










## Research Article

# Pantheon-DNA: Versatile encoding-decoding system with integrated adaptive NGS preprocessing algorithms for DNA data storage

Adriano Galindo Leal<sup>a, , \*</sup>, Thiago Yuji Aoyagi<sup>a, , André Guilherme Costa-Martins<sup>a, , Diego Trindade de Souza<sup>a, , Cristina Maria Ferreira da Silva<sup>a, , Eduardo Takeo Ueda<sup>a, , Marcelo Gonzaga de Oliveira Parada<sup>b, , Allan Eduardo Feitosa<sup>a, , André Fujita<sup>c, </sup></sup></sup></sup></sup></sup></sup></sup>

<sup>a</sup> Artificial Intelligence and Analytics Department, Institute for Technological Research, São Paulo, 05508-901, SP, Brazil

<sup>b</sup> Lenovo Tecnologia Brasil Ltda., Indaiatuba, 13.337-200, SP, Brazil

<sup>c</sup> Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 05508-090, SP, Brazil

## ARTICLE INFO

## Keywords:

DNA data storage  
Sequencing data preprocessing  
Encoding schemes  
Software engineering

## ABSTRACT

We introduce Pantheon-DNA, an end-to-end processing pipeline for DNA data storage that effectively addresses scalability challenges while efficiently managing large datasets, maintaining  $\geq 99.996\%$  retrievability at  $10\times$  coverage under both LER and HER in our tests. To prevent repetitive patterns in DNA sequences, which potentially cause chimeras at the molecular level and also hinder clustering algorithms, we propose a data arrangement scheme and a randomization procedure during encoding. We use block data architecture to enhance parallel processing and retrieval. The proposed sequencing data preprocessing pipeline utilizes prior knowledge of the data structure encoded in the DNA sequences to simplify conventional clustering routines and reduce computational complexity. The system's robustness and reliability are validated through an actual synthesis and sequencing experiment, which encodes and decodes 1.59 MB of data containing multiple files. Future enhancements will focus on refining error correction capabilities, particularly for indel recovery, as well as optimizing preprocessing efficiency and sensitivity.

## 1. Introduction

In the rapidly evolving field of computational biology and bioinformatics, DNA molecules have emerged as a breakthrough solution for digital data storage media, meeting the increasing demands for higher data volumes [1–3]. DNA has carried genetic information for billions of years thanks to its inherent stability and high-density storage capacity [4], offering a unique combination of compactness, durability, and longevity. Traditional storage media, such as magnetic tapes and hard drives, have limitations in lifespan, physical space, and energy consumption. In contrast, DNA presents a biosynthetic solution capable of storing a large amount of data in a small physical space for millennia without significant degradation [5].

The shift from conventional data storage to DNA-based storage introduces complexity and requires a combination of computational techniques and intricate molecular biology processes [6]. This challenge has motivated researchers and engineers worldwide to explore the potential

of DNA as a medium for storing digital data. To represent digital data as DNA molecules, data must be encoded into DNA sequences and then synthesized as physical molecules. To retrieve the data stored in DNA, the molecules must be sequenced, preprocessed, and then decoded back into digital data. Fig. 1 illustrates the complete DNA data storage workflow.

Currently, the main bottleneck in DNA writing and reading technology is the high cost and limited throughput of synthesis and sequencing, which is still insufficient to meet the demands of the digital world [6,7]. The constraints inherent to current technology preclude the synthesis of single DNA molecules with arbitrarily long sequences [12]. Consequently, most DNA data storage protocols employ short DNA sequences, generally limited to approximately 200 nucleotides, due to the limitations of base-by-base synthesis. Cycle-based chemistry imposes a per-base reaction time, so the total synthesis time scales linearly with the target length, which further constrains the practicable strand lengths on conventional instruments [9]. As a consequence, long data

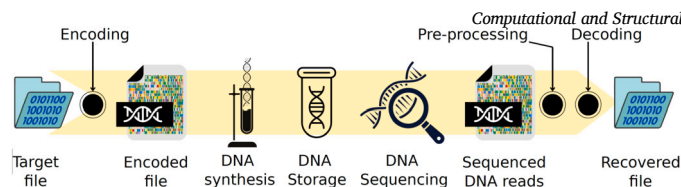
\* Corresponding author.

E-mail address: [leal@ipt.br](mailto:leal@ipt.br) (A.G. Leal).

URL: <https://ipt.br/en/> (A.G. Leal).

<https://doi.org/10.1016/j.csbj.2025.09.002>

Received 8 May 2025; Received in revised form 31 August 2025; Accepted 1 September 2025



**Fig. 1.** End-to-end workflow of DNA data storage. First, a target file is encoded into DNA sequences. DNA molecules containing these sequences are then synthesized and stored for future use. When reading the data, the molecules are sequenced, preprocessed, and finally decoded to retrieve the original file.

strings must be partitioned into smaller fragments for storage in DNA. Accurate organization and retrieval of the stored information require that such partitioning be accompanied by systematic indexing, typically achieved through the attachment of unique DNA tags and the use of primers to initiate replication or sequencing [1–3]. Synthesis parallelization introduces challenges in crosstalk and reagent delivery across large arrays; microfluidic compartmentalization and electrochemical arrays on complementary metal–oxide–semiconductor (CMOS) chips mitigate these issues by isolating features and enabling electronic addressing of reaction sites [8]. Although sequencing already operates as a highly parallelized process, downstream computation remains a bottleneck due to the volume of reads and heterogeneity in library construction strategies, quality profiles, and error biases [10,11].

The organization of short DNA sequences commonly used in data storage mirrors amplicon sequencing, which amplifies and sequences target regions using flanking primers [11]. Both biological amplicons and DNA storage data fragments feature conserved regions (primers/indexes) flanking variable sequences (payloads in data storage, gene parts in amplicons), necessitating *in silico* clustering of reads for data retrieval. High sequence coverage is a reliable indicator that the sequenced sample accurately represents the designed target sequence. Obtaining uniform coverage across all regions can be challenging due to biases in sample preparation, sequencing technologies, and biological characteristics of DNA [13].

Finding an optimal coverage level for DNA data storage that balances data recovery and operational costs is a challenge. High coverage means high information redundancy and enables data retrieval in high-error-rate channels; however, it also significantly increases sequencing costs and the computational complexity of processing sequencing data. However, working with low coverage in a complex DNA pool containing millions of different DNA sequences, even in low error-rate channels, can lead to increased sequence loss (zero coverage) for a fraction of the DNA pool. Such a condition may result in the complete loss of specific data segments, necessitating a robust error-correcting code (ECC) to reconstruct missing sequences and fix nucleotide substitution errors. However, implementing a robust ECC that addresses these gaps increases the complexity of data recovery. Achieving the appropriate balance requires careful planning and optimization of sequencing strategies to ensure that the benefits of growing coverage justify the additional sequencing and computational costs.

A computational preprocessing stage is required to treat raw sequencing data and condense sequence information. The workflow involves clustering similar sequences via local or global alignment between read pairs. It faces significant computational challenges exacerbated by the scalability issues of traditional clustering pipelines [14], [15], [16], [17]. Other challenges include the formation of molecular chimeras due to PCR (polymerase chain reaction) artifacts and *in silico* chimeras resulting from improper clustering. Molecular chimeras result from DNA polymerases that mistakenly combine DNA fragments from different origins during PCR amplification, leading to hybrid sequences that misrepresent the original data. *In silico* chimeras occur when clustering algorithms inaccurately group sequences or errors arise in aligning sequences within clusters, creating erroneous consensus sequences. These artifacts necessitate the use of bioinformatics tools and ECC to distinguish and correct such errors, ensuring the accurate recon-

struction of stored data and maintaining the reliability of DNA-based data storage systems.

Unlike biological data, the DNA sequences used in data storage have predictable diversity with known fragments. This predictability enables the simplification of bioinformatics pipelines to meet the demands of digital data storage and management. In this work, we introduce *Pantheon-DNA*, an end-to-end processing algorithm for DNA data storage that incorporates a codec (including encoding and decoding schemes) and decoding preprocessing algorithms, aiming to enhance retrievability and reduce computational costs. Encoded sequences are organized into blocks, each identified by unique flanking tags. A unique address tag also identifies each sequence within a block. The proposed preprocessing algorithm leverages this known sequence structure to simplify alignment and clustering procedures in a multilayered fashion: sequences are sorted by tags using local alignment rather than global alignment, and clustering is performed accordingly. Clustering is only performed if data is still missing. The ECC embedded in the sequences allows a lightweight parity check to verify whether the sequence needs no further processing. We also proposed two randomization schemes during encoding to prevent repetitive patterns in digital data from being retained in the DNA sequences, thereby avoiding the formation of chimeras and clustering issues during preprocessing. The method is tested under various simulation scenarios and also in an experiment involving DNA synthesis and sequencing, thereby validating the proposal in actual conditions.

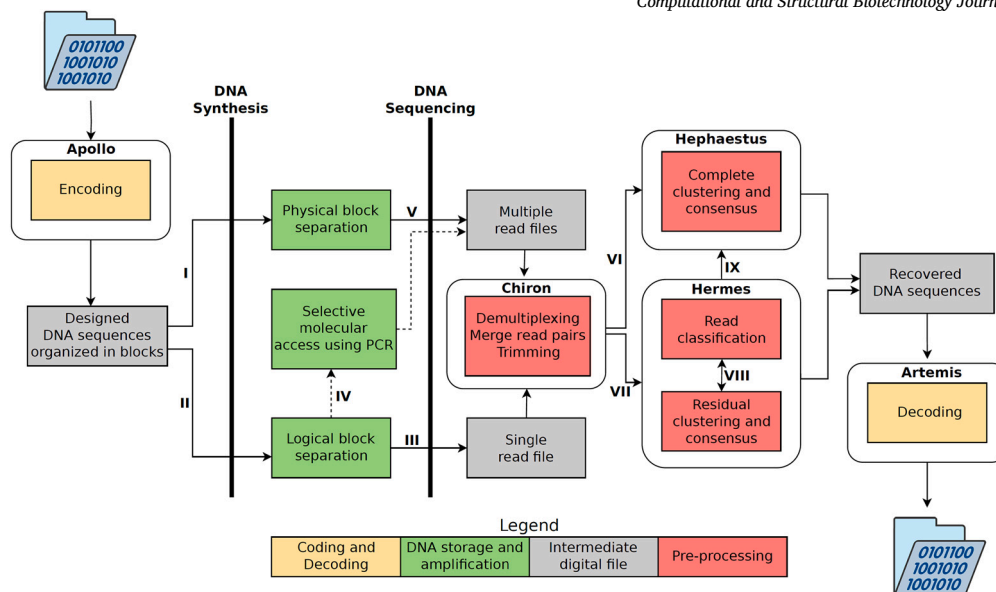
## 2. Proposed method

In this section, we provide a detailed description of the *Pantheon-DNA* workflow, which encompasses encoding and decoding schemes, as well as a versatile decoding preprocessing pipeline that handles sequencing reads from various sequencing platforms and strategies. Fig. 2 shows the *Pantheon-DNA* workflow, highlighting the possible routes of the proposed processing scheme. The proposal is divided into modules, namely *Apollo*, *Chiron*, *Hermes*, *Hephaestus*, and *Artemis*.

### 2.1. Encoding

The encoding process converts the input digital data into a set of DNA sequences. Each sequence will later be synthesized as a DNA strand. The module responsible for encoding is *Apollo*, the god of the sun and music in Greek mythology. In the same way that a good song is composed of a structured order of notes and rhymes, allegorically, the encoding process generates a structured order of nucleotide bases.

It is well known that certain DNA sequence patterns, such as long homopolymers and unbalanced GC content, introduce error bias in sequences during synthesis and sequencing [18–20]. However, implementing constrained codes to avoid them, such as those mentioned in [1,2,23,24], may be inefficient, as suggested by [21]. Instead, we adopt a simple bit-to-base encoding, utilizing two pattern-breaking methods (or randomization methods) [45]. By randomizing the binary data that form DNA sequences, we do not ensure that constraints such as those for homopolymer and GC content are met; however, we statistically reduce the occurrence of extreme, problematic cases. Additionally, randomization breaks any patterns that may be present in the binary data and prevents them from persisting in the DNA sequences. Repetitive patterns in DNA sequences are prone to forming chimeras during PCR and



**Fig. 2.** Detailed pipeline of Pantheon-DNA. Apollo encodes the input digital file is encoded into DNA sequences; the encoded set of sequences is partitioned into blocks, each one identified by a unique pair of flanking tags; after synthesized, the sequence blocks may be stored into separate physical containers (I) or be stored into a single physical container (II); when all blocks are in a single container, they are sequenced together (III) generating a single read file, which can then be logically separated by Chiron; they can also be molecularly separated via PCR (IV), which produces separate read files; when blocks are stored in separate containers, each block may be separately sequenced (V); both approaches (IV) and (V) alleviate computational cost of Chiron; Chiron output data may be conducted directly to Hephaestus clustering (VI) or to Hermes adaptive identification of sequences (VII); Hermes performs multiple rounds of read classification and data summarization via alignment and consensus; residual unclassified data from Hermes may further be conducted to Hephaestus (IX); output from both Hermes and Hephaestus are finally merged and sent to Artemis to be decoded.

also hinder clustering algorithms during read preprocessing, making it difficult to separate distinct sequences effectively.

We define *payload* as the fragment of a DNA sequence containing encoded digital data. We define *bitstring* as the bit sequence encoded into a single DNA sequence. In the following, we describe the entire encoding pipeline. The input data can be a single file or multiple files, presented as a single stream of bits. This stream is first partitioned into *data blocks* of approximately the same number of bits. The number of bits in each data block must be a multiple of the size of the bitstring. The last block may be padded with random bits to ensure it fits the block size. As elucidated in the following subsection, this division into blocks, or hierarchical addressing, is essential for restricting the computational load during decoding preprocessing and preventing an increase in computational time. Each data block is arranged as a two-dimensional array of bits to form a group of bitstrings. Here we apply our first pattern-breaking approach: the transversal arrangement. Instead of displaying the sequence of bits along the bitstrings, it is arranged across the bitstrings in each data block, scattering possible bit patterns that could repeat in different bitstrings. Then, for each data block, we generate redundant bitstrings by applying the Reed-Solomon error-correcting code [22] with bitstrings (or partitions of these bitstrings) as entire code symbols. This outer code enables, during decoding, the retrieval of occasional missing bitstrings.

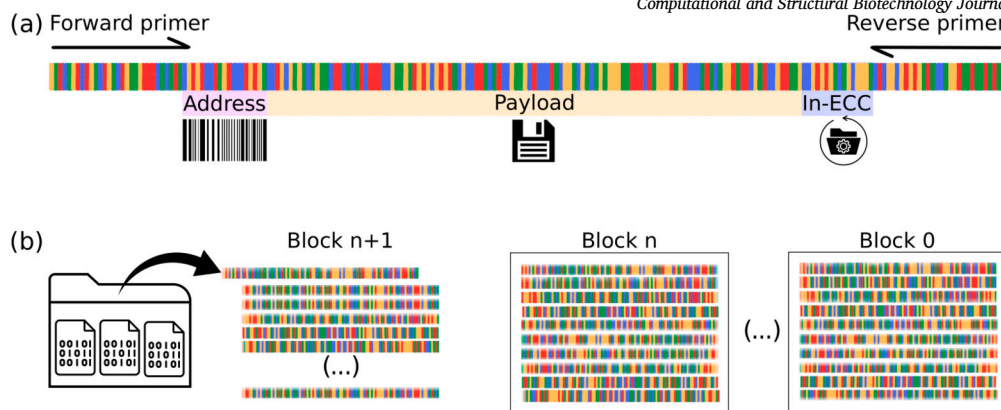
The bitstrings in each block must follow the order in which they were encoded to preserve the information. We identify this order by associating each bitstring with a unique sequential integer value. To further disperse remaining bit patterns across different bitstrings, we apply a second pattern-breaking approach: randomization by index/address. We use the unique index of each bitstring as a seed to generate a pseudo-random sequence of bits to be added (bitwise sum via XOR operation) to the bitstring itself. Next, each bitstring is mapped into a sequence of nucleotide bases (A, C, G, and T) to form the payload. As discussed previously, we opt to use a simple encoding approach, mapping every two bits to a nucleotide as {00, 01, 10, 11} to {A, C, G, T}.

We must also introduce means to reassemble the data during decoding. Since the physical DNA molecules are commonly mixed and stored in the same container after synthesis, we must be able to trace back

their position in the data block. Within each block, we attach an *address tag* to each payload. Each address tag encodes the unique address of each bitstring via a library of tags. This library of address tags is designed to maintain a certain edit distance between every two tags, reducing misclassification, especially in high-error-rate channels. Address tags can be concatenated to expand the total number of addresses per block. Then, the address and the payload are encoded with the Reed-Solomon code (inner code), appending a redundant sequence segment that enables detection and correction of substitution errors. Finally, the sequences are flanked with forward and reverse primer tags to be amplified via PCR, a process commonly necessary before sequencing to convert DNA molecules into double-stranded DNA and to increase the amount of DNA present. The content of each sequence is depicted in Fig. 3.

Suppose the DNA molecules of distinct data blocks are stored in the same DNA container. The amplification tags can also be used as *block tags* to distinguish data blocks. Distinct amplification/block tags identify sequences from different blocks, allowing them to be separated at the molecular level by PCR amplification or computationally during decoding preprocessing. Fig. 3b illustrates the block organization scheme. Focusing the computational clustering process on individual blocks, rather than on the entire data archive, markedly accelerates processing and constitutes a distinctive advantage of the approach. In addition to streamlining the workflow, the method enhances the overall efficiency of the DNA data storage system.

Independent work by Sharma et al. [27] reports comparable results, demonstrating that fixed-size encoded DNA blocks substantially reduce processing time. Block-specific primers underscore the high degree of control Pantheon-DNA affords over the data pool, enabling targeted access to designated archive segments — a capability beyond traditional media such as LTO tapes. The ability to amplify specific sections, including metadata and other high-priority blocks, is particularly valuable for improving coverage in crucial areas of the archive and thereby increasing the probability of successful data recovery. Furthermore, block structure opens up exciting opportunities for innovative storage strategies, such as developing DNA chips analogous to microarrays, in which



**Fig. 3.** Proposed data structure in DNA. (a) Each DNA sequence is composed of a flanking pair of amplification primers, also used to identify the data block the sequence belongs to; an address tag to identify the position of the sequence within the data block; the payload that encodes a piece of the digital data; and a redundant segment for inner code (ECC). (b) Digital data is partitioned into blocks before encoding properly; each block is distinguished from one another by a unique pair of amplification/block tags, enabling logical or molecular separation as depicted in Fig. 2, respectively, by routes (III) and (IV).

blocks are spatially organized. Moreover, the spatial organization facilitated by block strategies introduces the possibility of searching for files by encoded DNA patterns. Although direct identification without sequencing is outside the scope of this work, the block-structured layout could support probe-guided pre-enrichment and pattern-guided discovery prior to sequencing. These possibilities are prospective and will require targeted experimental validation; if realized, they may reduce retrieval time and cost.

By integrating encoding architecture into storage and post-sequencing processing strategies, this design is compatible with both chip-based spatially organized storage and pooled storage. Here, we demonstrate Illumina paired-end compatibility and outline other platforms as future work. The blocking strategy greatly benefits DNA chip storage through spatial organization. It is also fully compatible with DNA pooling storage methods, where molecules are stored *en masse* without any physical organization. This adaptability ensures that Pantheon-DNA's innovative features, designed to enhance data accessibility and interaction, are accessible across various storage paradigms.

## 2.2. Preprocessing NGS data

Besides the length limitation, current DNA synthesis technologies cannot grow a single DNA molecule with a specified sequence of nucleotide bases; instead, they produce a batch of thousands and even millions of copies (usually measured in nmol) with the same sequence [28]. Thus, millions of redundant molecules are produced during the synthesis process.

Depending on the synthesis strategy, a few copies may differ slightly from others due to errors in synthesis, while other mistakes can be introduced during the PCR amplification cycles. PCR artifacts include base substitutions, usually recovered by standard ECC methods, insertions and deletions (also known as *indels*), and chimeric sequences. Although base substitution tends to be an approximately stochastic event, the others are more related to specific DNA patterns, such as short repetitive segments and high similarity between different DNA sequences, respectively [28]. Thus, avoiding these patterns in the sequences reduces indel and chimera rates, consequently improving data recovery.

An NGS platform is needed to read large amounts of DNA sequences. NGS platforms yield a large volume of digital DNA sequences, also known as reads [17,3,20]. When reading stored DNA molecules, sequencing involves taking a sample of the material, resulting in a large volume of digital data. DNA molecules are processed and read in parallel; thus, a certain degree of redundancy, inherent in the synthesis process, will be present in the sequencing output. The sequenced output contains copies of each expected population of copies for encoded DNA sequences. Bioinformatics has long addressed this issue with read

processing techniques [29–33]. Here, we advocate that preprocessing is crucial for ensuring data integrity and poses a challenge for scaling the decoding to the vast amount of data required to implement DNA data storage as a widespread technology.

Traditional ECC methods cannot correct indels. ECC methods capable of correcting indels can be found in the literature [34–37], but with strict limitations. Furthermore, in general, such indel-correcting codes are known to have low capacity [38] (i.e., they require excessive redundancy). Multiple sequence alignment (MSA) and consensus methods can utilize coverage to handle indels and reconstruct the original sequence from the sequenced population. However, MSA is computationally intensive [39,40] and requires prior clustering of NGS reads, which is also a computationally intensive step [15], to assign reads to a specific sequence type.

Pantheon-DNA presents a novel decoding preprocessing approach specifically designed for DNA data storage. Our proposition is a multistage processing scheme, which, at each stage, processes reads and evaluates their integrity via a parity check to decide whether further processing is required or not. Early parity checks and address fills spare further computation on near-perfect reads and already-resolved sequences, reserving alignment and clustering for the minority that require heavier processing. With this approach, we are mitigating superlinear growth and achieving near-linear scaling in our experiments, which also enables parallelization.

Our preprocessing pipeline is flexible enough to handle sequencing data from different backgrounds, particularly in terms of how molecules are grouped when stored or sequenced. After synthesis, molecules from different data blocks may be stored in separate physical containers (Fig. 2-I) or be mixed in the same container (Fig. 2-II). When data blocks are stored separately, sequencing of each container produces separate read files (Fig. 2-V). When blocks are mixed, sequencing produces a single read file (Fig. 2-III), and block separation is performed computationally (logically). Additionally, when data is mixed, data from different blocks can be molecularly separated via PCR (Fig. 2-IV), allowing sequencing to be performed to isolate the blocks. NGS technologies require sample preparation protocols, including the insertion of foreign DNA segments into samples, known as sequencing indexes. Some sequencing strategies, known as paired-end and mate-pair sequencing, read the DNA fragment in both orientations. Both cases require prior processing to ensure that the indexes are entirely removed and that the pairs of reads are merged.

The preprocessing pipeline in Pantheon-DNA comprises three modules: Chiron, Hermes, and Hephaestus. Chiron, named after the legendary tutor of heroes and gods in Greek mythology, is the first to



process raw sequencing reads. Its primary role is to demultiplex<sup>1</sup> sequencing reads to their respective blocks by processing primer regions (block tags) and trimming them from the sequence. Tags are classified by semi-global alignment [41]; reorientation is applied when required. For Illumina paired-end data, PEAR [42] merges reads prior to demultiplexing. This processing standardizes reads before downstream preprocessing and decoding; subsequent stages are executed per block.

Hermes, drawing from the role of the divine messenger in Greek mythology, is responsible for locating the address tag of each read and associating it with the correct position within a block. This process occurs in two stages. In the first stage, Hermes identifies the address tag through exact string matching at a fixed position, followed by verification of both read length and integrity via error-correcting codes (ECC). Reads lacking a valid address tag are retained in a buffer for later processing by Hephaestus. In contrast, those with a valid tag but failing length or integrity checks are sent to a separate buffer for Hermes' second stage. Reads that pass this initial screening proceed to decoding, and their corresponding addresses are flagged as filled, thereby preventing redundant processing of subsequent reads with identical addresses.

After all reads have been processed in Hermes' first stage, Hermes' second stage processes the data accumulated in the buffers of each address separately. Reads found for each address are aligned [43] and a consensus is taken. Then, the length and parity of the resulting consensus are verified. If the sequence passes, it is retained for decoding, and the corresponding address is marked as filled. If not, all reads from MSA are kept in the data buffer for the Hephaestus stage.

As a smithing god responsible for forging armor and weapons, Hephaestus in Pantheon-DNA performs the last and heaviest processing stage of preprocessing. The remainder of the reads accumulated in the Hephaestus buffer are all clustered with abundance-based greedy clustering (AGC) [14,15], based on the sequence alignment scores. Then, the consensus sequences of each cluster are kept for decoding. This last stage is skipped if all address positions have already been filled.

Hephaestus does not require a planned address scheme to function; however, it takes longer to cluster and obtain consensus from sequenced reads. Additionally, the prior demultiplexing of reads into blocks with Chiron reduces the complexity of clustering, as it reduces the overall number of read pairs to be compared. Demultiplexing also opens the opportunity for parallelization, as the blocks are independent of each other. This strategy was adapted from common practices of metagenome and amplicon studies and optimized for high-error channels and higher coverage compared to Hermes.

### 2.3. Decoding

Preprocessed sequences are then decoded to retrieve the original digital data. As encoding and decoding are (reversed) paired processes, we call the decoding module Artemis, Apollo's twin sister, and the goddess of the moon and the hunt. Just as Apollo encodes data into DNA sequences, Artemis decodes it back into binary, meticulously hunting down errors and tracking erasures to reconstruct missing sequences. Her role in Pantheon-DNA symbolizes the precision and speed of an archer in the decoding process, aiming to retrieve data with high fidelity under the evaluated conditions. Decoding consists of the routine below, applied to every preprocessed read. A read is discarded whenever it fails any of these steps.

1. Verify sequence length (sequences from Hephaestus might not have the correct length).
2. Apply Reed-Solomon error detection and correction, and remove ECC redundancy.

<sup>1</sup> In computer science and telecommunications, multiplexing combines multiple signals into a single channel — in this case, the molecular medium — whereas demultiplexing is the reverse.

3. Decode the address tag, obtaining the numerical position of the read data in the block.
4. Map the DNA sequence back to a bitstring.

The decoded bitstrings are sorted by address position; missing segments are verified and filled with outer code. Outer-coded redundant bitstrings are filtered out. Finally, the binary data can be reassembled and retrieved by gathering the data from all blocks.

## 3. Results

In this section, we evaluate the performance of Pantheon-DNA in two main aspects: data retrievability and processing time. We perform four study cases: (i) simulation scenario<sup>2</sup> to compare the proposed method with literature methods; (ii) simulation scenario to break down the performance of the proposed method; (iii) simulation scenario with larger input data; and (iv) experiment with oligo synthesis.

The simulation of DNA synthesis and sequencing procedures is performed under different error rates and synthesis and sequencing conditions. For synthesis, we simulate a population of sequence copies for each encoded sequence, applying random nucleotide errors (insertions, deletions, and substitutions) along the sequences. The set of simulated numbers of copies for each sequence is a sample from a multinomial distribution, where the number of attempts is the total number of reads from sequencing. Additionally, in simulation, we test various scenario conditions, including average coverage (the number of copies of each DNA sequence) and error rates, using our set of scripts. We use ART [46] to simulate Illumina NGS sequencing outputs.

### 3.1. Scenario 1: Comparing with literature methods

This analysis involved four distinct pipelines: DNA Fountain [25], HEDGES [26], and two versions of the proposed Pantheon-DNA method. The following methods were implemented, with necessary adaptations for the testing environment based on available source code:

- **DNA Fountain:** Implemented with simple preprocessing (PEAR and length filtering). It does not handle reverse-complementary reads. Error detection uses Reed-Solomon (RS) codes, but not error correction. The code was modified to ensure execution.
- **HEDGES:** Modifications were made to the decoding algorithm to handle FASTQ files with randomly mixed reads and multiple oligo copies. A code rate of 0.5 and a strand length of 226 nt were used. Preprocessing consisted solely of merging R1 and R2 reads with PEAR.
- **Pantheon-DNA:** Two versions were tested: (i) transversal arrangement only and (ii) transversal arrangement plus address-based randomization.

Pantheon-DNA uses the following encoding settings:

- **Mapping:** A simple direct mapping is used (2 bits to 1nt); this scheme encodes 228 data bits in a 114nt payload for each DNA sequence.
- **Address:** library-based address, in which two tags of size 12nt are concatenated; the library of 12nttags is composed of 235 distinct tags with a minimum mutual edit distance of 6.
- **The inner code** is applied to the data after bit-to-base mapping; Reed Solomon code of order  $2^6 = 64$  is used so that three consecutive encoded bases represent one code symbol; error-correcting capacity is  $t = 4$ , generating a total of 24nt bases of redundancy.
- **The outer code** is applied to bits before being mapped into bases; Reed Solomon code with order  $2^{10} = 1024$  is used, and the error-

<sup>2</sup> In silico simulation of synthesis and sequencing.

**Table 1**  
Results for different DNA coding methods.

Scenario	Fountain	HEDGES	Pantheon (no rand.)	Pantheon (with rand.)
Zipped files	381,072 (200nt)	652,290 (226nt)	431,273 (222nt)	431,273 (222nt)
Nyanko	380,163 (200nt)	650,760 (226nt)	430,344 (222nt)	430,344 (222nt)

correcting capacity is  $t = 50$  so that up to 100 missing binary segments can be corrected at every block of 1 020 data segments (redundancy included).

- Block tags (PCR primers) of 30 nucleotides in size are appended to each end of the sequence.

The methods were evaluated using two test files:

- **Nyanko:** A 10.5 MB bitmap image file with a high degree of data repetition.
- **Zipped:** A 10.5 MB ZIP file containing a mix of file types (JPG, WAV, CSV), representing a heterogeneous dataset.

We simulate coverage 10 in the sequencing output. We simulate two scenarios of error rate:

- a scenario of low error rate (LER), with substitution rate per base at 0.3%, deletion rate per base at 0.1%;
- a scenario of high error rate (HER), with a substitution rate per base at 1.0% and a deletion rate per base at 0.5%.

For all Pantheon-DNA tests, the oligo size was 222 nt, and each block contained up to 48,000 oligos, resulting in 9 blocks for both files. Table 1 reports, for the two test files, the resulting oligo counts together with the strand lengths used by each method.

Under the simulated channels, both DNA Fountain [25] and HEDGES [26] achieve complete file reconstruction once coverage and error profiles fall within their operating envelopes. In higher-error cases, Pantheon-DNA maintains recoverability through a division of labor: the inner code rectifies substitutions at the read level; Hermes/Hephaestus provides alignment-based summarization to manage indels; and the outer code fills residual erasures. This layered design sustains full recovery at lower coverage than monolithic approaches, particularly when addressability and block partitioning are exploited.

Let  $p_{\text{ins}}$  and  $p_{\text{del}}$  denote insertion and deletion rates per base, and let  $c$  denote coverage per address. Under an independence approximation, the probability that a position is not affected by indels in at least  $\lceil c/2 \rceil$  reads is

$$P_{\geq \lceil c/2 \rceil} \approx \sum_{k=\lceil c/2 \rceil}^c \binom{c}{k} (1 - p_{\text{ins}} - p_{\text{del}})^k (p_{\text{ins}} + p_{\text{del}})^{c-k},$$

The expression provides a lower bound on the correctness of majority consensus prior to Reed–Solomon substitution correction. The bound highlights that consensus reliability increases with coverage and decreases with indel rates, complementing the empirical behavior observed in the simulations and the synthesis experiment.

Table 1 presents the total number of oligos generated by each encoding method for the “Zipped” and “Nyanko” test files, with the corresponding oligo size indicated in parentheses. The results demonstrate significant differences in encoding efficiency among the evaluated methods.

The HEDGES method utilized the largest number of oligos to encode the files, with 652,290 oligos for “Zipped” and 650,760 for “Nyanko,” both at a size of 226 nt. In contrast, both DNA Fountain and Pantheon-DNA proved to be more efficient in terms of data density. DNA Fountain generated 381,072 oligos (200nt) for “Zipped” and 380,163 oligos (200nt) for “Nyanko.” Meanwhile, Pantheon-DNA (in both versions)

**Table 2**  
Retrievability across methods and scenarios.

Scenario	DNA Fountain	HEDGES	Pantheon (no rand.)	Pantheon (with rand.)
Zipped LER	100%	100%	100%	100%
Nyanko LER	100%	100%	100%	100%
Zipped HER	0.73%	100%	99.999%	99.996%
Nyanko HER	3.53%	100%	99.998%	99.996%

used an intermediate number of oligos, with 431,273 for “Zipped” and 430,344 for “Nyanko,” but with a length of 222nt.

These results indicate that, while HEDGES produces a higher number of DNA strands per bit of information, DNA Fountain and Pantheon-DNA achieve a higher data density. This is a crucial factor for the feasibility and cost of DNA storage. The similarity in the number of oligos between the two versions of Pantheon-DNA suggests that the inclusion of randomization methods did not affect the total volume of encoded data.

Table 2 presents the retrievability results for the different DNA encoding methods under low and high error rate scenarios. In the low error rate (LER) scenarios, all tested methods were able to retrieve both the “Zipped” and “Nyanko” files entirely (100% retrieval), demonstrating robust performance under ideal conditions.

However, performance varied significantly in the high error rate (HER) scenario. HEDGES was the only tested method to retrieve the original digital files integrally. Pantheon-DNA, with or without randomization, retrieved more than 99.99% of the files, failing only on a very small portion of the data (failure in the order of one in hundreds of thousands) due to an inner code failure to detect and correct some errors, resulting in erroneous corrections. DNA Fountain retrieved only a small fraction of the data (as usually happens to the LT code when decoding fails).

The processing times for all methods under the LER and HER scenarios are detailed in Tables 3. Regarding encoding time, HEDGES and Pantheon-DNA, with or without randomization, required a similar amount of processing time, approximately 1 minute and 30 seconds. DNA Fountain took at least three times longer, ranging from 5 to 6 minutes, because it exhaustively generates random segments, discarding those that do not meet a predefined set of DNA sequence constraints.

For preprocessing, DNA Fountain took the shortest time, around 14 minutes, because we performed the simplest pipeline, as suggested in their paper, which only merges paired-end reads using the PEAR algorithm. For HEDGES, we also only performed the PEAR algorithm for preprocessing. Still, it took more time than in the DNA Fountain pipeline, of around 22 minutes, because HEDGES generates more oligos than DNA Fountain during encoding.

Pantheon-DNA took a long time for preprocessing because it includes more processing, but, as we will see, it saved time during decoding. Pantheon-DNA took approximately 27 minutes in the LER scenarios, excluding Hephaestus clustering, and around 56 minutes in the HER scenario, including Hephaestus clustering. Note the special case of preprocessing the Nyanko image on the HER scenario when using Pantheon-DNA without randomization. It took more than 4 hours to clusterize instead of the common 56 minutes in this scenario with Pantheon-DNA. This highlights the issue of highly repetitive sequences, specifically arising from the repetitive patterns of the binary data that was not shuffled during encoding. In such a case, Hephaestus took much longer to clusterize because each cluster has many more similar reads to compare with.

For decoding, Pantheon-DNA is by far “the fastest among the methods we evaluated under matched conditions because most redundancy is removed during preprocessing. Pantheon-DNA (with or without randomization) took 1 minute and 50 seconds in LER cases, and approximately 7 minutes in the HER cases. DNA Fountain took more time, from 70 to 80 minutes in the LER case, because its decoding involves a recursive approach for each completely decoded segment. When decoding failed in the HER case, only a few segments were completely

**Table 3**

Processing time by method and scenario (encoding/preprocessing/decoding).

Method & Scenario	Encoding	Preproc.	Decoding
DNA Fountain (Zipped LER)	5m25s	14m21s	70m56s
DNA Fountain (Nyanko LER)	6m13s	13m32s	79m40s
DNA Fountain (Zipped HER)	5m25s	14m37s	6m9s
DNA Fountain (Nyanko HER)	6m5s	13m49s	6m48s
HEDGES (Zipped LER)	1m29s	22m4s	37m53s
HEDGES (Nyanko LER)	1m29s	21m57s	37m17s
HEDGES (Zipped HER)	1m30s	22m6s	5h26m11s
HEDGES (Nyanko HER)	1m28s	22m3s	5h27m28s
Pantheon (no rand., Zipped LER)	1m24s	27m22s	1m46s
Pantheon (no rand., Nyanko LER)	1m23s	27m13s	1m44s
Pantheon (no rand., Zipped HER)	1m24s	56m26s	7m1s
Pantheon (no rand., Nyanko HER)	1m24s	4h6m16s	6m57s
Pantheon (with rand., Zipped LER)	1m31s	26m53s	1m54s
Pantheon (with rand., Nyanko LER)	1m35s	27m20s	1m55s
Pantheon (with rand., Zipped HER)	1m31s	55m58s	6m57s
Pantheon (with rand., Nyanko HER)	1m31s	56m3s	6m58s

Times use compact h/m/s notation without spaces.

decoded, resulting in a time of only 6 to 7 minutes. Although the only to integrally decodes the original file, HEDGES took the longest time to decode. It took 37 minutes to decode in the LER and more than 5 hours in the HER case.

Overall, we can see that Pantheon-DNA features the fastest processing time for each scenario (file and error rate), except in the case where DNA Fountain could not retrieve the file in the HER scenario.

### 3.2. Scenario 2: Pantheon-DNA's modules performances

We break down the performance contribution of each of Pantheon-DNA's modules. We encoded eight files in various formats (PNG and JPG image files, as well as text files), totaling 585 KB in size.

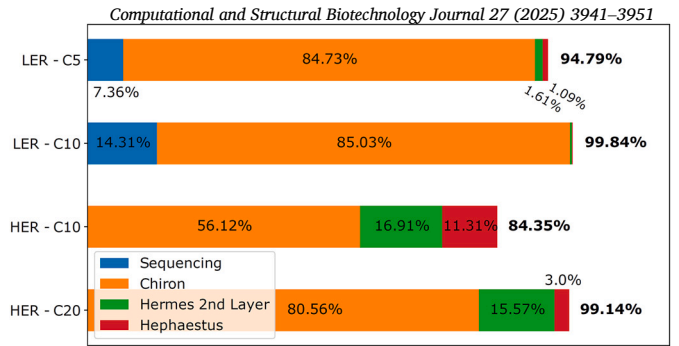
The following settings are used for encoding data:

- Mapping: A simple direct mapping is used (2 bits to 1 nt); this scheme encodes 180 data bits into a 90-nt payload for each DNA sequence.
- Address: library-based address, in which two tags of size 12nt are concatenated; the library of 12nt tags is composed of 235 distinct tags with a minimum mutual edit distance of 6.
- Inner code is applied to the data after bit-to-base mapping; Reed Solomon code with order  $2^6 = 64$  is used so that three consecutive encoded bases represent one code symbol; error-correcting capacity is  $t = 4$ , generating a total of 24 nt of redundancy.
- Outer code is applied to bits before being mapped into bases; Reed Solomon code with order  $2^{10} = 1024$  is used, and error-correcting capacity is  $t = 50$  so that up to 100 missing binary segments can be corrected at every block of 1020 data segments (redundancy included).
- Block tags (PCR primers) of 30 nucleotides in size are appended to each end of the sequence.

A total of 29,649 sequences of 198 nt are generated. The sequences are organized into 9,883 data blocks each. We tested the LER scenario with average coverages of 5 and 10, and the HER scenario with average coverages of 10 and 20, resulting in a total of four simulation scenarios. All scenarios are simulated with logical block separation, with all blocks mixed during sequencing.

We compare the performance of four variations of the proposed preprocessing method:

- Chiron-only (X): performs merging, block demultiplexing, and block tag trimming. This approach is a minimalist preprocessing method, but it leaves a large volume of data for decoding.



**Fig. 4.** Amount of data retrieved at each sub-stage of Chiron & Hermes (XE) approach (simulation scenario). Chiron (demultiplexing and primer trimming) retrieves the largest data during preprocessing. For high error rates (HER), Hermes' second layer (alignment and consensus separately for each address) and Hephaestus (clustering and global alignment) also play important roles in data retrieval.

- Chiron & Hephaestus (XH): performs Chiron and further summarizes each block's data by clustering. This approach is very similar to literature methods [15].
- Chiron & Hermes (XE): performs Chiron and further summarizes the data in each block by the adaptive Hermes stages (layers 1 and 2).
- Simple-Chiron & Hermes (sXE): Like Chiron & Hermes, block demultiplexing is performed by exact string matching rather than semi-global alignment.

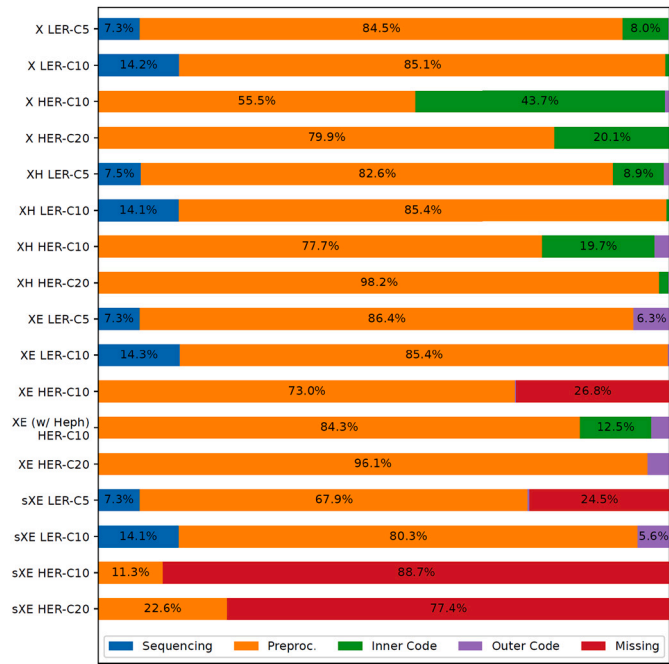
For all tests, except when mentioned, Hermes is used without Hephaestus as the last processing stage. However, Pantheon-DNA was designed to alternate between both dynamically. Hephaestus must handle the unclassified reads from Hermes, especially when some position-biased errors compromise the addresses.

**In silico results summary** Tables 2 and 3 consolidate retrievability and time under LER/HER at 10× coverage. For DNA Fountain, partial recovery in failure cases reflects the threshold behavior of LT decoding: the decoder's state improves progressively, whereas complete segment recovery typically occurs abruptly once a sufficient number of droplets has been observed [25].

**Retrieval performance:** Fig. 5 depicts the retrieval performance for each method test in the four simulation scenarios. It illustrates the contribution of each decoding stage to data retrieval. For example, taking the first bar of the figure, for the case of the Chiron-only approach in a low error rate scenario with coverage 5 (X LER-C5):

- 7.3% of encoded sequences (out of 29,649) are found flawless within sequencing data (reads are only subject to a simple pair-merging for this verification);
- 84.5% of encoded sequences are found flawless after preprocessing (demultiplexing, trimming, alignment, and consensus);
- 8.0% of encoded sequences are found flawless after error correction by inner code; and
- 0.2% (not visible) of encoded sequences are flawless after outer code correction.

Decoding with Chiron-only and Chiron & Hephaestus could retrieve all the encoded data at every simulation scenario. Chiron & Hermes failed to recover all the data for the scenario HER-C10 (26.8% of sequences were missing), and Simple-Chiron & Hermes failed integral retrieval for cases LER-C5, HER-C10, and HER-C20. In the LER scenarios, a small portion of the sequences is already error-free due to sequencing (7% to 14%). However, almost no intact sequences are found in sequencing reads in HER scenarios. Preprocessing plays a crucial role in data



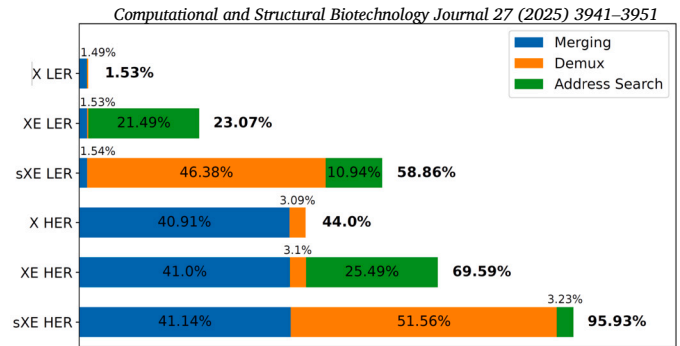
**Fig. 5.** Percentage of data retrieved at each decoding stage for four decoding strategies—X, XH, XE, and sXE—across multiple simulation scenarios defined by two error rates (HER and LER) and coverage levels ranging from C5 to C20. In all scenarios, preprocessing accounts for more than 50% of total data retrieval. Decoding failures are observed only in Hermes-based methods (XE and sXE), likely due to the substantial proportion of discarded data. The combination of Hermes with Hephaestus (XE w/ Heph) achieves complete data retrieval in all evaluated cases.

retrieval and is responsible for more than 50% of sequence retrieval in all cases. Inner code also proves to be relevant, mainly in HER scenarios.

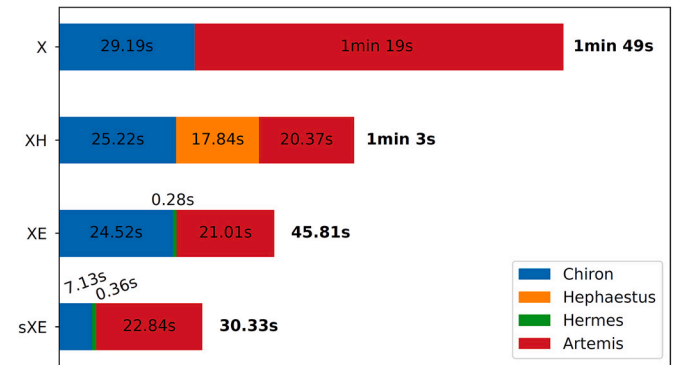
In practical terms, the outer code may appear marginal in some scenarios (e.g.,  $\leq 6.3\%$  recovery). It is worth recalling that the outer code was designed to recover up to 10% of missing sequences in the present experiment; under real-world conditions, outer codes remain essential to help ensure complete recovery under practical conditions. To assess the performance of Hermes, two aspects must be considered: (i) Hermes makes implicit use of inner code to verify sequence parity but does not correct errors, and (ii) when Hephaestus is not employed, Hermes delivers to Artemis only sequences that have passed integrity checks, thereby bypassing inner code error correction. All observed decoding failures occur with Hermes, likely due to the substantial amount of discarded data combined with the absence of inner code error correction. For instance, in scenario HER-C10, Hermes exhibited a 26.8% decoding failure rate. When Hephaestus operates alone, accounting for the sequences discarded by Hermes, the same scenario achieves full data retrieval. The outcome is even more unfavorable when Hermes is paired with Simple-Chiron, as Simple-Chiron discards an even larger number of reads.

The performance of each Hermes sub-stage is examined next. Fig. 4 illustrates the four scenarios described above, featuring Chiron and Hermes, all of which incorporate Hephaestus as a subsequent stage. Chiron retrieves the largest number of sequences, as the first layer of Hermes leaves the sequences unchanged and merely screens them. The second layer of Hermes—alignment and consensus performed separately for each address—becomes critical when error rates are high. Although Hephaestus recovers only a small fraction of the data and demands substantial processing time (as will be shown later), its inclusion proves decisive for complete data retrieval in the XE HER-C10 case presented in Fig. 5.

**Data waste:** As mentioned before, Hermes discards many reads compared to Hephaestus, which may have induced the decoding failure in



**Fig. 6.** Portion of data discarded at each decoding stage (simulation scenario). Pair-merging is very efficient (discarding a few reads) for a low error rate (LER) scenario but very inefficient for a high error rate (HER) scenario, discarding more than 40% of reads. An inefficient process is the simple demultiplexing with exact string matching, which discards more than 45% of reads in both high- and low-error rate scenarios. Address search based on exact string matching is markedly inefficient across both LER and HER scenarios.



**Fig. 7.** Time performance of different decoding preprocessing approaches (simulation scenario). Chiron-only leaves much data for Artemis to decode, yielding a high overall processing time. The essential difference between Chiron & Hephaestus (XH) and Chiron & Hermes (XE) lies in the increased Hephaestus clustering time. Simple-Chiron & Hermes (sXE) reduces the Chiron processing time of the XE approach but also reduces retrieval performance, as shown above.

some cases. We now assess the volume of data discarded at each stage of preprocessing for each approach.

The first three bars in Fig. 6 depict the volume of discarded reads of the methods applied to a low error rate scenario (LER). Hephaestus is omitted since it does not discard reads. Note that merging and alignment-based demultiplexing have low discard rates (both less than 2%). In the address-search stage, exact string matching accounts for approximately 20% of discarded reads under Chiron+Hermes and about 10% under Simple-Chiron (Fig. 6), which is not negligible. Simple demultiplexing, as expected, discards a very high amount of reads (more than 45%).

When considering high error rates (HER), merging becomes inefficient and discards more than 40% of reads. Discard rates in other processing stages also increased compared to LER, but not as much as in merging. The total discard rate for Simple-Chiron & Hermes, for example, becomes 95%.

**Time performance:** Time performance: Fig. 7 depicts the time elapsed during preprocessing and decoding for each preprocessing approach in the LER scenario with coverage 10 (LER-C10). The corresponding per-method, per-scenario measurements are summarized in Table 3.

Hermes offers substantial speed advantages over Hephaestus. While Hephaestus requires 17.84s to complete its task, Hermes performs the same operation in 280 milliseconds—over 60 times faster. Consequently, when Hermes is employed, the overall decoding time is largely





**Fig. 8.** Decoding time performance improvement when the sequencing dataset is partitioned, possible because of the block structure (simulation scenario). When sequencing data is processed as a single large block (as in literature methods), the clustering time (Hephaestus) is drastically longer (the long green bar is out of scale), and the overall process is approximately 29 times longer than the same method with block partitioning (as proposed).

determined by Chiron and Artemis. The execution time of Chiron is considerable: approximately 25–29 s when run without Hermes or Hephaestus, primarily due to the time needed to write the output to disk. In contrast, using a simple demultiplexing approach (Simple-Chiron)—which is not a robust method—reduces this to about 7 s. In Simple-Chiron, most of the time is spent on pair merging, whereas the remaining 18 seconds correspond to alignment-based demultiplexing, which constitutes the main computational cost of preprocessing. When Chiron is executed without subsequent processing stages, decoding in Artemis takes significantly longer than in other approaches. The increase in processing time is due to the absence of summarization or filtering, which results in Artemis receiving a considerably larger volume of sequences to process.

**Block partition:** Block partitioning confers a clear computational advantage of partitioning the set of sequences into blocks. Fig. 8 compares the time performance between the Chiron & Hephaestus approach as above and the same approach without block separation before clustering, both in the LER-C10 scenario. The same set of encoded sequences is used in both methods; the only difference is that Chiron does not perform demultiplexing, and sequencing reads must be clustered simultaneously.

Clustering without block separation closely resembles the approach reported in the literature [15] and related variants, which are faster than most all-pairs strategies [32].

The second approach requires 30 minutes and 33 seconds, as indicated by the second bar, which falls outside the plotted scale (denoted by the green bar). The difference between the two strategies is substantial—approximately a factor of 29—owing to the fact that the computational cost of clustering algorithms grows at a rate exceeding linear proportionality with the amount of data.

**Discussion:** In DNA data storage, searching for a single algorithm that efficiently manages all possible scenarios is akin to seeking a mythical silver bullet. The observation above highlights the need for a flexible suite of algorithms that can dynamically adapt to maximize data recovery while allocating computational effort appropriately to each task. Such flexibility is not just a feature; it is a necessity that significantly extends the lifespan of a codec and the data it preserves. A versatile codec system, capable of adjusting its strategy based on the error rates, coverage, and specific data set characteristics, is intended to support robust data retrieval across diverse conditions and workloads.

A diverse bioinformatics toolkit is essential for handling both paired-end and single-end strategies, as well as accommodating heterogeneous sequencing outputs and protocols. Such adaptability also supports future recovery scenarios, including disaster recovery that leverages platforms designed for biological applications. Careful, dataset-aware tool selection improves success rates and overall system reliability. At the same time, limitations in the current Hermes module warrant attention, particularly in tag classification for block demultiplexing and address identification. Semi-global alignment provides accurate comparisons at a high computational cost, whereas exact string matching is fast but insufficiently robust for sequence comparison.

**Table 4**

Runtime summary for the 90 MB experiment (LER).

Encoding	13 m 26 s
Preprocessing (total)	12 h 03 m 34 s
Chiron (multiplexed)	12 h 01 m 00 s
Decoding	14 m 57 s

Exact string matching in address searches offers speed but lacks robustness for reliable sequence comparison. Future developments should incorporate alternative address-matching strategies that raise retrieval performance while reducing discarded reads, thereby lowering the coverage threshold for complete recovery and improving operational efficiency. Such refinements would also diminish computational and material demands. Similarity metrics grounded in *k*-mer frequency [44] provide a probabilistic, continuously adaptive framework—an attribute that is critical for long-term preservation and accessibility of stored information. In addition, we anticipate that probe-guided pre-enrichment could narrow the search space prior to sequencing, thereby reducing material and computational costs; this concept remains prospective and will require targeted experimental design and validation.

### 3.3. Simulation scenario 3: Encoding large file

Scale-up experiment (90 MB, LER). We evaluate the full pipeline on a 90 MB corpus, reporting synthesis-normalized runtime, density, and reconstruction fidelity. The encoded file was a folder containing a ZIP file, JPG images, WAV audio files, and a CSV text file. The scenario used was LER.

Decoding was successful, with the file being integrally retrieved, a concise runtime summary for the 90 MB experiment appears in Table 4.

### Coding settings

- 74 blocks with up to 49,883 oligos each
- Total of 3,691,288 sequences of 222 nt
- Pantheon-DNA with the same parameters as in the other in silico test in the paper

Preprocessing time increased significantly due to multiplexing in Chiron, which alone took 12 hours and 1 minute. Owing to the larger number of block tags, tag identification is performed by alignment at this stage.

### 3.4. Real-data experiment

While simulated data is used to evaluate individual processing strategies performed at each error channel profile, we also validate our codec in a real-data experiment using DNA synthesis and sequencing. We encode a 1.59 MB data folder containing three different PDF files with text and images embedded. DNA sequences were synthesized, sequenced, and decoded using our proposed pipeline.

In an experimental study using actual data, we encoded a folder containing three PDF files, totaling approximately 1.59 MB, into DNA. This encoding process, performed by the Apollo system, was completed in 31 seconds and 490 milliseconds. The outcome was the production of 15 data blocks, each with 4,489 or 4,490 sequences, and an additional block of metadata. Metadata encapsulates a comprehensive array of codec parameters in JSON format. These parameters include the mapping scheme, inner and outer code settings, the positioning of files within blocks, a list of primers specific to each block, and additional codec settings, such as randomization configurations and file-related information, including permissions, directory structure, and checksums.

The encoded information and metadata were translated into 67,854 DNA sequences of 198 nt. Amplification of the DNA was achieved using a universal primer that targeted the entire archived data, followed

by sequencing on the Illumina MiSeq platform with the MiSeq Reagent Kit v2, employing a paired-end strategy. The sequencing endeavor comprised two runs, yielding an average coverage of 68.89 with a standard deviation of 18.14. In the sequencing data, notably, the missing rate of sequences hovered around 0.03%, with individual runs exhibiting approximately 0.09% missing rates. The modest discrepancy in sequence recovery, together with coverage fluctuations and the lack of overlap among missing sequences across runs, is attributable to molecular sampling biases inherent to the sequencing process. Artemis successfully recovered all missing sequences using the embedded outer code.

**Impact of the metadata block on storage density** Let  $N$  denote the total number of sequences,  $N_{\text{meta}}$  the number allocated to metadata,  $\ell$  the sequence length (nt), and  $\rho$  the effective payload bits per nt (after primers/tags/ECC). The fractional overhead due to metadata is  $\eta_{\text{meta}} = N_{\text{meta}}/N$ , yielding the net density

$$D_{\text{net}} = (1 - \eta_{\text{meta}}) \rho.$$

For the 1.59 MB experiment with  $N=67,854$  and  $\ell=198$ , the total number of synthesized bases is 13,435,092. Once  $N_{\text{meta}}$  and  $\rho$  are set by the sequence layout,  $D_{\text{net}}$  follows directly.

A significant initial step in the decoding process involves identifying and separating the metadata block. Chiron completed metadata identification in 1 minute 30 seconds, followed by 730 milliseconds for finalization. Subsequently, Hermes validated the metadata block and dispatched residual reads to Hephaestus for consensus; the combined operation completed in approximately 200 milliseconds, after which decoding proceeded in Artemis. The pipeline then handed control to Artemis for decoding. Artemis completed the decoding process in 21 seconds and 40 milliseconds.

We also assessed the performance of Hephaestus alone, without being preceded by Hermes, in both demultiplexed and non-demultiplexed scenarios. In the non-demultiplexed scenario, we do not use Chiron to separate data blocks, emulating what is most commonly practiced in the literature. In the demultiplexed scenario, Hephaestus clustered all blocks within 1 minute. Conversely, handling non-demultiplexed data markedly increased the computational demand, taking 9 hours, 11 minutes, and 28 seconds to cluster. The marked difference in processing time underscores the critical role of block partitioning in DNA data storage systems, making this strategy crucial for scalability in our tests. It is important to note that both scenarios performed Hephaestus without parallelization. Moreover, a clustering method is not highly parallelizable; therefore, the demultiplexed scenario, with each block being processed independently, could significantly benefit from multiple Hephaestus instances running in parallel. This experiment demonstrates the versatility of the Pantheon-DNA system, showcasing its potential in real-world data storage, particularly when utilizing heavy algorithms, and its ability to efficiently handle erasures, further proving its efficacy and flexibility in managing and recovering DNA-encoded data.

#### 4. Conclusion

The central concept of Pantheon-DNA is the robust encoding-decoding framework with adaptive algorithms designed for Next-Generation Sequencing (NGS) preprocessing. Integral to this framework is the standard address library and inner code for accelerating the preprocessing of reads and enhancing error correction capabilities by applying MSA only where and when necessary within a data block.

Furthermore, our unique block data architecture is specifically designed to promote parallelization and selective access in the processing of NGS data and the decoding phase. This architectural approach optimizes throughput and significantly improves the efficiency of data handling and retrieval. Our codec manages DNA constraints, such as homopolymers and GC content, alongside a suite of user-defined constraints. We promote data retrieval by applying inner code for substitution errors, utilizing sequence alignment and consensus methods for

insertions and deletions (indels), and employing outer code for erasures. Such comprehensive handling of potential DNA sequencing errors supports high-fidelity storage and retrieval under the tested conditions. A distinguishing feature of Pantheon-DNA is that it integrates the preprocessing pipeline as an intrinsic component of the codec, comprising the modules Chiron, Hermes, and Hephaestus. The system is adept at switching algorithms based on error rates, employing Multiple Sequence Alignment (MSA) only when necessary.

Moreover, maintaining a versatile bioinformatics toolbox to assist pre-decoding activities enhances codec compatibility with contemporary NGS platforms and sequencing strategies. Standardizing the read for downstream analysis augments the system's efficiency and significantly extends the codec's lifespan. Additionally, our codec workflow and data architecture were rigorously validated through an actual data experiment, successfully encoding and decoding 1.59 MB of data, which comprised multiple files.

The empirical test underscores the robustness and reliability of the Pantheon-DNA system. Future research directions for Pantheon-DNA entail continuously refining the error correction code, particularly by integrating an inner code capable of recovering insertions and deletions (indels). This development aims to diminish the necessity for MSA, streamlining the error correction process. Additionally, improvements in the Hermes module are essential for enhancing address recognition sensitivity and consequently reducing the volume of reads necessitating processing by the Hephaestus module for greedy clustering. Addressing these areas will significantly optimize the preprocessing pipeline, ensuring more efficient data handling.

In conclusion, Pantheon-DNA makes a substantive contribution to DNA data storage by demonstrating a scalable, reliable, and adaptable codec and preprocessing pipeline. The framework provides a solid foundation for future work and may, in time, influence storage architectures as sequencing technologies and workloads evolve.

#### CRedit authorship contribution statement

**Adriano Galindo Leal:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Thiago Yuji Aoyagi:** Software, Methodology, Investigation, Formal analysis, Conceptualization. **André Guilherme Costa-Martins:** Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Diego Trindade de Souza:** Writing – original draft, Software. **Cristina Maria Ferreira da Silva:** Writing – original draft, Software, Data curation, Conceptualization. **Eduardo Takeo Ueda:** Validation, Software, Methodology. **Marcelo Gonzaga de Oliveira Parada:** Project administration, Funding acquisition. **Allan Eduardo Feitosa:** Software, Methodology. **André Fujita:** Writing – review & editing, Validation.

#### Code availability

The Pantheon-DNA implementation is proprietary to Lenovo and the source code cannot be publicly released. For peer-review verification, we can provide a containerized, reproducible executable (inputs, configuration files, and scripts to regenerate the reported figures and tables) under NDA, upon reasonable request to the corresponding author.

#### Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation and revision of this manuscript, we utilized Overleaf, ChatGPT-5, and Grammarly to ensure clarity and grammatical precision, as English is our second language. The authors assume full responsibility for creating the primary content and ensuring technical accuracy and have appropriately cited all secondary sources used in this publication.

## Funding

We extend our heartfelt thanks to Lenovo Tecnologia Brasil Ltda for the financial support provided through the incentive law from the Ministério da Ciência, Tecnologia e Inovações of Brazil. We also gratefully acknowledge support from IPT (Institute for Technological Research) and São Paulo Research Foundation (FAPESP) Grant Numbers #2017/50343-2, #2019/01664-6, #2020/09850-0 and #2021/11.905-0.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

The authors would like to express their most profound gratitude and appreciation to Hildebrando Lima, Anderson Ribeiro Correia, Natalia Neto Pereira Cerize, Maria Cristina Machado Domingues, Bruno Marinero Verona, Denis Bruno Virissimo, Alessandro Santiago dos Santos, Silvia Elisabete Ferrari, Melissa Revoredo Braga, Fabricia Garcia Barbosa, Dirce Ap. Rosaboni, Terezinha Chinelato, Benedita Mangeli, Matheus Marques de Miranda, and Leonardo Rodrigues de Souza Melo.

## Data availability

The datasets that support the findings of this study are available from the corresponding author upon reasonable request. This includes: (i) raw reads and intermediate consensus outputs from the 1.59 MB wet-lab experiment; and (ii) the in-silico inputs, parameter files, and summarized outputs for all simulations, including the 90 MB scale-up. Where applicable, we will share logs and plotting scripts sufficient to reproduce the reported figures and tables.

## References

- Church GM, Gao Y, Kosuri S. Next-generation digital information storage in DNA. *Science* 2012;337(6102).
- Goldman N, et al. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature* 2013;494:77–80.
- Ceze L, Nivala J, Strauss K. Molecular digital data storage using DNA. *Nat Rev, Genet* 2019;20:456–66.
- Xu J, et al. Selective prebiotic formation of RNA pyrimidine and DNA purine nucleosides. *Nature* 2020;582:60–6.
- Coudy D, Colotte M, Luis A, Bonnet J. Long-term conservation of DNA at ambient temperature. Implications for DNA data storage. *PLoS ONE* 2021;16(11):e0259868.
- Carmean D, et al. DNA data storage and hybrid molecular-electronic computing. *Proc IEEE* 2019;107(1):63–72.
- Gervasio JHDB, et al. How close are we to storing data in DNA? *Trends Biotechnol* 2024;42(2):156–67.
- Nguyen BH, et al. Scaling DNA data storage with nanoscale electrode wells. *Sci Adv* 2021;7:eabi6714.
- Caruthers MH. The chemical synthesis of DNA/RNA: our gift to science. *J Biol Chem* 2013;288(2):1420–7.
- van Dijk EL, et al. Ten years of next-generation sequencing technology. *Trends Genet* 2014;30(9):418–26.
- He B, et al. Assessing the impact of data preprocessing on analyzing next generation sequencing data. *Front Bioeng Biotechnol* 2020;8(817).
- Kosuri S, Church G. Large-scale de novo DNA synthesis: technologies and applications. *Nat Methods* 2014;11:499–507.
- Sims D, et al. Sequencing depth and coverage: key considerations in genomic analyses. *Nat Rev, Genet* 2014;15:121–32.
- Westcott SL, Schloss PD. De novo clustering methods outperform reference-based methods for assigning 16S rRNA gene sequences to operational taxonomic units. *PeerJ* 2015;3:e1487.
- Rognes T, Flouri T, Nichols B, Quince C, Mahe F. VSEARCH: a versatile open source tool for metagenomics. *PeerJ* 2016;4:e2584.
- Rashtchian C, Makarychev K, Racz MZ, Ang S, Jevdjic D, Yekhanin S, et al. Clustering billions of reads for DNA data storage. In: *Neural information processing systems (NIPS)* 2017; 2017. p. 3360–71.
- Organick L, et al. Random access in large-scale DNA data storage. *Nat Biotechnol* 2018;36:242–50.
- Nakamura K, et al. Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Res* 2011;39(13).
- Ross G, et al. Characterizing and measuring bias in sequence data. *Genome Biol* 2013;14(R51).
- Heckel R, Mikutis G, Grass RN. A characterization of the DNA data storage channel. *Sci Rep* 2019;9(9663).
- Weindel F, et al. Embracing errors is more effective than avoiding them through constrained coding for DNA data storage. In: *2023 59th annual allerton conference on communication, control, and computing*; 2023. p. 1–8.
- Reed IS, Solomon G. Polynomial codes over certain finite fields. *J Soc Ind Appl Math* 1960;8(2):300–4.
- Grass RN, Heckel R, Puddu M, Paunescu D, Stark WJ. Robust chemical preservation of digital information on DNA in silica with error-correcting codes. *Angew Chem, Int Ed Engl* 2015;54:2552–5.
- Blawat M, et al. Forward error correction for DNA data storage. *Proc Comput Sci* 2016;80:1011–22.
- Erlach Y, Zielinski D. DNA fountain enables a robust and efficient storage architecture. *Science* 2017;355:950–4.
- Press WH, Hawkins JA, Jones Jr SK, Finkelstein LJ. HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints. *Biophys Comput Biol* 2020;117(31):18489–96.
- Sharma P, et al. Efficiently enabling block semantics and data updates in DNA storage. In: *2023 56th IEEE/ACM international symposium on microarchitecture*; 2023. p. 555–68.
- Hughes RA, Ellington AD. Synthetic DNA synthesis and assembly: putting the synthetic in synthetic biology. *Cold Spring Harb Perspect Biol* 2017;9:a023812.
- Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* 2006;22(13):1658–9.
- Ghods M, Liu B, Pop M. Dnaclust: accurate and efficient clustering of phylogenetic marker genes. *BMC Bioinform* 2011;12(1):1–11.
- Shimizu K, Tsuda K. Slidesort: all pairs similarity search for short reads. *Bioinformatics* 2011;27(4):464–70.
- Zorita E, Cusco P, Filion GJ. Starcode: sequence clustering based on all-pairs search. *Bioinformatics* 2015;31(12):1913–9.
- James BT, Luczak BB, Girgis HZ. MeShClust: an intelligent tool for clustering dna sequences. *Nucleic Acids Res* 2018;46(14):e83.
- Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. *Sov Phys Dokl* 1966;10(8):707–10.
- Davey MC, Mackay DJC. Reliable communication over channels with insertions, deletions, and substitutions. *IEEE Trans Inf Theory* 2001;47(2):687–98.
- Sima J, Gabrys R, Bruck J. Optimal systematic t-deletion correcting codes. In: *2020 IEEE international symposium on information theory (ISIT)*; 2020.
- Lenz A, et al. Coding over sets for DNA storage. *IEEE Trans Inf Theory* 2020;66(4):2331–51.
- Fertonani D, Duman TM, Erden MF. Bounds on the capacity of channels with insertions, deletions and substitutions. *IEEE Trans Commun* 2011;59(1):2–6.
- Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol* 1970;48(3):443–53.
- Smith TF, Waterman MS. Identification of common molecular subsequences. *J Mol Biol* 1981;147(1):195–7.
- Gusfield D. Algorithms on strings, trees and sequences. Cambridge University Press; May 1997.
- Zhang J, et al. PEAR: a fast and accurate illumina paired-end reAdmergeR. *Bioinformatics* 2014;30:614–20.
- Hirschberg DS. A linear space algorithm for computing maximal common subsequences. *Commun ACM* 1975;18(6):341–3.
- Bao J, Yuan R, Bao Z. An improved alignment-free model for DNA sequence similarity metric. *BMC Bioinform* 2014;15(321).
- Feitosa AE, Aoyagi TY, Leal AG, da Costa-Martins AG, da Silva CMF, de Souza DT, et al. Encoding digital data using oligonucleotides, United States Patent Number US 12,406,750 B2, granted September 2, 2025. Patent application 18/459,312 filed August 31, 2023; pre-grant publication US 2025/0078958 A1 (March 6, 2025). Available at: <https://patentcenter.uspto.gov/applications/18459312>.
- Huang W, Li L, Myers JR, Marth GT. ART: a next-generation sequencing read simulator. *Bioinformatics* 2012;28(4):593–4.