Supplementary information

Contents:

Appendix 1	Data retrieval performance on different media	···	1 / 44
Appendix 2	Sequence constraints for compatibility	···	3 / 44
Appendix 3	Variants enabled in SPIDER-WEB	···	5 / 44
Appendix 4	Performance comparison with early efforts with relevant purpose	···	8 / 44
Appendix 5	Construction details of the high-compatibility coding digraph and its components	···	15 / 44
Appendix 6	Mathematical rationality of the new data retrieval pipeline	···	16 / 44
Appendix 7	Primary factors influencing direct retrieval rate	···	18 / 44
Appendix 8	Prevention and mitigation strategies for sequence loss	···	21 / 44
Appendix 9	Tool selection for the conventional data retrieval pipeline	···	25 / 44
Appendix 10	Explanation and measurement of non-blocking retrieval mode	···	26 / 44
Appendix 11	Global constraint design for ultra-long data-coding sequences	···	30 / 44
Appendix 12	Construction details of the 2,950-nt sequences used in the demonstration	···	32 / 44
Appendix 13	Instantaneous data retrieval efficiency under non-blocking retrieval mode	···	36 / 44
Appendix 14	Analysis of data backlog risk and sequencing throughput release	···	38 / 44
Appendix 15	Optimisation of data structures and algorithms	···	41 / 44

1 Data retrieval performance on different media

1.1 Performance metrics in widely used storage media

A variety of performance metrics are used to characterize data retrieval efficiency across storage technologies. Three most fundamental and widely used ones are latency, throughput and access time, which capture the temporal, operational and application aspects of the data retrieval process, respectively. Their definitions and usage vary subtly across many research and industry documents. Herein, we adopt the classical definitions tailored to the performance context to ensure consistency[1]:

- Latency refers to the waiting time between issuing a read request and the data becoming ready. It primarily focuses on the "waiting" component, excluding the data transfer process itself.

- Throughput refers to the amount of data read per unit time. It mainly reflects the performance level a system can achieve in real-world operations (i.e., the standard performance at 1X).

- Access time refers to the duration between when a memory receives a read request and when the corresponding data retrieval operation is completed. It primarily reflects the overall time-cost to retrieve all the required data, which is suited to be used as a reference for various downstream applications.

Moreover, the relationships among the three metrics can be described by the following equation:

$$\text{throughput} = \frac{\text{amount of data}}{\text{access time} - \text{latency}} = \frac{\text{amount of data}}{\text{termination point of access time} - \text{termination point of latency}} \tag{S1}$$

These definitions provide a uniform framework for comparing the data retrieval efficiency of diverse data storage systems. As follows, we illustrate the retrieval characteristics of DNA-based data storage, and use these metrics to make a qualitative comparison to that of optical data storage, magnetic data storage, and semiconductor data storage, to assist the performance evaluation of SPIDER-WEB.

1.2 Definitions of the metrics on DNA-based data storage

To make a precise definition for the retrieval performance of DNA-based data storage, three points (i.e., the termination point of latency, the amount of data and the termination point of access time) must be declared at first and aligned to that of conventional storage to make the performances of various media comparable.

1.2.1 Termination point of latency

In conventional storage systems, latency corresponds to the arrival of the first bit at the interface, following mechanical or electronic delays. In contrast, DNA-based data storage lacks a well-defined read order or physical addressability. Moreover, its data retrieval pipeline progresses through a series of morphologically distinct states: from molecular form to symbolic nucleotide sequences, and eventually to binary representations. Under such circumstances, it is difficult to define the termination point of latency since the point of "data becomes ready" is unambiguous. Considering these issues, we define the termination of latency as the moment when molecular signals have been completely converted into sequencing reads (strings consisting of A/C/G/T).

Under this definition, the essential information substrate is available for decoding. This interpretation also addresses a likely objection, i.e., in traditional systems, latency ends when the first bit becomes available. An analogous condition can then be satisfied in DNA storage in principle, provided two assumptions hold: 1) the raw sequence reads are error-free, and 2) the corresponding binary message is independent (e.g., not entangled with other reads by operations such as Exclusive-OR[2]). Under these conditions, it would be possible to decode the binary message – and thus the first bit – directly from an individual sequence read. Although sequencing errors may hinder such direct access in practice, conceptually, the information is already embedded in the symbolic form, and the definition can be aligned to that of other types of data storage systems.

1.2.2 Definition of the amount of data

The amount of data is typically defined as the volume of user-requested content delivered from the storage medium to the host system, i.e., only the payload (i.e., the actual file content) is included, and auxiliary information is excluded, such as physical addressing metadata, transmission control headers, and low-level error-correcting bits. This separation is feasible for conventional storage architectures since they ensure strict physical ordering and addressing. By contrast, in DNA-based data storage, the DNA molecules are preserved and sequenced in an essentially unordered and stochastic manner. Under such circumstances, we define the amount of data as the total number of bits of binary messages of both the payload and the index segments. The inclusion of index segments is necessary since it guarantees the successful data reconstruction under the unordered retrieval framework of molecular data.

1.2.3 Termination point of access time

In conventional data storage systems, access time ends when the full set of requested data has been transferred and made available for processing. It is worth noting that this point is located at the interface between the physical storage layer and the system memory or buffer, but does not depend on the reconstruction of the original file structure or content by higher-level software. We adopt an analogous definition for DNA-based data storage, i.e., the end of access time is the moment when all pure binary messages have been successfully recovered from the available raw sequence reads. This set of messages forms a complete digital representation of the encoded data, excluding any auxiliary redundancy introduced for error correction. This definition is conceptually aligned with the controller-level definition in conventional systems. Once all pure binary messages are available, the symbolic and structural decoding of the stored data has been completed at the message level, even if data reassembly is deferred to a later stage. In other words, just as traditional systems regard the bitstream as the retrieval endpoint, we regard the collection of pure binary messages as the digital termination point of the DNA-based data retrieval pipeline.

1.3 Performance of DNA-based data storage

Building upon the definitions, we are now equipped to make an approximate comparison between the performance of DNA-based data storage and conventional data storage. While the data retrieval process in DNA storage is structurally distinct and biochemically mediated, the metric of data retrieval efficiency allows for a quantitative assessment of its throughput-equivalent characteristics.

A study in 2024 by Zhao et al.[3] provides one of the few time-resolved baselines of DNA-based data storage, reporting both the data size and the retrieval duration for individual experiments. The authors conducted two retrieval tasks: a 219-byte text file and a 4109-byte image file. For both cases, the reported retrieval time – including clustering, back-projection, and decoding – was one hour. Based on our workflow, the data retrieval efficiency for these tasks can be estimated as follows. For the 219-byte file, $E = 219 \times 8 \div 3600 \approx 0.5$ bit/s, and for the 4109-byte file, $E = 4109 \times 8 \div 3600 \approx 9.1$ bit/s. A detailed analysis in Appendix 4 further supports the credibility of this performance estimate.

While we attempted to include recent relevant efforts in our quantitative comparison, several practical limitations prevent direct performance alignment. Specifically, one study relies on GPU-based computation[4], making its reported efficiency difficult to compare directly with our CPU-based workflow and earlier CPU-based studies such as Zhao et al.[3] under equivalent resource constraints. Another study proposes a high-efficiency retrieval framework[5]; however, to our knowledge, its publicly released repository does not currently provide the encoding component, making it difficult to reconstruct the full encoding-decoding pipeline required for a complete and fair end-to-end assessment. We therefore discuss these works as important recent progress, but do not include them in the quantitative comparison in order to maintain methodological consistency.

1.4 Performances of optical, magnetic, and semiconductor data storage media

We also make a survey for the data retrieval throughput of three dominant storage media types: optical, magnetic, and semiconductor materials, as a reference to compare with DNA-based data storage. Each of them relies on a distinct physical mechanism for encoding and reading digital information, leading to fundamentally different characteristics as follows.

Optical storage encodes data as a series of microscopic pits and lands on the surface of a disc. These surface variations cause differential light reflection when being scanned by a laser, further interpreted as binary data. Retrieval is achieved by detecting the reflected light using photodiodes. Modern optical media include Compact Disc (CD), Digital Versatile Disc (DVD), and Blu-ray Disc (BD), differentiated primarily by laser wavelength and data density. According to IEC 60908:1999, the standard data retrieval throughput for CD is 150 kilobytes per second. DVD, as specified in ISO/IEC 16448:2002, achieves 1.385 megabytes per second, while BD reaches 4.5 megabytes per second as per the Blu-ray Disc™ Format White Paper. Here, we define the typical throughput range for optical storage media as 150 kilobytes to 4.5 megabytes per second.

Magnetic storage encodes binary data via the orientation of magnetized domains on a physical medium. A read head senses the polarity of these domains during retrieval. Common implementations include magnetic tape (MT) and hard disk drives (HDD). For magnetic tape systems, Linear Tape-Open (LTO-1) delivers a standard throughput of 20 megabytes per second[6]. Magnetic tape is predominantly used for archival purposes due to its cost-effectiveness, despite slower retrieval speeds. By contrast, HDDs provide faster access. While their maximum throughput is constrained by mechanical movement, improvements in interface and cache design have increased standard throughput to the 100-250 megabytes per second range under SATA III specifications. Here, we define the typical throughput range for magnetic storage media as 20 to 250 megabytes per second.

Semiconductor storage media use arrays of metal-oxide-semiconductor memory cells fabricated on silicon chips. We focus here on non-volatile flash memory, which preserves data in the absence of power. The most common implementation is the solid-state drive (SSD) based on NAND flash technology. SSD throughput is governed by both the underlying flash architecture and the interface protocol. According to the JEDEC JESD218B standard, consumer-grade SSDs achieve 550 megabytes per second under the SATA III interface and up to gigabytes per second under NVMe PCIe 4.0. Here, we define the typical throughput range for semiconductor storage media as 550 megabytes to 7 gigabytes per second.

2 Sequence constraints for compatibility

2.1 Mathematical formulation of the regionalised constraints

According to the early efforts[2,7–11], two kinds of regionalised constraints are widely investigated: the homopolymer run-length-limited constraint and the regionalised GC content constraint.

In a DNA sequence, a homopolymer run refers to a maximal consecutive sub-sequence of the same symbol and the number of nucleotides in each run is called its run-length. For example, $v = \mathrm{AAAATTCGG}$ contains four runs with run-lengths as 4,2,1,2 accordingly. Typically, in DNA-based data storage, long runs should be avoided. A homopolymer run-length-limited constraint is of the form "the maximal run-length is at most h".

Another regionalised constraint, known as the regionalised GC content, requires that the GC-ratio in any consecutive sub-sequence of length k is bounded within an interval. The parameter k is called the window length and usually $k \geq h$. Given $0 \leq \epsilon_1 \leq \epsilon_2 \leq 1$ and the window length k, a regionalised GC content constraint is of the form "in any consecutive sub-sequence of length k, the sum of the numbers of G and C is within the interval $[\epsilon_1 \times k, \epsilon_2 \times k]$".

Additionally, we sometimes come across a third kind of constraint regarding undesired motifs. Such motifs usually have a serious impact on specific operations or storage environments. Let Θ be a set of undesired motifs. A DNA sequence v is Θ-free if each motif in Θ does not appear in v as a consecutive sub-sequence. Usually, we only consider undesired motifs with length less than or equal to the window length k.

2.2 Mathematical formulation of the global constraints

In addition to regionalised constraints that apply within sliding windows, two global constraints might be required to ensure the compatibility of encoded DNA sequences. For each individual DNA sequence, these constraints apply to the whole sequence rather than a sliding window.

To reduce the risk of amplification errors, such as mispriming, template slippage, or undesired recombination during molecular assembly[12–14], it is often required that no sufficiently long subsequence appears more than once within the same DNA sequence. Let v be a DNA sequence of length ℓ. Given a length threshold n, the fragment repetition constraint requires that no exact subsequence of length at least n appears more than once within v. Formally, for all $i, j \in [1, \ell - n + 1]$ with $i \neq j$, we require $v[i : i+n-1] \neq v[j : j+n-1]$.

In addition, reverse complement DNA segments can form hairpins or duplexes during synthesis or storage[15,16], compromising the stability of the following molecular workflows. Let v be a DNA sequence of length ℓ. Given a length threshold n, the reverse complement constraint requires that no subsequence of length at least n should appear as its reverse complement (overlapping situation has been ruled out). To formalise this constraint, we define the reverse complement operation. Let the Watson-Crick complement satisfy: $\overline{A} = T$, $\overline{T} = A$, $\overline{C} = G$, and $\overline{G} = C$. For any DNA subsequence $u = (x_1, x_2, \ldots, x_n)$ of length n, its reverse complement is defined as $\mathrm{RC}(u) = (\overline{x}_n, \overline{x}_{n-1}, \ldots, \overline{x}_1)$. That is, $\mathrm{RC}(u)$ is obtained by reversing the order of the nucleotides and replacing each with its Watson-Crick complement. Then for any two non-overlapping subsequences $u_1 = v[i : i+n-1]$ and $u_2 = v[j : j+n-1]$, where $1 \leq i < i+n-1 < j < j+n-1 \leq \ell$, the constraint requires that $u_2 \neq \mathrm{RC}(u_1)$.

2.3 Mathematical formulation of the mutual distance constraints

Unlike global constraints, which evaluate the internal structure of each sequence individually, mutual constraints impose distance requirements between sequences within a set. These mutual constraints also enable advanced storage functionalities, including fuzzy search[17] and information editing[18] at the molecular scale. For all DNA sequences in a pool, a library, or a mixture, these constraints apply to the whole sequence rather than a sliding window.

When generating a set of sequences $\{v_1, v_2, \ldots, v_M\}$, it is often desirable to enforce a minimum pairwise distance, such as Hamming distance[19] or Levenshtein distance[20], to reduce the risk of cross-contamination and to enhance sequence-level distinguishability. Given a threshold d and a distance metric, the sequence difference constraint requires that the distance between any two sequences must be at least d, expressed formally as: $\forall 1 \leq i < j \leq M$, $\mathrm{D}(v_i, v_j) \geq d$.

2.4 Implementation in SPIDER-WEB

All aforementioned sequence constraints, including both regionalised and global types, are explicitly supported in the SPIDER-WEB framework. Their implementation, however, is integrated at different stages of the encoding or generation process depending on the nature of the constraint.

For all the regionalised constraints, enforcement is integrated directly into the generation of the coding digraph. Any vertex (i.e., k-mer) that would lead to a violation of these constraints is excluded during the construction process. This ensures that all walks through the digraph inherently satisfy the local sequence requirements, and no post hoc screening is needed.

Global constraints such as the fragment repetition constraint and the reverse complement constraint are handled during the message encoding process or sequence generating process. In these cases, edges that would result in constraint violations are temporarily masked during graph traversal. This dynamic screening process[2,21] ensures that the generated DNA sequences comply with the specified global constraints while preserving the structure of the underlying digraph. It is important to note that under

particularly strict regionalised constraints, the resulting digraph may become too sparse to support certain encoding tasks. This can lead to failures in encoding or to a limited number of producible sequences. To address this, one may modify the virtual vertex design or relax constraint parameters to restore feasibility.

For the mutual constraints, SPIDER-WEB currently supports them in the sequence generating process only. In this mode, each newly generated sequence is compared with the set of previously accepted sequences. It is retained only if it satisfies the constraint with respect to all existing members in the set.

2.5 Usage in this work

This study involves three categories of sequence constraints:

- **regionalised constraints**: homopolymer run-length, regionalised GC content, and undesired motifs;
- **Global constraints**: fragment repetition and reverse complement;
- **Mutual distance constraints**: Hamming distance and Levenshtein distance.

Among these, the regionalised and global constraints are extensively discussed and evaluated throughout both the Main Text and the supplementary sections. Mutual distance constraints are fully implemented in our framework and available for use, but are not elaborated in this manuscript, as they are not central to the questions addressed.

3 Variants enabled in SPIDER-WEB

3.1 Complete algorithm generation process

The algorithm generation process in SPIDER-WEB is responsible for constructing a customised coding digraph: a central structure used both for the message encoding process, sequence decoding process, sequence correcting process (one of the processes in the data retrieval pipeline), and sequence generating process (producing DNA sequences of a given length from the coding digraph without binary messages). This process consists of a sequence of deterministic steps that embed biochemical constraints, information density, and error-resilience directly into the graph architecture. The procedure consists of the following steps (Fig. S1):

1. Enumerate all possible k-mers over the DNA alphabet (A, C, G, T). Each k-mer is evaluated against a set of predefined sequence constraints. Only k-mers that satisfy all constraints are retained as vertices in the graph.

2. Construct a directed graph (digraph) by connecting retained k-mers based on suffix-prefix overlap. An arc from vertex u to v is added if the $(k-1)$-suffix of u matches the prefix of v.

3. Identify and remove vertices with insufficient in-degree or out-degree to ensure adequate local connectivity. This step will continue until no vertices are removed.

4. Among the resulting connected components, select the the largest connected component[22] that preserves maximal path diversity and eliminates the instability of information density.

5. If fast-mode encoding is required, apply optional disambiguation to further reduce walk ambiguity. The resulting graph structure supports deterministic and high-throughput encoding via direct walk traversal.

6. Assign a deterministic digit-to-nucleotide mapping to the outgoing arcs of each vertex. This mapping establishes a one-to-one correspondence between digit strings and DNA sequences (see Fig. 1c).

7. If global constraints are required, construct an optional global screening component to enforce global constraints that are not addressed by regionalised k-mer filtering (see Appendix 2). This component is saved alongside the coding digraph.

8. Save the final coding digraph and any additional components as the complete infrastructure for the following tasks.

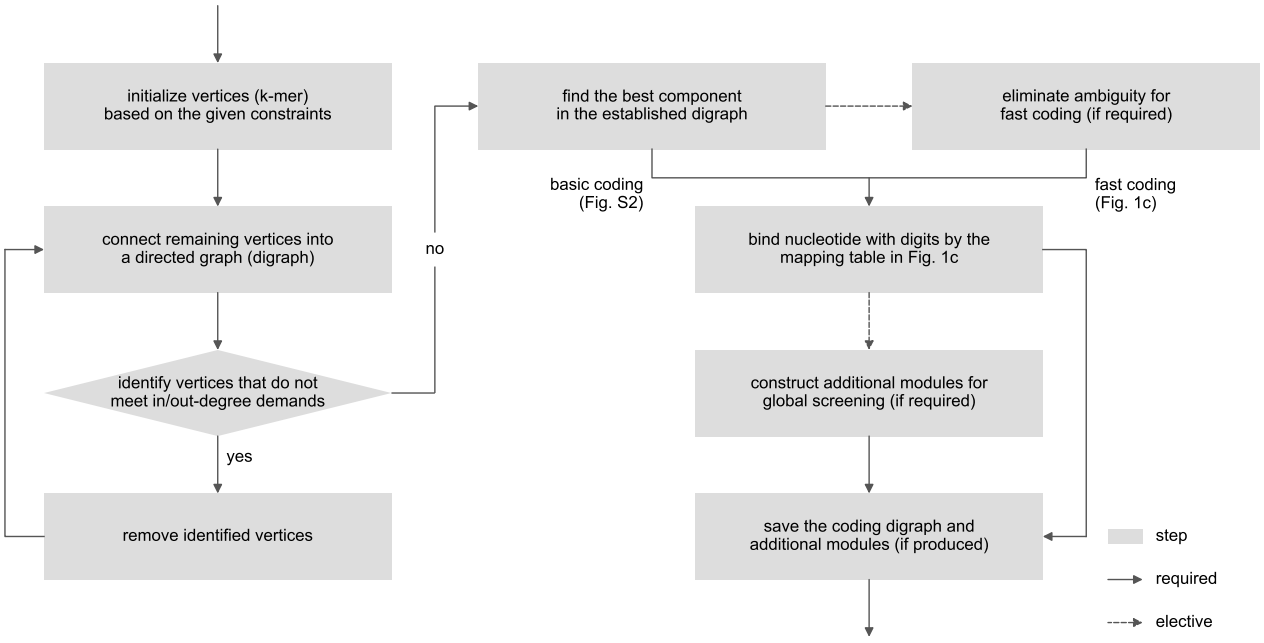

Fig. S1 | Complete pipeline of coding algorithm generation. It provides a list of variations that have been implemented in the code repository for different purposes.

A potential concern regarding SPIDER-WEB is that the complexity of the generation steps may lead to prohibitive computational costs. To address this, we conducted a benchmark evaluation under a typical constraint set with an observation window of length 10. Specifically, we considered all practical combinations of widely used regionalised constraints, including the homopolymer run-length-limited constraint and the regionalised GC content constraint (detailed in Appendix 2).

Fig. S2 reports the runtime required to generate the coding digraphs under the specified constraint settings. In all evaluated cases, the generation completes within 30 seconds. These results demonstrated that SPIDER-WEB enables efficient algorithm generation, offering a practical and scalable alternative to manual encoding design, especially when dealing with complex or application-specific constraint sets.

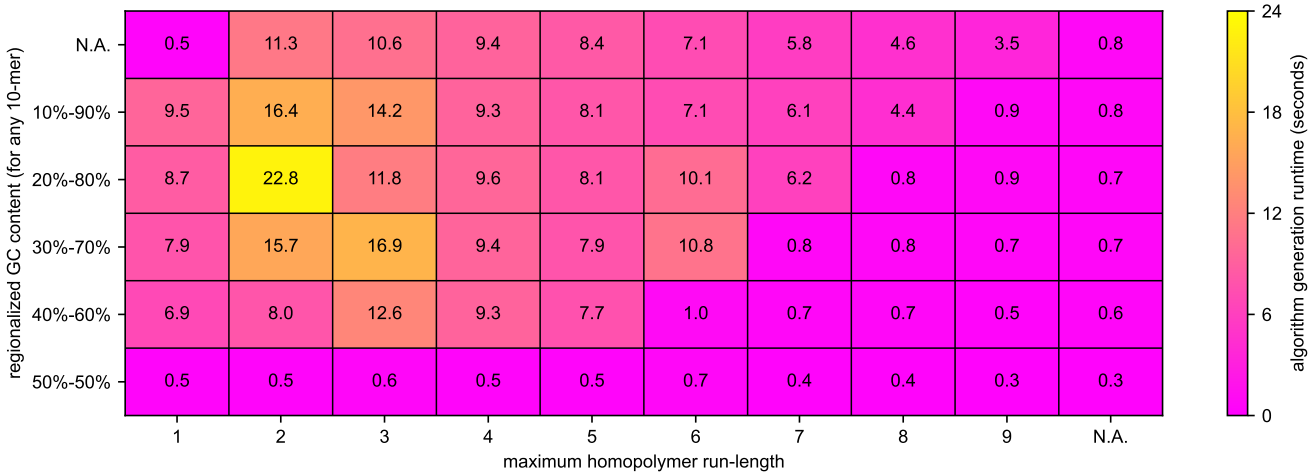

Fig. S2 | Algorithm generation runtime under commonly used constraints. Each case includes a combination of the homopolymer run-length and the regionalised GC content constraints, with an observation window of length 10.

3.2 Coding digraph with basic mode and fast mode

To accommodate varying balances between encoding efficiency and computational complexity, SPIDER-WEB provides two distinct modes of message encoding: basic mode and fast mode. Both modes share the same underlying coding digraph and apply symmetrically to sequence decoding, but they differ in how bit-streams are mapped onto walks through the graph structure.

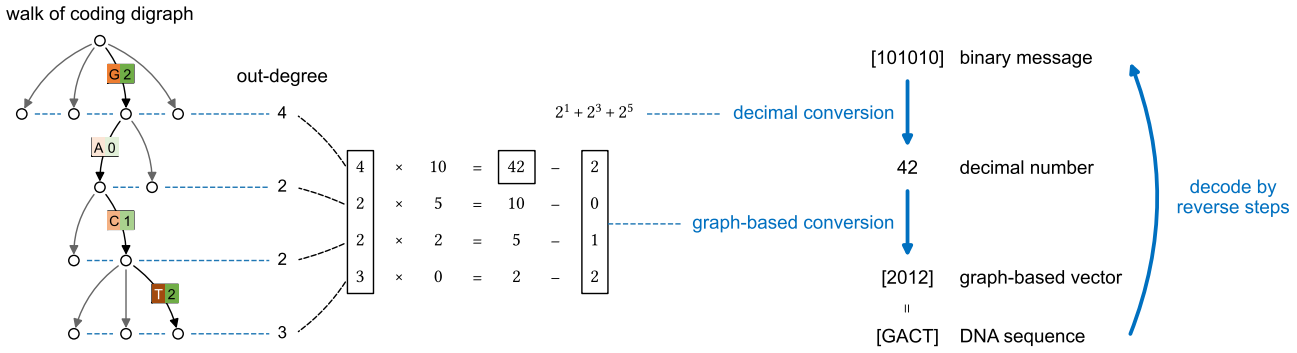

Fig. S3 | Message encoding and sequence decoding process using the basic mode. A binary message $[101010]$ is first converted to a decimal number 42, which is then mapped to a graph-based vector $[2012]$ using successive divisions guided by vertex out-degrees. The resulting vector is finally translated into a DNA sequence [GACT] via arc label mapping.

The basic mode (Fig. S3) prioritises maximal information density and encodes a binary message into a compact walk over the digraph, starting from a given virtual vertex. This approach involves three intermediate transformations: the binary message is first converted into a decimal integer, which is then translated into a graph-based vector according to vertex-specific out-degrees, and

finally mapped to a nucleotide sequence via arc labels. While optimal in terms of storage efficiency, this method incurs a higher computational cost due to the need for base conversions and backward traversal during decoding.

The fast mode (Fig. 1c), by contrast, emphasises computational speed and enables streaming-like, stepwise message encoding. Instead of converting the full message into a single integer, the encoder determines the number of bits to read at each vertex based on its out-degree. This bit-length is typically logarithmic in the out-degree (i.e., no bit for out-degree 1, 1 bit for out-degree 2; 2 bits for out-degree 4), and the corresponding bits are directly translated into nucleotide arcs. This mode reduces encoding delay and supports fast decoding, particularly suited for real-time or hardware-constrained applications.

In both modes, the sequence decoding process is deterministic and proceeds by reversing the walk on the coding digraph, with the predetermined virtual vertex. Because each arc is uniquely labelled and the graph is constructed to be prefix-free, the original binary message can be fully reconstructed once the DNA sequence is traversed.

3.3 Additional components for global constraints

To support the enforcement of global constraints, SPIDER-WEB includes two classes of additional components. The first class consists of constraint-detecting components (e.g., "create_palindrome_map" function in the code repository), which are designed to rapidly determine whether a given DNA sequence satisfies given global constraints. These components are lightweight and can be integrated into any functional stage, including message encoding, sequence decoding, and sequence generating. They also support sequence correcting, if needed. The second class consists of remedial components (e.g., "create_coexisting_map" function in the code repository), which are invoked when the presence of global constraints prevents a target function – including message encoding, sequence decoding, and sequence generating – from completing successfully. Instead of modifying existing walks or relaxing constraints, these components attempt to re-plan a new suitable virtual vertex from which the desired operation can be re-executed under the given global constraints. Together, these two classes of components provide flexible and robust support for constraint management throughout our framework.

4 Performance comparison with early efforts with relevant purposes

To contextualise the performance of SPIDER-WEB within broader algorithmic developments, we compared it along four dimensions: (1) maximising information density under complex sequence constraints, (2) leveraging algorithmic structure to support error tolerance, (3) improving data retrieval efficiency through tailored combinations of algorithmic modules, and (4) improving data retrieval efficiency through alternatives to conventional pipelines. Where the implementation, experimental setup, and computational configuration could be reasonably aligned with our framework, we performed quantitative comparisons; otherwise, we restricted the discussion to qualitative comparison based on algorithmic design. For each dimension, we selected one representative and conceptually related prior effort to clarify the design rationale of SPIDER-WEB. Notably, we do not conduct an exhaustive survey, and while we assert the merits of SPIDER-WEB, we do not claim its superiority across all settings. In fact, different methods are suited to different application scenarios, each with its own considerations in terms of compatibility, flexibility, scalability, and implementation complexity.

4.1 Maximising information density under sequence constraints

To evaluate how SPIDER-WEB performs in maximising information density under complex sequence constraints, we selected a representative baseline for comparison. Our goal is to benchmark against methods that are explicitly designed to (1) accommodate arbitrary regionalised constraint combinations and (2) achieve the highest possible information density under those constraints. Within this scope, a natural choice is DNA Fountain[2], which introduced the widely adopted screening-based approach to constraint compliance and remains one of the most prominent methods employing this strategy.

DNA Fountain attempts to maximise information density by generating large pools of candidate DNA sequences using the droplet operation, which are constructed by applying the bit-wise exclusive-OR operation to a randomly selected subset of binary message fragments. Constraint-compliant DNA sequences are then selected through a constraint-based screening process. While this approach is highly flexible and adaptable to a wide range of constraints, its final information density is influenced by two main factors: the degree distribution used during droplet encoding, and the strictness of the screening criteria[21].

Fig. S4 presents a direct comparison of average information density and standard deviation for both DNA Fountain and SPIDER-WEB (in basic and fast coding modes) across all tested combinations of homopolymer run-length-limited and regionalised GC content constraints, under an observation window of length 10.

Under weak or moderate constraints, DNA Fountain achieves relatively high information densities, with values reaching up to 1.744 bit/nt in favourable conditions. This reflects the ability of DNA Fountain to effectively utilise unconstrained sequence space when the screening burden is low[21]. While the theoretical maximum for information density is 2 bits/nt, DNA Fountain inherently requires the generation of additional redundant sequences in order to satisfy the conditions necessary for successful decoding. These include sufficient fragment coverage and convergence requirements within the peeling decoder, both of which contribute to a reduction in net information density even under relaxed constraint settings. As the constraints become more stringent, particularly when multiple conditions are applied simultaneously, such as narrow GC content windows combined with strict homopolymer limits, the performance of DNA Fountain deteriorates markedly. In several tested configurations, the average information density drops below 1.6 bit/nt. Moreover, the standard deviation across repeated trials increases significantly, reflecting the inherent stochasticity and instability of the screening-based approach in the presence of multiple overlapping constraints. These fluctuations arise from the fact that the survival rate of valid sequences in the candidate pool becomes increasingly unpredictable. As a result, a large number of droplets often need to be generated to obtain a sufficient subset that satisfies all specified constraints. This increases the difficulty of maximising information density and compromises the stability of information density across files of the same size[21].

On the contrary, SPIDER-WEB maintains consistently high information density across all constraint combinations, with standard deviations ≤ 0.001. This stability arises from its algorithmic design, in which constraint enforcement is intrinsically embedded into the structure of the coding digraph. Rather than relying on constraint-based screening when randomly producing random DNA sequences, SPIDER-WEB guarantees that all produced sequences inherently satisfy the constraints, thereby enabling the system to consistently approach the theoretical information density. Even in the fast mode, which sacrifices a small fraction of capacity in exchange for improved decoding speed, the variance remains negligible, and the overall information density still surpasses that of screening-based methods under moderate to strict constraints.

These results demonstrated that SPIDER-WEB offers a principled and reliable mechanism for constraint-compliant message encoding, avoiding the unpredictability and inefficiencies associated with sampling-based screening approaches when constraint combinations become stringent.

4.2 Leveraging algorithmic structure to support error tolerance

To evaluate the capability of SPIDER-WEB to support error tolerance via algorithmic structure, we compared it with HEDGES[23], a graph-based decoder that shares conceptual similarities. Both SPIDER-WEB and HEDGES utilise sequence graphs to guide error correction. Nevertheless, their decoding strategies differ: SPIDER-WEB employs a local search mechanism, exhaustively generating candidate sequences around the expected decoding path, while HEDGES applies a global A-star search[24], selecting the optimal decoding path by scoring and ranking all potential trajectories through the graph.

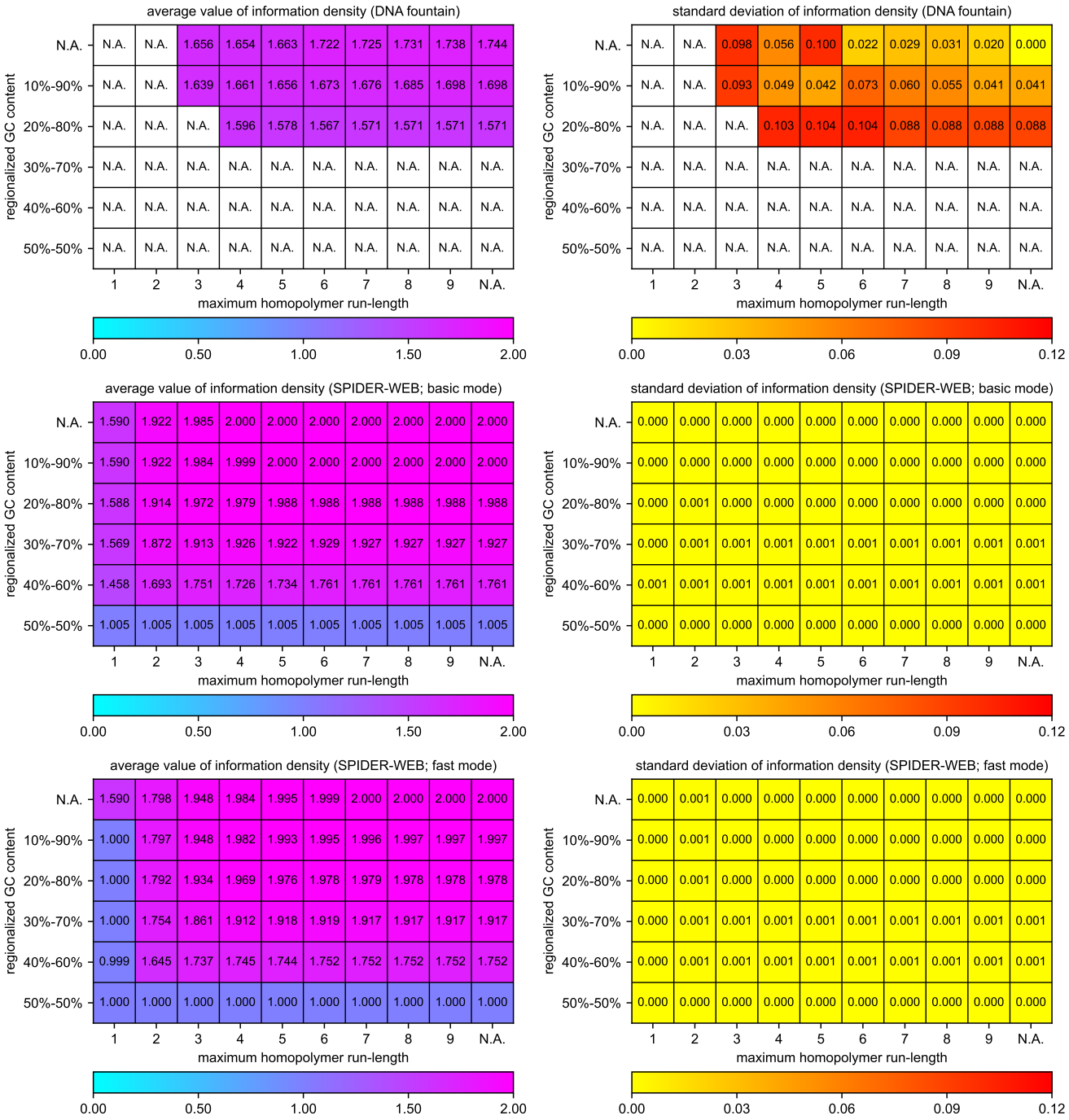

Fig. S4 | Information density comparison of DNA Fountain, SPIDER-WEB (basic mode), and SPIDER-WEB (fast mode). Each DNA sequence is 200 nt in length, encoding a total of 400,000 bits of digital data. Index ranges and bit-level logical redundancy (e.g., error-correcting codes) are excluded from the calculation. For each constraint combination, 20 independent trials were conducted to evaluate the average information density and its variability. The results for DNA Fountain were obtained using the default parameters provided by its original implementation (see Supplementary Materials in its paper). "N.A." in the DNA Fountain results indicates that valid output could not be generated during the message encoding process, since it is failed to find any valid sequence within 1,000 consecutive screening attempts.

To enable as closely equivalent a comparison as possible, we re-implemented HEDGES in Python, since its original design was not directly compatible with our experimental framework. Given the inherent inefficiency of Python compared to compiled languages, we avoided comparing raw execution time. Instead, we used the vertex access times during the sequence correcting process as a platform-independent measure of computational effort. Furthermore, due to the high runtime of HEDGES, we limited testing to 400 randomly sampled decoding cases per error rate.

Fig. S5a shows the vertex access times by HEDGES when correcting 200-nt DNA sequences under identical sequence design and simulated error conditions. As expected, when no errors are introduced, HEDGES visits exactly 200 vertices, confirming correctness (also 100% success rate; Fig. S5b). However, as errors accumulate, the vertex access times increase dramatically. In the worst case, we observed up to 20,351,643 vertex access times for a single correcting attempt. Compared with the corresponding SPIDER-WEB results (Fig. 2b), HEDGES consistently exhibits higher correcting complexity and greater variability. This indicates that global search introduces significant instability in practical error-correcting scenarios.

In addition to computational overhead, HEDGES also demonstrates degraded decoding accuracy at higher error rates. The probability of successfully reconstructing the original sequence drops sharply as errors increase. This explains why the original HEDGES design requires additional layers of protection, such as Reed-Solomon code[25], to ensure sufficient fidelity. In contrast, our strategy enables robust error-correcting performance without reliance on any logical redundancy.

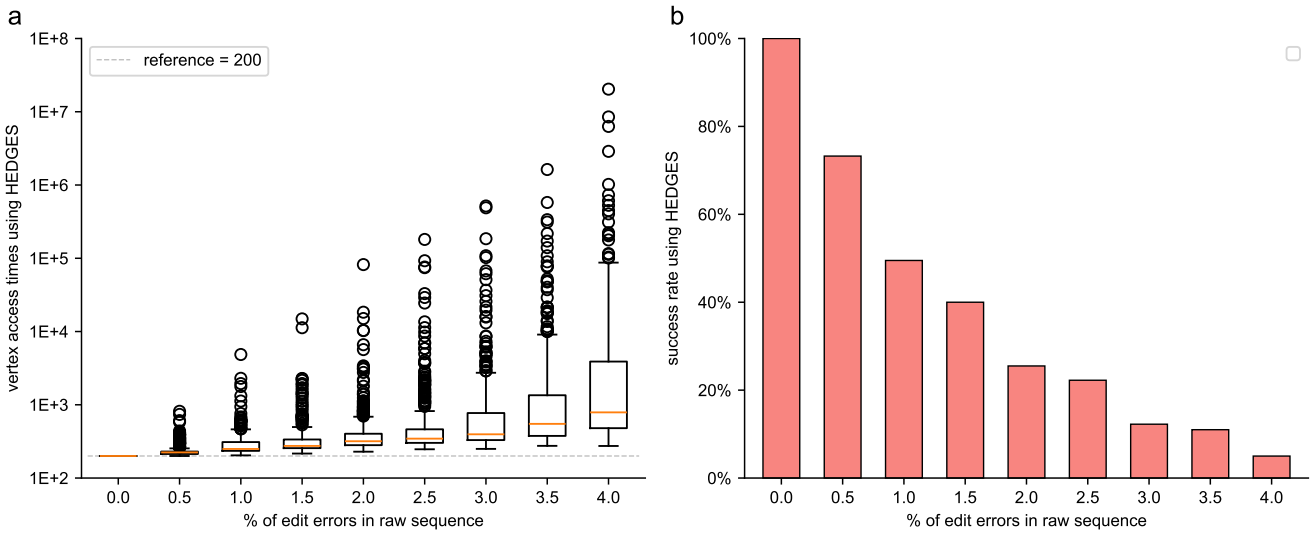

Fig. S5 | Error-correcting performance of HEDGES. Here, the imposed constraints are aligned with those used in the construction of the high-compatibility coding digraph. (a) Vertex access times under varying error rates. Since both the sequence content and the introduced errors are identical, the results can be directly compared to those in Fig. 2b. (b) Retrieval accuracy using HEDGES. Since HEDGES returns a single sequence, while SPIDER-WEB outputs a set of sequence candidates, a direct comparison is not applicable.

4.3 Improving data retrieval efficiency through tailored combinations of algorithmic modules

For this dimension, we selected Composite Hedges Nanopores (CHN)[3] as a representative baseline. CHN is directly relevant to this comparison because it targets rapid and portable DNA data readout on single-molecule nanopore sequencers, where high insertion and deletion rates remain a major obstacle to efficient retrieval. Rather than introducing a single standalone algorithm, CHN combines two previously developed concepts into a unified codec: the path-search-based error-correcting principle of HEDGES and the composite DNA letter representation for increased logical information density[23,26]. The resulting design is further adapted to nanopore-derived reads through downstream filtering, alignment, clustering, and assembly procedures. Thus, CHN provides an informative example in which mature algorithmic modules are strategically coordinated to facilitate rapid readout on a single-molecule nanopore sequencer.

For controlled and reproducible analysis, we restructured CHN in the code repository accompanying this study. The provided Python script, i.e., code_comparable.py, consolidates the computational procedures from the original code repository into a single, self-contained implementation. This evaluation-oriented consolidation addresses practical limitations encountered during direct benchmarking, such as undocumented behaviours, dispersed scripts, and loosely connected processing steps. The relevant modifications were made solely for evaluation purposes and should not be regarded as a fully equivalent reimplementation of the original CHN system. The restructured implementation was further checked through automated tests covering both individual modules and their multi-module integration. Communication with the authors of the corresponding publication[3] supported the appropriateness of using

the restructured implementation for rapid-readout evaluation.

We specified the implementation parameters and experimental settings used for the CHN-based evaluation. Following the original implementation, each 36-byte message segment was encoded into a 40-byte Reed-Solomon codeword with four parity symbols[25], and a 34-nt barcode was appended to the 5'-end as the address information for each degenerate sequence. For the CHN-specific parameters, we used the default settings adopted in the released workflow: the number of generated sequences per degenerate sequence (i.e., the "resolution" parameter) was set to 2, the modulus parameter for valid composite letters (i.e., the "sigma" parameter) was set to 8, and the number of bits per sliding-window step (i.e., the "step" parameter) was set to 2. Constraint screening was also kept consistent with the original design. Because our goal was to evaluate data retrieval efficiency across different file sizes and error rates, we tested input files from 1 KB to 16 KB, doubling the file size at each step (five settings). The simulated error rate was varied from 0.0% to 4.0% in increments of 0.5% (nine settings), consistent with the error-rate range used in the preceding experiments. For each file size, five random replicates were performed to assess stability, resulting in a total of 5×9×5=225 experiments.

During data preparation, we extensively debugged our CHN implementation and observed a non-negligible occurrence of encoding failures (Tab. S1). This phenomenon indicates that, even after excluding sequence errors introduced by wet-lab procedures, the data retrieval process may fail to stably recover the expected digital data. Given that unique decodability under an error-free setting is the prerequisite for any storage system, we analysed these failures. At the aggregate level, encoding failures were observed in 51.799% of the examined random cases under the tested settings. Further inspection showed that, apart from a limited number of cases in which encoding could not proceed because the imposed constraints left no valid degenerate nucleotide available, most failures arose because the predefined sequence length could not accommodate the information content of the binary message while maintaining unique decodability. Given that our objective was to evaluate the data retrieval efficiency of CHN under settings consistent with its original design, we did not increase the predefined sequence length, although doing so could reduce this issue. Furthermore, to remove potential evaluation bias introduced by encoding failures, the encoded random digital data used in the subsequent 225 retrieval experiments included only random binary messages that were both encodable and uniquely decodable.

Tab. S1 | The count of CHN encoding failures across random trials. Encoding failures arise from two sources. First, the imposed constraints may leave no valid degenerate nucleotide at a given sequence position, preventing further encoding of the binary message. Second, even after encoding is completed, the resulting degenerate sequence may either fail to recover the expected binary message under an error-free setting or yield multiple candidate binary messages. Since the original design resolves multiple-candidate cases by selecting the first entry in the retrieved message list, rather than ensuring a unique decoded message, we regard such cases as violations of the principle of unique decodability.

file size (kilobyte)	message number	failure count				
		trial 1	trial 2	trial 3	trial 4	trial 5
1	29	32	26	34	33	39
2	57	52	76	59	61	62
4	114	138	124	151	94	101
8	228	257	263	256	251	259
16	456	456	467	434	529	496

For each tested case in the CHN retrieval performance evaluation, the number of sequence copies supplied to the retrieval pipeline was increased incrementally from one. For each tested copy number, all distinct encoded sequences were supplied at the same copy number, so the retrieval results were not affected by the sequence-copy imbalance reported in the original study[3]. The incremental increase in sequence copy number was terminated when one of three conditions was met: the stored digital data were fully retrieved at the current copy number; the fraction of retrieved message segments remained unchanged after three consecutive one-copy increases; or increasing the copy number led to a lower fraction of retrieved message segments. These stopping criteria were introduced to avoid unexpectedly prolonged computation caused by individual cases, while still capturing the minimum sequence copy number required for successful retrieval whenever full retrieval was achievable under the tested setting. If a fully retrieved case was recorded when the search was terminated, the same case was subsequently processed by SPIDER-WEB and used for comparison. To keep the simulated error characteristics comparable, the sequences generated by SPIDER-WEB were constrained to have the same length as those generated by CHN. Considering that CHN was implemented in Python, SPIDER-WEB used its C++ implementation for the sequence-retrieval step and its Python implementation for the decoding step to make the comparison as fair as possible. To maintain a controlled computational comparison, the number of threads used by each processing step was fixed to one.

For each case in which full recovery was achieved, we recorded two measurements for both CHN and SPIDER-WEB: the minimum sequence copy number required for full recovery and the corresponding retrieval runtime. For a given file, we further defined the required total number of sequencing reads as the number of distinct encoded sequences multiplied by the required sequence copy number. This metric represents the upstream sequencing burden placed on the sequencing device, with larger values indicating that more sequencing reads are required before successful data recovery can be achieved. Using these measurements, we first analysed

CHN retrieval behaviour under the current controlled experimental setting and compared it with previously reported observations. We then compared CHN and SPIDER-WEB in terms of retrieval runtime and the required total number of sequencing reads under matched settings. In particular, we examined how the runtime difference between the two pipelines changed as the stored file size increased, thereby assessing their relative scaling behaviour.

We first examined whether the restructured CHN implementation produced retrieval behaviour broadly consistent with previously reported performance. As shown in Fig. S6a, all fully recovered cases achieved data retrieval efficiencies above the minimum value (0.5 bit/s) reported in the original publication[3]. The average retrieval efficiency was 8.2638 bit/s, close to the previously reported high-end value of approximately 9 bit/s. A subset of cases further exceeded the reported high-end value, which may partly reflect the limited statistical resolution of the originally reported performance, as well as evaluation-oriented optimizations introduced during restructuring, which reduced certain repeated computations without changing the core encoding and retrieval logic. Together, these results indicate that the restructured CHN implementation reasonably captures the expected performance range of the original design and can serve as a representative baseline for subsequent retrieval-efficiency comparisons.

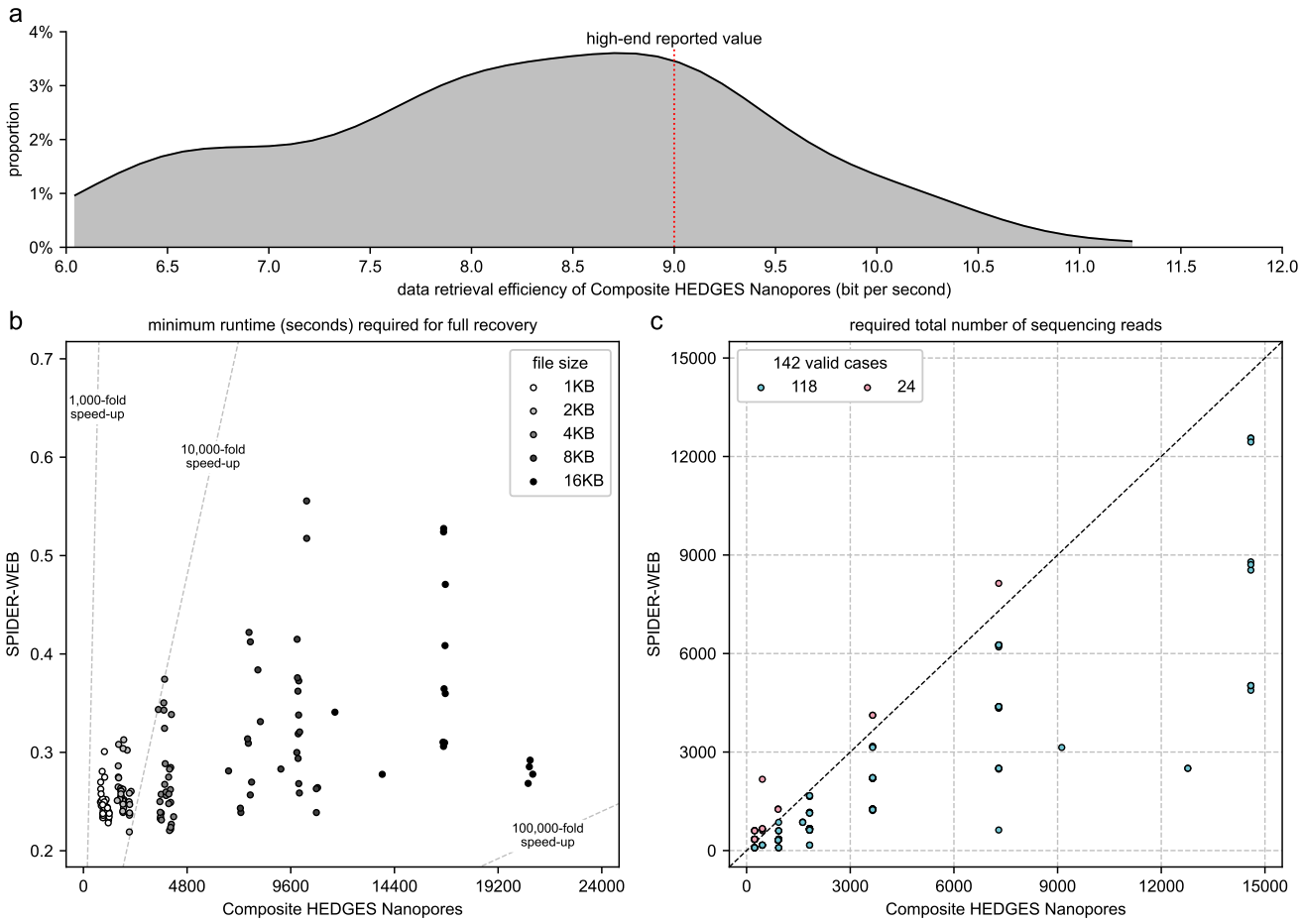

Fig. S6 | Differences in data retrieval between Composite HEDGES Nanopores and SPIDER-WEB. (a) Distribution of the data retrieval efficiency of Composite HEDGES Nanopores (CHN) across all cases in which full recovery was achieved. A total of 142 cases achieved full recovery before the stopping criteria were triggered. The red dashed line indicates the previously high-end reported value. (b) Comparison of the minimum runtime required for full recovery by SPIDER-WEB and CHN across randomly generated digital files of different sizes. Each point represents one fully recovered case, with point styles indicating file size. Dashed reference lines indicate fold differences in runtime between the two pipelines. (c) Comparison of the required total number of sequencing reads for full recovery by SPIDER-WEB and CHN for the same recovered cases. The required total number of sequencing reads was defined as the number of distinct encoded sequences multiplied by the minimum sequence-copy number required for full recovery. The dashed diagonal line indicates equal sequencing-read requirements between the two pipelines.

We then compared CHN and SPIDER-WEB in terms of retrieval runtime under matched settings. As shown in Fig. S6b, for the same randomly generated digital files, SPIDER-WEB consistently achieved more than three orders of magnitude higher data retrieval

efficiency than CHN. The maximum observed speed-up reached 76,706.7-fold across the fully recovered 16-kilobyte cases. This runtime advantage generally increased with stored file size, reflecting the different scaling behaviours of the two pipelines. This trend is consistent with the conclusion drawn from Fig. 4f. The magnitude of this speed-up was also affected by the simulated error rate: for the same stored file, higher error rates had a stronger effect on SPIDER-WEB retrieval than on CHN retrieval. Consequently, the runtime speed-up was moderately reduced under higher-error settings.

We also compared CHN and SPIDER-WEB in terms of the required total number of sequencing reads for full recovery. As shown in Fig. S6c, CHN required more sequencing reads than SPIDER-WEB in most matched cases, with 118 of the 142 fully recovered cases falling below the diagonal reference line. This indicates that, for the same stored digital file, SPIDER-WEB usually imposed a lower upstream sequencing burden than CHN. Because simulated error rate had a clear effect on SPIDER-WEB retrieval, the required number of sequencing reads for SPIDER-WEB increased under higher-error settings. The matched cases in which SPIDER-WEB required more reads than CHN were mainly concentrated at smaller file sizes.

Taken together, these comparisons indicate that SPIDER-WEB is better suited than CHN for real-time DNA data retrieval scenarios under the tested settings. CHN provides an informative rapid-readout baseline by combining HEDGES, composite DNA letters, and nanopore-oriented downstream processing, but its retrieval process still depends on progressively increasing sequence-copy input and exhibits substantial runtime overhead in matched full-recovery cases. By contrast, SPIDER-WEB achieved consistently higher retrieval efficiency, required fewer sequencing reads in most matched cases, and showed a scaling advantage as file size increased. From the encoding perspective, SPIDER-WEB also provides a more engineering-oriented design: its encoding process avoids the observed unique-decodability failures, supports direct sequence retrieval through structured indexing and local repair, and reduces dependence on downstream clustering or global reconstruction procedures. These properties make SPIDER-WEB more aligned with practical real-time retrieval, where both computational latency and upstream sequencing burden are critical constraints.

4.4 Improving data retrieval efficiency through alternatives to conventional pipelines

We considered the recently proposed LCRC-based accompanying indexing and data recovery framework[5], which is directly relevant to efficient data retrieval in DNA-based data storage. As clarified in Appendix 1, we restrict the comparison with LCRC here to a qualitative analysis of algorithmic design.

The LCRC-based framework addresses data retrieval efficiency by replacing a high-cost sequence-clustering step in the conventional recovery pipeline with structured indexing and staged read processing. During readout, low-error reads are rapidly identified through short-component-code correlation, followed by starting-site inference through the Chinese remainder theorem and an index double-check step. Reads with more severe insertions and deletions can then be processed by alignment to the LCRC, which serves as a hidden reference for read positioning and payload polishing. The stored data are finally reconstructed using consensus and error-correcting codes (ECC). In this design, an important efficiency gain arises from converting part of the unordered-read reconstruction task into a structured index-identification problem, thereby reducing the computational burden associated with sequence clustering.

Therefore, we treat LCRC as a conceptually related but non-equivalent effort rather than as a direct quantitative baseline. The major distinction is where the acceleration occurs in the recovery pipeline and whether usable output can be generated before downstream aggregation is completed. In LCRC, the acceleration mainly operates at the sequence-indexing and read-assignment level. The extent to which sequence clustering can be bypassed depends on whether the accompanying index can be reliably identified under the observed error profile; when correlation-based identification fails, reads still require alignment to the LCRC reference for positioning and polishing. Moreover, successful read assignment does not by itself complete sequence recovery, which remains dependent on sufficient coverage, consensus formation, and error-correcting codes. Hence, LCRC improves the conventional retrieval pipeline by reducing the cost of read assignment, but it does not fundamentally remove downstream dependencies that remain key bottlenecks for non-blocking retrieval.

SPIDER-WEB addresses a different and more stringent retrieval objective. Rather than accelerating only the assignment of reads to positions, SPIDER-WEB embeds sequence constraints, valid transitions, and error-tolerant neighbourhoods directly into the coding digraph. This enables noisy reads to be processed through graph-native local repair and candidate generation without sequence clustering, global alignment to a long reference, or coverage-complete consensus before producing useful intermediate output. In other words, SPIDER-WEB shifts correction and reconstruction from an aggregation-dependent stage to a read-level, graph-guided stage. Thus, for retrieval settings where latency, early output, and minimal blocking stages are central, SPIDER-WEB provides a more direct and less aggregation-dependent strategy than LCRC-based recovery, because candidate-level information can be generated at the read-processing stage rather than after coverage-driven consensus and ECC completion.

4.5 Conclusion

The comparative analyses clarify the design rationale of SPIDER-WEB at two levels. At the level of general coding principles, the comparison with DNA Fountain and HEDGES shows how SPIDER-WEB differs from representative approaches for constraint-compatible information encoding and error correcting. At the level of real-time retrieval-oriented systems, the comparison with CHN and LCRC-based recovery further clarifies where SPIDER-WEB is positioned among recent attempts to reduce retrieval latency and downstream reconstruction costs.

At the level of general coding principles, SPIDER-WEB demonstrates the advantage of a unified coding solution. Compared with DNA Fountain, SPIDER-WEB does not rely on screening-based sampling to obtain sequences satisfying biochemical constraints. Instead, biochemical constraints are embedded directly into the coding digraph, so constraint-compatible paths can be generated by construction rather than obtained through repeated rejection. Compared with HEDGES, SPIDER-WEB does not treat error correction as a separate layer that must be supplemented by external error-correcting codes. Its graph-based correcting strategy allows noisy reads to be mapped back to plausible coding paths, thereby integrating error tolerance within the same algorithmic structure. These features explain why SPIDER-WEB can balance sequence compatibility, information density, and error tolerance within a unified solution rather than optimizing these factors through separate algorithmic modules.

The comparison with real-time retrieval-oriented efforts highlights a complementary distinction. CHN demonstrates how existing algorithmic modules can be combined for rapid nanopore-based readout, but under the matched settings examined here it still showed substantial runtime overhead and required more sequencing reads than SPIDER-WEB in most fully recovered cases. LCRC-based recovery and related indexing-based methods reduce the cost of conventional sequence clustering, but complete recovery still depends on downstream coverage, consensus formation, and error-correcting codes. By embedding valid transitions, sequence constraints, and error-tolerant neighbourhoods directly into the coding digraph, SPIDER-WEB supports read-level local repair and candidate generation before coverage-complete consensus or global reconstruction is achieved. This makes SPIDER-WEB more aligned with practical real-time retrieval, where low latency, reduced sequencing burden, and non-blocking access to usable intermediate output are central requirements. Although we did not include a quantitative comparison with machine-learning-based approaches such as DNAformer[4], such methods can be interpreted as part of the same broader effort to move beyond conventional sequence-clustering-heavy recovery pipelines. In this sense, DNAformer is conceptually related to LCRC-based recovery: both aim to reduce or replace sequence clustering as a major bottleneck in data recovery. The difference is that DNAformer further introduces machine-learning-based inference and depends on GPU acceleration.

Overall, SPIDER-WEB is distinguished by its graph-based integration of encoding, error correction, and direct data retrieval. This integration makes it a more engineering-oriented framework for DNA-based data storage, particularly in settings where real-time retrieval and low computational latency are critical. Rather than improving a single isolated stage, SPIDER-WEB coordinates encoding and retrieval so that stored information can be accessed more directly and with fewer downstream blocking dependencies.

5 Construction details of the high-compatibility coding digraph and its components

5.1 Constraint setting

The high-compatibility coding digraph was built under a set of regionalised constraints (Appendix 2) with an observation window of length 10. To reduce the risk of errors during the DNA synthesis and DNA sequencing workflows[27,28], the maximum homopolymer run-length was limited to 2. To improve PCR amplification efficiency[29], the regionalised GC content was fixed at 50%. Moreover, platform-specific error-prone motifs – including "GGC" and "GGT" in high-throughput sequencing[30,31], and "AG", "GA", "CT", "TC" in single-molecule sequencing[32] – were explicitly excluded during digraph construction. To ensure practical decoding performance, this digraph was generated in fast mode (Appendix 3). Beyond the regionalised constraints, we activated two global constraint components: the fragment repetition constraint and the reverse complement constraint, both configured with a threshold of $k = 10$ (Appendix 2). This coding digraph and its global-constraint-augmented variant serve for simulation analysis and experiment validation.

5.2 Record of the digraph construction process

The construction of the high-compatibility coding digraph involves a series of constraint-driven screening and graph structural optimisation steps. We begin with a full enumeration of all possible 10-mers (4^{10}; 1,048,576). These were then screened through a sequence of the given regionalised constraints:

1. Homopolymer run-length constraint: Removed 371,740 10-mers.
2. regionalised GC content constraint: Removed an additional 187,300 10-mers.
3. Undesired motif constraint: Excluded 482,714 10-mers based on sequencing platform-specific error-prone motifs.

After these three steps, 6,822 valid 10-mers remained, which were then used to construct the initial vertices of a digraph.

In the arc trimming process, additional constraints were applied to prune structurally incompatible vertices. Specifically, 1,404 vertices were removed because they either lacked a valid out-degree or in-degree, or formed isolated cycles. The resulting intermediate digraph contained 5,418 vertices, with the following out-degree distribution:

- 1,454 vertices with out-degree 1;
- 2,628 vertices with out-degree 2;
- 1,336 vertices with out-degree 3.

To enable the fast mode, all vertices with out-degree 3 were adjusted to behave as out-degree-2 vertices during the vertex traversal. And to ensure that every selected vertex has at least one incoming arc (thus minimising decoding performance asymmetry across virtual vertices), an additional 1,120 vertices were removed. The final high-compatibility coding digraph thus contained 4,298 vertices (1,332 with out-degree 1 and 2,966 with out-degree 2). The entire generation process required 10.127 seconds to complete.

5.3 Record of the associated component construction process

In addition to digraph construction, several additional components were instantiated to support global constraint screening (for the message encoding process and the sequence generating process), and the fallback strategy during the message encoding process (used when message encoding becomes infeasible due to global constraints).

- The non-palindrome mode component checks whether the reverse complement of a given vertex is also present in the digraph. To accelerate this operation, each 10-mer is encoded as a base-4 vector (A = 0, C = 1, G = 2, T = 3), forming a matrix of shape (4298×10). The reverse complement is computed by subtracting from 3 and reversing along the row axis. Hamming distance is then used to determine palindromic relationships. This operation took 0.0221 seconds to complete.
- The fragment repetition constraint requires no additional component. It is implemented at runtime by checking whether a vertex has been revisited. If so, the corresponding outgoing arc is temporarily masked.

6 Mathematical rationality of the new data retrieval pipeline

Here, we provide a concise mathematical overview of data retrieval pipelines, covering both conventional approaches in DNA-based data storage and our novel approach in SPIDER-WEB.

6.1 Conventional approaches

Assume the data has been synthesised into M *information strings*, where each string is replicated into multiple copies. By DNA sequencing, we obtain a list of strings, referred to as the *reads*. In the conventional approach, the first step is to cluster these reads. Using any standard clustering algorithm, the reads are divided into several clusters, where the reads in the same cluster are very likely to be generated from a common information string.

Suppose in one cluster there are N distinct reads, denoted as $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$. Retrieving the original information string $\boldsymbol{x}$ based on these reads is a well-known research area in the coding theory community, known as the *sequence reconstruction problem*, first introduced by Levenshtein[20]. There is also a closely related topic known as the *trace reconstruction problem*. The main difference is that the sequence reconstruction problem considers a "combinatorial channel" where there is a maximum bound on the number of errors, whereas the trace reconstruction problem considers a "probabilistic channel" where every symbol in a read could be independently erroneous with a given probability distribution function.

The formal setting of the sequence reconstruction problem is as follows. For a given alphabet set Σ, consider a code $C \subseteq \Sigma^n$, which is a subset of all the length-n strings over the alphabet set Σ. Suppose a codeword $\boldsymbol{x} \in C$ is sent through several identical channels. The receiver collects N distinct erroneous reads $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$ and wants to reconstruct $\boldsymbol{x}$ based on these reads, the knowledge of the code C, and the property of the channel. Let $B(\boldsymbol{x})$ be the set of all possible outputs when $\boldsymbol{x}$ is sent through the channel, known as the *error ball* centred at $\boldsymbol{x}$. Levenshtein[20] first showed that the minimum number of reads to guarantee unique reconstruction of $\boldsymbol{x}$ is $N+1$, where

$$N = \max_{\boldsymbol{x}\neq\boldsymbol{y},\boldsymbol{x},\boldsymbol{y}\in C} |B(\boldsymbol{x}) \cap B(\boldsymbol{y})|.$$

Here, the value N is the largest size of the intersection of two error balls centred at distinct codewords in C, usually referred to as the *unique reconstruction threshold*.

In the coding theory community, there are two main research directions regarding the sequence reconstruction problem:

1) **The classic problem**: Calculate the unique reconstruction threshold for a given code (mostly for $C = \Sigma^n$ or C is a code with a preset minimum distance) and a given channel (with a given type of error and the maximum number of errors), and then design the corresponding efficient reconstruction algorithms when the number of reads exceeds the unique reconstruction threshold.

2) **The reverse problem**: Given a channel and the threshold value N, design a *reconstruction code* C which can guarantee unique reconstruction with any $N+1$ distinct reads, i.e., a code for which $\max_{\boldsymbol{x}\neq\boldsymbol{y},\boldsymbol{x},\boldsymbol{y}\in C} |B(\boldsymbol{x}) \cap B(\boldsymbol{y})| \leq N$. The redundancy of the code, usually defined by $n - \log|C|$, is desired to be as small as possible.

These theoretical problems are quite non-trivial and challenging. For example, regarding the classic problem for the edit error channel (a combination of insertion, deletion, and substitution errors), which is very close to the DNA-based data storage setting, so far, the results on the reconstruction threshold are very rare. The most recent known result analyses the case of single-substitution-single-insertion and single-substitution-single-deletion[33]. The reverse problem is even more difficult since analysing the optimal redundancy of the reconstruction code requires advanced tools from extremal combinatorics or graph theory. It is proven that in the single-deletion channel $\log\log n + O(1)$ bits of redundancy are necessary if we want unique reconstruction by only two reads[34].

Note that all these problems following the argument of Levenshtein's reconstruction threshold are indeed a worst-case analysis for the sequence reconstruction problem. If we consider the average case instead, then with high probability, one can expect unique reconstruction even when the number of reads is far less than the threshold. This is the reason why in most known experiments of DNA-based data storage[2,9,21,23], the *coverage depth* does not have to be as large as the reconstruction threshold. However, the theoretical foundations for the average case analysis are quite limited. For this direction, there are two related recent papers: Wang et al.[35] proposed a new sufficient condition for unique reconstruction that accounts for both the number of reads and their pairwise distances; Papadopoulou et al.[36] analysed the probability of correct reconstruction under a majority-vote decoder when the number of reads is below the threshold.

However, most of the theoretical contributions from the coding theory community have not been incorporated into current DNA-based data storage techniques. One of the main reasons is that, except for the Varshamov-Tenegolts code[37] correcting one deletion error or one insertion error, the other known codes for deletion/insertion errors are not yet very applicable. For example, the best-known theoretical two-deletion correcting codes from Guruswami, et al.[38] have redundancy $4\log n$ when the codeword length n is sufficiently large. However, in the current DNA-based data storage framework, a string typically contains 200-300 nucleotides, and for such a short codeword length, the codes do not perform well. Therefore, in most known DNA-based data storage experiments, while

various types of error-correcting codes have been used as "outer codes" to encode the information strings, the "inner codes" inside a string itself are seldom used. As a consequence, for sequence reconstruction, the most commonly used strategy is the straightforward symbol-wise majority-vote decoder, i.e., multiple sequence alignment.

In this work, since we are considering constrained systems with multiple constraints, we will have more clues for the reconstruction of an information string, and thus, it is expected to have a better strategy than the majority-vote decoder. Thus, we develop a new approach in SPIDER-WEB as follows.

6.2 Our new approach in SPIDER-WEB

There are at least three motivations for designing a new data retrieval pipeline for constrained systems. First, we want to get rid of the clustering process in order to improve efficiency. Second, the strings we synthesize are characterised by a constrained system, and the theoretical analysis for the reconstruction threshold of a given constrained system is almost blank, even in the coding theory community. Third, and most importantly, we have additional clues due to the constraints when reconstructing each string, and thus we can apply this side information and have a strategy better than the symbol-wise majority-vote decoder.

Our new approach in SPIDER-WEB works as follows.

- Step 1. Individually decode each read based on the coding digraph of the constrained system. A proper constrained string could be mapped into a directed path in this coding digraph. Suppose we find that a read violates the digraph at a certain coordinate i, then we know for sure that there must be some error within the range $i-k+1$ to i (here k refers to the length of the string represented by each vertex in the digraph), and then we run a local brute-force search for such an error. The correction is by analysing how to modify the string in this local interval such that it can be modified into a directed path on the digraph. Note that there might be multiple possibilities for correcting one string, and we will keep all the possible candidates within a list.

- Step 2. After the decoding for all reads, combine all the candidates and then rank all these possible candidates based on the number of their appearances. Pick the M strings that appear the most times as the original M information strings.

We explain why this approach works with high probability, and analyse the situations in which the new approach might fail.

First, we analyse how Step 1 works. If there is no error in a read, then in this ideal case, the list simply contains the correct string itself. If there is only one error, or there are multiple errors but they are adequately separated (two neighbouring errors are of distance at least k), then the local brute-force search works and the list will contain the correct string plus some other possible candidates since the decoding possibility might not be unique[2,9,21,23]. Thus, the correct string $\boldsymbol{x}$ will appear in almost every candidate set when we decode a read $\boldsymbol{x}'$ originated from $\boldsymbol{x}$. The exceptions include mainly two cases. One case is that the erroneous read happens to be another proper string satisfying all the constraints, and in this case the list only contains the incorrect string. Yet the probability of such an event is very small, especially when multiple constraints have made the coding digraph quite sparse. The other case is that there exist multiple errors concentratively distributed within a small interval, which may lead to failures of local brute-force search. Again, the probability of such an event is very small and we can also just discard this read once we notice a local brute-force search failure.

Second, we discuss the selection criterion in Step 2. If there is only one information string, since this correct string will appear in most lists, it is natural to pick the one which appears the most times as the decoding output. When there are multiple information strings, a potential failure might arise from the following situation: the sequencing procedure has captured a large amount of reads which are generated by an information string $\boldsymbol{x}$, but only very few copies of reads which are generated by the other information string $\boldsymbol{y}$. The correct $\boldsymbol{x}$ will certainly rank high in the combination of all lists, but it could happen that some incorrect $\boldsymbol{x}'$, appearing in the list when decoding a certain fraction of reads related to $\boldsymbol{x}$, might rank higher than the other correct information string $\boldsymbol{y}$. We claim that while this is possible, the probability of such an event is still very small (Fig. 3). The mathematical explanation is that the value $|B(\boldsymbol{x}) \cap B(\boldsymbol{x}')|$, where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are two distinct codewords, is way less than the size of an individual error ball. Thus, to obtain many reads generated from $\boldsymbol{x}$ such that their decoding list also contains $\boldsymbol{x}'$ only occurs with a small probability. Moreover, such a failure could be avoided if we could forbid the extreme imbalance in the number of reads for all information strings.

The reasons above are mainly intuitive. Yet, our experiments and simulations have shown the superiority of our new approach. In particular, skipping the clustering process makes our approach way more efficient than the conventional approach.

7 Primary factors influencing direct retrieval rate

In developing the SPIDER-WEB data retrieval pipeline, we adopt the working hypothesis that the top m most frequently occurring sequence candidates represent the intended m direct retrieval targets. This hypothesis is supported by both empirical observations (Fig. 2e) and mathematical reasoning (Appendix 6). However, for practical deployment, particularly for applications scaling to terabyte data volumes and beyond, it is not sufficient to rely solely on theoretical plausibility. Given a target sequence diversity d and an error rate e, it is necessary to quantitatively understand how many sequence copies c are needed to ensure that the frequency ranking yields accurate recovery of all intended sequences. Addressing this question provides not only validation of the pipeline's underlying hypothesis, but also concrete empirical values that can serve as predictive guidelines when planning retrieval experiments at a large scale. The following analysis systematically explores how the required number of sequence copies scales with both error rate and sequence diversity, thereby translating a general statistical principle into actionable benchmarks for DNA-based data retrieval.

7.1 Definition recap

As introduced in the Main Text, the direct retrieval rate is measured as the proportion of intended sequences within a specified number of retrieved candidates. In the standard setting, the number of retrieved sequences is chosen to match the intended diversity, so that a rate of 1.0 corresponds to complete recovery of the intended set. Building on this definition, here we revisit the conditions under which the frequency-ranking approach achieves high direct retrieval rates. Specifically, we examine how the required number of sequence copies C depends on both the error rate e and the sequence diversity D. By quantifying these dependencies, we provide empirical support for the scaling relation

$$C \sim (\log D)^e$$

described in Methods, and translate it into practical benchmarks for large-scale data retrieval.

7.2 Effect of sequence diversity

We first inspected whether the direct retrieval rate depends on the sequence diversity. Across a broad range of d values, from 10^1 to 10^8, the simulation results reveal that curves corresponding to different diversities nearly overlap (Fig. S7). This indicates that the required number of sequence copies C is largely insensitive to sequence diversity under fixed error rates.

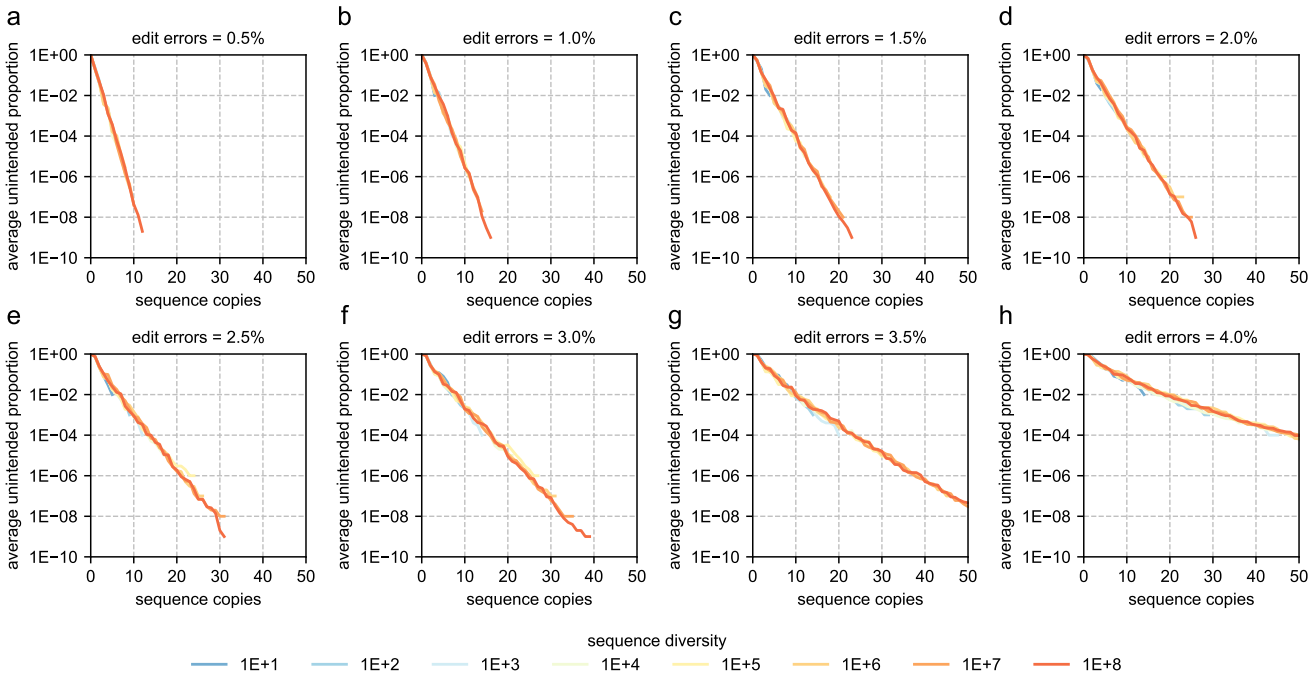

Fig. S7 | Effects of sequence diversity and error rate on the average unintended proportion across different sequence copies, grouped by sequence diversity. (a) - (h) correspond to sequence diversities ranging from 10^1 to 10^8.

However, to achieve lossless data retrieval at a given diversity, the average unintended proportion must fall below the threshold of $1 / D$, ensuring that unintended sequences are statistically excluded from the retrieval set. Under this stricter criterion, a residual

dependence on D becomes apparent. Intuitively, the required copies increase in a logarithmic-like fashion with D: expanding diversity by several orders of magnitude only marginally raises the sequence copy requirement, yet the trend reflects the need to suppress increasingly rare but statistically possible unintended candidates.

7.3 Effect of error rate

The influence of error rate is clearly observed when sequence diversity is fixed, and the edit error rate is systematically varied (Fig. S8). Across all tested diversities, increasing the error rate from 0.5% to 4.0% leads to a consistent upward shift of the retrieval curves, such that more sequence copies are required to reach the same unintended-proportion threshold. For example, under $D = 10^5$, raising the error rate from 1.0% to 2.0% roughly doubles the copy requirement needed for lossless retrieval.

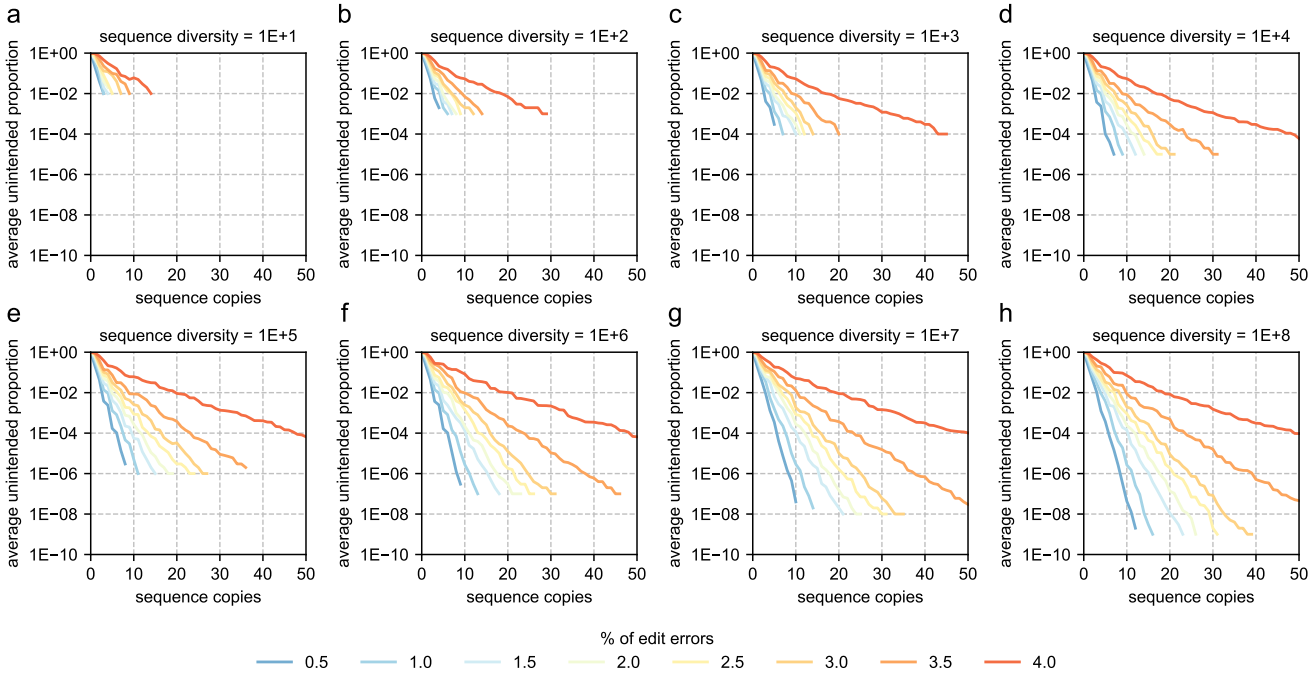

Fig. S8 | Effects of sequence diversity and error rate on the average unintended proportion across different sequence copies, grouped by sequence diversity. (a) - (h) correspond to sequence diversities ranging from 10^1 to 10^8.

Unlike sequence diversity, which contributes only a slow logarithmic-like correction, error rate directly dictates the vertical scaling of the curves. This reflects the combinatorial growth of unintended variants generated by edit errors, which accumulate multiplicatively as the error rate increases. As a result, retrieval performance is far more sensitive to error rate than to diversity, underscoring the importance of error suppression strategies in practical implementations of large-scale data retrieval.

7.4 Empirical scaling relation

The simulation results can be rationalised by considering the competition between intended sequences and unintended variants in the frequency ranking. For a sequence set with sequence diversity D and sequence copy number C, each intended sequence is expected to appear with frequency proportional to $C \,/\, D$. In contrast, unintended variants arise stochastically from edit errors. While most variants occur only once or a few times, the most frequent unintended variant sets the effective threshold for lossless data retrieval.

Extreme-value theory[39] predicts that the maximum of a large set of random frequencies increases approximately as $\log D$ for distributions with exponential-type tails. This behaviour accounts for the residual logarithmic-like dependence on diversity observed in the simulations: as D grows, the chance that at least one unintended variant attains deceptively high counts rises slowly with $\log D$. In contrast, the effect of error rate is multiplicative. Each edit error can occur at multiple positions and generate multiple unintended variants, and when errors accumulate their combined impact compounds. Accordingly, the fitted scaling $C \sim (\log D)^e$ emerges, with e increasing monotonically as the error rate rises. In this interpretation, the logarithmic term reflects the extreme-value behaviour associated with sequence diversity, while the exponent captures the combinatorial expansion of unintended variants caused by errors.

7.5 Empirical parametrisation of required sequence copies

To obtain a quantitative form for predicting the required sequence copies, we parametrised the simulation results using a two-parameter model. Specifically, for a given error rate, the dependence of sequence copy number on the error rate is expressed as

$$C = f(p;\ \theta_1, \theta_2) = \left(\frac{-\log_{10} p}{\theta_1}\right)^{1/\theta_2}$$

where p denotes the unintended proportion (equivalently, the file-scale error rate), θ_1 and θ_2 are fitted constants. In the simulations, p was set to $1\ /\ D$, so that recovery is considered lossless once the average unintended proportion falls below this threshold. It is important to note that, in this parametrisation, C does not represent the required sequence copies in an absolute sense, but rather the copy number that satisfies a chosen unintended-rate threshold.

Tab. S2 | The fitted constant θ from non-linear least-squares fitting of the empirical scaling law. It is obtained using The `curve_fit` function of SciPy package[40] with default method (i.e., Levenberg-Marquardt[41,42]).

error rate	0.5%	1.0%	1.5%	2.0%	2.5%	3.0%	3.5%	4.0%
fitted constant θ_1	0.6788	0.5030	0.4784	0.4173	0.3654	0.2993	0.3059	0.2291
fitted constant θ_2	1.0274	1.0358	0.9331	0.9271	0.9145	0.9333	0.8171	0.7399
coefficient of determination	0.9943	0.9951	0.9962	0.9955	0.9940	0.9956	0.9970	0.9919
root mean square error	0.2169	0.2748	0.3540	0.4606	0.6413	0.6525	0.7364	1.2808

The fitted constants reveal a consistent trend across error rates. Both θ_1 and θ_2 decrease as the edit error rate increases, indicating that higher error conditions reduce the effective scaling strength of $-\log_{10} p$. Since θ_1 governs the horizontal shift of the scaling curve and θ_2 controls its curvature, reductions in these values effectively flatten the response, meaning that larger copy numbers are required to achieve the same unintended-proportion threshold. This behaviour reflects the combinatorial growth of unintended variants under increasing error rates: as more variants accumulate, the model compensates by demanding higher sequencing depth to suppress them. Despite this systematic shift in parameter values, the empirical model achieves excellent fits across all error rates, with the coefficient of determination $R^2 > 0.99$ and root mean square errors close to unity or lower, confirming that the parametrised scaling law robustly captures the combined influence of error rate and sequence diversity.

In practical use, ensuring that $p < 1\ /\ D$ provides an operational criterion for obtaining the required C under a given error rate. Once the average error rate e_a of a sequence collection is obtained, the required sequence copies can be estimated as

$$\left\lceil f(\frac{1}{D};\ \theta_1, \theta_2) \right\rceil + \epsilon,$$

where $\epsilon \geq 0$ provides flexibility for further adjustment, and the θ_1, θ_2-pair is selected according to e_a from Tab. S2, using the next higher listed error rate when e_a falls between two tabulated values. This operational rule translates the fitted parameters into actionable copy-number estimates. The resulting estimates were subsequently used to construct Fig. 2f. This empirical model provides a practical bridge between simulation-based intuition and experimental planning, enabling reliable prediction of required sequencing depth for large-scale data retrieval.

8 Prevention and mitigation strategies for sequence loss

8.1 Sources of sequence loss

Sequence loss may arise from multiple causes across synthesis, sequencing, and handling workflows[43–45]. Inadvertent losses are commonly associated with synthesis failures, incomplete oligonucleotide assembly, or chemical defects, as well as insufficient sequencing depth or systematic read dropout. Additional factors include physical mishandling during sample transfer or purification, contamination, preferential amplification during polymerase chain reaction (PCR), and long-term chemical degradation.

Notably, sequence loss can also be deliberately introduced to fulfil specific experimental objectives. For example, serial dilution experiments[2,21] progressively reduce the copy number of DNA molecules to explore the achievable physical density of the proposed advances, while accelerated ageing experiments[9,46] subject samples to stress conditions to estimate their maximum preservation time under ideal storage environments. In these designs, irreversible information loss is intentionally used as a threshold or endpoint indicator. Such experimental objectives, however, are unrelated to standard data retrieval workflows and should not be conflated with the operational failure modes that affect practical system performance.

8.2 Prevention strategies

Although sequence loss may affect the data retrieval performance, it is not inherently unavoidable. A combination of design strategies, workflow controls, and analytical safeguards can substantially reduce its likelihood and mitigate its impact before the retrieval stage.

First, from the sequence design perspective, avoiding certain high-risk sequence patterns can reduce synthesis-stage failures and amplification bias[2,21,23]. For example, minimising homopolymer runs and avoiding extreme GC content have been shown to improve synthesis yield and PCR uniformity, thereby lowering the risk of sequences being lost prior to retrieval.

Second, standard wet-lab workflows already incorporate multiple quality control steps that allow sequence loss to be detected before the data retrieval stage. Such quality control checkpoints[47], including intermediate yield measurements and sequencing-based validation, enable the early identification of missing sequences. When detected, corrective actions such as re-synthesis or targeted re-amplification can be initiated, preventing such losses from propagating downstream.

Third, a systematic assessment of the relationship between sequencing depth and the diversity of distinct sequences can avoid misinterpreting insufficient sequencing coverage as genuine sequence loss. For instance, in our work, we demonstrated that the expected retrieval completeness can be quantitatively predicted given a known library complexity and sequencing throughput (Fig. 2f), allowing experimental designs to ensure adequate coverage and reduce false positives in loss detection.

8.3 Mitigation strategies

From a results-oriented perspective, sequence loss in DNA-based data storage can manifest in three principal forms. The first arises when a given sequence exhibits an observed error level exceeding its error-correction capability, leading to its classification as uncorrectable and subsequent discarding during retrieval. The second is encountered when a sequence is filtered out at the sequencing stage due to failing other quality criteria, such as low basecalling quality scores, regardless of its actual error count. The third is observed when a sequence is absent from the readout, representing actual physical loss.

For the first and second cases, enhancing the intrinsic error tolerance of each sequence can remarkably reduce the probability of it becoming unprocessable or allow sequences with lower basecalling quality scores[48,49] to be still used in subsequent data retrieval. This includes bit-level error tolerance (e.g., Reed-Solomon code) and sequence-level error tolerance (e.g., our study). For the third case, the well-accepted method is to introduce outer codes, which add logical redundancy across multiple DNA sequences[8] or binary messages[2,9,50] so that the entire dataset can be reconstructed without recovering every individual DNA sequence.

In addition, given that particular experimental objectives or immature laboratory techniques may lead to uncontrollable errors and loss rates, a range of post-retrieval recovery strategies has also emerged. These include both domain knowledge-driven methods and AI-based inference techniques capable of repairing missing or damaged content at the application layer, for example, reconstructing text via spellchecker engine or large language models[51,52], and restoring corrupted images through classical image reconstructing methods or machine learning models[53–57].

8.4 SPIDER-WEB with outer codes

A key technical contribution of SPIDER-WEB lies in integrating and optimising multiple coding strategies within a unified framework for DNA-based data storage. Given that outer codes are widely adopted in the broader coding domain, evaluating their interaction with SPIDER-WEB is both natural and necessary.

8.4.1 Compatibility between SPIDER-WEB and outer codes

In typical designs of DNA-based data storage, an outer code is sometimes applied, adding a prescribed amount of logical redundancy across payloads. At the same time, the address portion of the original binary messages is excluded from the redundancy construction. As illustrated in Fig. S9a, we construct a modular integration scheme between SPIDER-WEB and outer codes, showing the independence of the outer and inner codes. For SPIDER-WEB, the presence or absence of redundancy introduced by an outer code in the

bit sequences during the encoding process is immaterial based on such process; its decoding process likewise remains unaffected. For certain specialised outer codes, such as fountain codes, the address serves a different role: it encodes the seed required to reconstruct the corresponding droplet, which is generated by applying the bit-wise exclusive-or operation to multiple original binary messages. This distinction does not alter the modular integration principle, but it does mean that the interpretation of the address field can differ from that in more conventional outer codes.

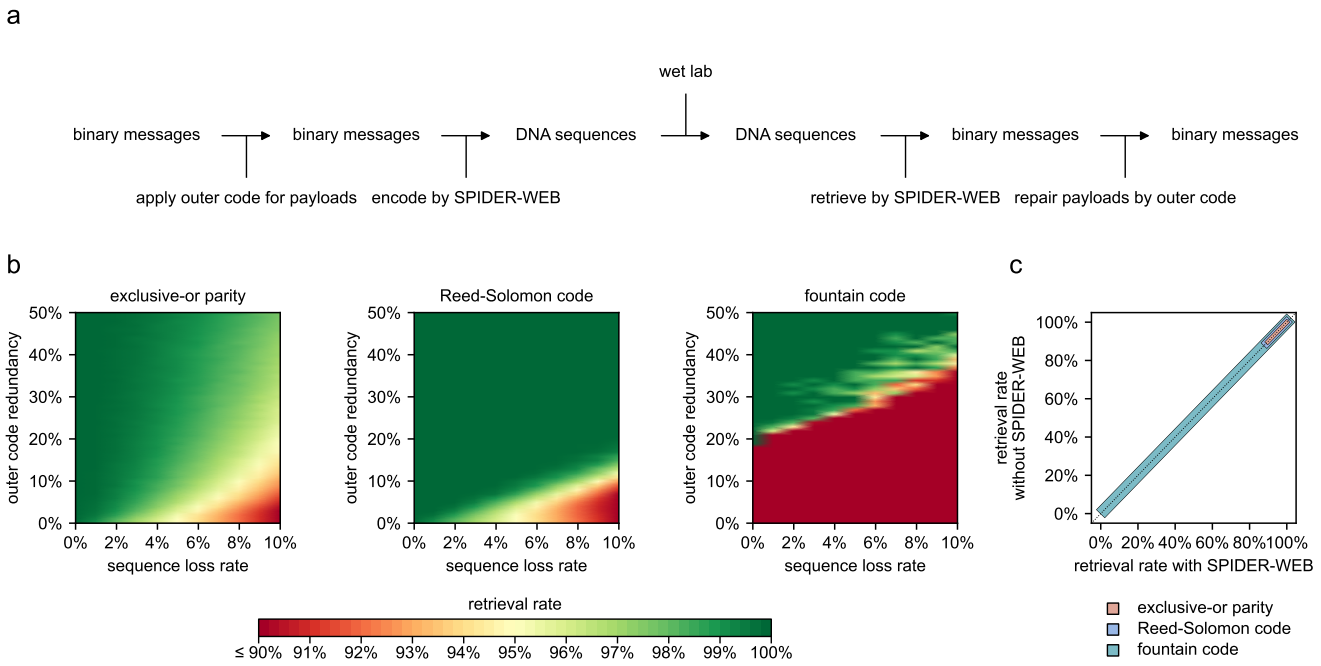

Fig. S9 | **Inner-outer code integration and its loss tolerance.** (a) Independent combination of outer code and SPIDER-WEB (as inner code). (b) Recovery performance of three commonly used outer codes in DNA-based data storage under varying logical redundancy and sequence loss rates. Sequence loss was simulated at the bit level to ensure independence from any inner code. (c) Equivalence between pure outer code and "SPIDER-WEB + outer code" in addressing sequence loss.

To provide a baseline for subsequent joint analysis with SPIDER-WEB, we evaluated the loss tolerance of exclusive-or parity[16,50,58], Reed-Solomon code[9,23,57,59], and fountain code[2,26,60,61] in the absence of any inner-code participation. Detailed implementation can be found in "code_correcting.py" in the "works" folder of the provided GitHub repository. Using these outer codes, we generated 100 datasets, each consisting of 1,000 binary messages of length 80 bits. Logical redundancy is introduced at levels ranging from 1% to 50% (in 1% increments). For each redundancy level, sequence loss rates are set from 0% to 10% (in 1% increments), and the number of recoverable binary messages is recorded. This design yielded a total of 150,000 experimental configurations.

Fig. S9b summarises the measured performance of three well-established outer codes under varying levels of logical redundancy and sequence loss. For exclusive-or parity, the ideal case is straightforward: introducing a given percentage of logical redundancy should, in principle, enable perfect recovery under the same percentage of sequence loss. This follows from its construction, where an additional binary message is generated as the bitwise exclusive-or of n original messages; loss of any single one among these $n+1$ messages can be perfectly repaired. Nevertheless, when two or more messages from the same parity group are lost, the lossless recovery of this parity group becomes impossible. Consequently, achieving reliable recovery at a given loss rate typically requires redundancy exceeding the loss fraction, and even then, perfect recovery cannot be guaranteed. In contrast, Reed-Solomon codes, which operate over Galois fields to correct erasures, exhibit experimental performance closely matching their theoretical limits. The behaviour of fountain codes differs remarkably from the above. Due to their rateless droplet-based construction with performance determined by the degree distribution, a baseline redundancy of approximately 20% is required to achieve successful decoding even in the absence of sequence loss. Beyond this threshold, only a modest additional redundancy is needed to sustain high recovery probabilities, even at loss rates up to 10%. While fountain codes require higher redundancy than the other two codes at low loss rates, their capacity to adapt flexibly when the loss rate is unknown can offer a practical advantage in certain scenarios.

When SPIDER-WEB was incorporated as the inner code, we repeated the complete set of 150,000 experiments under identical conditions mentioned above. As shown in Fig. S9c, the results are indistinguishable from those obtained with the outer codes alone. This confirms that SPIDER-WEB is independent to any outer code: its operation does not alter outer-code performance, and the

presence of an outer code does not interfere with the normal encoding or decoding processes of SPIDER-WEB.

8.4.2 Impact on data retrieval efficiency

Given that the central performance metric of this work is data retrieval efficiency, we investigated the potential influence of incorporating outer codes on this metric.

The basic comparison scheme is illustrated in Fig. S10a. Under identical configuration settings, we compared the efficiency of retrieving 10 datasets, each consisting of 1,000 binary messages of length 80 bits, with and without outer codes, at different error rates (from 0.5% to 4.0%, as mentioned in the Main Text). This design yielded a total of 679,500 experimental configurations.

In theory, the sequencing depth required to achieve lossless retrieval may differ slightly between these two cases because the error rate can affect the sequence ranking stage. In contrast, the use of an outer code does not require the retrieval of every individual sequence to reconstruct the dataset. Let n denote the sequencing depth required without an outer code, with a corresponding runtime t_n, and let m denote the sequencing depth required with an outer code, with a corresponding runtime t_m. We quantify the effect of outer codes on efficiency using the normalised runtime growth rate defined as $t_m \ / \ t_n$.

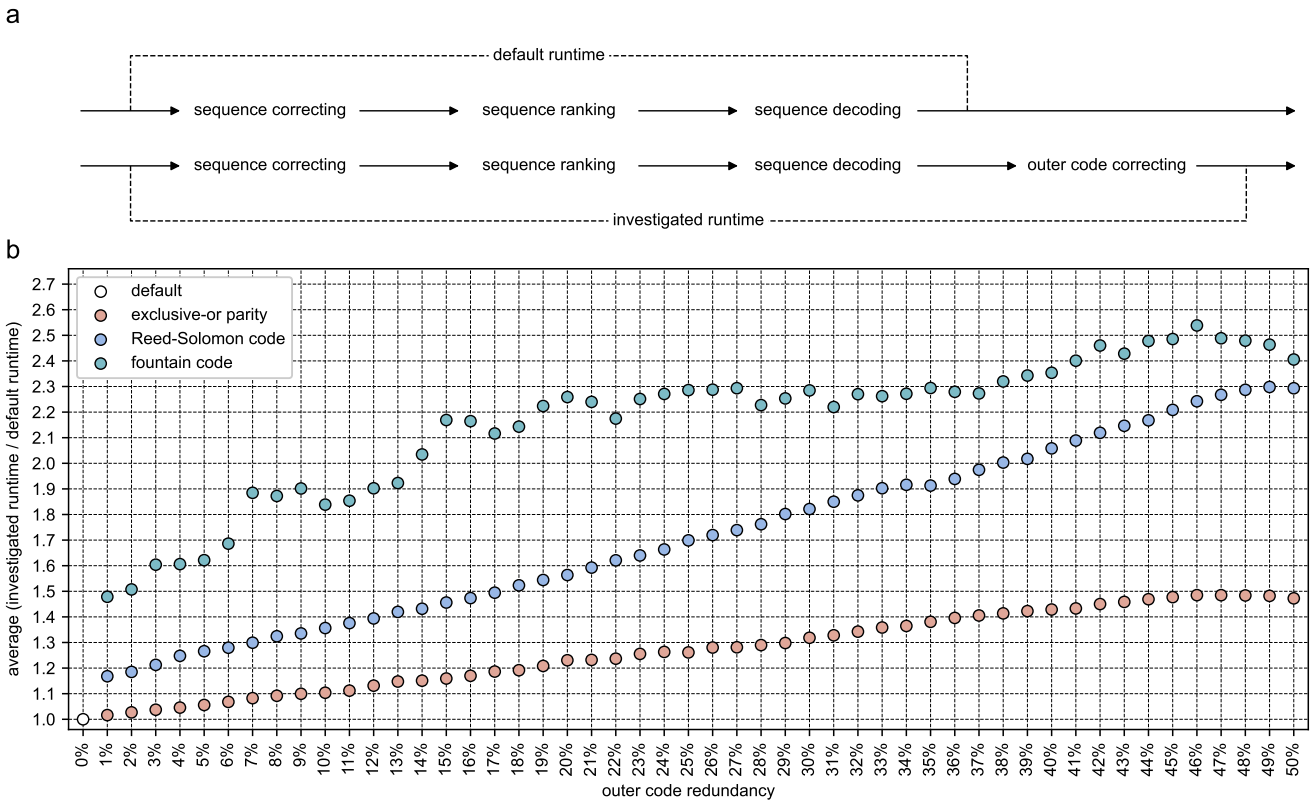

Fig. S10 | Evaluation methods and result analysis for handling runtime differences. (a) Retrieval workflows with and without an outer code, along with their corresponding time breakdowns. For pipelines incorporating an outer code, the only additional step is "outer code correcting". Both configurations were evaluated under identical conditions to compare total runtime. (b) Normalised runtime growth rate at different redundancy rates, expressed relative to the corresponding runtime without an outer code.

Fig. S10b summarises the results for three representative outer codes across redundancy rates ranging from 0% to 50%. Overall, the normalised runtime growth rates increase with redundancy, as expected, yet the growth patterns differ slightly among the three codes. Exclusive-or parity exhibits a nearly linear increase with redundancy, whereas Reed-Solomon codes introduce a more moderate slope. Fountain codes show a slight baseline shift even at low redundancy, reflecting their inherent decoding overhead, followed by a gradual increase. Even in the most extreme case tested, with 50% redundancy, the average runtime increases by at most 2.54 times the baseline without outer codes. By comparison, for a dataset of 1,000 sequences, the runtime of SPIDER-WEB retrieval is at least three orders of magnitude lower than that of a conventional retrieval pipeline (Fig. 4a). Thus, the marginal delay introduced by outer codes is negligible relative to the computational cost of conventional data retrieval workflows.

8.4.3 Necessity of outer codes for SPIDER-WEB

Drawing on nearly one million numerical simulations, we have demonstrated that SPIDER-WEB can be combined orthogonally with any mainstream outer code to address sequence loss, and that adding an outer code does not significantly impact the data retrieval efficiency achieved by SPIDER-WEB. Yet, the decision to employ an outer code should ultimately be guided by the storage context and intended use case. When the primary goal is to ensure reliability in scenarios where sequence loss is anticipated during storage, incorporating an outer code is a reasonable option (which may also include post-retrieval recovery strategies, depending on the nature of the stored content). Conversely, if the objective is real-time data retrieval, the inclusion of an outer code may disrupt the benefits of streaming readout, thereby limiting the ability to realize the real-time data retrieval fully.

9 Tool selection for the conventional data retrieval pipeline

In our study, we benchmarked SPIDER-WEB against the conventional data retrieval pipeline to contextualize its performance under comparable conditions. Tool selection for the conventional pipeline was based on established precedents in recent study.

According to Zhao et al.[3], the data retrieval pipeline employed MMSeqs2 for clustering and MUSCLE5 for multiple sequence alignment. We initially adopted this configuration for our own experimental validation. However, during execution, we observed significant runtime instability, primarily stemming from the high computational cost of MUSCLE5, which rendered it impractical for large-scale retrieval tasks.

In contrast, Qu et al.[62] reported that USEARCH offered both stable and efficient performance in the sequence clustering and multiple sequence alignment tasks. Motivated by this, we carried out a controlled evaluation of two configurations: "MMSeqs2 + MUSCLE5" versus "MMSeqs2 + USEARCH". Both setups were applied to the same sequencing reads, and we assessed their performance in terms of direct retrieval rate and runtime.

Owing to the prohibitive runtime of MUSCLE5, we limited this benchmark to the first technical replicate of three distinct experimental replicates (Methods), each spanning three error-prone PCR rounds (#0-#2) and four sequencing depths (10X, 20X, 30X, and 40X), resulting in $3 \times 3 \times 4 = 36$ test samples. The results are summarized in Fig. S11.

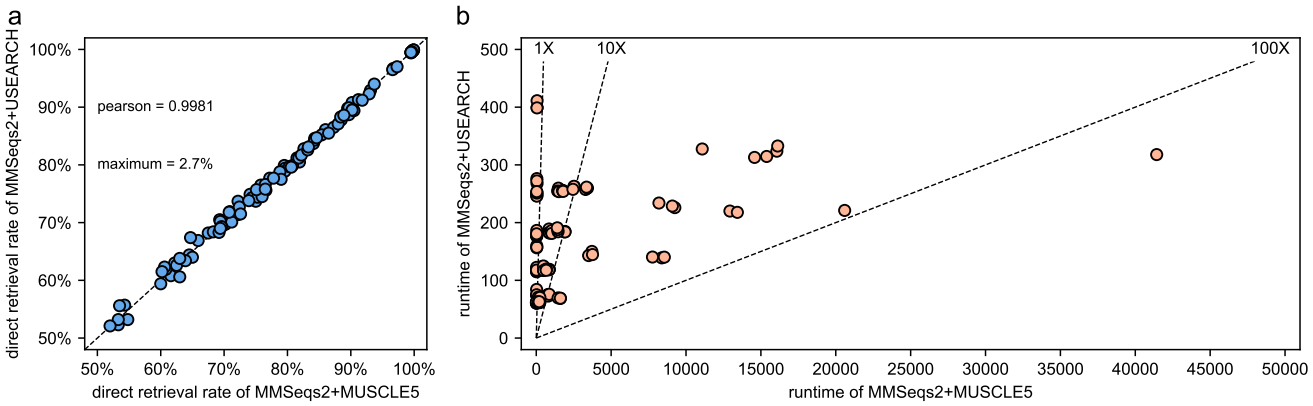

Fig. S11 | The performance difference between "MMSeqs2+MUSCLE5" and "MMSeqs2+USEARCH". (a) Direct retrieval rate, defined as the proportion of 1000 high-confidence retrieved DNA sequences that match 1000 intended DNA sequences after completing the pipeline. (b) Runtime comparison of the two pipelines under consistent experimental conditions.

As shown in Fig. S11a, the direct retrieval rate remained nearly identical between the two configurations, suggesting that the sequence consensus produced by MUSCLE5 and USEARCH did not significantly differ in terms of sequence recovery quality. However, the runtime analysis reveals a stark contrast (Fig. S11b): USEARCH achieved an average speed-up of 12.32×, with the most extreme case reaching 130.35× faster than MUSCLE5.

This performance disparity is largely attributable to the algorithmic design of USEARCH, which relies on heuristic and greedy strategies optimised for large-scale, high-throughput sequence handling. By contrast, MUSCLE5 emphasises alignment accuracy and refinement, resulting in a higher computational overhead that is unnecessary for consensus extraction in this context. Given the high redundancy in sequencing reads and the relatively narrow variation within clusters, the accuracy advantage of MUSCLE5 is marginal in our pipeline, while its runtime cost is substantial.

Based on these findings, we selected "MMSeqs2 + USEARCH" as the conventional baseline to compare against the pipeline components of SPIDER-WEB (sequence correcting + sequence ranking), as it preserves retrieval accuracy while offering orders-of-magnitude gains in computational efficiency – making it the most appropriate baseline for performance comparison.

10 Explanation and measurement of non-blocking retrieval mode

In DNA-based data storage, conventional data retrieval pipelines typically operate in a blocking fashion, where computational analysis begins only after sequencing is fully completed. While suitable for bulk data reconstruction, such blocking designs introduce substantial latency and fail to exploit the continuous nature of modern sequencing outputs, especially in single-molecule platforms that stream reads in real time. To address this, SPIDER-WEB introduces a non-blocking retrieval mode, in which computational processes such as sequence correction, candidate ranking, and message decoding are interleaved with ongoing sequencing. This design enables early identification of intended sequences before the entire sequencing run is finished, thereby transforming the retrieval pipeline into a real-time, feedback-driven process. Non-blocking mode is particularly advantageous for interactive or latency-sensitive applications – such as rapid data access, streaming visualisation, or responsive storage systems – where minimising response time is critical. Here, we analyse the feasibility of this mode and quantify its performance characteristics through simulation and measurement.

10.1 Feasibility analysis of non-blocking retrieval mode

In the conventional data retrieval pipeline of DNA-based data storage, processing typically proceeds in a strictly sequential fashion after sequencing is complete. As shown in Fig. S12, this pipeline comprises four stages: sequence clustering, sequence alignment, sequence decoding, and message correcting, with each step beginning only after the previous one has fully concluded. Such stage-by-stage dependency introduces unnecessary latency and limits opportunities for early-stage interpretation.

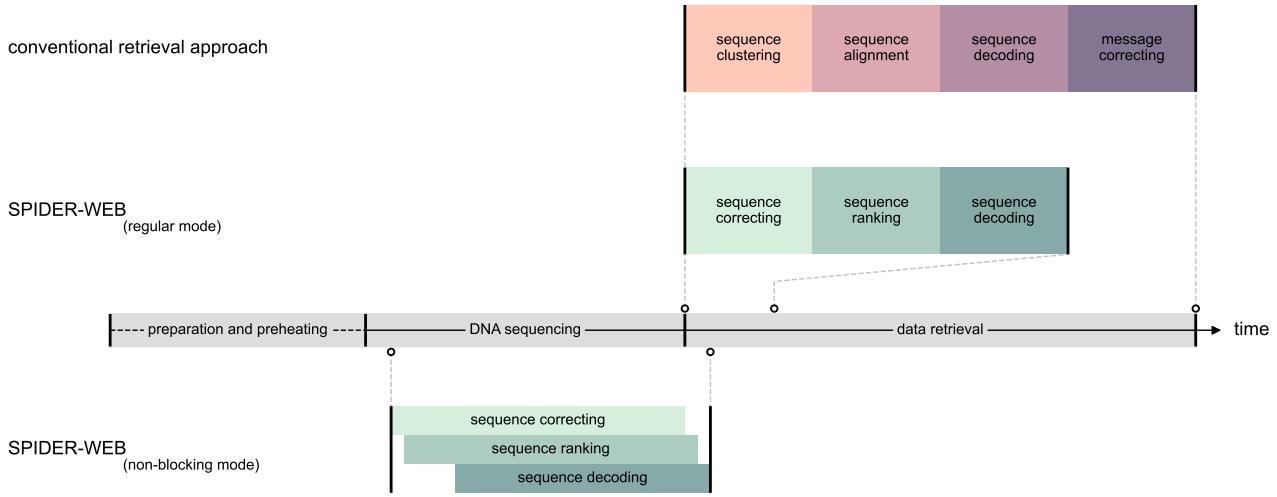

Fig. S12 | Different data retrieval approaches in DNA-based data storage. This illustration contains the conventional data retrieval pipeline, the regular mode of data retrieval approach in SPIDER-WEB, and the non-blocking mode of data retrieval pipeline in SPIDER-WEB. In current DNA sequencing technologies, the regular mode is primarily suited for high-throughput sequencing platforms, whereas the non-blocking mode is mainly compatible with single-molecule sequencing platforms.

SPIDER-WEB re-conceptualises this structure by organising data retrieval into three steps: sequence correcting, sequence ranking, and sequence decoding. In the standard (regular) mode, these steps are still triggered only after sequencing has finished. However, they are not inherently bound to a rigid temporal order, and crucially, they do not require the sequencing process to be fully completed before initiation. This observation creates the foundation for a new operational paradigm.

Building on this flexibility, SPIDER-WEB introduces a non-blocking retrieval mode in which computational tasks proceed concurrently with ongoing sequencing. In this mode, sequencing reads, particularly from single-molecule sequencing platforms, are immediately fed into the sequence correcting process. The sequence candidates are then incrementally added to a dynamic ranking dictionary-like structure that tracks the frequency of all corrected candidates in real time.

As DNA sequencing progresses, the ranking structure is updated in real time. Once the frequency of any candidate surpasses a predefined threshold, calibrated against the expected maximum level of false positives (see next subsection), SPIDER-WEB can begin decoding and visualisation of the corresponding digital message without waiting for the sequencing run to complete. This early-exit mechanism transforms the retrieval pipeline into a feedback-driven loop, allowing preliminary results to emerge in parallel with ongoing sequencing. Compared to blocking pipelines, this design markedly improves responsiveness. It enables access to stored information while sequencing is still underway, making the system particularly suitable for interactive or latency-sensitive applications such as rapid media preview, responsive querying of DNA-based archives, and other scenarios where fast turnaround is critical.

It is important to clarify that the distinction between the regular and non-blocking modes also extends to how data retrieval efficiency is measured. As illustrated in Fig. S12, in the regular mode, DNA sequencing is already complete, and SPIDER-WEB operates on a static pool of reads. In this setting, data retrieval efficiency reflects the pure computational throughput of SPIDER-WEB (Methods, Appendix 1), as each DNA sequence can be processed immediately without waiting for upstream input. By contrast, in the non-blocking mode, SPIDER-WEB must synchronise with the DNA sequencing platform in real time. Here, the moment when a DNA sequence becomes available is determined not solely by SPIDER-WEB but also by the physical capture of DNA molecules by the sequencing platform, the detection of raw signals, and their subsequent translation into base calls. Thus, the observed data retrieval efficiency in this mode is effectively constrained by the slowest stage of the entire pipeline (see Appendix 14). While the regular mode quantifies the intrinsic data retrieval capacity of SPIDER-WEB, the non-blocking mode quantifies the end-to-end throughput achievable in conjunction with sequencing hardware. This distinction is critical when interpreting efficiency metrics and evaluating real-time performance. For this reason, both metrics are reported in our study: the regular mode isolates the intrinsic efficiency of the retrieval algorithm itself, whereas the non-blocking mode characterises the practical, system-level efficiency achievable in real time. Together, they provide a complete view of SPIDER-WEB's retrieval performance under different usage scenarios (see Fig. 5-6).

10.2 Measurement of non-blocking retrieval mode

To further evaluate the feasibility of non-blocking retrieval, we performed a set of simulations centred on one of its most critical decision parameters: the maximum frequency of a false-positive sequence candidate recorded in the ranking dictionary-like structure. This metric is decisive because, under non-blocking operation, once any corrected sequence surpasses a preset frequency threshold, it is provisionally accepted as the intended target and decoding is triggered. A false-positive candidate that happens to accumulate disproportionately high frequency could therefore cause premature or erroneous decoding. The statistical behaviour of such false positives under different sequencing conditions thus provides a direct measure of non-blocking safety margins.

Our simulations extended the scaling experiments described in Methods (Fig. 2e), covering regimes from kilobyte-scale to gigabyte-scale data. For each setting, we systematically measured the average of the maximum frequency attained by false-positive candidates as a function of the number of available sequence copies. These experiments were stratified by both edit error rate (0.5-4.0%) and sequence diversity (10^1-10^8), thereby jointly capturing the two dominant factors influencing the abundance and distribution of erroneous candidates. The results are presented in Fig. S13 (grouped by error rate) and Fig. S14 (grouped by diversity).

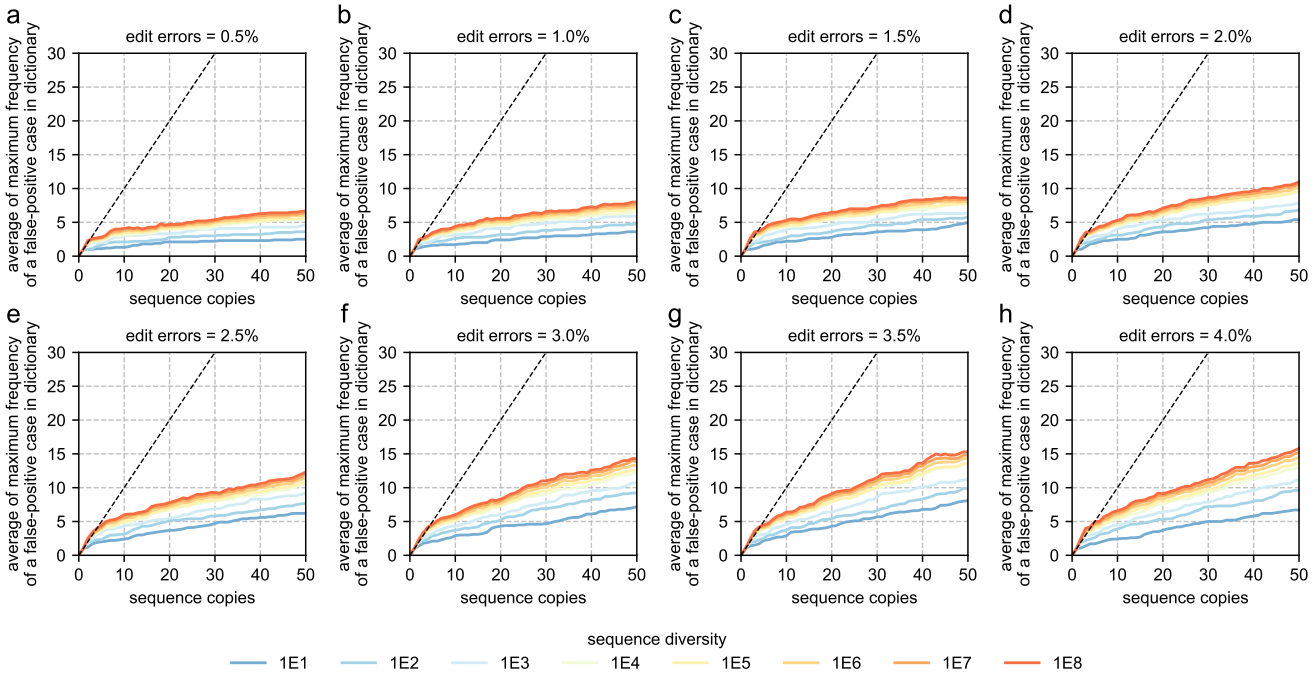

Fig. S13 | Effects of sequence diversity and error rate on the average of maximum frequency of a false-positive sequence candidate in dictionary across different sequence copies, grouped by error rate. (a) - (h) correspond to error rates ranging from 0.5% to 4.0%. The dashed line represents the average sequence copy number.

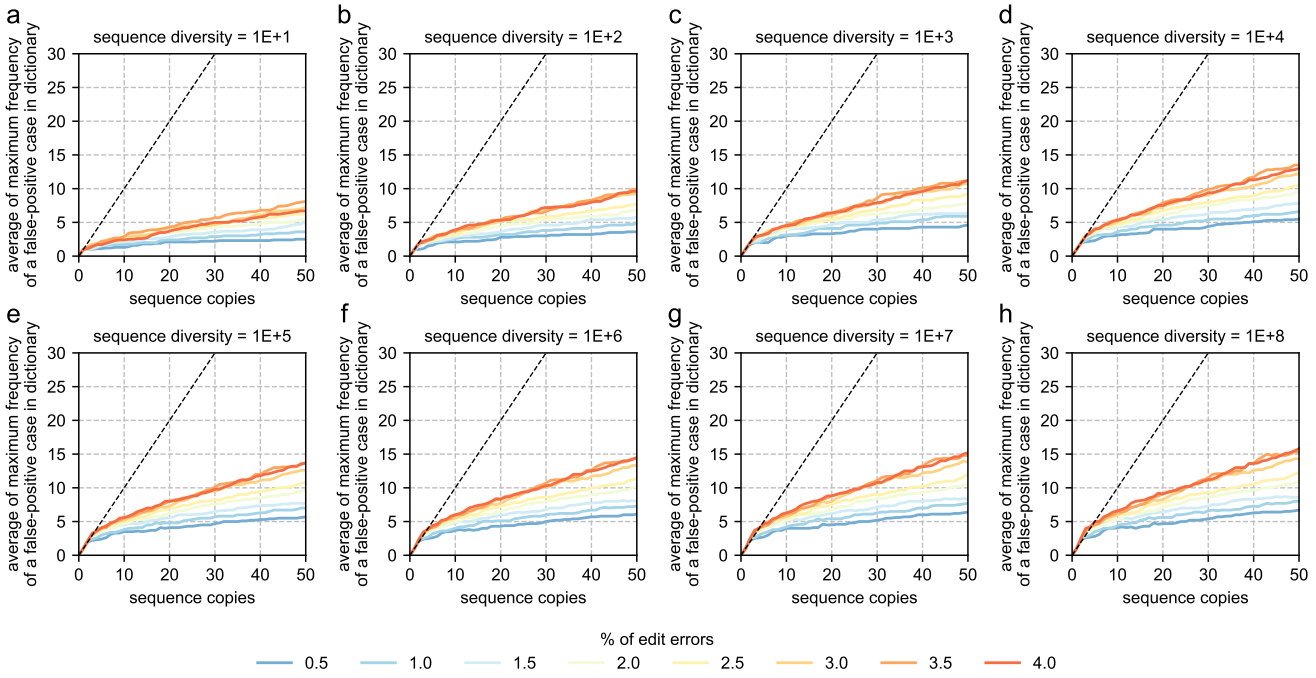

Fig. S14 | Effects of sequence diversity and error rate on the average of maximum frequency of a false-positive sequence candidate in dictionary across different sequence copies, grouped by sequence diversity. (a) - (h) correspond to sequence diversities ranging from 10^1 to 10^8. The dashed line represents the average sequence copy number.

Three robust observations follow:

- **Strict separation:** Across all tested conditions, the solid curves remain well below the dashed baseline. The maximum frequency attained by any falsepositive candidate is always smaller than the contemporaneous average copy count of an intended sequence. This guarantees a safety margin for a depthaware trigger.
- **Widening margin with depth:** As sequencing proceeds and the average copy number increases, the gap between the dashed line and the solid curves monotonically widens. Intuitively, counts for the true target grow roughly linearly with depth, whereas counts for any single false candidate grow much more slowly due to the dispersion of errors over many candidates. Consequently, the risk that a false candidate temporarily "overtakes" the true target decreases with depth.
- **Shift with noise and diversity, but preserved ordering:** Higher editerror rates and larger sequence diversity shift the solid curves upward, indicating more unintended aggregation. However, the ordering relative to the dashed baseline does not change: even in the most challenging regimes, the maximum falsepositive frequency stays well below the average sequence copy number, and the separation still increases with depth.

These properties directly inform the decision rule for real-time data retrieval with non-blocking mode. Let N denote the current estimate of the average copy number per intended sequence (the dashed baseline) and $F_{\max}(N)$ denote the worstcase upper envelope of the solid curves under the operative error rate and sequence diversity. A safe, depthaware threshold $\Theta(N)$ can be chosen such that

$$F_{\max}(N) < \Theta(N) \leq N.$$

When the top candidates observed frequency first reaches $\Theta(N)$, sequence decoding can be triggered without waiting for the full run. Since the gap $N - F_{\max}(N)$ increases with depth, $\Theta(N)$ can be implemented as an adaptive linear-in-depth rule anchored to the dashed baseline, which preserves statistical robustness at early depths and becomes increasingly conservative as more reads accrue.

In practice, the thresholds can be selected by referencing the upper envelope of the solid curves in Figures S13-S14 under matched error rate and diversity settings, and then placing $\Theta(N)$ strictly between that envelope and the dashed baseline. This strategy ensures that early exits are triggered by true signals while maintaining a negligible probability of premature decoding by a false-positive candidate.

10.3 Final form of non-blocking mode

We believe that what constitutes the final form of non-blocking mode in the data retrieval process depends on the future trajectory of DNA sequencing technologies. From a conservative perspective, we assume that DNA sequencing will remain based on its current operational principles – namely, high-throughput short-read sequencing or single-molecule long-read sequencing – with only incremental improvements in throughput rather than paradigm-shifting innovations. Under this scenario, as illustrated in Fig. 5b and 5c, the total time required for data retrieval is largely dominated by the DNA sequencing process itself. As a result, the termination point of the entire retrieval pipeline will, in practice, asymptotically converge to the time point when the sequencer halts – effectively making real-time decoding possible as sequencing progresses. From a speculative perspective, it is conceivable that future innovations in DNA sequencing may involve technologies co-designed with data storage in mind, potentially featuring ultra-low-latency interfaces, hardware-accelerated decoding logic, or novel readout modalities tailored for data retrieval. Together, these perspectives frame non-blocking mode not as a fixed capability, but as an evolving design space: one whose ultimate performance boundaries are defined by the interplay between molecular readout mechanisms and our method.

11 Global constraint design for ultra-long data-coding sequences

We devised an extended design framework for constructing long DNA sequences. These sequences satisfy both stringent regionalised and global constraints. With this capability, SPIDER-WEB retains full compatibility with short-segment-based storage scenarios and further extends its applicability to long-sequence-based storage scenarios.

11.1 Emerging challenges in constructing long sequences

Compared to short segments, long sequences face emerging constraints as they are typically assembled from multiple short segments. Such a shift in workflow introduces two major risks that can significantly reduce the yield of correctly assembled sequences[63]:

- **Segment duplication:** Repeated short segments within the sequence of identical length can cause ambiguous breakpoints during assembly. Such ambiguities may thus lead to incorrect segment ordering or misalignment.
- **Palindrome formation:** Reverse-complementary patterns within or across short segments can form secondary structures. These structures can interfere with ligation or polymerase-based assembly, obstructing the completion of full-length sequences.

Both risks become more severe as sequence length increases, since longer sequences require more assembly steps and make these risks more likely to occur. When these risks arise in localised regions within a long sequence, they are particularly difficult to eliminate during assembly, potentially requiring substantial redesign or alternative assembly strategies. Therefore, our constraint design incorporates two independent global checks (as illustrated in Fig. S15): (1) within any spacing of length n, no duplicate k-mers are allowed, where k is the predefined window size; and (2) within any spacing of length m, no k-mer is allowed to appear as the reverse complement of another k-mer in the same span. These constraints are enforced throughout the design process to mitigate duplication and palindrome-related risks.

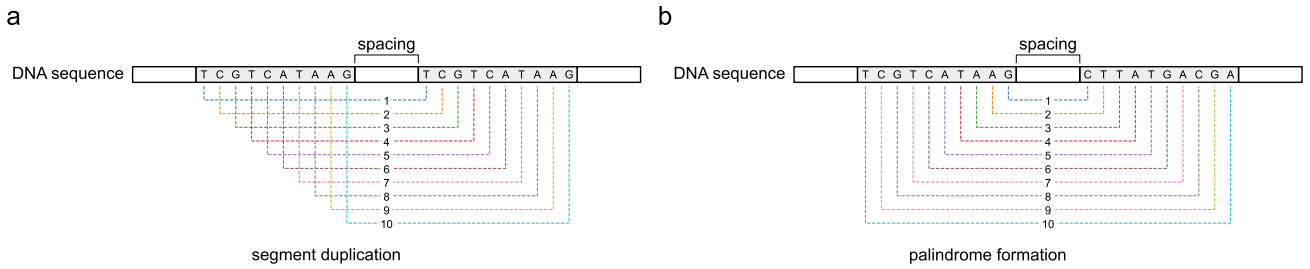

Fig. S15 | Illustration of global constraints. (a) Illustration of segment duplication. (b) Illustration of palindrome formation. The length of segment (k) is set as 10.

11.2 Construction framework

While the SPIDER-WEB variants described in Appendix 3 can remove repeated segments or palindromes of length equal to or exceeding the window size during encoding using pre-constructed components, such guarantees cannot be maintained when building ultra-long sequences. This limitation arises because dynamically introduced constraints during encoding may cause the process to fail, which is a risk emphasised in the original report of HEDGES[23] and was also observed in our DNA Fountain experiments (Fig. S4). Hence, our extended design framework partitions the target message into blocks, encodes each block as multiple short-segment candidates, and then determines the final assembly through filtering and trial combinations. The resulting assembled sequences represent the complete message while satisfying both regionalised and global constraints.

11.2.1 Short segment preparation

The long binary message is first divided into multiple short blocks of equal length. Each short block is then encoded through a predefined coding digraph with global-constraint detection components enabled (Appendix 3). During encoding, if a potential violation of global constraints is detected, the accessibility of subsequent vertices is temporarily tested to ensure that the encoding path can still be completed. The process will terminate early if a situation similar to that described for HEDGES (where no valid continuation exists) is encountered. To collect all available candidates, the encoding is performed from every vertex in the coding digraph as a starting (or virtual) vertex[8,21]. The k-mer corresponding to the chosen starting vertex is recorded at the beginning of the short segment, serving as a reference for locating the initial position in the coding digraph during decoding. A user-defined fixed-length parameter is applied during encoding to ensure that all short segments have identical lengths. After this step, each short block typically yields a large set of segment candidates for subsequent selection.

11.2.2 Pre-screening strategy

Suppose each short block has a segment candidates and a long sequence contains b blocks, the total number of possible assemblies can be as large as a^b. Although exhaustively evaluating all combinations to obtain the optimal result would be ideal, in some cases such enumeration could require computation times exceeding one year on a single laptop. Therefore, an efficient strategy is required to obtain assemblies that meet all constraints without exhaustive search.

We provide the following two strategies, which can be applied independently or in combination:

- **Conflict-minimised candidate selection:** This strategy prioritises short segments with minimal sequence conflicts across candidate sets, thereby reducing the likelihood of duplication or palindrome formation in the subsequent assembly feasibility assessment. For each candidate set, we use a window of length l ($l \geq$ window size) to obtain the set of l-mers from both the forward and reverse-complement strands of every short segment in the set. For any given short segment within the current candidate set, we compute the intersection size between its l-mer set and the l-mer sets derived from all short segments in other candidate sets. Candidates are ranked by this intersection size in ascending order, where a lower intersection size indicates a lower probability of segment duplication or palindrome formation with segments from other candidate sets. Based on the tolerable computational workload, only the subset of short segments with the smallest intersection sizes is retained from each candidate set, reducing the candidate size n to a computationally acceptable value and improving assembly efficiency.

- **Non-coexisting neighbour identification:** This strategy prioritises the early identification of mutually incompatible short segments across neighbouring candidate sets, enabling rapid pruning of the search space during assembly. For any given short segment in a candidate set, we identify the short segments in its left and/or right adjacent candidate sets (neighbouring sets) that cannot be selected if this segment is chosen. During the subsequent assembly feasibility assessment, selecting a candidate can immediately reduce the available options in neighbouring candidate sets and result in empty sets that trigger early termination.

It should be noted that the decision to apply these pre-screening strategies depends entirely on the characteristics of the candidate sets. If the candidate set sizes are already small, or if the global constraints between different candidate sets can be readily satisfied, it can be sufficient to either randomly select or exhaustively enumerate candidates, and then evaluate each assembled sequence without employing any pre-screening strategy. On the other hand, it is also possible that no valid sequence emerges for an extended period. In such cases, beyond relaxing global constraint requirements, the most straightforward approaches are to reduce the segment length or decrease the number of segment candidates to be assembled.

11.2.3 Connectors between short segments

In scenarios where the sequence error rate is very low, connector sequences between short segments are not strictly necessary. Yet, under more complex and error-prone conditions, connectors play an important role in distinguishing each data-coding short segment within a long sequence. In addition, they can serve other purposes, such as defining the orientation of double-stranded constructs or providing auxiliary signals for downstream processing. During assembly, the connectors are inserted between every two consecutive short segments in the prescribed order, producing a continuous long sequence that retains the individual segment boundaries.

11.2.4 Assembly feasibility assessment

All long sequence candidates, assembled from the combinations of the selected short segment candidates, are first required to satisfy the duplication- and palindrome-avoidance constraints $\mathbb{C}_{(n,m,k)}$. Here, k denotes the predefined window size. These global constraints specify that (i) no duplicate k-mers are allowed within any spacing n, and (ii) no k-mer may appear as the reverse complement of another k-mer within any spacing m.

On top of this hard discrimination, if valid sequences can be generated at a relatively high frequency, we apply a soft optimisation procedure to further select those with lower overall duplication and palindrome density. In this step, the focus shifts from enforcing a minimum distance between identical or reverse-complement segments to evaluating the density of such events across the entire sequence, quantified by the penalty score:

$$\text{score} = \sum_i \frac{1}{\text{distance}_i}$$

where distance_i is the spacing between the i-th pair of identical or reverse-complement segments (as illustrated in Fig. S15). This formulation converts local duplication/palindrome risks into a continuous, additive penalty rather than a binary pass/fail decision, enabling more flexible optimisation.

12 Construction details of the 2,950-nt sequences used in the demonstration

Although the strategies in Appendix 11 were individually validated during the design stage, in the initial application of this framework, we purposely avoided complex procedures. After generating short-segment candidates, we assembled long sequences by randomly selecting one segment from each candidate set and directly applying hard discrimination. When a selected sequence satisfied the global duplication- and palindrome-avoidance constraints $\mathbb{C}_{(60,20,10)}$, we retained it immediately and terminated the search. This simple yet effective procedure ensured that valid ultra-long sequences could be generated without additional optimisation.

12.1 Detailed constraint parameters and available short segment candidates

In our demonstration, the synthesis of long sequences was designed with direct nanopore sequencing in mind, which required careful consideration of sequencing error rates since the sequence correcting in SPIDER-WEB occurred only after segmentation. Such segmentation was not a core function of SPIDER-WEB but rather a practical adaptation to enable fully autonomous real-time decoding without external software. While poly-A tails are commonly used in nanopore sequencing workflows for adapter binding and transcript enrichment[64], we adopted this concept in our design, using poly-T stretches as connectors (or anchors) to delimit encoded short segments within a long sequence. Each 48-bit unit block was independently encoded through SPIDER-WEB using all virtual vertices in the coding digraph, with the chosen starting vertex recorded at the beginning of the sequence. Constraint-checking modules ensured that no repeated or palindromic subsequences longer than 10 nt appeared in any candidate. To further improve retrieval robustness, the length of each encoded short segment was extended to 110 nt to accommodate potential ambiguity near connectors, and candidates whose first or last 5 bases contained T residues were excluded to reduce anchor-related misinterpretation.

Under these design parameters, each unit block produced a candidate set of short segments whose size is illustrated in Fig. S16. On average, 384.9 candidates were obtained per unit block (minimum 13, maximum 682). By multiplying the candidate set sizes across all unit blocks, each long sequence yielded on the order of 3.68×10^{52} possible assemblies (minimum 6.62×10^{44}, maximum 1.29×10^{54}). Such an enormous combinatorial space provided a sufficiently rich pool from which sequences satisfying the global constraints could be reliably identified.

12.1.1 Random selection and termination condition

In our demonstration, only hard discrimination was applied. We did not employ the soft optimisation procedure, as its performance had not yet been sufficiently correlated with the practical difficulty of sequence assembly. Since segment duplication is the most critical factor influencing assembly, we empirically set the global duplication- and palindrome-avoidance constraints to $\mathbb{C}_{(60,20,10)}$.

For each long sequence, one short segment was randomly selected from each candidate set and concatenated using poly-T anchors. The resulting sequence was then tested against $\mathbb{C}_{(60,20,10)}$. If any constraint was violated, another random combination would be tested until a valid long sequence was found, which was then retained and subjected to post-processing. Fig. S17 and Fig. S18 show, for each assembled long sequence, the distribution of the minimum spacing of all contained 10-mers with respect to segment duplication and palindrome formation, consistent with our imposed constraints.

12.1.2 Post-operation

To uniquely label each valid long sequence, SPIDER-WEB was used to generate 60-nt address segments appended to both ends of the sequence. Candidate addresses were screened by Levenshtein distance to ensure a minimum pairwise separation of 18 nucleotides (corresponding to 30% error redundancy), both between the two addresses of the same sequence and across different sequences. In addition, to mitigate the reduced accuracy typically observed at the beginning of nanopore sequencing reads, 200-nt buffer segments were generated by SPIDER-WEB and inserted at both termini[65]. Thus, each final construct followed a symmetric layout: a 200-nt buffer segment was placed at both termini, followed inward by a 60-nt address segment on each side, with the payload (assembled long sequence) located in the middle. Buffer, address, and payload were concatenated in the prescribed order using 10-nt poly-T anchors, resulting in a 2,950-nt DNA sequence (Fig. 6a). All final constructs were verified to satisfy $\mathbb{C}_{(60,20,10)}$.

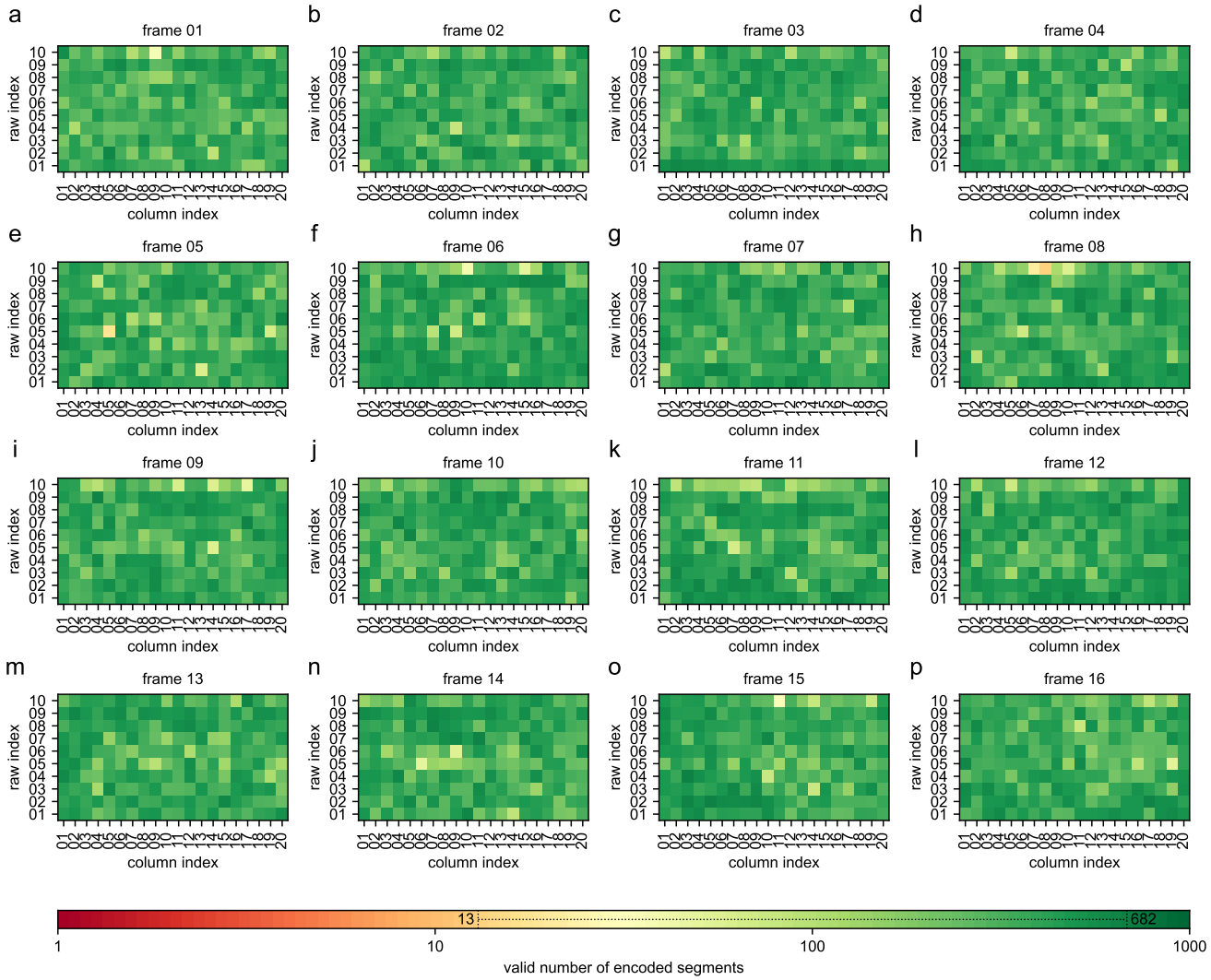

Fig. S16 | Valid number of encoded segments in each frame. The encoded animation consists of 16 frames, each partitioned into 10 rows and 20 columns of 3×2 unit blocks (3,200 unit blocks in total), with each unit block represented by one encoded segment. (a)-(p) display the valid number of encoded segments for the corresponding 16 frames. Across all frames, the number of valid segments ranges from a minimum of 13 to a maximum of 683, and 384.9 on average.

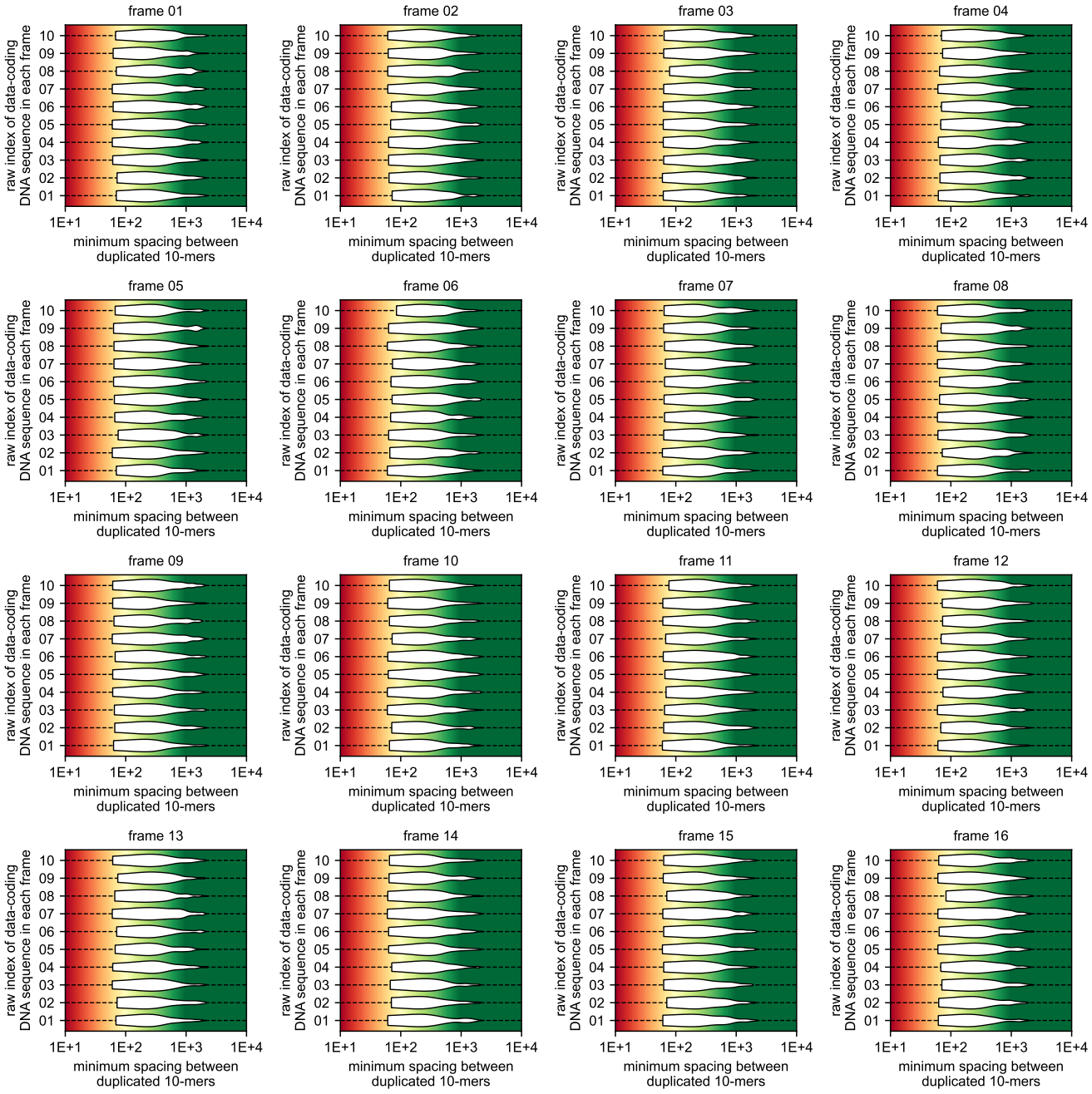

Fig. S17 | The distribution of minimum spacing between duplicated 10-mers. Each sample in the violin represents a type of 10-mer in the observed DNA sequence.

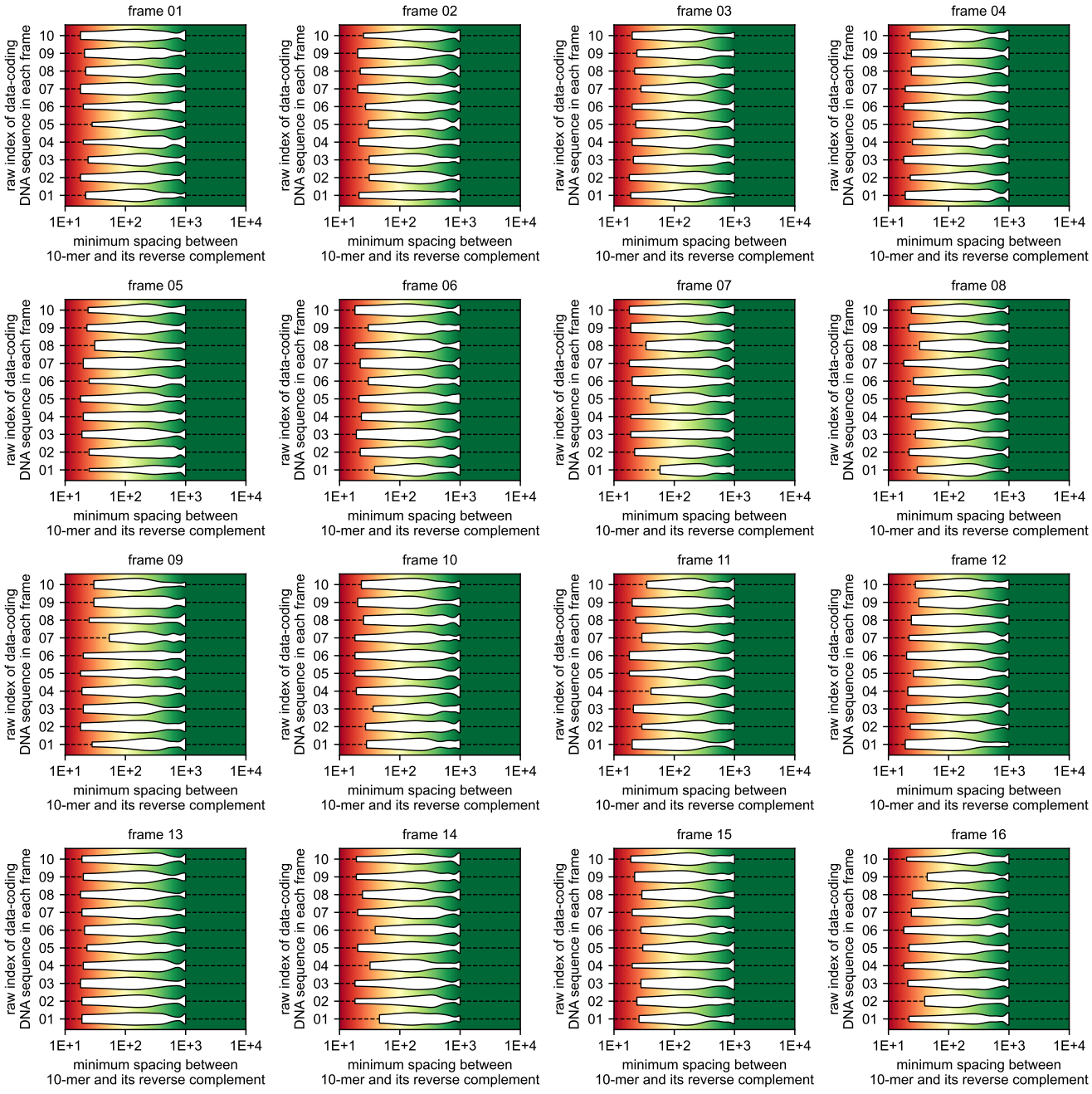

Fig. S18 | The distribution of minimum spacing between a 10-mer and its reverse complement segment. Each sample in the violin represents a type of 10-mer in the observed DNA sequence.

13 Instantaneous data retrieval efficiency under non-blocking retrieval mode

To further dissect the system behaviour, we evaluated *instantaneous data retrieval efficiency* (hereafter referred to as instantaneous efficiency) in the context of the non-blocking retrieval mode. Unlike aggregate runtime benchmarks, this metric captures the time-resolved gain in data retrieval efficiency as sequencing and basecalling proceed, thereby providing a closer alignment with real operational conditions.

13.1 Coarse-grained calculation of instantaneous efficiency

In the non-blocking mode, FASTQ files are streamed from the sequencing platform in chronological order. After each file becomes available, the raw DNA sequences (or sequencing reads) are parsed and progressively integrated into the reconstructed output. At every update, we calculated the increase in the retrieval rate and divided it by the elapsed time interval since the previous update. This ratio defines the instantaneous efficiency, reflecting the marginal throughput achieved at each step. Based on the five independent replicate trials described in the Main Text, we applied this calculation to obtain time-resolved trajectories of instantaneous efficiency for each trial (Fig. S19a).

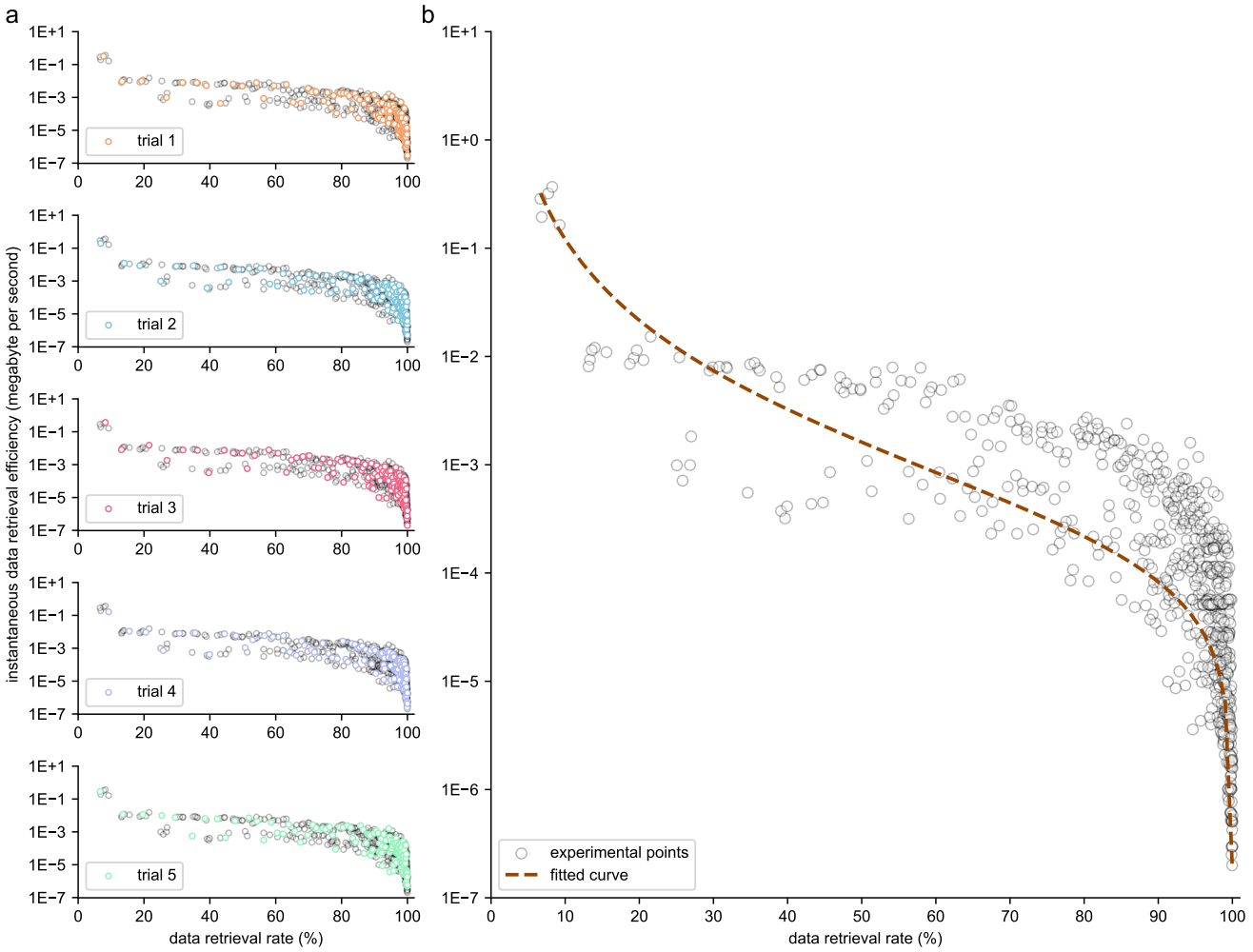

Fig. S19 | Observation and modelling of instantaneous efficiency. (a) Observations of instantaneous efficiency for five independent replicate trials. (b) Trajectory modelling based on the aggregated experimental points.

13.2 Characteristic phases of instantaneous efficiency

As the retrieval rate increases, instantaneous efficiency shows an initial sharp drop, followed by an intermediate stable phase, and finally another decline (Fig. S19a). A plausible mechanism is as follows: in the early stage, when most data blocks remain unretrieved, the chance of capturing intended blocks is high, resulting in high instantaneous efficiency; in the intermediate stage, after the "easy"

blocks – intended blocks occurring far more often than any potential false positives – have been recovered, a substantial remainder sustains a stable phase of efficiency; in the late stage, only a small fraction of unretrieved blocks persists, and recovery becomes difficult due to scarcity (reflecting sequence-dependent molecular biases such as synthesis yield or sequencing preference) and elevated error rates, with some blocks discarded as uncorrectable and others obscured by false positives.

Inspired by depletion-based models in information retrieval and ecological sampling[66,67], we approximated the relationship between retrieval rate (x) and instantaneous efficiency (y) with the functional form

$$y = \frac{a \times (1-x)}{x^b},$$

where a and b are hyperparameters controlling scaling and curvature. In this analysis, we aggregated the five replicate trials and fitted this equation using the `curve_fit` function in the SciPy package[40] with the default Levenberg-Marquardt method[41,42] (Fig. S19b). The resulting parameters were $a = 5442.8$ and $b = 2.312$. This agreement supports our interpretation of the three characteristic phases and aligns well with the actual retrieval dynamics.

13.3 Relation to coarse-grained efficiency approximation workflow

The trajectory of instantaneous efficiency reflects the combined influence of multiple factors, including sequence error rates, molecular biases such as sequence-dependent synthesis yield or sequencing preference, the frequency with which the sequencing platform outputs FASTQ files, the fragmentation strategy for long reads (which is not intrinsic to SPIDER-WEB), and even the overall size of the retrieved digital data. Therefore, its fine-grained pattern differs substantially from the coarse-grained efficiency approximation workflow defined in Methods, which abstracts low-level processes and focuses on the dominant performance determinants of the SPIDER-WEB data retrieval pipeline.

Yet, the coarse-grained efficiency approximation workflow offers a reliable reference when interpreting the onset of instantaneous efficiency. Specifically, the first observed value of instantaneous efficiency is not yet confounded by the above complexities and therefore provides a clean benchmark for system-level performance. On the nanopore sequencing platform, this workflow predicted retrieval efficiencies corresponding to a standard laptop in the range of 96.0-683.2 kilobytes per second (Fig. 5e). Experimentally, the first instantaneous efficiency values we measured fell between 167.7 and 377.3 kilobytes per second (Tab. S3), fully consistent with the range anticipated by the coarse-grained approximation, despite including the overhead of sequence fragmentation. This agreement indicates that while the subsequent trajectory reflects additional layers of molecular and platform-specific variation, the initial instantaneous efficiency faithfully captures the baseline performance predicted by the coarse-grained workflow.

Tab. S3 | First observed instantaneous efficiency across replicate trials. The first instantaneous efficiency is reported in kilobytes per second, and the corresponding observation time is reported in seconds, measured from the release of the first FASTQ file by the sequencing platform.

Replicate trial	1	2	3	4	5
First instantaneous efficiency	328.03	198.79	377.29	167.65	292.83
Observation time	0.0044	0.0065	0.0041	0.0103	0.0043

14 Analysis of data backlog risk and sequencing throughput release

In real-time non-blocking data retrieval scenarios, an essential concern is whether the computational stage of SPIDER-WEB has sufficient capacity to keep pace with, or exceed, the upstream processes. If its processing power is limited, data produced earlier in the pipeline can accumulate unprocessed in memory or storage, creating a data backlog that undermines the effective throughput of the entire system. Conversely, if SPIDER-WEB operates at a rate equal to or faster than the upstream flow, the output of the preceding pipeline can be fully utilised without additional latency.

This consideration is particularly relevant to nanopore sequencing platforms, where data are streamed continuously rather than delivered as a completed batch. In such settings, DNA sequencing, basecalling, and higher-level data retrieval (e.g., SPIDER-WEB) operate concurrently with signal acquisition, forming a fully pipelined process. The key evaluation criterion is not whether individual stages can be optimised in isolation, but whether downstream computation can consistently keep pace with upstream data generation. Assessing whether SPIDER-WEB sustains this pace therefore provides a direct measure of its compatibility with real-time, non-blocking operation, and enables attribution of system latency under streaming conditions.

14.1 Benchmarking SPIDER-WEB against upstream processes

To establish a coarse benchmark for SPIDER-WEB, we selected its throughput under a 4.0% error rate. This choice is motivated by the fact that currently reported nanopore sequencing platforms typically exhibit error rates below this level[68]. Thus, the 4.0% setting provides a conservative yet realistic bound. The SPIDER-WEB data retrieval pipeline consists of three steps: sequence correcting, sequence ranking, and sequence decoding, where ranking relies primarily on constant-time hash lookups, and decoding incurs the same cost as correcting when sequences are error-free. Accordingly, the computational bottleneck of SPIDER-WEB lies almost entirely in the correcting stage. Based on the results shown in Fig. 5b, the minimum stable throughput of SPIDER-WEB under these conditions is 7,800,482.1 nucleotides per second (hereafter simplified as 7.80×10^6).

For context, the two principal upstream steps preceding SPIDER-WEB are DNA sequencing and basecalling. Nanopore sequencing operates at a theoretical maximum rate of approximately 450 nucleotides per second per channel. Nanopore basecalling throughput is more variable, depending on factors such as sequence length, hardware configuration, model complexity, and parallelisation strategy. Nevertheless, benchmarking reports indicate a maximum stable throughput of approximately 3.47×10^6 nucleotides per second on a single GPU instance, which is calculated by

$$\frac{10^9}{\text{hours per Gbase}\times3600\times\text{GPUs}}$$

where all parameters required for the calculation are reported in Tab. S4.

When compared with these values, a single CPU thread running SPIDER-WEB is equivalent to the combined output of 17,334 sequencing channels. This implies that the minimum configuration of SPIDER-WEB can sustain the throughput of six PromethION Flow Cell, assuming all flow cells in the platform are functional. Moreover, the throughput of SPIDER-WEB is roughly two-fold higher than that of a GPU basecaller. Considering that CPUs can host multiple threads, that CPU acquisition and deployment costs are substantially lower than those of GPUs, and that SPIDER-WEB natively supports multi-threaded execution (Fig. 5d), it follows that SPIDER-WEB can readily exceed basecalling throughput.

Therefore, in the context of nanopore sequencing platforms, SPIDER-WEB data retrieval pipeline remains free from data backlog and introduces negligible computational burden relative to all upstream steps, confirming that it does not pose a practical throughput bottleneck under realistic sequencing conditions.

14.2 Runtime distribution in our demonstration

To complement the coarse benchmarking analysis, we further examined the actual runtime distribution of the SPIDER-WEB retrieval process in a case study based on a nanopore sequencing platform (CycloneSEQ G400-ER). Across 5 independent replicate trials, we measured the relative contribution of each step to the overall runtime. As shown in Fig. S20, the SPIDER-WEB operating factor (i.e., actual runtime relative to planned availability) remained consistently low across the entire retrieval stage, without noticeable dependence on the progression of sequencing or basecalling. To quantify this observation, we discretised the retrieval stage into intervals of width 0.1 and compared the density distributions across stages. The maximum Wasserstein distance[69] observed between any two stage-specific distributions was only 0.0015245, indicating that distributional differences were negligible. The overall density remained below 10% throughout the process, with an average contribution of 2.998% of the total data retrieval time. Together, these results demonstrate that SPIDER-WEB introduces only a negligible computational burden, regardless of the retrieval stage, and that the dominant share of the overall runtime is instead spent waiting for upstream processes.

This observation highlights two important points. First, under current sequencing conditions, the computational stage of SPIDER-WEB remains unsaturated: the actual runtime is far below its theoretical maximum throughput, and its operating factor shows no stage-dependent accumulation. This unsaturation indicates substantial headroom, meaning that SPIDER-WEB can seamlessly scale with future increases in sequencing throughput without becoming a new bottleneck. Second, the dominant source of latency arises

Tab. S4 | Nanopore basecalling performances. The relevant data are derived from Tables 1 and 2 of the AWS HPC Blog (named Benchmarking the Oxford Nanopore Technologies basecallers on AWS), systematically obtained by Stefan Dittforth, Doruk Ozturk, and Michael Mueller.

instance type	GPUs	basecaller	hours per Gbase	single-GPU throughput (nucleotide per second)	maximum
g5.xlarge	1	dorado v0.2.4	0.21	1.32×10^6	
p4d.24xlarge	8	dorado v0.3.0	0.01	3.47×10^6	✓
g5.2xlarge	1	dorado v0.2.4	0.21	1.32×10^6	
p4d.24xlarge	8	dorado v0.2.4	0.01	3.47×10^6	✓
g4dn.xlarge	1	dorado v0.2.4	0.52	5.34×10^5	
g5.12xlarge	4	dorado v0.2.4	0.05	1.39×10^6	
g5.4xlarge	1	dorado v0.2.4	0.21	1.32×10^6	
g4dn.2xlarge	1	dorado v0.2.4	0.5	5.56×10^5	
g5.48xlarge	8	dorado v0.2.4	0.03	1.16×10^6	
g5.24xlarge	4	dorado v0.2.4	0.05	1.39×10^6	
g4dn.metal	8	dorado v0.2.4	0.06	5.79×10^5	
g4dn.12xlarge	4	dorado v0.2.4	0.12	5.79×10^5	
g5.8xlarge	1	dorado v0.2.4	0.21	1.32×10^6	
p3.2xlarge	1	dorado v0.2.4	0.17	1.63×10^6	
p3.8xlarge	4	dorado v0.2.4	0.04	1.74×10^6	
p3.16xlarge	8	dorado v0.2.4	0.02	1.74×10^6	
p3dn.24xlarge	8	dorado v0.2.4	0.02	1.74×10^6	
g4dn.4xlarge	1	dorado v0.2.4	0.49	5.67×10^5	
g5.16xlarge	1	dorado v0.2.4	0.21	1.32×10^6	
g4dn.8xlarge	1	dorado v0.2.4	0.52	5.34×10^5	
g4dn.16xlarge	1	dorado v0.2.4	0.54	5.14×10^5	
g5.xlarge	1	guppy v6.4.8	0.28	9.92×10^5	
g4dn.xlarge	1	guppy v6.4.8	0.6	4.63×10^5	
g5.2xlarge	1	guppy v6.4.8	0.27	1.03×10^6	
g5.12xlarge	4	guppy v6.4.8	0.07	9.92×10^5	
p4d.24xlarge	8	guppy v6.4.8	0.01	3.47×10^6	✓
g5.4xlarge	1	guppy v6.4.8	0.26	1.07×10^6	
g4dn.2xlarge	1	guppy v6.4.8	0.59	4.71×10^5	
g5.24xlarge	4	guppy v6.4.8	0.07	9.92×10^5	
g4dn.metal	8	guppy v6.4.8	0.07	4.96×10^5	
g5.48xlarge	8	guppy v6.4.8	0.03	1.16×10^6	
g4dn.12xlarge	4	guppy v6.4.8	0.14	4.96×10^5	
g5.8xlarge	1	guppy v6.4.8	0.26	1.07×10^6	
g4dn.4xlarge	1	guppy v6.4.8	0.57	4.87×10^5	
p3.2xlarge	1	guppy v6.4.8	0.22	1.26×10^6	
p3.8xlarge	4	guppy v6.4.8	0.06	1.16×10^6	
p3dn.24xlarge	8	guppy v6.4.8	0.02	1.74×10^6	
p3.16xlarge	8	guppy v6.4.8	0.03	1.16×10^6	
g5.16xlarge	1	guppy v6.4.8	0.26	1.07×10^6	
g4dn.8xlarge	1	guppy v6.4.8	0.61	4.55×10^5	
g4dn.16xlarge	1	guppy v6.4.8	0.63	4.41×10^5	

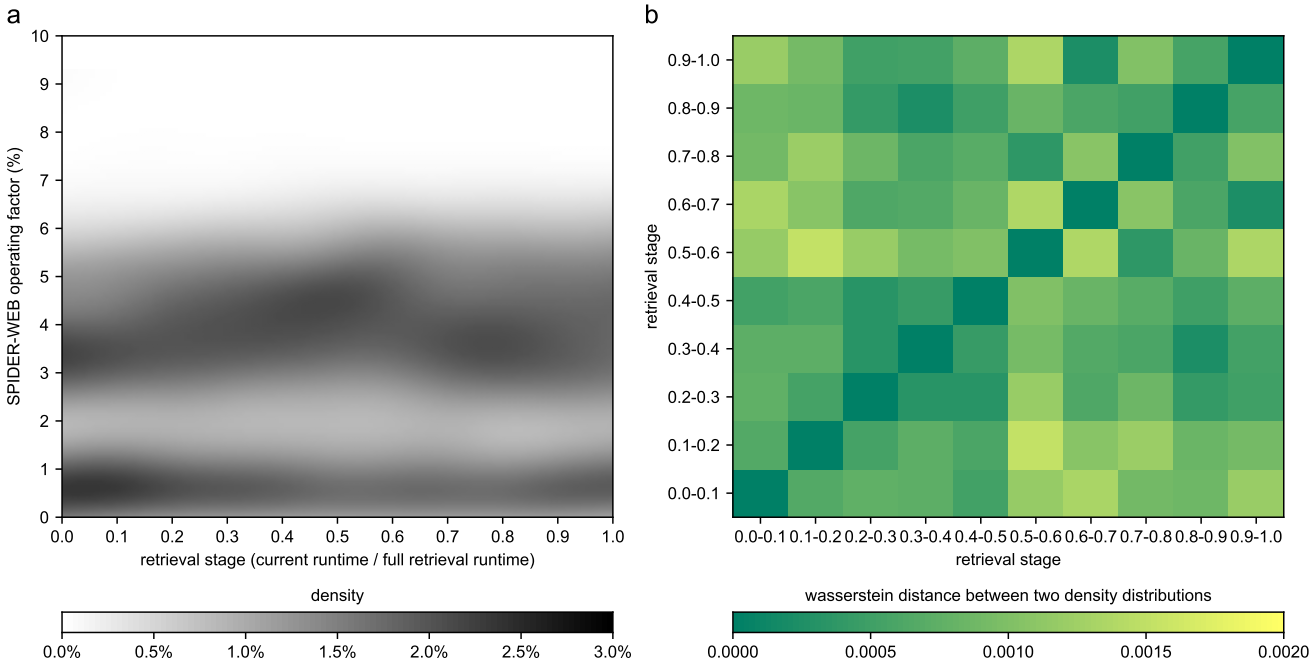

Fig. S20 | Operating factor of SPIDER-WEB in the real-world data retrieval demonstration. The operating factor is defined as the actual runtime of SPIDER-WEB relative to its planned availability; a lower value indicates that a larger fraction of the total retrieval time is spent waiting for upstream processes rather than on computation. (a) Density plot of the operating factor across the retrieval stage. (b)Pairwise Wasserstein distances between stage-specific density distributions, calculated by discretising the retrieval stage into bins of width 0.1.

from upstream processes, namely DNA sequencing and nanopore basecalling, rather than from SPIDER-WEB itself. In other words, the system-level bottleneck in real-time data retrieval lies in molecular readout and signal translation, while SPIDER-WEB introduces negligible additional burden.

14.3 Concluding remarks

Taken together, these analyses demonstrate that SPIDER-WEB not only fully releases the sequencing throughput of existing platforms but also unlocks the potential of nanopore sequencing platforms for non-blocking real-time data retrieval in DNA-based data storage. Within this regime, system latency is primarily governed by upstream data generation rather than post-sequencing computation, effectively shifting the dominant performance constraint towards sequencing or earlier experimental stages. Importantly, this alignment is achieved without requiring paradigm-level changes in sequencing technology. As readout throughput continues to improve incrementally, the computational design of SPIDER-WEB allows it to scale accordingly, preserving compatibility with evolving platform capabilities. These results establish a system-level integration of sequencing and reconstruction processes, supporting real-time operation while avoiding the accumulation of computational backlog.

15 Optimisation of data structures and algorithms

For graph-related calculations, the adjacency matrix is conventionally used to represent a digraph. For the window length k mentioned above, the shape of an adjacency matrix is $(4^k, 4^k)$. When k reaches 8, the file size will achieve 16.0 gigabytes with the integer format[70], which is unable to be allocated (both MATLAB and Python platforms). Considering that such an adjacency matrix is an extremely sparse matrix, only 4 positions in each row (4^k) can be actually used. Using a compressed matrix, such as compressed sparse row, can save memory space exponentially, but it brings trouble to massive matrix operations (e.g. singular value decomposition). It implies that the computing time of capacity approximation, coding algorithm generation, and graph-based search may be intolerable. Thus, we have completed some design and optimisation at the software level to make the creation and calculation of huge digraphs possible.

15.1 Index definition in the programming

Here we declare that the index in this section is defined as the zero-based numbering[71]. Therefore, the initial element of a vector (γ) is assigned the index 0, denote as $\gamma[0]$. Besides, the Numpy package[72] accepts negative indices for indexing from the end of the vector. For example, we have a vector $\gamma = (1, 2, 3, \cdots, 10)_{1\times 10}$, $\gamma[0] = 1$ and $\gamma[-1] = \gamma[9] = 10$.

15.2 Representation of digraph

Let A = 0, C = 1, G = 2, and T = 3, the index of a vertex in $\mathcal{V}_4^k$ can be defined as a decimal number:

$$\text{index}(\boldsymbol{u}) = \sum_{p=1}^{k} \boldsymbol{u}[p] \times 4^{k-p}.$$

Set $i = \text{index}(\boldsymbol{u})$, then $\boldsymbol{u}$ is i-th vertex of $\mathcal{V}_4^k$, also denote as $\mathcal{V}_4^k[i]$. Hence, based on the de Bruijn graph of order k, for any arc $(\mathcal{V}_4^k[i], \mathcal{V}_4^k[j])$, i and j must satisfy $i \bmod 4^{k-1} = \lfloor j/4 \rfloor$.

Practically, some arcs are trimmed during special processes. As the simplest design, a special compressed matrix with 4^k rows (for vertex indices) and 4 columns (for outgoing arcs) can be constructed. For this matrix, if the element of x-th row and y-th column is 1, $\mathcal{D}$ represented by this matrix contains an arc from x-th vertex to $(x \times 4 + y) \bmod 4^k$ vertex; otherwise, $\mathcal{D}$ lacks this arc. However, this work involves massive matrix, graph, and tree operations. The above matrix will perform a modular operation every time the vertices are accessed in turn. Hence, we further construct a special compressed matrix, named accessor (α), containing 4^k rows and 4 columns. If $(\mathcal{V}_4^k[i], \mathcal{V}_4^k[j])$ is an arc of $\mathcal{A}$, $\alpha[i, j \bmod 4] = j$. When accessing a location in the row, it is directly to obtain the follow-up row corresponding to the current row and column. Besides, the value of the remaining elements of α is set as -1. For any related operation, it will be ignored. In practice, we only need to allocate a vector of size $4^k + 1$, so that the operation of these irrelevant values takes place in the last position without affecting the calculation. After calculations, by setting the value at the last position to an invalid number, the expected vector will be obtained.

To illustrate the difference between adjacency matrix and accessor, we represent the digraph in Fig. 2a by the following matrices:

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}_{16\times 16}
\Longleftrightarrow
\begin{pmatrix}
-1&-1&-1&-1\\
4&-1&-1&7\\
8&-1&-1&11\\
-1&-1&-1&-1\\
-1&1&2&-1\\
-1&-1&-1&-1\\
-1&-1&-1&-1\\
-1&13&14&-1\\
-1&1&2&-1\\
-1&-1&-1&-1\\
-1&-1&-1&-1\\
-1&13&14&-1\\
-1&-1&-1&-1\\
4&-1&-1&7\\
8&-1&-1&11\\
-1&-1&-1&-1
\end{pmatrix}_{16\times 4}.$$

By doing so, the memory usage of a digraph in this work can be reduced from 4^{k+k} to 4^{k+1}. In our simulation experiments ($k = 10$), the allocated memory is decreased from 4.0 terabytes to 16.0 megabytes (262,144 times).

15.3 Use of rapid search in digraph

In the generation task of SPIDER-WEB, breadth-first search[73] is used widely. Meanwhile, in the sequence correcting task of SPIDER-WEB, a major repetitive operation is to detect whether the corrected DNA sequence is on the walk that satisfies established constraints. Thus, the optimisation of search operation in a digraph can significantly reduce the time required for not only coding algorithm generation but also sequence correcting.

Through accessor, the search operation can be converted to matrix operations, we therefore construct a rapid search for layers and walks. Here, we introduce stride s to represent the stride from the root vertex to the leaf vertex, then provide customised algorithms below. Normally, in the breadth-first search[74], the vertex acquisition of the next layer depends on the next vertex of each vertex in the current layer. Accessor considers parallelising the search of each layer, so as to speed up the search.

Taking i-th vertex ($\mathcal{V}_4^k[i]$) as the root vertex in the digraph $\mathcal{D}$, the initial vector can be $\gamma_0 = (0,0,\cdots,1,\cdots,0,0)_{1\times 4^k+1}$. In this vector, only i-th element is 1. For the next layer (level 1), we initialise γ_1 as an all-zero vector and fill the locations (represented by the value in $\alpha[i]$) of $\alpha[i]$ in 1. That is,

$$\gamma_1[\alpha[i,j]] = 1$$

where $j \in \{0,1,2,3\}$. Expansively, the s-layer vector can be defined through s iterations. In each iteration, all the elements in the vector are initialised as 0. From s-th iteration to $(s+1)$-th iteration,

$$\gamma_{s+1}[\alpha[i]] = \sum_{f=0}^{3} \gamma_s[f\times 4^{k-1} + \lfloor \frac{i}{4} \rfloor],$$

where j is each location with the value of 1 in γ_s. Using the "where" function in the Numpy package, the search process can be executed in parallel (from 4^k iterations to 4 iterations).

15.4 Design of accelerator for data retrieval

The accelerator for sequence correcting is based on a multi-threaded framework, which adopts a two-tier parallel architecture, utilising three main threads to establish a pipelined workflow for data processing. Each main thread cyclically executes three stages: data reading, data parallel processing, and data writing. Before performing a stage, each main thread checks whether another thread is already executing the same stage. If a specific stage is occupied, the thread selects another available stage, ensuring exclusive access to each stage.

The specific workflow of the three steps is as follows:

1. **Data reading process**: The main thread checks the read buffer status. If the buffer is idle and no other thread is reading, it loads fixed-size sequences from input files. After loading, the file handle is immediately released to reclaim resources and prevent prolonged file occupation.

2. **Data parallel processing**: When the read buffer contains data, the main thread spawns multiple sub-threads, each independently assigned a sequence error correction task. Sub-threads fetch data from the read buffer, process it, and store results in the result buffer. To enhance efficiency, a task-stealing strategy is implemented: idle sub-threads can retrieve pending tasks from the task queue of other threads, achieving load balancing.

3. **Data writing process**: The main thread checks the result buffer. If data is ready and no other thread is writing, it sequentially writes data to the output files. Upon completion, the next data reading cycle is triggered to form a closed-loop pipeline.

This design achieves efficient resource management through dynamism and decoupling. Decoupled read and result buffers enable efficient coordination between stages, while batched data loading and buffer constraints eliminate memory overflow risks. Dynamic separation of I/O and computation ensures disk latency does not degrade processing efficiency, with at least one main thread consistently performing computation in the data parallel processing to prevent CPU idling. Sub-thread counts adapt to hardware resources, supporting scalability from single machines to distributed environments.

References

1. Shastri, V., Rangan, P. V., Rajaraman, V., Pittet, A. & Kumar, S. S. Performance issues in CD-ROM-based storage systems for multimedia. In *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, 618–621 (IEEE, 1996).
2. Erlich, Y. & Zielinski, D. DNA fountain enables a robust and efficient storage architecture. *Science* **355**, 950–954 (2017).
3. Zhao, X. *et al.* Composite HEDGES nanopores codec system for rapid and portable DNA data readout with high INDEL-correction. *Nature Communications* **15**, 9395 (2024).
4. Bar-Lev, D., Orr, I., Sabary, O., Etzion, T. & Yaakobi, E. Scalable and robust DNA-based storage via coding theory and deep learning. *Nature Machine Intelligence* 1–11 (2025).
5. Zhang, Y., Qin, R., Ge, Q., Guo, Q. & Chen, W. From spacecraft ranging to massive dna data storage: Composite ranging codes as indices and error correction references. *Science Advances* **12**, eaec1469 (2026).
6. Perlmutter, M. The lost picture show. *IEEE Spectrum* **54**, 26–31 (2017).
7. Church, G. M., Gao, Y. & Kosuri, S. Next-generation digital information storage in DNA. *Science* **337**, 1628–1628 (2012).
8. Goldman, N. *et al.* Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature* **494**, 77 (2013).
9. 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 Edition* **54**, 2552–2555 (2015).
10. Blawat, M. *et al.* Forward error correction for DNA data storage. *Procedia Computer Science* **80**, 1011–1022 (2016).
11. Gabrys, R., Kiah, H. M., Vardy, A., Yaakobi, E. & Zhang, Y. Locally balanced constraints. In *2020 IEEE International Symposium on Information Theory (ISIT)*, 664–669. IEEE (IEEE, Los Angeles, CA, USA, 2020).
12. Judo, M. S., Wedel, A. B. & Wilson, C. Stimulation and suppression of pcr-mediated recombination. *Nucleic Acids Research* **26**, 1819–1825 (1998).
13. Clarke, L., Rebelo, C., Goncalves, J., Boavida, M. & Jordan, P. PCR amplification introduces errors into mononucleotide and dinucleotide repeat sequences. *Molecular Pathology* **54**, 351 (2001).
14. Hommelsheim, C. M., Frantzeskakis, L., Huang, M. & Ülker, B. PCR amplification of repetitive DNA: a limitation to genome editing technologies and many other applications. *Scientific Reports* **4**, 5052 (2014).
15. Brownie, J. *et al.* The elimination of primer-dimer accumulation in PCR. *Nucleic Acids Research* **25**, 3235–3241 (1997).
16. Organick, L. *et al.* Random access in large-scale DNA data storage. *Nature Biotechnology* **36**, 242–248 (2018).
17. Bee, C. *et al.* Molecular-level similarity search brings computing to dna data storage. *Nature Communications* **12**, 1–9 (2021).
18. Lin, K. N., Volkel, K., Tuck, J. M. & Keung, A. J. Dynamic and scalable DNA-based information storage. *Nature Communications* **11**, 1–12 (2020).
19. Hamming, R. W. Error detecting and error correcting codes. *The Bell System Technical Journal* **29**, 147–160 (1950).
20. Levenshtein, V. I. Efficient reconstruction of sequences. *IEEE Transactions on Information Theory* **47**, 2–22 (2001).
21. Ping, Z. *et al.* Towards practical and robust DNA-based data archiving using the yin-yang codec system. *Nature Computational Science* **2**, 234–242 (2022).
22. Bollobás, B. *Modern graph theory*, vol. 184 (Springer Science & Business Media, New York, 2013).
23. 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* **117**, 18489–18496 (2020).
24. Hart, P. E., Nilsson, N. J. & Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* **4**, 100–107 (1968).
25. Reed, I. S. & Solomon, G. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics* **8**, 300–304 (1960).
26. Anavy, L., Vaknin, I., Atar, O., Amit, R. & Yakhini, Z. Data storage in DNA with fewer synthesis cycles using composite DNA letters. *Nature Biotechnology* **37**, 1229–1236 (2019).
27. Minoche, A. E., Dohm, J. C. & Himmelbauer, H. Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and genome analyzer systems. *Genome Biology* **12**, 1–15 (2011).
28. Shafir, R., Sabary, O., Anavy, L., Yaakobi, E. & Yakhini, Z. Sequence reconstruction under stutter noise in enzymatic DNA synthesis. In *2021 IEEE Information Theory Workshop*, 1–6 (IEEE, 2021).
29. Benita, Y., Oosting, R. S., Lok, M. C., Wise, M. J. & Humphery-Smith, I. Regionalized GC content of template DNA as a predictor of PCR success. *Nucleic Acids Research* **31**, e99–e99 (2003).
30. Nakamura, K. *et al.* Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Research* **39**, e90–e90 (2011).
31. Meacham, F. *et al.* Identification and correction of systematic error in high-throughput sequence data. *BMC Bioinformatics* **12**, 1–11 (2011).
32. Wick, R. R., Judd, L. M. & Holt, K. E. Performance of neural network basecalling tools for Oxford nanopore sequencing. *Genome Biology* **20**, 129 (2019).
33. Abu-Sini, M. & Yaakobi, E. On Levenshtein's reconstruction problem under insertions, deletions, and substitutions. *IEEE Transactions on Information Theory* **67**, 7132–7158 (2021).
34. Chrisnata, J., Kiah, H. M. & Yaakobi, E. Correcting deletions with multiple reads. *IEEE Transactions on Information Theory* **68**, 7141–7158 (2022).
35. Wang, C., Yaakobi, E. & Zhang, Y. How to find simple conditions for successful sequence reconstruction? In *2024 IEEE Information Theory Workshop (ITW)*, 627–632 (IEEE, 2024).
36. Papadopoulou, V., Rameshwar, V. A. & Wachter-Zeh, A. On the expected number of views required for fixed-error sequence reconstruction. In *2024 IEEE Information Theory Workshop*, 639–644 (IEEE, 2024).
37. Varšamov, R. & Tenengolts, G. A code which corrects single asymmetric errors. *Avtomat. i Telemeh* **26**, 4 (1965).
38. Guruswami, V. & Håstad, J. Explicit two-deletion codes with redundancy matching the existential bound. *IEEE Transactions on Information Theory* **67**, 6384–6394 (2021).
39. Leadbetter, M. R., Lindgren, G. & Rootzén, H. *Extremes and related properties of random sequences and processes* (Springer Science & Business Media, 2012).
40. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* **17**, 261–272 (2020).
41. Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics* **2**, 164–168 (1944).
42. Marquardt, D. W. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics* **11**, 431–441 (1963).
43. Chen, Y.-J. *et al.* Quantifying molecular bias in DNA data storage. *Nature Communications* **11**, 1–9 (2020).
44. Matange, K., Tuck, J. M. & Keung, A. J. DNA stability: a central design consideration for DNA data storage systems. *Nature Communications* **12**, 1358 (2021).
45. Gimpel, A. L., Stark, W. J., Heckel, R. & Grass, R. N. A digital twin for DNA data storage based on comprehensive quantification of errors and biases. *Nature Communications* **14**, 6026 (2023).
46. Song, L. *et al.* Robust data storage in DNA by de Bruijn graph-based de novo strand assembly. *Nature Communications* **13**, 1–9 (2022).
47. Meiser, L. C. *et al.* Reading and writing digital data in DNA. *Nature Protocols* **15**, 86–101 (2020).
48. Ewing, B., Hillier, L., Wendl, M. C. & Green, P. Base-calling of automated sequencer traces usingphred. i. accuracy assessment. *Genome Research* **8**, 175–185 (1998).
49. Ewing, B. & Green, P. Base-calling of automated sequencer traces using phred. ii. error probabilities. *Genome Research* **8**, 186–194 (1998).
50. Bornholt, J. *et al.* Toward a dna-based archival storage system. *IEEE Micro* **37**, 98–104 (2017).
51. Zan, X. *et al.* A hierarchical error correction strategy for text dna storage. *Interdisciplinary Sciences: Computational Life Sciences* **14**, 141–150 (2022).

52. Zhang, C. *et al.* Parallel molecular data storage by printing epigenetic bits on DNA. *Nature* **634**, 824–832 (2024).

53. Pan, C. *et al.* Rewritable two-dimensional dna-based data storage with machine learning reconstruction. *Nature Communications* **13**, 2984 (2022).

54. Ruan, C. *et al.* Robust dna image storage decoding with residual cnn. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5 (IEEE, 2024).

55. Zheng, X. *et al.* A generative adversarial network for multiple reads reconstruction in dna storage. *Scientific Reports* **14**, 32071 (2024).

56. Weng, Z. *et al.* Massively parallel homogeneous amplification of chip-scale dna for dna information storage (mphac-dis). *Nature Communications* **16**, 667 (2025).

57. Qu, G., Yan, Z., Chen, X. & Wu, H. Dna data storage for biomedical images using helix. *Nature Computational Science* 1–8 (2025).

58. Tomek, K. J. *et al.* Driving the scalability of DNA-based information storage systems. *ACS Synthetic Biology* **8**, 1241–1248 (2019).

59. Antkowiak, P. L. *et al.* Low cost DNA data storage using photolithographic synthesis and advanced information reconstruction and error correction. *Nature Communications* **11**, 5345 (2020).

60. Koch, J. *et al.* A DNA-of-things storage architecture to create materials with embedded memory. *Nature Biotechnology* **38**, 39–43 (2020).

61. Welzel, M. *et al.* DNA-Aeon provides flexible arithmetic coding for constraint adherence and error correction in DNA storage. *Nature Communications* **14**, 628 (2023).

62. Qu, G., Yan, Z. & Wu, H. Clover: tree structure-based efficient DNA clustering for DNA-based data storage. *Briefings in Bioinformatics* (2022).

63. Nguyen, M. T., Gobry, M. V., Sampedro Vallina, N., Pothoulakis, G. & Andersen, E. S. Enzymatic assembly of small synthetic genes with repetitive elements. *ACS Synthetic Biology* **13**, 963–968 (2024).

64. Chen, Z. *et al.* Controlled movement of ssDNA conjugated peptide through mycobacterium smegmatis porin A (MspA) nanopore by a helicase motor for peptide sequencing application. *Chemical Science* **12**, 15750–15756 (2021).

65. Wang, Z. *et al.* The precise basecalling of short-read nanopore sequencing. *bioRxiv* (2024).

66. Flajolet, P., Grabner, P., Kirschenhofer, P., Prodinger, H. & Tichy, R. F. Mellin transforms and asymptotics: digital sums. *Theoretical Computer Science* **123**, 291–314 (1994).

67. Atkins, P. W., De Paula, J. & Keeler, J. *Atkins' physical chemistry* (Oxford university press, 2023).

68. Marx, V. Method of the year: long-read sequencing. *Nature Methods* **20**, 6–11 (2023).

69. Kantorovich, L. V. Mathematical methods of organizing and planning production. *Management Science* **6**, 366–422 (1960).

70. Zuras, D. *et al.* Ieee standard for floating-point arithmetic. *IEEE Std* **754**, 1–70 (2008).

71. Seed, G. M. *An introduction to object-oriented programming in C++: with applications in computer graphics* (Springer Science & Business Media, 2012).

72. Van Der Walt, S., Colbert, S. C. & Varoquaux, G. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* **13**, 22–30 (2011).

73. Moore, E. F. The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching.*, 285–292. Harvard University (Harvard University Press, USA, 1959).

74. Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. *Introduction to algorithms* (MIT press, Cambridge, Massachusetts, USA, 2009).