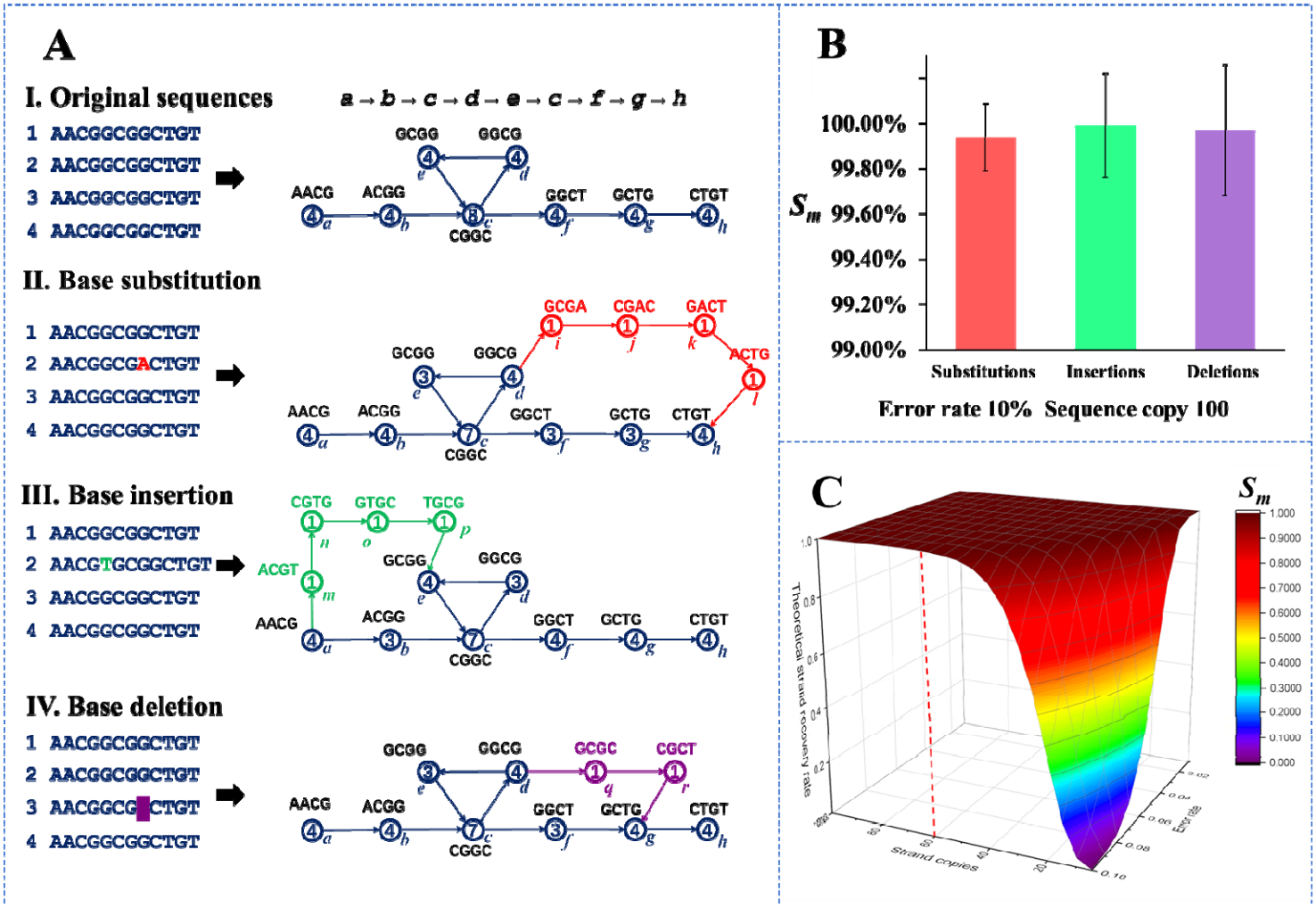
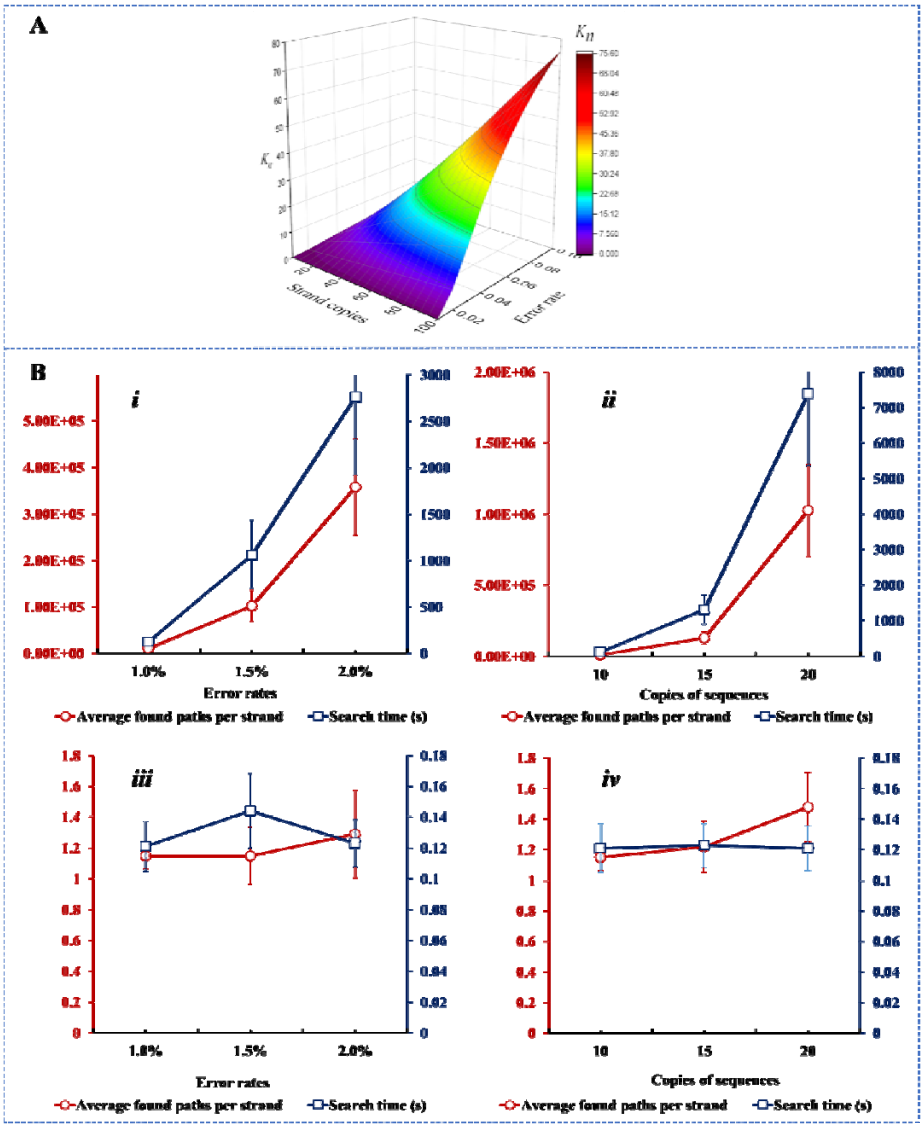

A
I. Original sequences
a → b → c → d → e → c → f → g → h
1 AACGGCGGCTGT
2 AACGGCGGCTGT
3 AACGGCGGCTGT
4 AACGGCGGCTGT
GCGG
GGCG
AACG
ACGG
GGCT
GCTG
CTGT
CGGC
II. Base substitution
1 AACGGCGGCTGT
2 AACGGCGACTGT
3 AACGGCGGCTGT
4 AACGGCGGCTGT
GCGA
CGAC
GACT
ACTG
III. Base insertion
1 AACGGCGGCTGT
2 AACGTGCGGCTGT
3 AACGGCGGCTGT
4 AACGGCGGCTGT
CGTG
GTGC
TGCG
ACGT
IV. Base deletion
1 AACGGCGGCTGT
2 AACGGCGGCTGT
3 AACGGCGCTGT
4 AACGGCGGCTGT
GCGC
CGCT
B
100.00%
99.80%
99.60%
99.40%
99.20%
99.00%
Sm
Substitutions
Insertions
Deletions
Error rate 10% Sequence copy 100
C
Sm
Theoretical strand recovery rate
Strand copies
Error rate

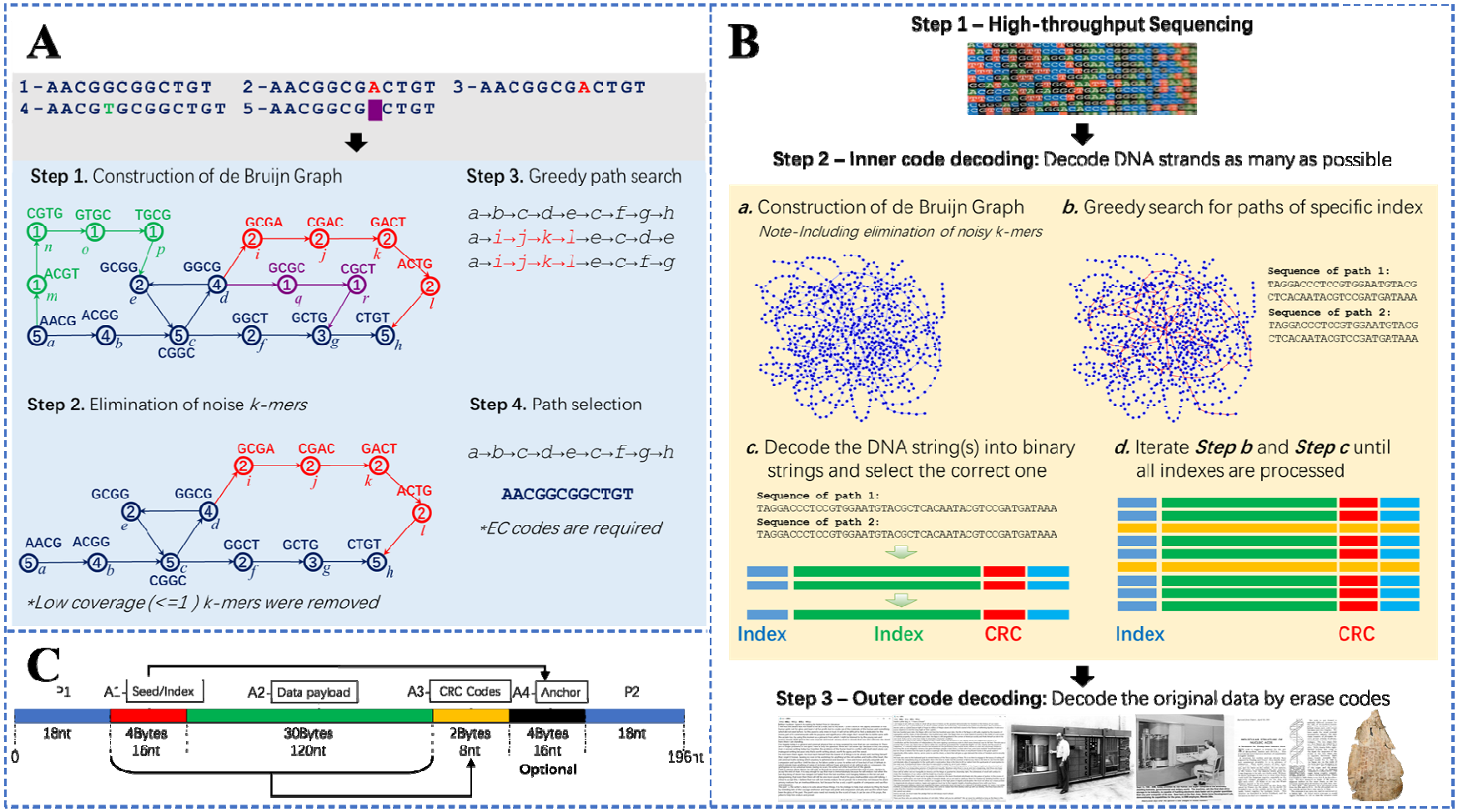
Fig. 2 Path search complexity caused by base errors can be significantly released by elimination of low coverage *k*-mers. A – Massive noisy *k*-mers are introduced by sequence errors. **B** - Greedy path search burden with the enlarged de Bruijn graph can be remarkably reduced by elimination of low coverage *k*-mers. The number of paths found and search time increased exponentially as the rates of errors (**B-*i***) or the number of sequence copies (**B-*ii***) increased; The greedy search time and revealed number of paths decreased remarkably after elimination of low coverage *k*-mers (**B-*iii*** and **B-*iv***).


A
K_n
K_c
Strand copies
Error rate
B
i
ii
iii
iv
Error rates
Copies of sequences
Average found paths per strand
Search time (s)

732

Fig. 3 Framework of DBG-GPS and its combination with an erasure outer code for large-scale data storage solutions

A – A proposed general framework for de Bruijn graph-based strand decoding with multiple error-rich sequences. For correct reconstruction of DNA strand sequences, we propose a four-step framework including: 1. Construction of de Bruijn Graph; 2. Elimination of noisy *k*-mers; 3. Greedy path search; 4. Path selection by traditional EC codes. **B** –Integration of DBG-GPS with an erasure outer codes for large-scale data storage solutions. **C** - Structure of the DNA strand. There are four parts in the designed DNA droplet strand: A1 – The index of the DNA droplet. The seed of the DNA strand can be calculated by pseudorandom function using the value of A1 as random seed. codes which is used for inferring the indexes of the data chunks involved in the present droplet; A2 – Data encoding part; A3 – Cyclic Redundancy Check codes (CRC) which are used for identification of the correct path if multiple paths were found after the path searching using the A1 as an index; A4 – Anchor cods which is generated using specific hashing function based on the value of A1.


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 (<=1) 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
Step 1 – High-throughput Sequencing
Step 2 – Inner code decoding: Decode DNA strands as many as possible
a. Construction of de Bruijn Graph
Note-Including elimination of noisy k-mers
b. Greedy search for paths of specific index
Sequence of path 1:
TAGGACCCTCCGTGGAATGTACG
CTCACAATACGTCCGATGATAAA
Sequence of path 2:
TAGGACCCTCCGTGGAATGTACG
CTCACAATACGTCCGATGATAAA
c. Decode the DNA string(s) into binary strings and select the correct one
Sequence of path 1:
TAGGACCCTCCGTGGAATGTACGCTCACAATACGTCCGATGATAAA
Sequence of path 2:
TAGGACCCTCCGTGGAATGTACGCTCACAATACGTCCGATGATAAA
Index
Index
CRC
d. Iterate Step b and Step c until all indexes are processed
Index
CRC
Step 3 – Outer code decoding: Decode the original data by erase codes
C
P1
A1 Seed/Index
A2 Data payload
A3 CRC Codes
A4 Anchor
P2
18nt
4Bytes 16nt
30Bytes 120nt
2Bytes 8nt
4Bytes 16nt Optional
18nt
0
196nt

Tab. 1 Decoding speed of DBG-GPS in comparison with clustering and multi-alignment (CL-MA) based method. The time of clustering stage in CL-MA based decoding was estimated based on a recent study by Cyrus Rashtchian *et al.*[17] The consensus calling stage used majority voting algorithm used in a recent study by Antkowiak, P.L. *et al.*[12]

DBG-GPS		**CL-MA**	
Stages	**Time**	**Stages**	**Time**
k*-mer counting** ***(Jellyfish Single CPU)	31.33 ± 0.65s	**Clustering**[17]	~11.45s
Strand decoding ***(Python Script)***	**36.84 ± 0.91s**	**Multi-alignment** ***(Muscle[18], IO time subtracted)***	**3874.48 ± 30.82s**
		Consensus calling ***(Majority voting[12])***	15.42 ± 0.40s
Total	73.17 ± 1.56s	**Total**	~3901.35s

Fig. 4 Experimental verification the robustness of DBG-GPS by error-prone PCR.

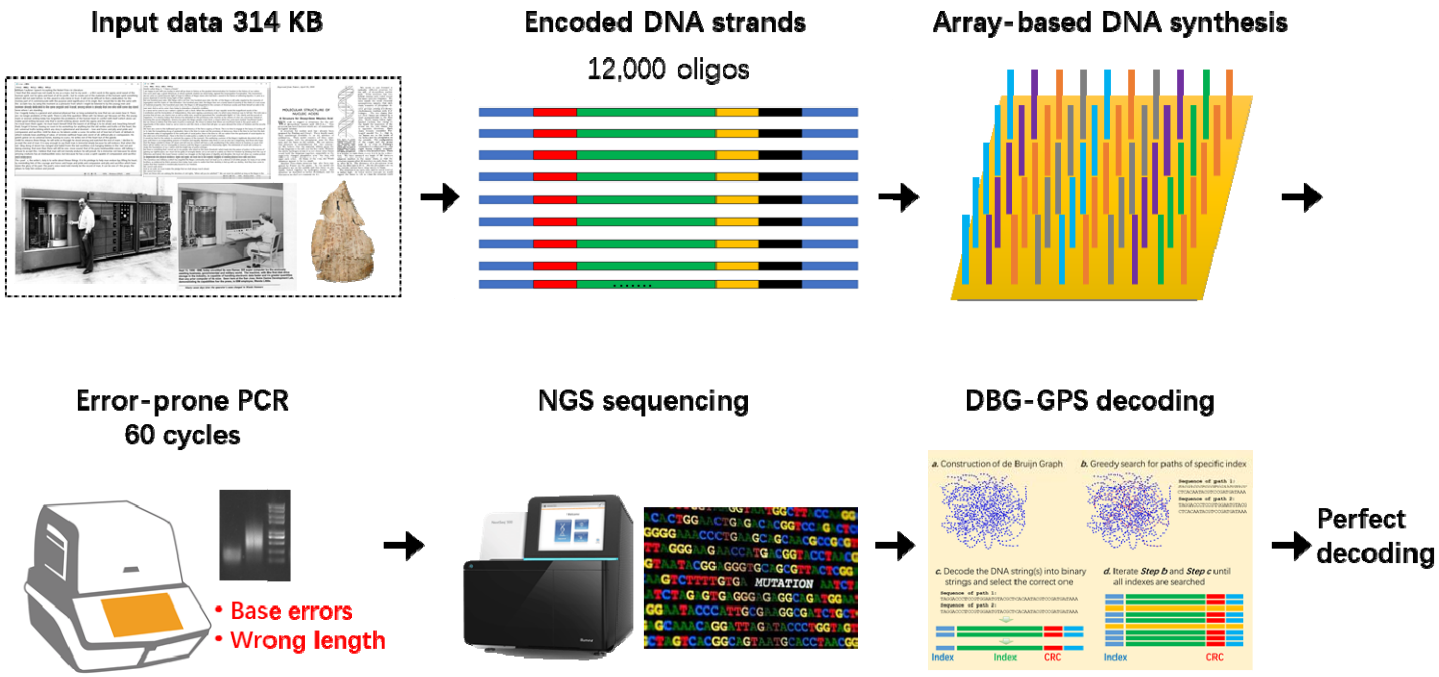
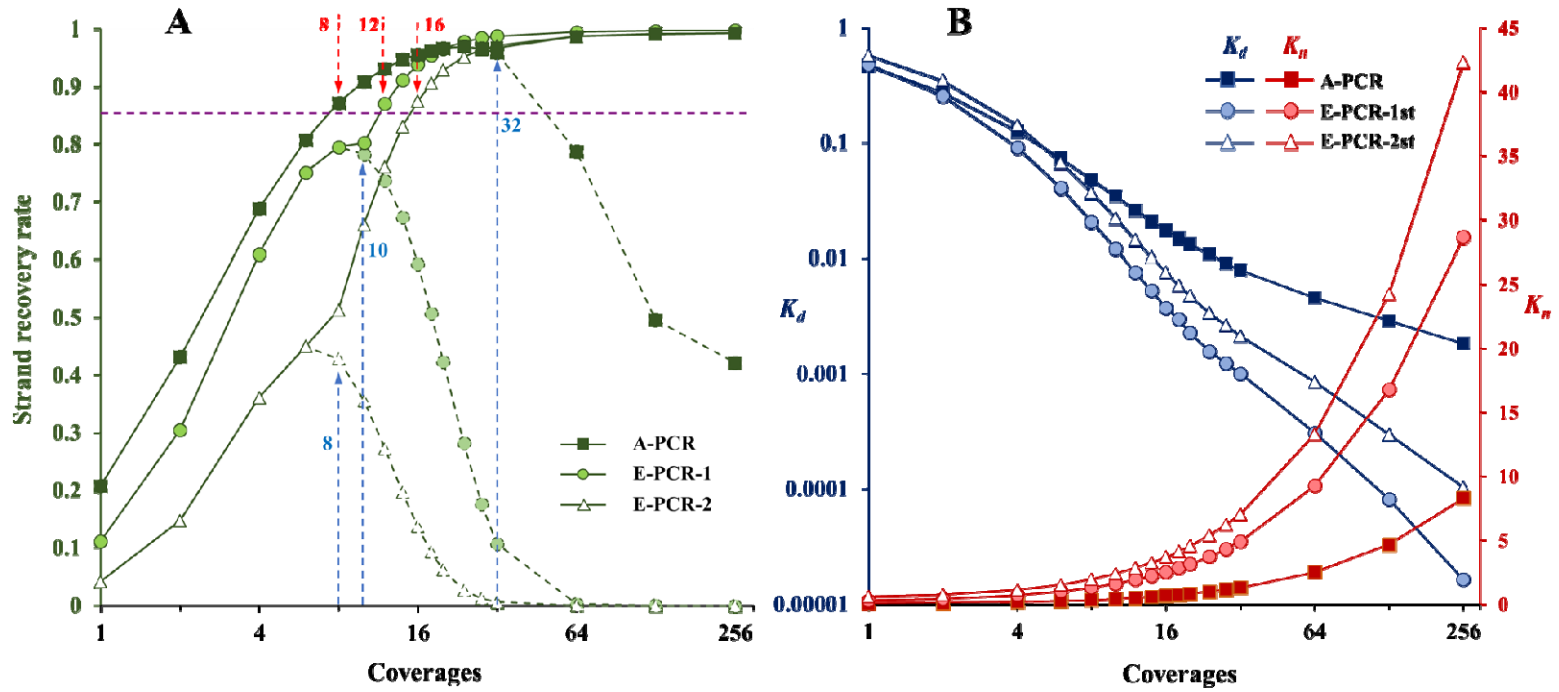


Fig. 5 Low-coverage data decoding by random sub-sampling of all sequencing reads.

For each coverage, we run 12 times of independent sub-sampling. The average strand recovery rates (**A**), K_d and K_n values (**B**) calculated with variant coverages illustrated. The dashed curves in Fig. A are the results without elimination of low coverage *k*-mers. The horizontal dashed purple line stands for the decoding limits by the outer Fountain codes. The numbered dashed red line with arrows point out the data point that data can be recovered by the outer Fountain codes and the numbers are the corresponding minimal coverages for decoding. The blue dashed lines with arrows point out the data points that a *k*-mer elimination process was applied.


A
B
Strand recovery rate
Coverages
A-PCR
E-PCR-1
E-PCR-2
8
12
16
10
32
K_d
K_n
E-PCR-1st
E-PCR-2st

759

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 have to be corrected in another dimension making the efficiency of such treatments very low.

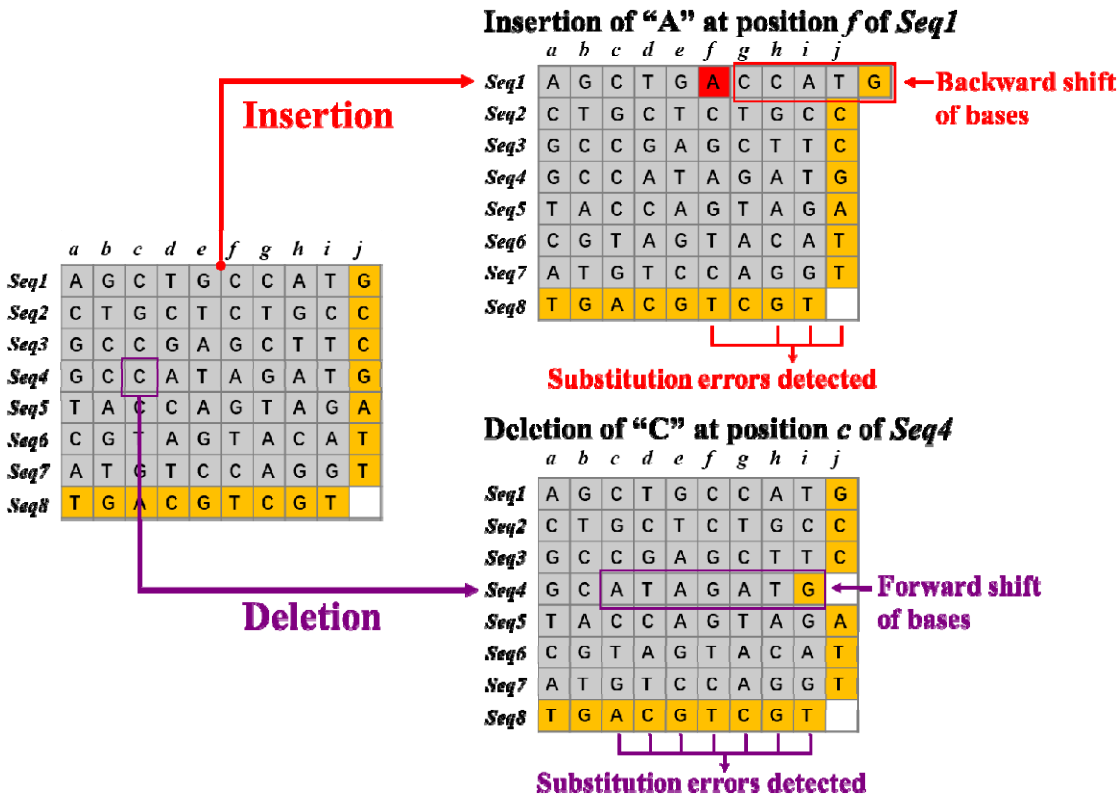


Fig. S 2 The sizes of de Bruijn graph increased remarkably by increasing the error rates (A) or the number of sequence copies (B).

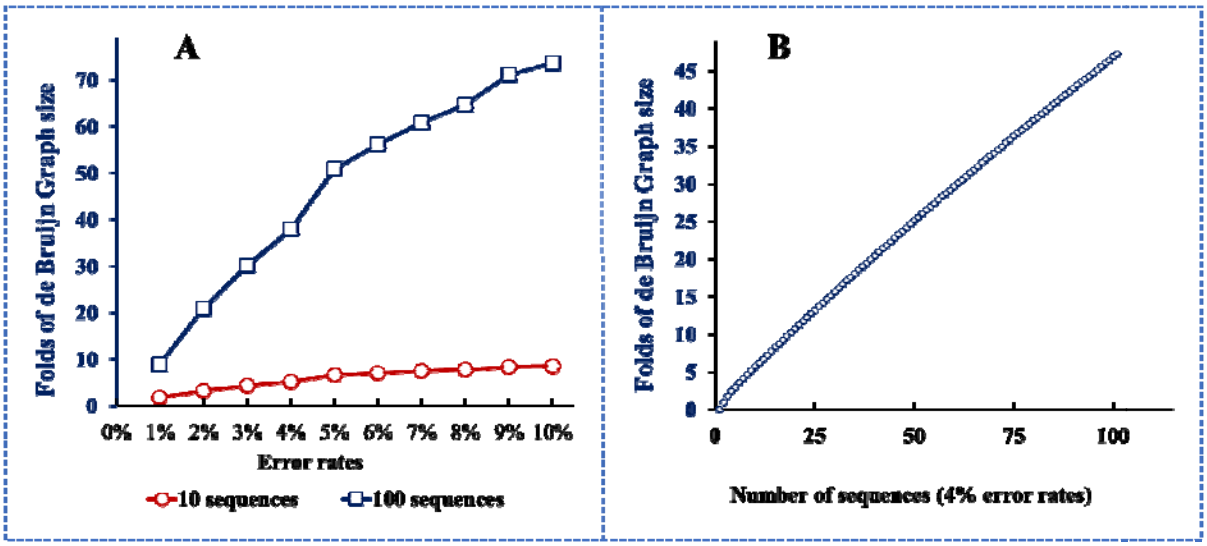


Fig. S3 Coverage distributions of correct *k*-mers and noisy *k*-mers revealed by monte carlo simulation. The simulation is performed with 10,000 DNA strands, a λ value (i.e. the expectation value) of 50 for strand copies, a base error rate of 4% (a common error rate)

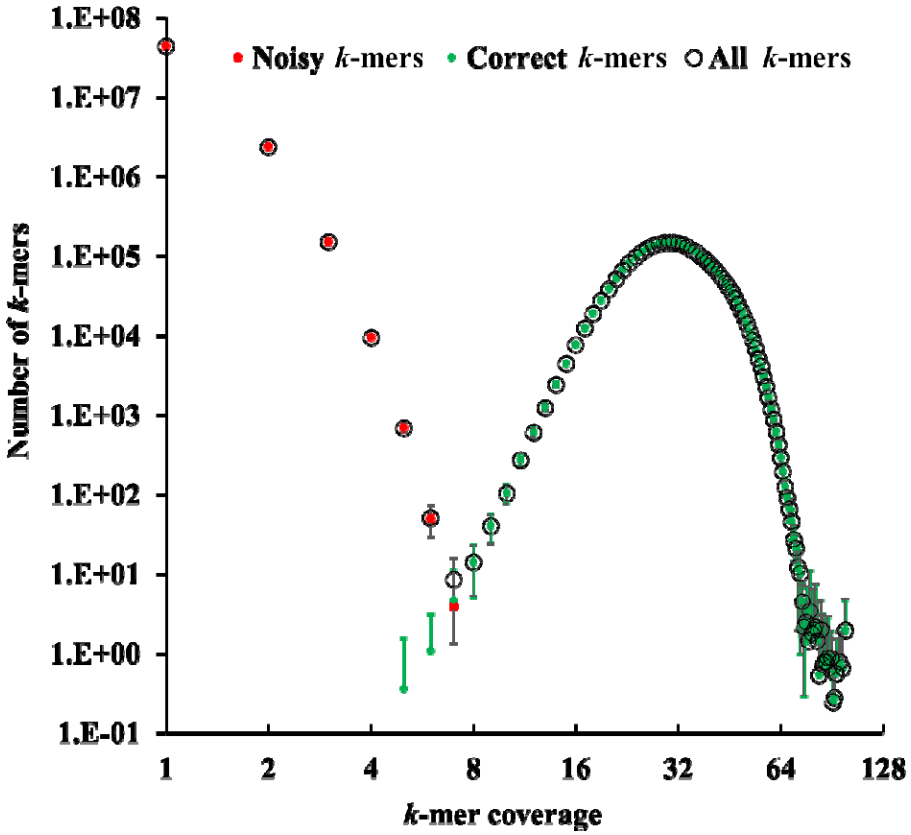


Fig. S4 Both-way search strategy shows fast decoding than one-way search. The simulation is performed with 10,000 DNA droplets. 100 sequence copies with random base errors in rates of 10% are generated for decoding test.

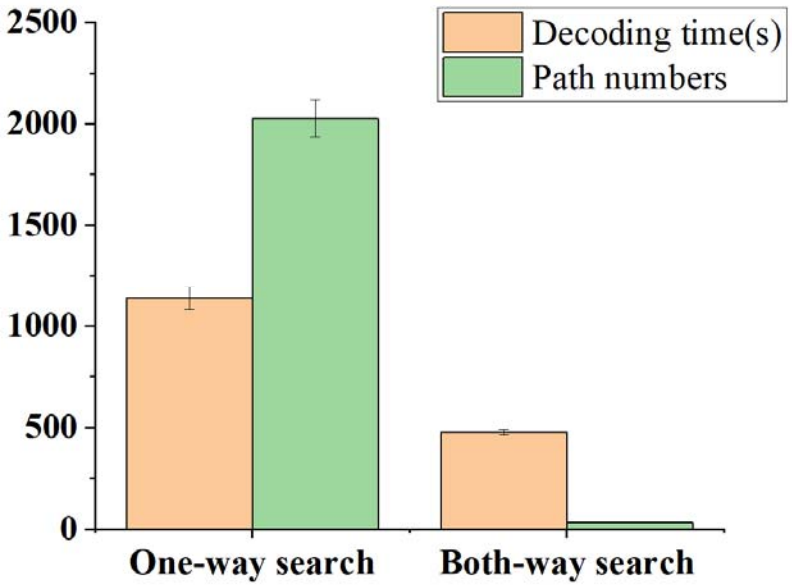


Fig. S5 Details of de Bruijn graph-based greedy searching algorithms.

The detailed description is presented in **Methods** section.

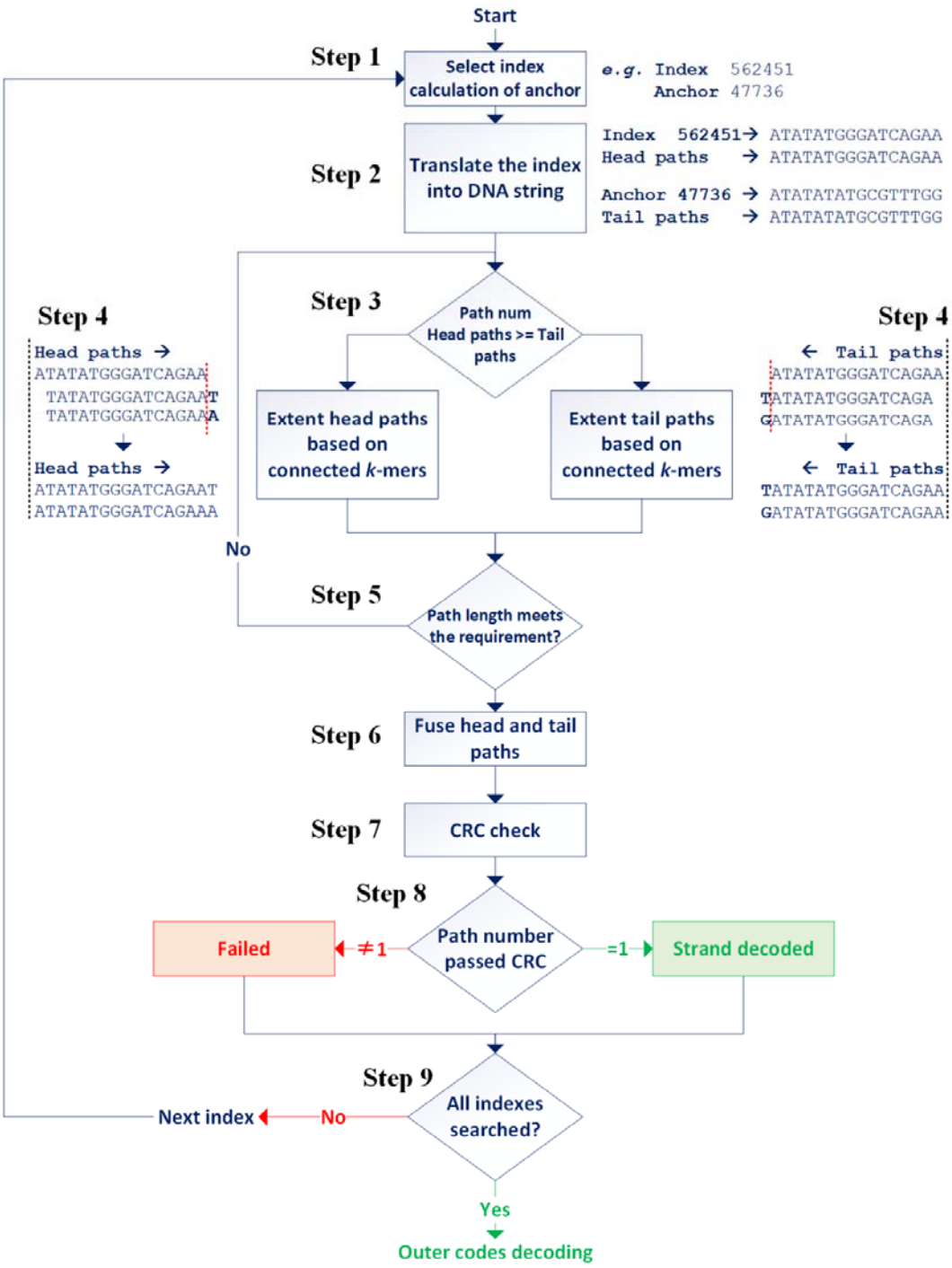


Fig. S6 The algorithm shows excellent linear decoding complexity while scaling up.

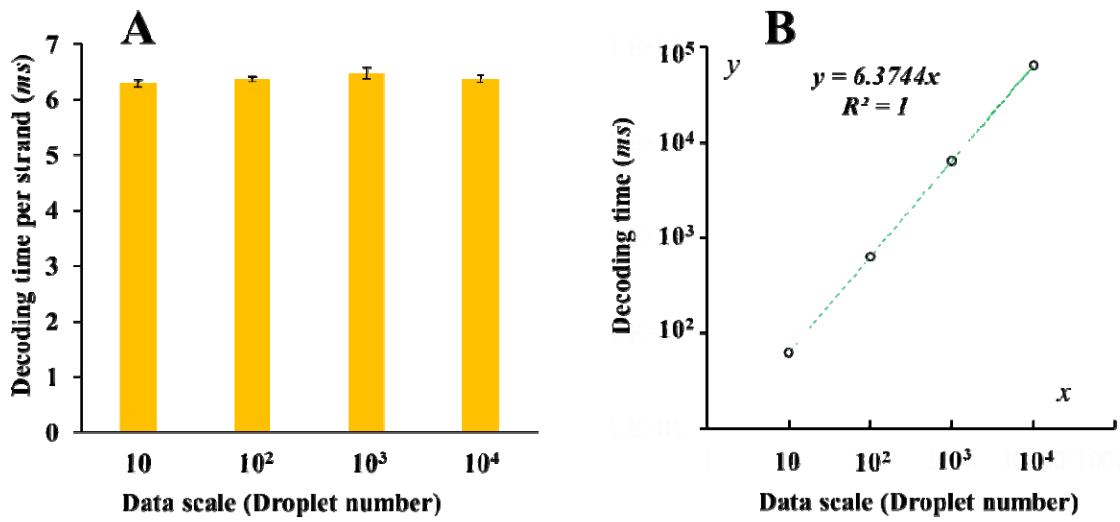


Fig. S7 Gel analysis of error-prone PCR products. A – The first-round error-prone PCR. B – the second-round error-prone PCR using 1 µl first round PCR products (A) as templates. Although the quality of the two PCR products, especially the second-round (B), is very low, we can accurately retrieve the original data without any mistakes in both cases. *Note, the designed length of DNA strands is 200bp.*

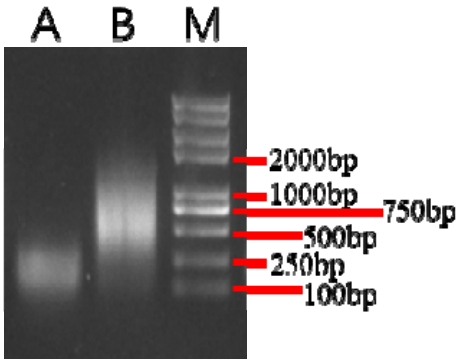
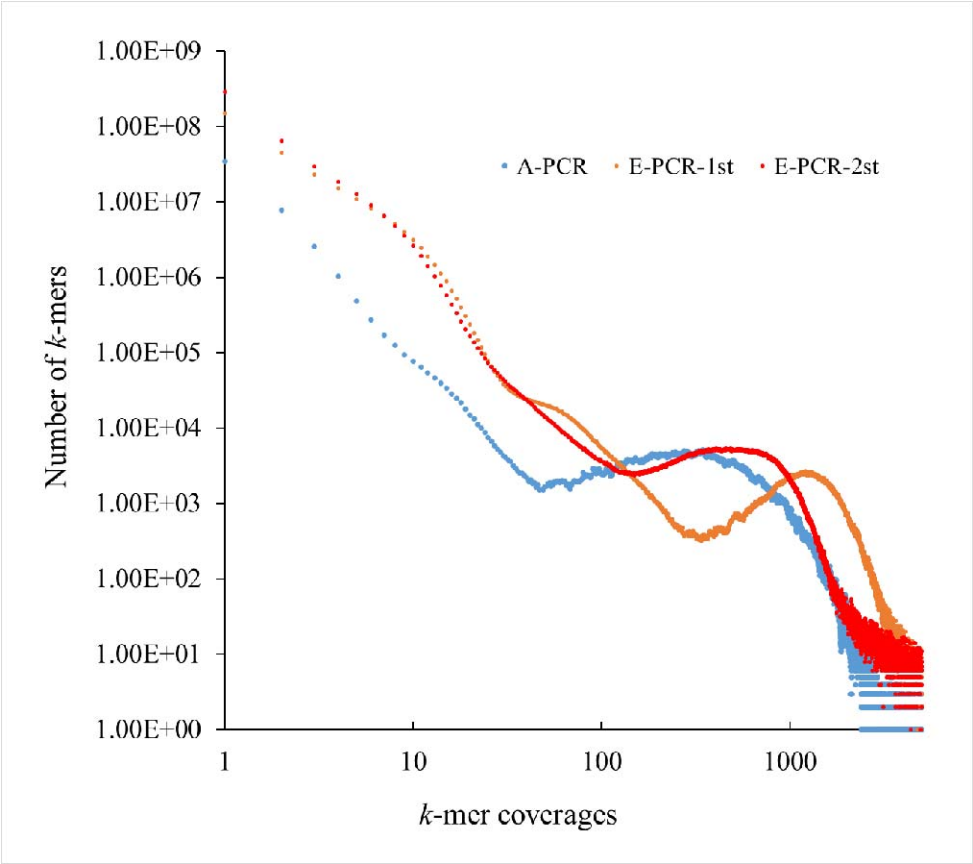


Fig. S8 *k*-mer coverage distributions of three data retrieval experiments by accurate PCR (A-PCR) and error-prone PCR (E-PCR-1st and E-PCR-2st). This figure is used as reference for setting the cut-offs of low coverage *k*-mers.



Tab. S1 Theoretical decoding capacities of DBG-GPS with different *k*-mers sizes

The formula used for decoding capacity estimation was described in **Methods**.

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