EURECOM
Sophia Antipolis

SORBONNE UNIVERSITÉ

A Dissertation

In Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy from Sorbonne University

Scalable and Accurate Algorithms for Computational Genomics and DNA-based Digital Storage

by

Yiqing YAN

Co-supervisor:
Raja Appuswamy
EURECOM, Sophia Antipolis, France

Thesis Director:
Paolo Papotti
EURECOM, Sophia Antipolis, France

Data Science Department
Eurecom/Sorbonne University, France
February, 2023

Acknowledgements

First of all, I would like to thank Prof. Raja Appuswamy for hosting me in his laboratory during these three years which have been enriching my eyes and broadening my views. I really appreciate the opportunity to participate in the adventure of OligoArchive and MoSS projects. Thanks for his professional suggestions and kind encouragement. The rigorous academic attitude and persistent enthusiasm for learning will continue to influence me in my career.

Later on, I would like to thank all the collaborators, specially Dr. Nimisha Chaturvedi, Dr. Giulio Franzese, Dr. Eddy Ghabach and Eugenio Marinelli. It is really pleasant to cooperate with them and I appreciate all their feedback. I would also like to thank Nimesh Pinnamaneni and Sachin Chalapati, the CEO and CTO from Helixworks, for their solid support of DNA synthesis and sequencing work. Then, an acknowledgement goes to my reviewers, Prof. Pascal Barbry at Institute Pharmacology Moléculaire Et Cellulaire (IPMC) and Prof. Marc Antonini at Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis (I3S). I am honoured to have them as my reviewers and thanks a lot for all the positive advice and criticism that help me to continue learning.

A big thank to all of my colleagues at EURECOM who made the work environment so pleasant, and an extra special thank to those who have become my friends here in France.

Furthermore, I would also like to thank my parents and my parents-in-law for having always supported me, especially during the hard COVID-19 period. Last but not least, I am indebted to my husband Guiming and my son Kellen for being always staying close with me.

Sophia Antipolis, February 2023 ***Yiqing YAN***

Abstract

Cost reduction and throughput improvement in sequencing technology have resulted in new advances in applications such as precision medicine and DNA-based storage. However, the sequenced result contains errors. To measure the similarity between the sequenced result and reference, edit distance is preferred in practice over Hamming distance due to the indels. The primitive edit distance calculation is quadratic complex. Therefore, sequence similarity analysis is computationally intensive. In this thesis, we introduce two accurate and scalable sequence similarity analysis algorithms, i) Accel-Align, a fast sequence mapper and aligner based on the seed–embed–extend methodology, and ii) Motif-Search, an efficient structure-aware algorithm to recover the information encoded by the composite motifs from the DNA archive. Then, we use Accel-Align as an efficient tool to study the random access design in DNA-based storage.

In computational genomics, sequence alignment, the process to determine the reads locations in the reference genome, has traditionally been a computationally intensive task. Modern sequence alignment algorithms use the seed–filter–extend (SFE) methodology and rely on filtration heuristics to reduce the overhead of edit distance computation. However, filtering makes assumptions about error patterns, has inherent performance–accuracy trade-offs, and requires careful manual tuning to match the dataset. Motivated by algorithmic advances in randomized low-distortion embedding, we introduce seed–embed–extend (SEE), a novel methodology for developing sequence mappers and aligners. While SFE focuses on eliminating sub-optimal candidates, SEE focuses instead on identifying optimal candidates. To achieve this, SEE transforms the read and reference strings from edit distance regime to the Hamming regime by embedding them using a randomized algorithm, and uses Hamming distance over the embedded set to identify optimal candidates. To show that SEE performs well in practice, we present Accel-Align, an SEE-based short-read sequence mapper and aligner that is 2-12× faster than the state-of-the-art aligners on commodity CPUs, without any special-purpose hardware, while providing comparable accuracy.

In DNA-based storage, recovering the stored information is also a computationally intensive task. In this work, we propose an efficient approximate-edit-similarity-based decoding and consensus algorithm – Motif-Search. It is a structure-aware decoder customized for our end-to-end DNA-based storage workflow. In this workflow, the input binary data is encoded by the composite motifs framework to scale logical density (bits written per cycle) which results in

potential reduction of the synthesis cost. Afterwards, the oligos are synthesized by our new enzymatic motif ligation techniques that can scale DNA synthesis in the DNA write pipeline, and are sequenced by our assembly-free, Nanopore-based motif read out that can scale DNA sequencing in the DNA read pipeline. Using the proposed methods, we present an end-to-end experiment where we store the text “HelloWorld” at a logical density of 84 bits/cycle (14–42× improvement over state-of-the-art work). Moreover, Motif-Search is able to fully recover the data from the error-prone reads with a low sequencing coverage of 20×.

To study the random access in DNA-based storage, we use Accel-Align as an efficient tool to determine the corresponding reference payload and primer of each read. Based on that, we find that file-based random access can suffer from the polymerase chain reaction (PCR) bias issue if file sizes are not uniform and improper binding can occur during PCR for the short primers. To solve these issues, we propose the block-based random access methodology in which each block has a uniform size. We elaborately design 24 left primers and 26 right primers of 20-mers and select 4 from each of them respectively as the validated primers for the trial of the TPCH database storage. It shows that the block-based design eliminates the PCR bias and improves the proportion of the reads with correct primers. As a result, more than 99.9% reads are correctly distributed by the decoder for each primer pair.

Abrégé

La réduction des coûts et l'amélioration du débit de la technologie de séquençage ont entraîné de nouvelles avancées dans des applications telles que la médecine de précision et le stockage basé sur l'ADN. Cependant, le résultat séquencé contient des erreurs. Pour mesurer la similitude entre le résultat séquencé et la référence, la distance d'édition est préférée en pratique à la distance de Hamming en raison des indels. Le calcul de la distance d'édition est complexe quadratique. Par conséquent, l'analyse de similarité de séquence nécessite beaucoup de calculs. Dans cette thèse, nous introduisons deux algorithmes d'analyse de similarité de séquence précis et évolutifs, i) Accel-Align, un mappeur et un aligneur de séquence rapide basé sur la méthodologie seed–embed–extend, et ii) Motif-Search, une structure-efficace algorithme conscient pour récupérer les informations codées par les motifs composites à partir de l'archive ADN. Ensuite, nous utilisons Accel-Align comme un outil efficace pour étudier la conception d'accès aléatoire dans le stockage basé sur l'ADN.

En génomique computationnelle, l'alignement de séquences, le processus permettant de déterminer les emplacements de lecture dans le génome de référence, a traditionnellement été une tâche informatique intensive. Les algorithmes d'alignement de séquences modernes utilisent la méthodologie seed–filter-extend (SFE) et s'appuient sur l'heuristique de filtrage pour réduire la surcharge du calcul de la distance d'édition. Cependant, le filtrage fait des hypothèses sur les modèles d'erreurs, présente des compromis performances-précision inhérents et nécessite un réglage manuel minutieux pour correspondre à l'ensemble de données. Motivés par les avancées algorithmiques dans l'intégration aléatoire à faible distorsion, nous introduisons seed–embed–extend (SEE), une nouvelle méthodologie pour développer des mappeurs et des aligneurs de séquences. Alors que SFE se concentre sur l'élimination des candidats sous-optimaux, SEE se concentre plutôt sur l'identification des candidats optimaux. Pour ce faire, SEE transforme les chaînes de lecture et de référence du régime de distance d'édition au régime de Hamming en les intégrant à l'aide d'un algorithme randomisé, et utilise la distance de Hamming sur l'ensemble intégré pour identifier les candidats optimaux. Pour montrer que SEE fonctionne bien dans la pratique, nous présentons Accel-Align, un mappeur et un aligneur de séquences à lecture courte basé sur SEE qui est 2 à 12× plus rapide que les aligneurs de pointe sur les processeurs de base, sans tout matériel spécialisé, tout en offrant une précision comparable.

Dans le stockage basé sur l'ADN, la récupération des informations stockées est également une

tâche de calcul intensif. Dans ce projet, nous proposons un algorithme efficace de décodage et de consensus basé sur la similarité d'édition approximative – Motif-Search. Il s'agit d'un décodeur sensible à la structure personnalisé pour notre flux de travail de stockage basé sur l'ADN de bout en bout. Dans ce flux de travail, les données binaires d'entrée sont codées par le cadre de motifs composites pour mettre à l'échelle la densité logique (bits écrits par cycle), ce qui entraîne une réduction potentielle du coût de synthèse. Ensuite, les oligos sont synthétisés par nos nouvelles techniques de ligature de motifs enzymatiques qui peuvent faire évoluer la synthèse d'ADN dans le pipeline d'écriture d'ADN, et sont séquencés par notre lecture de motif sans assemblage, basée sur Nanopore, qui peut faire évoluer le séquençage d'ADN dans le pipeline de lecture d'ADN. En utilisant les méthodes proposées, nous présentons une expérience de bout en bout où nous stockons le texte "HelloWorld" à une densité logique de 84 bits/cycle (amélioration de 14 à 42 × par rapport à l'état de l'art) . De plus, Motif-Search est capable de récupérer entièrement les données des lectures sujettes aux erreurs avec une faible couverture de séquençage de 20×.

Pour étudier l'accès aléatoire dans le stockage basé sur l'ADN, nous utilisons Accel-Align comme un outil efficace pour déterminer la charge utile de référence correspondante et l'amorce de chaque lecture. Sur cette base, nous constatons que l'accès aléatoire basé sur les fichiers peut souffrir du problème de biais de la réaction en chaîne par polymérase (PCR) si les tailles de fichiers ne sont pas uniformes et qu'une liaison incorrecte peut se produire pendant la PCR pour les amorces courtes. Pour résoudre ces problèmes, nous proposons la méthodologie d'accès aléatoire basée sur les blocs dans laquelle chaque bloc a une taille uniforme. Nous concevons minutieusement 24 amorces gauches et 26 amorces droites de 20-mers et sélectionnons 4 de chacune d'entre elles respectivement comme amorces validées pour l'essai du stockage de la base de données TPCH. Il montre que la conception basée sur les blocs élimine le biais de PCR et améliore la proportion de lectures avec des amorces correctes. En conséquence, plus de 99,9% des lectures sont correctement distribuées par le décodeur pour chaque paire d'amorces.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Sequencing is the process to determine the sequence of the nucleic acids. In 1977, British biochemist Frederick Sanger invented the chain termination sequencing method, marking the birth of the first generation of DNA sequencing technology [2]. Over the past few decades, many other sequencing technologies have been developed. They provide a new way to address biological and medical problems, such as gene expression profile analysis, chromosome counting, epigenetic change detection, and molecular analysis.

Recent research shows that the genomics data would far exceed the growth of the data generated by the other three major generators of Big Data: astronomy, YouTube, and Twitter in the year 2025 [3]. Sequencing is usually considered as the first and fundamental step in genomics data analysis. Throughput and price are the two main important factors in the area of massive genomics data analysis. Today, we can rapidly sequence millions of DNA simultaneously. It requires less than 24 hours to sequence the whole genome of a person [4]. As tracked by National Human Genome Research Institute (NHGRI) [5], the sequencing cost follows Moore's Law which halves every two years as shown in Figure 1.1. For instance, the sequencing-cost-per-genome has dropped from over a billion dollars to a mere $1000 [4] in the past decades. Thanks to the price reduction and throughput improvement, sequencing brings a promising future for novel applications, such as precision medicine [6] and DNA-based digital storage [7, 8].

Although modern sequencing technologies do provide ultra-high throughput with lower cost, they do not have correctness guarantee. Sequencing errors such as substitutions, insertions, and deletions can lead to the difference between the sequenced object and the result. In the meantime, the sequenced object might be biased against the original reference. Thus, sequence analysis is an approximate string similarity analysis tolerating errors and bias. Several metrics have been widely used to describe the similarity between sequences, such as

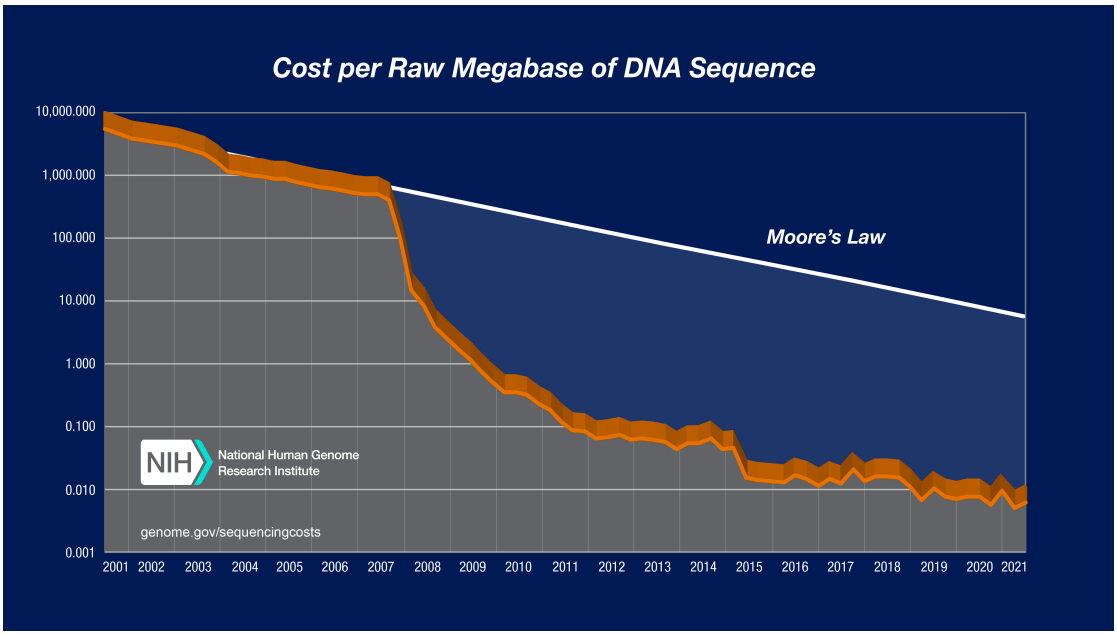


Figure 1.1: Sequencing cost per megabase - 2021 from NHGRI.

Hamming distance, edit distance (Levenshtein distance), longest common subsequence (LCS) distance, Damerau-Levenshtein distance and Jaro distance. Edit distance is the most common metric to measure the DNA sequence similarity. It is even expanded as a parameterizable metric where each operation is assigned a cost in practise. However, the primary edit distance computation is quadratic complex. Thus, the dynamic programming based algorithms, such as Smith-Waterman algorithm [9], Needleman-Wunsch algorithm [10], are not scalable for extremely large genomic dataset involving billions of such computations.

Therefore, it has been essential and indispensable to design accurate and scalable sequence similarly analysis algorithms for various computational genomics and DNA-based digital storage applications in the biological and informatics research.

1.2 Related Terms

1.2.1 DNA

DNA, or deoxyribonucleic acid, is the genetic material of most living organisms. It is a macromolecule composed of structural building blocks called nucleotides. There are four types of nucleotides, namely, Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). They function as the fundamental units of the genetic code. The quantity and order of nucleotides represent the distinct genetic information of each organism. Although the DNA sequences differ among different species, the DNA sequences among different individuals of the same species are

very similar. For instance, Escherichia coli has only around 4.7 million base pairs in its DNA, while Human DNA contains about 3 billion base pairs. The previous work found that no more than 0.5% of bases are different between any two persons. Hence, a common DNA sequence reference would be used to represent the human DNA sequence.

Apart from the DNA existing in nature, it is also possible to synthesize artificially designed DNA fragments through oligonucleotides (oligo) assembly. The first synthesized oligo came out in the 1950s by Michelson and Todd [11]. Later on, phosphoramidite-based synthesis [12] and enzymatic synthesis [13, 14, 15] have been invented. However, due to the decrease in reaction efficiency, synthetic purity and yield with DNA chain extension, the synthesized oligo length is limited to hundreds of bases.

Wherever the DNA sequence originates, it can be presented as a string of characters drawn from an alphabet $\Sigma = \{A, C, G, T\}$ computationally. Consequently, DNA sequence related problems can be interpreted as string manipulation or approximate matching problems in the Computer Science domain.

1.2.2 Reads

Sequencing technologies are capable of producing reads of hundreds of base pairs, or hundreds of thousands of base pairs. However, these are still too short compared to the genome, such as the human genome which contains 3 billion base pairs. Current sequencing technologies cannot fully sequence DNA sequences at once. So they break DNA into smaller fragments before sequencing each fragment. The inferred sequences corresponding to the DNA fragments are called *reads*. Due to the sequencing errors, and the bias between the fragments and references such as the variants in genomics or the synthesis errors in DNA-based storage, the reads are not exactly the same as the references. They might contain substitutions, insertions and deletions.

There are two types of reads according to the sequencing modes as shown in Figure 1.2.

- *Single-end reads*. In single-end sequencing, the sequencer reads a fragment from only one end to the other, generating the sequence of base pairs.
- *Paired-end reads*. The paired-end sequencing starts at one strand, finishes this direction at the specified read length. Then, it starts another round of reading from the opposite end of the fragment.

Paired-end sequencing doubles the read size compared to single-end sequencing. It improves the capability of identifying the relative positions of various reads in the genome. It is substantially more effective than single-end sequencing in resolving structural rearrangements such

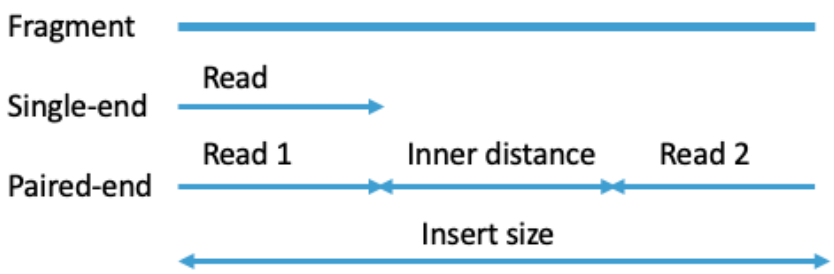


Figure 1.2: Single-end and paired-end reads.

as gene insertions, deletions, or inversions. It also improves the assembly of repetitive regions. However, paired-end sequencing is more expensive and time consuming. Not all experiments require this level of precision.

1.2.3 Sequencing Technologies

DNA sequencing is the process to determine the sequence of nucleotides.

The First-generation Sequencing Technology

Walter Fiers first sequenced the DNA of a complete gene in 1972 [16]. In parallel to Fiers' achievement, Fredrick Sanger kept working on an alternative DNA sequencing method and developed the first DNA sequencing method called "chain termination method" in 1977 [2]. This technique is capable of generating reads of approximately 1000 bp [17] at an extremely low sequencing error rate ranging from 0.001% to 0.01% [18]. Although Sanger's method has low throughput and high sequence cost, it is still used as the golden-standard nowadays due to its high accuracy.

The Second-generation Sequencing Technology

The second-generation sequencing technology, or next-generation sequencing (NGS), is known for its high throughput. It pioneered the introduction of sequencing-by-synthesis which enables to capture newly added bases which carry a special marker to determine the DNA sequence during DNA replication. It has three important features:

- High throughput. The next-generation sequencing can sequence millions of DNA molecules in parallel.
- Short read length. When the read length increases during the sequencing process, the

sequencing quality drops. The read length of the next-generation sequencing does not exceed 500 bp.

- Low error rate. The next-generation sequencing error rate is less than 1% [19, 20].

The Third-generation Sequencing Technology

The third-generation sequencing technology was emerged from a proof-of-principle instrumentation concept published by Watt Webb's group in Cornell University in 2003 [21]. Compared with the previous two generations, its biggest feature is single-molecule sequencing (SMS), which does not require PCR amplification during the sequencing process. Two SMS devices have achieved commercial status, coming from Pacific Biosciences (PacBio) and Oxford Nanopore Technologies, respectively:

- The Single Molecule Real Time (SMRT) technique from PacBio generates reads of 1000~100,000 bp in length. The reads contain up to 20% sequencing errors, including mostly insertions, deletions and some substitutions [22, 23]. The more recent PacBio instruments can generate 10 kbp to 25 kbp high-fidelity (HiFi) reads at an error rate ~1% [24].
- Oxford Nanopore Technology produces reads of hundreds of kbp with up to 15% error rate [25].

1.3 Sequence Similarity Analysis in Computational Genomics

1.3.1 Reference

For computational genomics problems, reference is a long assembly DNA sequence used as a standard for comparison in basic research and clinic settings. A reference genome is a database of nucleic acid sequences. It is equivalent to one or several long strings. As an instance, the human reference genome contains over 3 billion base pairs as 22 pairs of autosomes and 1 pair of sex chromosome. It is regarded as 24 long strings of 3 billion characters in total. Although one person's genome differs from another, more than 99% bases are the same among humans. The reference genome contains representative information of the population rather than specific information of an individual. Hence, the reference genome can effectively reflect the DNA information of the species.

Reference genome plays an important role in biological analysis. Since the reads generated by next-generation sequencing are very short, they need to be ordered and assembled appropriately to represent the sequenced genome before they can be used for further analysis. If the reference genome is known, the short reads can be directly aligned to the reference genome

to know their actual position on the genome. Otherwise, it is a computationally expensive problem to recover the complete DNA sequence from a large number of short reads.

1.3.2 Sequence Alignment Problem

In computational genomics, the sequence similarity problem that we work on is the sequence alignment algorithm. Sequence alignment algorithm aims to locate the origin of each read in the reference genome. It is defined as below.

Definition 1 *Sequence alignment: Given the reference genome R and read set Q, sequence alignment is to find out the approximate matching positions for each read* $q \in Q$ *in the reference genome R.*

In another word, sequence alignment is a process of looking up the query q in the target sequence R. A sketch of sequence alignment is shown in Figure 1.3. As explained in Sec.1.2.3, the sequenced reads are hundreds to hundreds of thousand bp in length carrying 1% to 20% substitutions, insertions and deletions depending on the sequencing technology. From the perspective of the sequencing errors and variants, sequence alignment can be perceived as an approximate string matching problem. Edit distance is the most common metric to measure the sequence similarity. However, the primary edit distance computation is quadratic complex based on the dynamic programming. To speed up, the mainstream sequence alignment algorithms adopt the seed-filter-extend (SFE) strategy [26, 27] instead of the traditional dynamic programming algorithm. The idea is to perform dynamic programming calculations only on the regions with high likelihood, rather than on the entire long genome. It first extracts some short substrings (seeds) from the sequenced fragments (reads), and then quickly searches these seeds in the pre-established genome index. The positions found by the lookup are called candidate positions. Finally, it performs dynamic programming (extension) on the regions near these candidate positions on the genome against the sequenced fragments to determine the final matching positions. This strategy can significantly speed up the sequence alignment process. Many famous aligners have been implemented based on it. However, we found that despite the fact that modern gapped read aligners (e.g. BWA-MEM [28], Bowtie2 [29] Minimap2 [30], SNAP [31] and HISAT2 [32])) can map thousands of reads to a reference genome per second, the sheer size of modern short-read sequencing dataset often makes sequence alignment one of the most time consuming steps in genomic data analysis.

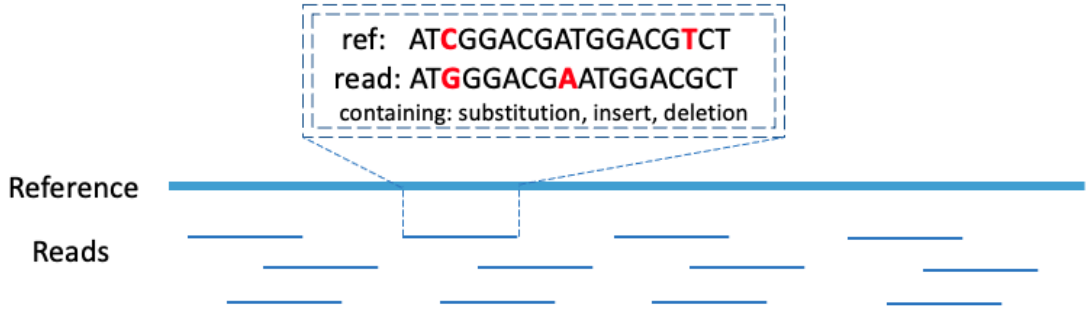


Figure 1.3: Sequence alignment in computational genomics.

1.4 Sequence Similarity Analysis in DNA-based Storage

1.4.1 Overview about DNA-based Storage

As reported by International Data Corporation (IDC), more than 90% of the data is generated in recent years and the "digital universe" is forecast to grow to 163 zettabytes by 2025, which is 10 times of that in 2016. The explosion in the rate of data generation has also led to an explosion in the storage requirement. According to the estimation, it will require more than 1000 kg of wafer-level silicon to store all global data, while the real supply would be only 108 kg in 2040. Thus, the traditional silicon-based storage media such as HDD and tapes can hardly meet the requirement. DNA becomes a promising candidate for “cold” data archive thanks to its high density, long duration and eternal relevance [33].

The first DNA-based digital storage was demonstrated in 1988 by Joe Davis [34]. They stored 35 bits of data in Escherichia coli DNA. DNA-based storage usually has four components, namely encoding, synthesis, sequencing and decoding. A template of data writing and reading procedures is shown in Figure 1.4.

- To *write* the data, the binary input data is encoded into a sequence represented by adenine (A), guanine (G), cytosine (C) and thymine (T) based on the encoding algorithm, such as the Goldman approach [7] and DNA Fountain [35]. Secondly, the designed sequence is synthesized into the oligos. Due to the limit of synthesis technology, the synthesized oligo length can be no more than a few hundreds nucleotides. Thus, data is converted into a set of short oligos instead of a single long oligo. Finally, the synthesised oligos are stored, for example in tube.
- To *read* the data, the stored oligos are extracted and sequenced by the sequencing technologies. Then, this sequenced result that is usually error-prone, is passed into the decoder to recover the original data. The corresponding decoding algorithms should be elaborately designed by taking into account the characteristics of encoding, synthesis

and sequencing.

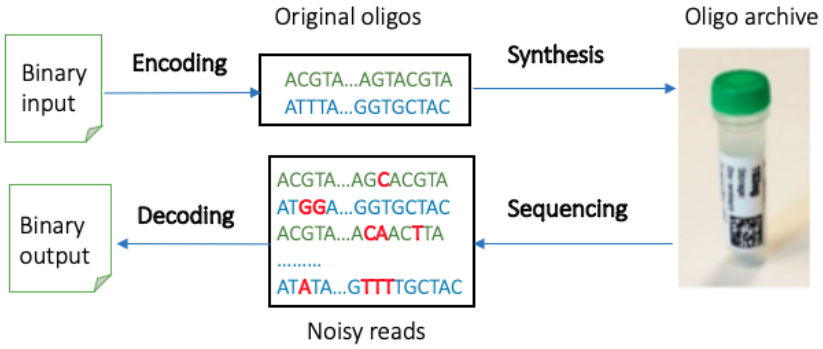


Figure 1.4: A template of DNA-based storage workflow.

1.4.2 Sequence Similarity Problems in DNA-based Storage

Sequence Alignment

Due to the errors introduced during synthesis and sequencing procedures, the reads may differ from original encoded oligos. The difference could be large depending on the synthesis and sequencing methodologies. Suppose we have the knowledge of the original encoded oligos, we need to answer the following questions: i) what is the coverage pattern across oligos, ii) how much is the error rate, and iii) what is the error pattern. This information can help to optimize sequencing and synthesis protocols at the "hardware" level, and consensus calling and decoding methods at the "software" level. Therefore, the sequence alignment algorithm introduced in Sec.1.3 could be also applied in DNA-based storage to compare the reads with the original encoded oligos, in order to understand the characteristics and statistics of the errors introduced during the process of synthesis and sequencing. Compared with the sequence alignment in genomics, there are several differences.

- Because of the synthesis limitation, the oligo could be no more than several hundred base pairs. Therefore, the read length is close to the reference length and each oligo could be fully sequenced at once. On the contrary, the read length in genomics data is much shorter compared to the reference length, thus sequencing cannot be performed at once.
- In DNA-based digital storage, we only need to find the oligo which each read maps to while the traditional computational genomics expects to have the exact aligned position. Thus, these algorithms could tolerate lower position-wise accuracy to achieve better performance.

- It is not always necessary to report the CIGAR field for DNA-based digital storage. The CIGAR field is a sequence representing the base lengths, and the associated operations that are used to indicate whether each base in the read is a match/mismatch/insertion/deletion. On the contrary, traditional computational genomics requires CIGAR for downstream variant calling to indicate variants. Consequently, it is not mandatory to implement the accurate dynamic programming extension in the DNA-based storage.

Consensus Calling and Decoding

Both synthesis and sequencing are approximate in nature and prone to errors. Hence, the sequenced result retrieved back from the DNA archives are noisy copies of the original sequences. The decoding algorithm is also an approximate edit similarity based methodology.

The state-of-the-art decoders work in two stages, consensus calling and inference. During DNA synthesis, each original encoded oligo can be synthesized with duplication. Before sequencing, library preparation steps, like PCR, are typically used to amplify the pool of oligos by create multiple copies of each oligo to ensure successful sequencing. As a result, an original oligo has multiple copies of the reads eventually. Consensus calling attempts to group similar reads and obtain the consensus to achieve higher confidence. Although the grouped reads belong to the same original oligo, the reads are different from each other due to the randomized errors. Measuring the similarity of the reads in the same group can then be considered as a sequence similarity analysis problem. Prior work [36] also suggests that the consensus calling problem can be modelled as a database edit similarity join problem. Following the consensus calling procedure, the aggregated result is then converted back to a binary output.

Moreover, with motif-based encoding design, the motifs could be inferred directly before the consensus calling. In motif-based encoding design, the oligos are composed of motif blocks which are pre-fabricated short oligonucleotide sequences. Therefore, we can approximately align the motif libraries to the reads so as to determine each read's motif combination.

1.5 Thesis Outline

With the development of sequencing technologies, massive reads are generated while containing errors. The goal of this thesis is to build accurate and scalable sequence similarity analysis solutions for computational genomics and DNA-based digital storage applications. We introduce a fast and accurate sequence aligner –Accel-Align in Sec.2, and a DNA-based storage decoder – Motif-Search in Sec.3. Then, we apply Accel-Align in the sequence analysis to design DNA-based storage random access in Sec.4.

Chapter2: Accel-Align: a Fast Sequence Aligner. In this chapter, we first introduce the background of the sequence alignment algorithm and state-of-the-art techniques. In particular, we present that previous studies have primarily concentrated on *seeding-filtering-extension(SFE)* methodology which uses filtering to eliminate candidates. This chapter, instead, proposes a new methodology *seeding-embedding-extension(SEE)* that uses randomized algorithms to embed the reference string at each candidate location. We explain each stage in detail, particularly the embedding stage. We illustrate $2N$ and $3N$ embedding algorithms with their limitation caused by randomization, and propose two corresponding approaches for improvement. Finally, experiments with both the simulated and the real genomics dataset were conducted to explore the accuracy and performance of Accel-Align, comparing with the state-of-the-art methods. Accel-Align is also compared to BWA-MEM in DNA-based storage experiment and it shows the outstanding performance.

Chapter3: Edit Similarity-Based Decoder in DNA-based Storage. This chapter provides a brief introduction about the origination of DNA-based Storage. We propose a new encoding logic called *composite motifs* to scale the logical density and designs the corresponding structure-aware decoder *Motif-Search*. Since errors may exist in the reads, Motif-Search does the approximate edit similarity decoding. We present an end-to-end workflow and shows that it is capable of fully recovering the original data with low sequence coverage $20x$.

Chapter4: Sequence Analysis for Random Access in DNA-based Storage. This chapter studies the PCR-based random access in DNA-based Storage based on the sequence alignment method Accel-Align. It shows that PCR bias impacts the selection sensitivity of small files in a big pool. We find that non-specific primer binding happened during PCR can lead to the silent data corruption with short primers. As a solution, we present an elaborate primer design and a block-wise random access where the base storage unit has a fixed size. As a result, the random selection precision has been improved to be above 99%.

Chapter5: Conclusion. This chapter summarizes the work of the full thesis, the main contribution and the innovation of this work. At last, it also highlights the limitations of the work and proposes the further potential research directions.

Chapter 2

Accel-Align: a Fast Sequence Aligner

2.1 Introduction

In the recent decades, sequence alignment has been one of the major interesting research subjects due to its importance in genomics. Recent researches demonstrated that sequence alignment accounts for more than 30% of the overall time of the GATK (Genome Analysis ToolKit) best practice workflow [37, 38, 39].

The sequencing sequence alignment problem is defined as follow. Given a set of reads S and the reference genome R, the sequencing sequence alignment is to find out the best matching position on the reference genome R for each $s \in S$. The aligners can be classified into *ungapped* or *gapped* depending on whether they use Hamming or edit(Levenshtein) distance for computing mismatch between reads and the reference [40]. As modern sequencers can produce both substitution and indel errors, gapped aligners are preferred in practice over their ungapped counterparts.

Modern gapped read aligners, like Bowtie2, BWA-MEM, and Minimap2, can map thousands of reads per second to the reference genome. However, as sequencing datasets continue to grow at a rapid pace, even these state-of-the-art aligners face scalability bottlenecks due to a crucial design aspect that is universal across all gapped read aligners–the use of edit-distance as a string comparison metric. Computing edit distance between two sequences is a computationally-expensive task that takes approximately quadratic time in the length of the input sequences. Given that sub-quadratic computation of edit distance is extremely unlikely [41], the brute force approach of trying to align a read at each position in the reference is infeasible even for a single read due to sheer number of edit-distance computations that would be required. Thus, all modern aligners focus on minimizing the number of such computations using a *seed-filter-extend* (SFE) strategy for performing alignment [40].

SFE strategy works by first indexing the reference genome and storing the occurrence locations

of short string sequences, which are also referred to as seeds or k-mers, typically in a hash table. Each read is processed in three steps. First, seeds are extracted from the read and the hash table is used to look up potential mapping locations in the reference genome. Second, filtering techniques are used to eliminate as many candidate locations as possible to minimize the overhead of the extension stage, during which the entire read is aligned at each of unfiltered candidate locations using the edit-distance-based approximate string matching algorithms.

Although state-of-the-art filtering techniques provide an order of magnitude improvement over unfiltered extension, they all suffer from two major limitations. First, all current filtering techniques are based on *elimination*. Their goal is to improve performance by eliminating some candidate locations without forwarding them to the extension stage. They cannot provide *selection* which is able to identify a candidate that is likely to be the actual match, or *ordering* which sorts candidates in the order of likelihood. This is particularly problematic for seed sequences that inevitably occur at hundreds or thousands of different locations in the reference sequence due to tandem repeats, or transposon-induced duplication. For such sequences, filtering techniques are less effective, as they can neither identify optimal candidates, nor eliminate candidates without significantly increasing the probability of misalignment due to accidental elimination of a true match. The second problem with filtering is that the thresholds and parameters used by these filtering techniques are often manually picked based on empirical analysis, as they are dependent on the type and pattern of errors introduced by the sequencing technology. Hence, filtering techniques are inherently non-portable heuristics and not technology-independent design principles.

We introduce a new design principle for constructing sequence mappers and aligners, henceforth referred to as *Seed-Embed-Extend* (SEE). SEE builds on recent theoretical advances in the design of randomized algorithms that can perform embedding from edit distance into Hamming distance with very low distortion ([42]). These algorithms provide a one-to-one mapping f that can be used to transform a set of strings S into another set of strings S', such that the worst case ratio between Hamming distance of any two strings $f(x)$ and $f(y)$ in S', and the edit distance of their equivalent strings x and y in S is very low. SEE uses seeding to identify candidate locations similar to SFE aligners. However, instead of using filtering to eliminate candidates, SEE uses randomized algorithms to embed the reference string at each candidate location. SEE then uses Hamming distance between the embedded candidates and the embedded read to rank the candidates based on likelihood of being an actual alignment target, and chooses candidates with the highest rank for extension. Thus, instead of eliminating many bad candidates, SEE focuses on quickly selecting a few good ones.

To show that SEE works well in practice, we present *Accel-Align*–an SEE-based short-read sequence mapper and aligner that can provide both extension-free mapping and base-to-base alignment with CIGAR and MAPQ. In doing so, we show that a naive implementation of

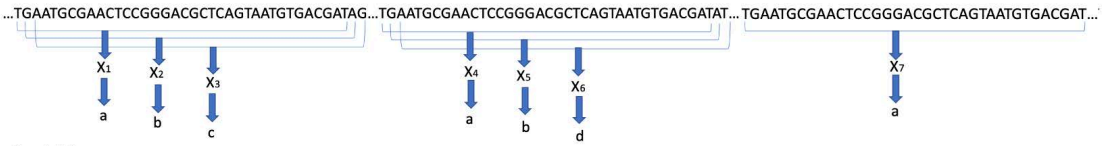
SEE will result in the embedding step becoming a computational bottleneck, and describe several optimizations that Accel-Align uses to implement SEE-based alignment effectively. Using experimental results from both simulated and real datasets, we show that embedding is capable of picking locations that are likely to be the correct alignment targets with very high accuracy. Using the SEE-approach to sequence alignment, Accel-Align is up to 9× faster than BWA-MEM, 12× faster than Bowtie2, and 3× faster than Minimap2, while providing comparable accuracy without using any special purpose hardware. We believe that SEE specifically, and embedding in general, is a robust technique that opens up new optimization opportunities not only for sequencing alignment, but also for several other computational biology problems that rely on edit distance.

2.2 Indexing and Seeding

2.2.1 Indexing

As Accel-Align uses seeding, it requires the reference genome to be indexed before execution. Similar to other aligners, we construct an index over the reference genome in a separate, offline phase. The index is a hash table of key-value pairs, where the key and value are both 32-bit unsigned integers. In order to populate the hash table, we extract k-mers from each position of the reference genome. As the reference sequence usually contains only 4 characters, namely A, T, C and G, we convert each character in the extracted k-mer into a two-bit equivalent representation. Any k-mers that contain 'N' characters are not added to the index. The k-mer length is a configurable parameter in Accel-Align, but we set it to 32 to enable a k-mer to fit in a single 64-bit integer. We hash the k-mer to generate the key by using a simple modulo-based hash function that maps the 64-bit integer into one of M buckets, where M is a large prime number that fits in a 32-bit integer. The 32-bit reference location offset from where the k-mer was extracted is the value associated with the key.

As the hash table is repeatedly used for looking up candidates during alignment, it is important to physically store these key–value pairs efficiently. We do this by using a chained hash table implementation based on two flat 32-bit integer arrays. With our construction, there are at most M different keys and $N-k+1$ different values, where N is the length of reference genome. As multiple k-mers can hash to the same key, each key can correspond to multiple position values. We gather all position values for each key, sort them individually, and store all such sorted values together, in key order, in a single *position* array. We represent the keys implicitly by an offset in a separate 4GB *key* array, and in each key-array entry, we store the cumulative count of candidate positions for all keys smaller than that key. Thus, as the position array is ordered by key, all the candidate locations indexed between the offsets K and $K+1$ in position table belong to the key K. The process is illustrated in Figure 1. Thus, the entire index,



Key table

Index	...	a	a+1	...	b	b+1	...	c	...	d	...
Accumulated frenquency		n_1	n_1+3		n_2	n_2+2		n_3		n_4	

Value table

Index	...	n_1	n_1+1	n_1+2	...	n_2	n_2+1	...	n_3	...	n_4	...
Location		l_1	l_2	l_3		l_4	l_5		l_6		l_7	

Figure 2.1: An example to build index. X_i represents a 32-mer extracted from reference genome. a, b, c, d are the keys calculated from X_i. n_i represents the number of keys smaller than the corresponding key. For example, n_1 is the number of keys smaller than a. Specifically, X_1, X_4 and X_7 have the same key a. So there are n_1+3 keys smaller than $a+1$. Thus, l_1, l_2 and l_3 indexed with n_1, n_1+1, n_1+2 in value table are the locations for X_1, X_4 and X_7.

represented using the key and position arrays, is saved in a single file on disk, and loaded in memory whenever alignment starts.

2.2.2 Seeding

During seeding phase of alignment, we extract all non-overlapping k-mers of each read. Then for each k-mer, we compute the key, and use the key to extract the list of candidate positions. The positions are adjusted using the offset of the k-mer into the read to get normalized candidate positions. Then, we merge the candidate lists across k-mers to produce the final list of normalized positions that does not have any duplicates. One way of performing this merging is to gather all candidates in an array, sort it, and then find unique elements. However, such an approach would take $O(N_l \lg N_l)$ time if N_l is the total number of candidates across all k-mers.

We avoid sorting by exploiting the physical organization of our hash table index. The position array of the hash table contains a key-ordered list of candidate positions, where each key's candidates are stored in sorted order. Thus, during seeding, the candidates retrieved for each k-mer will also be in sorted order. We maintain a min-priority queue of size N_k, where $N_k = readlength/k$ is the number of k-mers in each read, and initialize it with the first candidate location of each k-mer. Then, we pop the minimum value from the queue and push the next candidate from the same k-mer as the popped one into the queue. We repeat the pop–and–push steps until the all candidates have been processed. This approach allows us to process all candidates in $O(N_l)$ time without sorting as long as the number of k-mers is small enough to keep the overhead of min-priority-queue negligible, which we found to be the case

for short-read alignment using an empirical evaluation.

In the case of single-end reads, we don't apply any filtering to the merged candidate lists. Thus, all candidates are passed to the embedding stage. For paired-end reads, however, we use a configurable pairwise-distance threshold for identifying candidates from one read that have a matching pair within the specified distance in the other read. All such candidate pairs are passed to the embedding stage.

2.3 Embedding

After candidate locations are identified by seeding, Accel-Align moves to embedding, the second stage of SEE. The goal of embedding is to transform both the candidate strings from the reference genome, and the query string which is the read, into different strings such that the edit distance between the original strings can be approximated using the Hamming distance between new strings. We have implemented two randomized embedding algorithms in Accel-Align.

2.3.1 $3N$-embedding

The first algorithm was proposed by [42] who showed that given two strings x, y of length N taken from an alphabet $\sum$ such that $d_E(x, y)$, the edit distance between x and y, is less than K, there exists an embedding function f: $\sum^N \rightarrow \sum^{3N}$, such that the distortion $D(x, y) = d_H(f(x), f(y))/d_E(x, y)$ lies in $[1, O(K)]$ with at least 0.99 probability, where $d_H(x, y)$ is the Hamming distance between the embedded strings. In other words, [42] proposed a randomized algorithm that can embed strings of length N into strings of length $3N$ such that Hamming distance of embedded strings is at most square of the edit distance between original strings. Recent studies have demonstrated that this algorithm, which we henceforth refer to as *3N-embedding (3NE)*, works well in practice for performing edit similarity joins for even relatively large edit distances([43].)

Accel-Align uses 3NE for embedding both candidate strings from the reference and the read itself. Listing 1 shows the pseudo-code for the embedding algorithm. The input string is a DNA sequence of length N consisting of four possible characters (A,C,G,T). The output is an embedding string of length $3N$ consisting of the four characters and possibly multiple repeats of a pad character (P). In each iteration, the algorithm appends a character from the input string, or the pad if it runs out of the input string, to the output string. Then, it uses a random binary bit string to decide if the input index should be advanced. The net effect of this algorithm is that some input characters appear uniquely in the output string, while others are randomly repeated multiple times. Using the theory of simple random walks, [42] established that the randomization in this algorithm will result in strings that differ by a small

edit distance converging quickly to produce embedded strings that have a small Hamming distance.

Algorithm 1 $3N$-embedding

Input: A string $S \in \{A, C, G, T\}^N$, and a random string $r \in \{0,1\}^{3N}$
Output: The embedded string $S^{'} \in \sum^{3N}$

```
1: i ← 0
2: for j = 0 → 3N − 1 do
3:     if i < N then
4:         S'_j ← S_i
5:     else
6:         S'_j ← P
7:     end if
8:     i ← i + r_j
9: end for
10: return S'
```

2.3.2 $2N$-embedding

An initial implementation of 3NE in Accel-Align showed us early on that despite the simplicity of the algorithm, it was computationally intensive to embed billions of candidate locations across millions of reads. We describe several optimizations later in Section 2.4 that reduced the overhead of embedding, but one of the first optimizations we designed was a variant of the embedding proposed by [42], which we refer to as *2N-Embedding (2NE)*. Listing 2 shows the pseudocode for 2NE which is conceptually similar to 3NE with the exception that each character in the input string is copied to the output string at most two times. Thus, 2NE implements a mapping f: $\sum^N \rightarrow \sum^{2N}$ instead of $\sum^N \rightarrow \sum^{3N}$. After implementing 2NE in Accel-Align, we found that [44] had also developed it in parallel, performed a theoretical analysis of its optimality, and used it for the edit similarity join application. As we show in our evaluation, we found 2NE to be functionally comparable to 3NE in terms of accuracy, and slightly better in terms of performance as it reduces the embedding time due to a reduction in the embedded string length from $3N$ to $2N$.

2.3.3 Embedding Limitation

However, in practice, a "bad" random bit sequence could, in some cases, lead to a large distortion. To illustrate this, let us consider an example with two strings, "CTGACTGA" (#1) and "CTCACTGA" (#2). The two strings have an edit distance of 1. Given below are three different random sequences, and the embedded versions of these two strings for each random sequence in Table 2.1, Table 2.2, Table 2.3. Although the edit distance of the original strings is 1, the Hamming distance between embedded strings can vary dramatically and even be

Algorithm 2 $2N$-embedding

Input: A string $S \in \{A, C, G, T\}^N$, and a random string $r \in \{0,1\}^N$
Output: The embedded string $S^{'} \in \sum^{2N}$

```
1: j ← 0
2: for i = 0 → N − 1 do
3:     S'_j ← S_i
4:     j++
5:     if r_i = 1 then
6:         S'_{j+1} ← S_i
7:         j++
8:     end if
9: end for
10: for j = j + 1 → 2N − 1 do
11:     S'_j ← P
12: end for
13: return S'
```

inflated to 11 as shown in example 3.

Table 2.1: Example 1: 1 mismatch

Random seq for A	1110001000101000
Random seq for C	0010111101000001
Random seq for G	0010000001110110
Random seq for T	0110000110101111
Embedded #1	CTTGACCTTGGAPPPP
Embedded #2	CTTCACCTTGGAPPPP

Similarly, mismatches or indels at the beginning of strings will also lead to higher distortion than those at the end of a string. For instance, let us consider the string "CTGACTGC". Compared to #1, it differs only in its last character. Thus, the edit distance between them is 1. When the two strings are embedded, the embedded strings will be be identical for the initial set of characters except the last one. When the embedding algorithm reaches the last character, depending on whether the random bit is 0 or 1, the embedded strings will differ by 1 or at most 2. However, if we consider "ATGACTGA", which also has an edit distance of 1, but differs from #1 in the first character, the Hamming distance of their embedded strings will depend entirely on the random string. For instance, it will be embedded to "AATTGACCTTGGAAPP" using the random string in Example 1, with a Hamming distance of 10, or "ATTGAACTTGGAPPPP" using the random string in Example 2, with a distance of 1.

Table 2.2: Example 2: 4 mismatches

Random seq for A	0010111101100100
Random seq for C	0111010011001100
Random seq for G	0110001101101001
Random seq for T	1100100101100000
Embedded #1	CTTGAACTTGGAPPPP
Embedded #2	CTTCCAACTGGAPPPP

Table 2.3: Example 3: 11 mismatches

Random seq for A	1001011101001111
Random seq for C	1010110111011000
Random seq for G	1001100101001100
Random seq for T	1101111101101110
Embedded #1	CCTGGAACCTTGAAPP
Embedded #2	CCTCACCTTGGAPPPP

Thus, we proposed two optimization during embedding: multiple embedding and chain embedding.

2.3.4 Multiple Embedding

A simple strategy for dealing with distortion caused by embedding is to perform embedding multiple times with the goal that a high distortion produced by a "bad" random string will be overridden by a low distortion outcome from another random string. In the context of Accel-Align, this translates into the following per read operations: (i) $C \times R$ embedding operations for embedding C candidate locations R times, (ii) embed the query read itself R times, and (iii) $C \times R$ Hamming distance computations to identify the best candidate. A naive implementation of multiple embedding will also require $((C+1) \times 2N \times R)$ bytes of memory per read, where N is the length of read (each of the C candidates and the read itself have to be embedded R times, with each embedding producing a string of length $2N$).

During experimentation, we found that the computational and memory requirements of multiple rounds of embedding were high. Thus, we implemented a pipelined version of multiple embedding which works as follows. First, we embed the read R times. Then, we process each candidate one at a time by embedding it using a random string and computing the edit distance from the embedded read based on the same random string. After computing

Algorithm 3 Embedding

Input: A reference string $R \in \{A, C, G, T\}^m$, a querying read string $Q \in \{A, C, G, T\}^n$, a normalized candidate start position s_r with $|M|$ matches between the reference and read whose corresponding start and end indexes are s_{ri}, e_{ri} and s_{qi}, e_{qi}, and N times to embed

Output: The candidate's embedded Hamming distance d

$d_{min} = MAX$
for $l = 0 \leftarrow N - 1$ **do**
  $j = 0$, string $\hat{Q}$, $\hat{R}$
  for $k = 0 \rightarrow n - 1$ **do**
    if $k \notin [s_{qi}, e_{qi}), \forall i \in [0, |M|)$ **then**
      $\hat{Q}_j = \hat{Q}_k$
      $\hat{R}_j = \hat{R}_{s_r + k}$
      $++j$
    end if
  end for
  string $\hat{Q}', \hat{R}' \leftarrow$ the embedded string of $\hat{Q}$, $\hat{R}$
  $d \leftarrow$ the Hamming distance between $\hat{Q}'$ and $\hat{R}'$
  $d_{min} = min(d, d_{min})$
  if $d_{min} == 0$ or $d_{min} == 1$ **then**
    return d_{min}
  end if
end for
return d_{min}

R Hamming distances, we only keep the minimum Hamming distance per candidate, and use this to identify the candidate with the lowest overall minimum. We adopted this approach as it integrates seamlessly with two lower-level optimizations already performed by Accel-Align.

First, the embedding algorithm in Accel-Align does not generate the entire embedded string for each candidate. Rather, given a candidate location and random string, it generates one embedded character at a time, compares it with the corresponding character in the embedded read, updates the Hamming distance, and discards the character. This results in a CPU-cache-efficient embedding implementation. Second, the embedding algorithm is parameterized with a threshold so that it stops embedding as soon the threshold is exceeded. Instead of storing the embedded distance of all candidates, Accel-Align already dynamically tracks the lowest and second-lowest distances, and uses the latter as the threshold parameter. Our pipelined implementation of multiple embedding exploits both these optimizations to efficiently track the minimum embedding distance for each candidate.

We further optimize embedding by doing an early-stop for a candidate as soon as we find a random string under which the Hamming distance is computed to be less than or equal to 1. If the embedded Hamming distance is 0, the two embedded strings must be the same, so the original strings are same and edit distance is 0. If the embedded Hamming distance is 1, there

is 1 bit different in the embedded strings, same for the original strings, and the edit distance is 1. In either case, there is no need to do an additional round of embedding with a different random string as we have already found the minimum distance.

2.3.5 Chain Embedding

Let us consider the string x to represent a read, and string y to represent a candidate in the reference genome. Originally, Accel-Align embedded the entire read and an entire candidate string of length equal to the read. However, any candidate y identified by Accel-Align must have at least one k-mer that produced an exact match between the reference and the read which led to this candidate being identified as a potential match during seeding. If two strings are identical, their edit distance, and hence their embedded Hamming distance, will be zero. Thus, the embedded Hamming distance of all exact matching k-mers would already be zero. This implies that we only need to embed the non-matching parts of the read and the reference. We refer to such an approach as *chain embedding*, as it is reminiscent of the way aligners like Minimap2 use chaining to align gaps between exact matching regions.

Chain embedding improves both performance and accuracy. It improves performance as it reduces the length of the string that needs to be embedded. On the accuracy front, as mentioned earlier, the distortion of the randomized embedding algorithm depends on the edit distance value K. For any read x and a reference candidate y, let x_i represents the i-th non-matching substring in the read, y_i represents the corresponding non-matching part in the reference. These substrings are the parts that are found outside or between exact matching k-mers. The edit distance between them $d_E(x_i, y_i)$ is K_i, and $\sum d_E(x_i, y_i) = d_E(x, y) = K$. Original Accel-Align embeds x and y as a whole. Thus, the overall distortion is bounded by $[K, O(K^2)]$. Our modified Accel-Align with chain embedding, in contrast, embeds each substring separately. As each K_i is smaller than K, this should lower the distortion for each chain embedding, thereby improving accuracy. Putting together multiple and chain embedding techniques, Algorithm 3 shows the pseudo-code for the improved embedding algorithm.

2.3.6 Candidate Selection

We use one of the two embedding algorithms with the multiple embedding and chain embedding optimization described above to embed all candidates and the read. Then, we compute the Hamming distance between each embedded candidate and the embedded read. We refer to this distance as the *embedding distance*. While doing so, we dynamically keep track of the top two candidates with least embedding distance and forward them to the third phase for further extension, scoring, and mapping quality computation. This step is the most important

difference between SEE and SFE techniques. While SFE focuses on heuristics for eliminating candidates, SEE provides a way to rank candidates and directly select the most likely ones based on Hamming distance, which a scalable metric that can be computed in linear time.

2.4 Extension and MAPQ Computation

Accel-Align can be configured to run in alignment-free mapping mode where only the identified candidate location is reported, or full-alignment mode where base-by-base extension is performed and the CIGAR string is reported. For the mapping mode, we pick the best candidate, which is the one with the least embedding distance, as the target. Then, we embed the first seed of the read and the k-mers in reference genome at multiple positions around the final candidate position, and pick the position with the least embedding distance. This is done to take into account indels in the first few characters of a read.

For the full-alignment mode, originally Accel-Align did a global alignment between the read and the substring of same length in the reference's candidate position. But we found the lack of soft clipping to adversely affect accuracy of downstream variant calling. Accel-Align extracts a substring of length longer than read length and performs "glocal" alignment using lib-ksw [45] on either end to support soft clipping. The matching score is set to 2, mismatching, gap-open and gap-extension penalty are set to 8, 12, 2, to compute the alignment score and CIGAR. In addition to the CIGAR, Accel-Align also reports a mapping quality (MAPQ) that represents the degree of confidence in the alignment for each read. Accel-Align uses the cumulative Hamming distance obtained from chain embedding for identifying the top two candidates. If d_1 is the least embedded Hamming distance and d_2 is the second least, the MAPQ is computed as $MAPQ = 60 * (1 - d_1/d_2)^2$.

2.5 Optimizations

A naive SEE implementation would perform seeding, embedding, and extension as described so far in sequence. However, during initial experimentation Accel-Align, we found that while embedding reduced the overhead of extension, the computational task of embedding and Hamming distance computation added non-negligible overhead. Thus, in addition to the 2NE algorithm described in Section 2.3.2, we implemented three other optimizations, namely, *pipelining*, *early-stop*, and *prioritizing*, that reduced the overhead of embedding without any change in functionality or accuracy.

Pipelining. Instead of embedding all candidates and then computing the Hamming distance, our first optimization is to pipeline these steps. We do this by modifying the embedding step so that the read is embedded first. Then each candidate location is embedded one by

one, and the embedding algorithm simply updates the embedding distance in each iteration by comparing the output character of the candidate generated in that iteration with the corresponding character in the embedded read. This pipelining of embedding and distance computation provides three major benefits. First, as embedded strings are no longer generated in their entirety, it reduces memory consumption and associated overheads of allocating and freeing memory for storing embedded candidates. Second, it reduces the overhead caused by a needless second loop over the embedded candidates to calculate the Hamming distance. Third, as the output character generated by the algorithm is used immediately for distance computation, it improves processor cache utilization.

Early-stop. Using pipelining to produce the embedding distance for each candidate enables us to apply the second optimization based on the observation that only the top-two candidates with the least embedding distance are selected for further extension. Thus, if we have already encountered a candidate with a very low embedding distance, there is no point in continuing the embedding process for another candidate whose distance has already exceeded the previously observed minimum. Thus, we parameterize the embedding algorithm with a threshold such that the algorithm stops embedding a candidate as soon its embedding distance exceeds the threshold. Instead of storing the embedded distance of all candidates, we dynamically track the lowest and second-lowest distances, and simply use the latter as the threshold parameter.

Prioritizing. Our third optimization is a policy that drives pipelining and early-stop mechanisms. It is based on the intuition that if candidates with low embedding distance are prioritized before others, the overall cost of embedding will be low. This is due to the fact that the threshold will be set to a relatively low value during the early stages of embedding. As a result, early-stop will be applied to most candidates. However, the embedding distance of candidates is not known to us in advance. Therefore, we use candidate counting, a technique used by SFE aligners for count filtering ([46]), to prioritize candidates based on the assumption that candidates with higher counts or votes are more likely to have lower embedding distance, and more likely to be picked as the best candidate. Thus, we modify the seeding phase to associate with each candidate location a count of the number of k-mers that produced that location during the hash lookup. During embedding, we first embed the candidate with the highest count followed by all other candidates. It is important to note here that we still embed all candidates, albeit in a different order. Thus, unlike SFE aligners, we do not filter out candidates based on k-mer counting.

2.6 Results

Accel-Align is implemented in C++ and configured to use 2NE algorithm by default as it was found to be faster than 3NE with comparable accuracy. Accel-Align uses Intel Thread Building

Blocks for parallelizing both index generation and alignment. In this section, we present an evaluation of Accel-Align using both simulated and real data to compare its performance and accuracy with respect to three state-of-the-art short-read aligners, namely, BWA-MEM (v0.7.17; [28]), Bowtie2 (v2.3.5; [29]), and Minimap2 (v2.17; [30]). We also present an evaluation of Accel-Align using the real DNA-based storage data to compare with BWA-MEM (v0.7.17; [28]).

All experiments were run on a server equipped with a quad-core Intel(R) Core(TM) i5-7500 CPU clocked at 3.40GHz, and 32GB RAM. In an offline phase not reported here, we used each aligner to pre-index the reference genome. Then, in each alignment experiment, we run the aligner five times and gather execution statistics. As all aligners read the index from secondary storage, the first run is typically "cold" as data is not in memory. Hence, we ignore the first run. As we found the performance of the last four runs to be stable with all aligners, we only report the average of last four execution times.

2.6.1 Benchmark with simulated genomic reads

For benchmarking Accel-Align, we used Mason2 [47] to generate simulated reads from the hg37 reference genome (hs37-1kg) together with an alignment file describing the exact coordinate of each read. We used Accel-Align, BWA-MEM, Bowtie2, and Minimap2 (short-read mode) to align the reads and measured the end-to-end wall clock time for alignment. Using the Mason2 generated alignment file as our ground truth, we also evaluated the accuracy of each aligner in terms of the fraction of reads correctly mapped; we consider a read to be correctly mapped if the reported alignment overlaps with the Mason-provided one by at least ten percentage of read length.

Aligner comparison

Table 2.4 reports the performance and accuracy of the four aligners for a 10M, 100bp, single-end simulated read dataset generated by Mason2. In terms of performance, it can be seen that Accel-Align clearly outperforms the other aligners, as it is 8.2× faster than Bowtie2, 5.8× faster than BWA-MEM, and 2.4× faster than Minimap2.

Table 2.4: Evaluation on simulated single-end data

	BWA-MEM	Bowtie2	Minimap2	Accel-align
Exec. time (HH:MM:SS)	00:06:32	00:09:19	00:02:48	00:01:08
%Correctly mapped	97.37%	97.23%	96.52%	97.15%

Table 2.5 reports the performance and accuracy for a 10M, 100bp, paired-end dataset generated

by Mason2. Accel-Align outperforms the other aligners by an even larger margin here, as it is 14× faster than Bowtie2, 10× faster than BWA-MEM, and 3× faster than Minimap2. In terms of accuracy, all four aligners are comparable as a majority of reads are correctly mapped in both the single-end and paired-end datasets. BWA-MEM and Bowtie2 offer marginally better accuracy than Minimap2 and Accel-Align. These results demonstrate that at comparable accuracy, Accel-Align can provide an order of magnitude improvement in performance over some state-of-the-art aligners.

Table 2.5: Evaluation on simulated paired-end data

	BWA-MEM	Bowtie2	Minimap2	Accel-align
Exec. time (HH:MM:SS)	00:14:12	00:19:10	00:04:53	00:01:21
%Correctly mapped	98.54%	98.51%	98.09%	98.45%

Varying read length

The computational cost of the embedding step is proportional to the read length, as each read and at least one candidate are converted from length N into a new string of length $2N$. To test the sensitivity of performance with respect to read length, we used Mason2 to generate pair-end datasets with 10M reads, where each dataset was configured with a read length of either 150bp or 200bp. Figure 2.2 shows the throughput, the number of reads processed per second per thread, of the four aligners under various read lengths. Clearly, Accel-Align outperforms the other aligners at all read lengths as it provides 8–12× improvement over Bowtie2, 6–9× over BWA-MEM, 2.3–3× over Minimap2 across a range of read lengths. About the accuracy shown in Figure 2.3, we found that BWA-MEM is always most accurate. Accel-Align is close to BWA-MEM and Bowtie2 while highly outperforms Minimap2 especially when read length is short, e.g. 100bp. It catches up the accuracy when read length increases.

Alignment-free Mapping

Both Accel-Align and Minimap2 can be configured to run in alignment-free mapping mode where they report the position without the CIGAR string. The mapping mode completely eliminates the overhead of edit-distance computation. Although such mapping is useful in several applications that do not require base-by-base alignment, for example during the error characterization study in DNA storage, we use it in this context to isolate the benefit of embedding.

To compare Accel-Align with Minimap2 in mapping mode, we used the two aligners to perform alignment-free mapping of the paired-end Mason2 datasets. Figure 2.4 shows the execution time for various read lengths. Comparing Figures 2.2 and 2.4, we can make two observations.

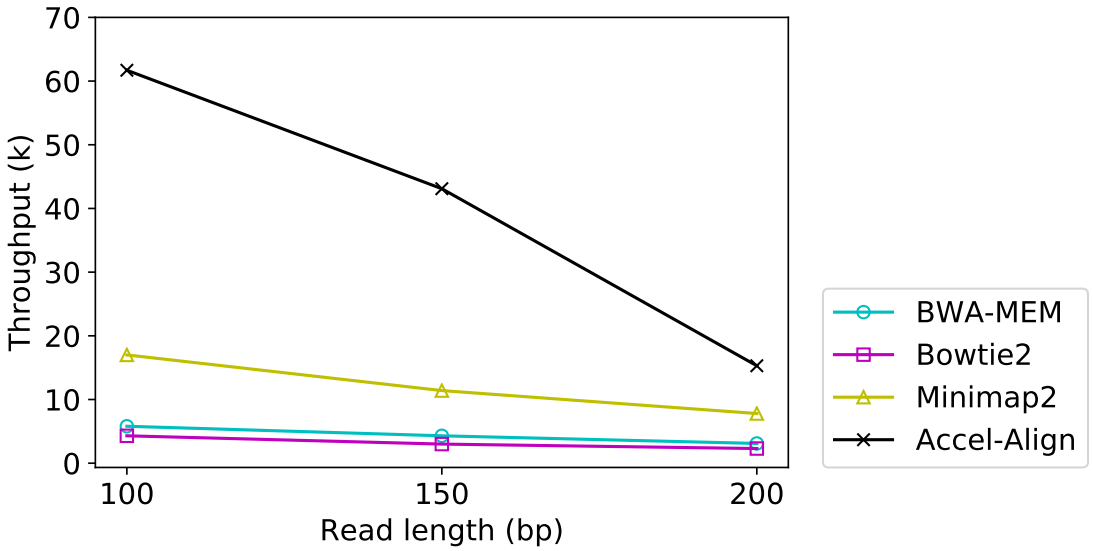


Figure 2.2: Throughput for 100bp, 150bp and 200bp pair-end simulated datasets.

First, alignment-free mapping provides a further 6% to 42% reduction in time over base-to-base alignment depending on the read length. Accel-Align maps 10M, 100bp, paired-end reads in 78 seconds, thus, mapping more than 650,000 reads per second on a simple quad-core processor without requiring any special hardware. These results demonstrate the benefit of using embedding in sequence mapping. Second, Accel-Align is 2-3.4× faster than Minimap2 at all read lengths (Figure 2.4) with alignment-free mapping while offering slightly better accuracy (Table 2.5). This demonstrates that our optimizations have eliminated any the computational overheads associated with embedding, making Accel-Align a competitive alternate to state-of-the-art sequence mappers.

Impact of embedding

To further isolate and understand the benefit of embedding in Accel-Align, we run Accel-Align in three modes:

- *No-embedding*: a no-embed mode where embedding is not used, and all candidate locations identified by seeding are directly forwarded for extension.
- 2*N*-*embedding*: the default mode using 2NE.
- 3*N*-*embedding*: using 3NE instead of 2NE.

Table 2.6 shows the performance and accuracy results for these three modes under the 10M, 150bp, Mason2 pair-end, simulated-read dataset.

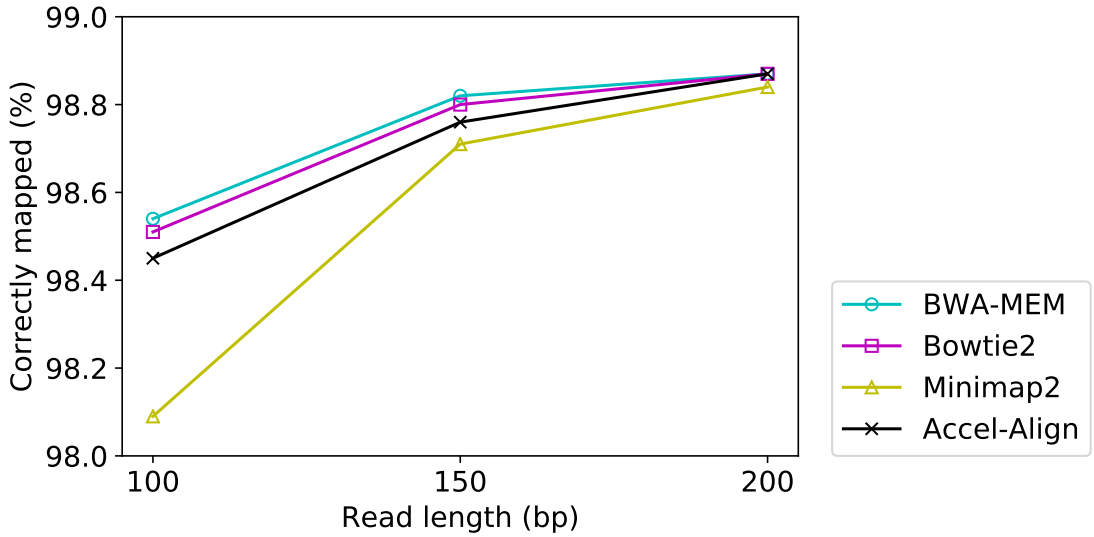


Figure 2.3: Accuracy of 100bp, 150bp and 200bp pair-end simulated datasets.

Table 2.6: Comparison of $2N$-embedding and $3N$-embedding

	No-embedding	$2N$-embedding	$3N$-embedding
Exec. time (HH:MM:SS)	00:29:24	00:02:56	00:03:16
correctly mapped	98.78%	98.56%	98.54%

Comparing 2NE and 3NE cases, we can see that 2NE provides a 11% improvement in performance with no discernible difference in accuracy. Comparing 2NE and the no embedding cases, we see that embedding provides a 10× reduction in execution time as it is able to identify the optimal candidate location without relying on edit distance computations at a marginal 0.2% lower accuracy.

2.6.2 Benchmark with real genomic reads

To evaluate the accuracy of Accel-Align on real data, we used the human whole-exome sequencing dataset NA12878 (accession No.: SRR098401). We built a pipeline similar to prior work [48] to detect variants using GATK HaplotypeCaller (v4.1.0) [39] as illustrated in Figure.2.5. We used the four aligners to align 150M 151bp paired-end reads in NA12878 to the hg37 reference genome. Then, we used the SureSelect Human All Exon v2 target captured kit bed file (ELID: S0293689) for capturing variant locations, and took high confidence variant calls (v2.19) from Genome in a Bottle (GiaB) consortium for validation.

We compare the aligners with respect to several metrics as shown in Table 2.7. The results for Accel-Align using the default parameters is shown in the column AA-32-mer. The execution

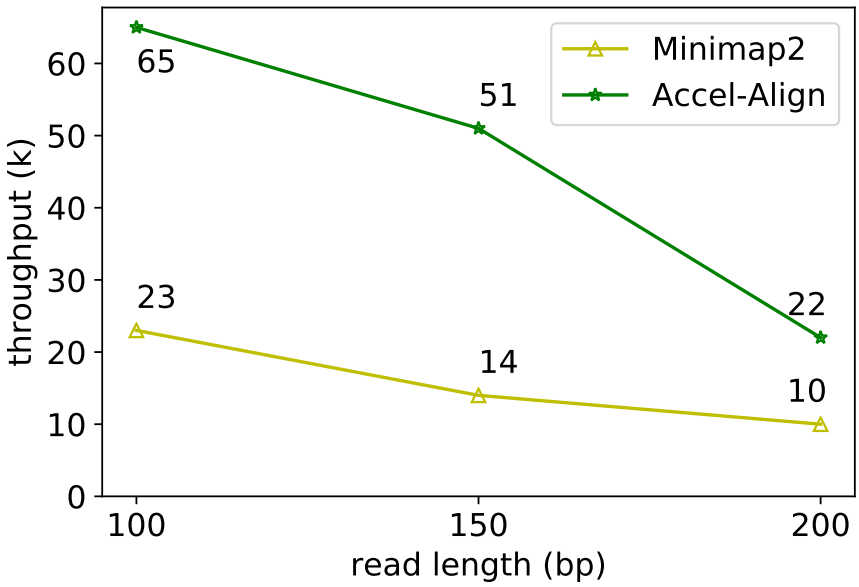


Figure 2.4: Alignment-free mapping time for 100bp, 150bp and 200bp pair-end read.

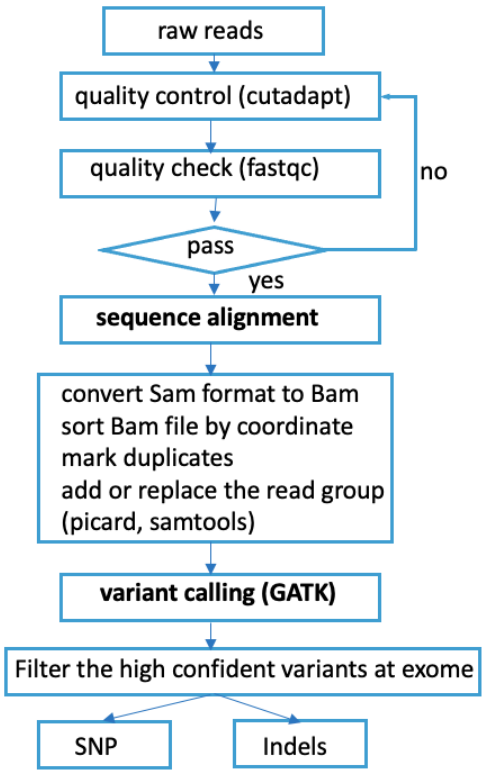


Figure 2.5: Variant calling pipeline.

Table 2.7: Evaluation on real data

	BWA-MEM	Bowtie2	Minimap2	AA-32-mer	AA-25-mer
Execution time	7:43:54	7:31:16	04:43:39	01:29:34	03:42:26
Ti/Tv	2.84	2.86	2.85	2.85	2.85
Precision	0.989	0.996	0.992	0.992	0.992
Recall	0.992	0.989	0.992	0.991	0.991
F-score	0.990	0.992	0.992	0.992	0.992
Fraction Mapped	99.40%	97.13%	99.31%	97.00%	98.02%
TP(SNP)	23521	23457	23521	23516	23518
FP(SNP)	235	79	163	176	207
FN(SNP)	165	229	165	170	168
F-score(SNP)	0.991	0.993	0.993	0.993	0.993
TP(InDels)	1223	1213	1223	1225	1223
FP(InDels)	49	30	27	31	32
FN(InDels)	35	45	35	33	35
F-score(InDels)	0.966	0.970	0.975	0.974	0.974

time reports the wall-clock time taken by various aligners for aligning 85M paired-end reads (or 170M reads in total). Accel-Align provides a speedup of 5× over Bowtie2 and BWA-MEM, and 3× over Minimap2, similar to the Mason2 dataset. The second metric is transition-to-transversion ratio (Ti/Tv), which is a key metric in detecting SNVs and should fall between 2.6–3.3 for this dataset, which is the case with all aligners. The three later metrics report precision, recall, and F-score values based on the GiaB truth set contains 23,686 SNVs and 1,258 InDels contributing to a total of 24,944 variants for the NA12878 exome. The precision, recall, and F-score of BWA-MEM, Minimap2, Accel-Align are comparable except Bowtie2 has a higher precision and lower recall which means that it detected less TP variants although detect less FP variants as well. Accel-Align and Bowtie2 aligned ~ 97% of reads while BWA-MEM and Minimap2 aligned more than ~ 99%. This could be the potential reason that Bowtie2 reported less variants and is one of the direction that Accel-Align is aimed to improve. We also showed the SNP and InDels separately. Accel-Align had better accuracy to detect InDels that it detected even 2 more TP InDels than BWA-MEM and Minimap. However, it needs to improve the capability to detect SNP as detected 5 less TP SNP.

To better understand the difference between detected variants, we show a Venn diagram of variants detected by various aligners in Figure 2.7 and Figure 2.6 for SNP and InDels separately. Clearly, 97.7% of Indels and 94.2% SNP variants are captured by all aligners. We find that there are 7 variants detected by the other three alignment tools, but not by Accel-Align. Upon further inspection, we found this to be due to two reasons. First, although embedding identifies the correct candidate location in most cases, there are reads for which it chooses the wrong candidate location. We found this to be particularly problematic for reads that map with

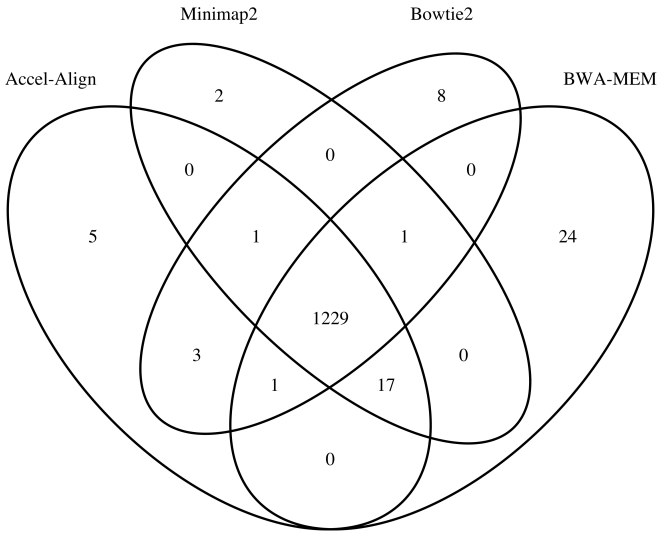


Figure 2.6: Venn diagram of InDels variants detected by various aligners and Accel-Align (32-mer).

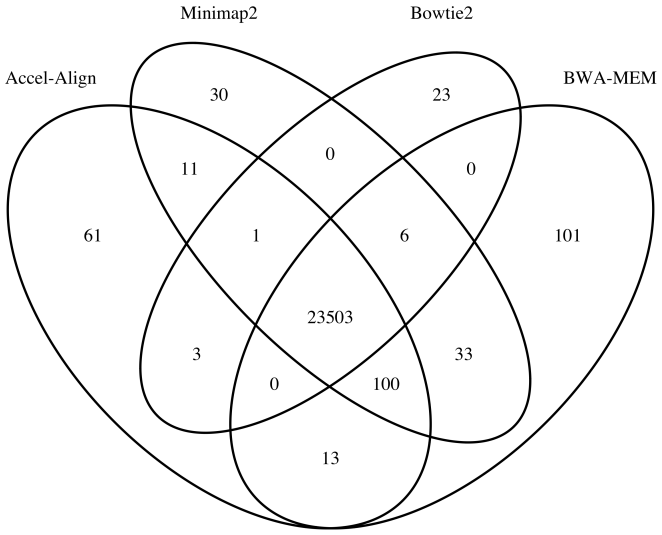


Figure 2.7: Venn diagram of SNP variants detected by various aligners and Accel-Align (32-mer).

a low edit distance to multiple locations in the reference. In such cases, we found that the hamming distance of embedded candidates is not spaced apart for embedding to identify a clear target. The second reason is the use of large k-mer length (32) in Accel-Align compared to the read length in NA12878 (151bp) which resulted in reads with four erroneous k-mers being unmapped. Table 2.7 shows the percent of mapped reads, and as can be seen, Accel-Align aligns 2% fewer reads compared to BWA-MEM and Minimap2.

To understand the impact of k-mer size on overall accuracy, we modified Accel-Align to use 25-mers instead of 32-mers. Table 2.7 shows the results obtained using the 25-mer setting under column AA-25-mer. Comparing it with AA-32-mer, we have two important observations. First, the overall execution time increases by 146% compared to AA-32-mer case. Figure 2.8 shows a breakdown of execution time across the three stages when 25-mers or 32-mers are used. Clearly, this increase in time can be attributed almost entirely to seeding and embedding as 25-mers produce 4× more candidates than 32-mers. Figure 2.8 also shows the fraction of each stage over the whole processing time. The seeding and embedding fractions are close to each other for 25-mer and 32-mer cases. As SEE does not filter out any candidates, the overhead of candidate normalization, counting, and duplicate elimination performed during seeding increases, similarly for embedding. Despite this, embedding is able to identify the candidates, avoid needless extension, and still provide 2-7× reduction in execution time over other aligners. Second, the fraction of mapped reads increases by 1.1% by using 25-mers instead of 32-mers. However, the overall variant detected by 25-mers not increase. In terms of the overall F-score as shown in Table 2.7, the 25-mer case provides similar accuracy to the 32-mer case. This is because compared to BWA-MEM and Minimap2, it still mapped 1% less reads.

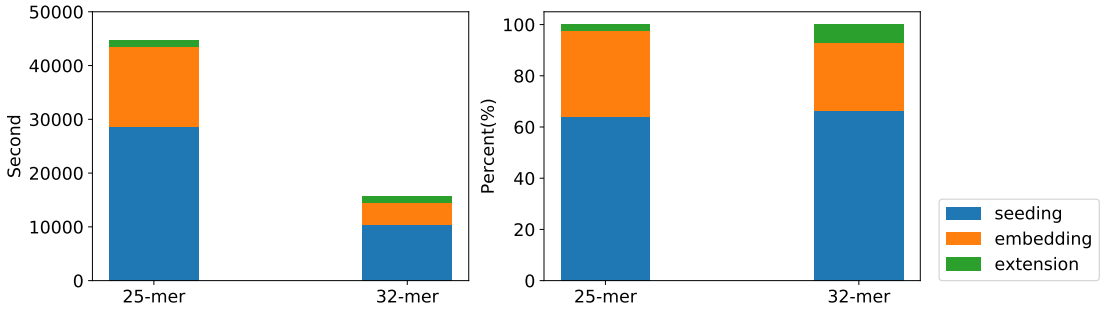


Figure 2.8: the percent of seeding, embedding and extension over the total processing time.

2.6.3 Benchmark with real reads from DNA-based storage

Apart from being a mapper and aligner for genomic data, Accel-Align can be used to determine the original reference oligos of the reads sequenced from DNA-based storage. By aligning the reads to the reference oligos, it can be used to study the error characterization, which is

often the first and important step to understand the DNA synthesis and sequencing pattern. To perform the comparison, we sequenced 44k oligos of 200bp and generated 43M reads in the work [49]. Subsequently, we aligned the 43M reads to the original oligos with Accel-Align [50, 51] and BWA-MEM (v0.7.17; [28]. The alignments were performed on a local server equipped with a 12-core CPU Intel(R) Core(TM) i9-10920X clocked at 3.50GHz, 128GB of RAM.

Table 2.8: Comparison between BWA-MEM and Accel-Align.

	BWA-MEM	Accel-Align	AA-align-free
Exec. time (HH:MM:SS)	00:08:11	00:05:03	00:04:16
%Mapped	99.9988%	99.9999%	99.9999%

Table 2.8 shows that Accel-Align saves 60% time than BWA-MEM, and Accel-Align alignment free mode (introduced Sec.2.6.1) saves 20% alignment time in addition. We further break down the Accel-Align processing time and find that 80% time is consumed by dumping the SAM out file (27GB). Accel-Align only consumes 42 second if we skip the output writing step. It indicates that although the alignment performance scales up with multiple threads, writing the output to disk can be the bottleneck since it allows only one thread to write output to avoid collision.

Table 2.8 also shows that both aligners aligned more than 99.99% reads to a reference oligo, indicating a very high quality of the generated read set, while Accel-Align slightly aligned 0.0011% more. We also report the error distribution in Figure 2.9 and Figure 2.10 which shows a histogram of edit distances between the reads and references. As can be seen, 96.97% reads have edit distance less than 10, indicating that the error rate is less than 6%.

Figure 2.11 and Figure 2.12 show the coverage histogram (number of oligos that have a given coverage). Each reference oligo is covered by at least one read, with a median coverage of 951× for both BWA-MEM and Accel-Align. We deliberately sequenced the oligos at such high coverage to test recovery at various coverage levels.

Among all these studies, we find that Accel-Align have similar result compared with BWA-MEM. Thus, we can adopt it later to accelerate the error characterization when we study DNA-based storage, for example, we used it in our last work [49].

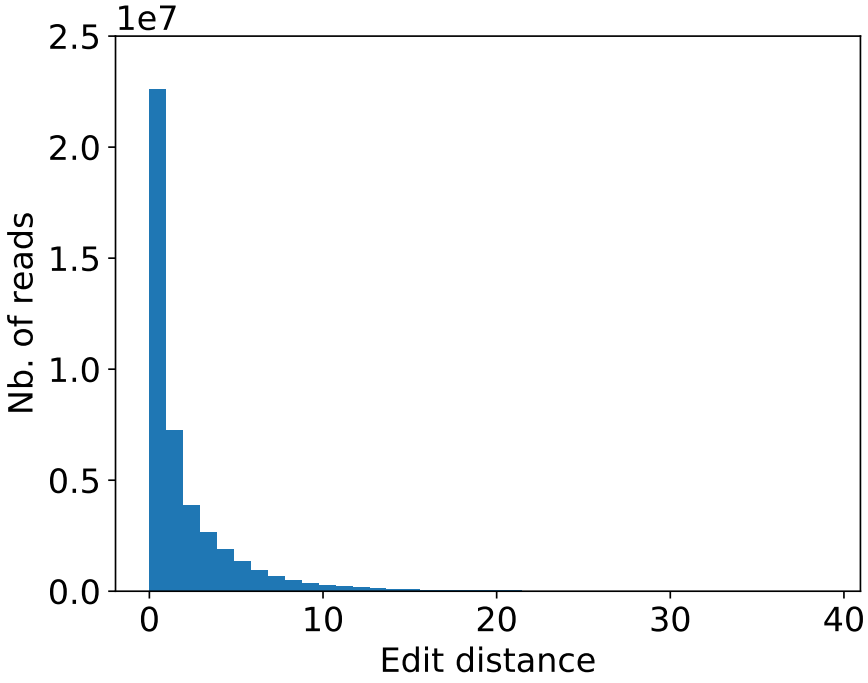


Figure 2.9: Distribution of edit distance with BWA-MEM.

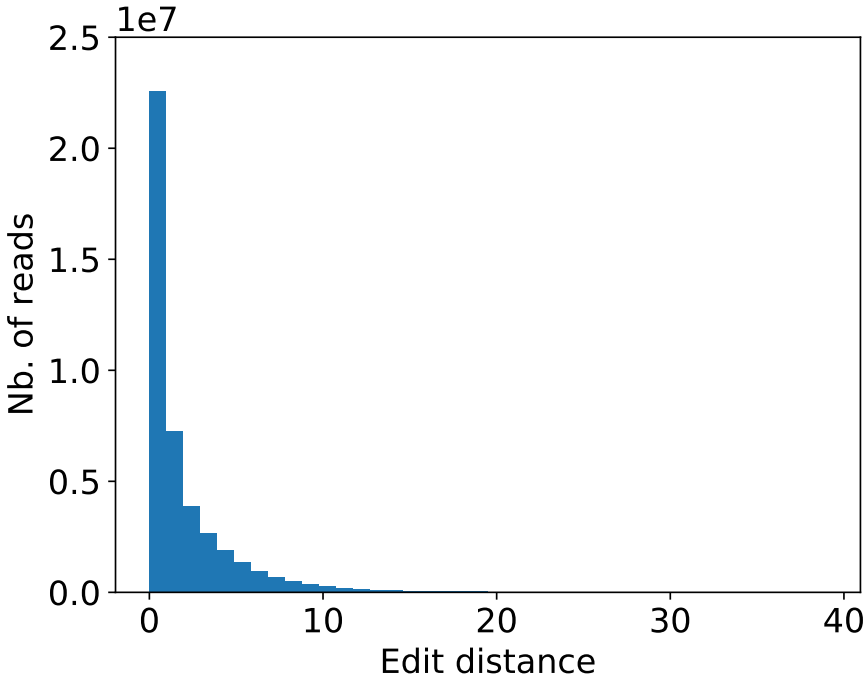


Figure 2.10: Distribution of edit distance with Accel-Align.

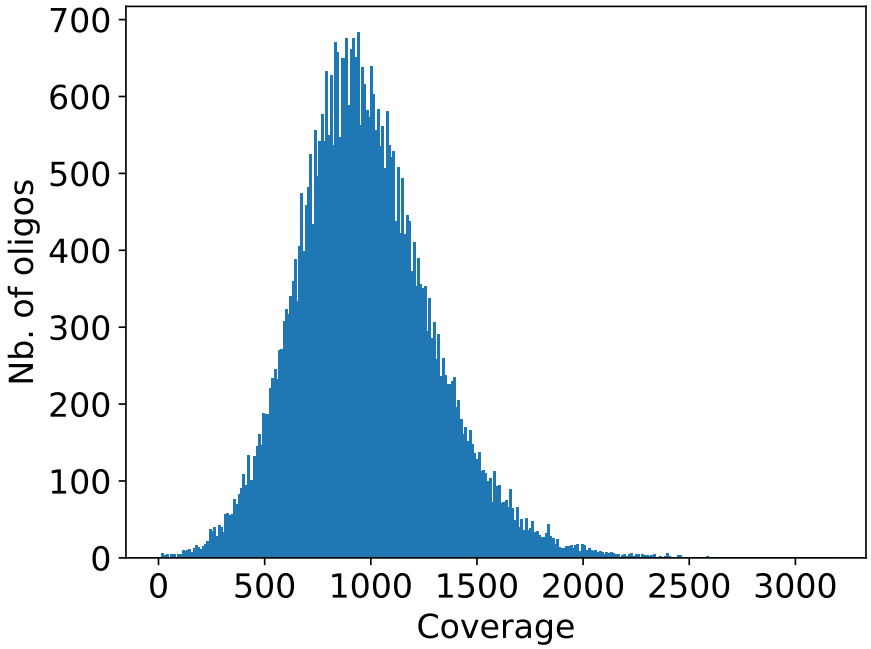


Figure 2.11: Histogram of coverage across oligos with BWA-MEM.

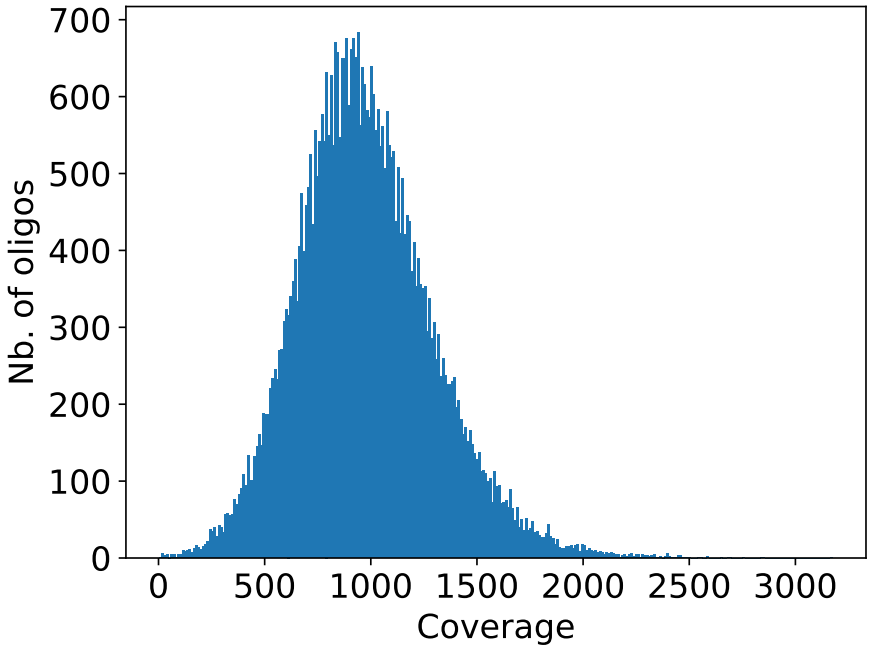


Figure 2.12: Histogram of coverage across oligos with Accel-Align.

2.7 Discussion

As sequencing technologies continue to increase read length while improving throughput and accuracy, we believe that randomized embeddings open up new avenues for optimization that cannot be achieved by using edit distance. We implemented Accel-Align and showed that it is up to 3× faster than Minimap2, 9× times faster than BWA-MEM, and 12× times faster than Bowie2, while providing comparable accuracy. Accel-Align clearly demonstrates the potential of using embedding for short-read sequence alignment. The SEE method used by Accel-Align is a design principle that is parameter free, applicable to any sequencing technology, and can deal with both indels and mismatches without requiring any customization. Further, the benefit of SEE methodology and low-distortion embedding are not limited to short-read alignment; any computational biology problem that is limited by the overhead of edit distance can benefit from embedding. As embedding transforms strings from edit to Hamming regime, computational tools like Locality Sensitive Hashing can also be applied on the resulting strings. Thus, the techniques presented in this paper have a much broader scope as they can be used for other applications like spliced RNA-seq and bi-sulfite alignment, multiple sequence alignment, and even sequence assembly.

Chapter 3

Edit Similarity-Based Decoder in DNA-based Storage

3.1 Introduction

The growing adoption of Big Data Analytics and Artificial Intelligence has led to an explosion in the rate of data generation and storage. A recent survey by the International Data Corporation reports that the digital datasphere is forecast to grow to 125 zettabytes by 2025 [52] and is anticipated to exceed silicon supply in 2040 [53]. As traditional storage media is unable to keep pace with the rate of data growth [33], synthetic DNA has become an increasingly attractive archival storage medium due to its high density, stability, longevity and absence of technical obsolescence compared with electronic media [7, 35, 54, 55, 56].

In most prior work on DNA-based digital storage, DNA synthesis is based on phosphoramidite chemistry, a DNA synthesis technology that has been optimized over several decades to perform highly-accurate, base-by-base synthesis of short DNA strands by making phosphodiester bonds between nucleotides. There are three Key Performance Indicators (KPIs) that can be used to evaluate the efficiency of DNA synthesis:

- bits written per cycle (also called *logical density* [57, 58]),
- bits written per oligo,
- coupling reactions per oligo.

The efficiency of writing data to DNA depends on the number of synthesis cycles (x) to grow the strand and available repeating units (m) for addition at each cycle. The information capacity of the oligo (N bits) can be derived as

$$N(bits) = x \times log_2 m \tag{3.1}$$

While base-by-base synthesis methods can perform 200 or more coupling cycles(x), the

number of available subunits to add at each cycle is four (nucleotides), thereby limiting bits per synthesis cycle to two, and the information capacity of an oligo to a few hundred bits. While the quality, quantity, cost, and rate of DNA synthesis provided by base-by-base chemistry is suitable for biological research, it is far from ideal for the DNA storage use case. This has resulted in synthesis emerging as a major bottleneck in DNA storage.

In this work, we introduce the *composite motifs* framework to scale logical density well beyond the limit of 2 bits per synthesis cycle. Composite motifs are inspired by recent advances in motif-based approaches to DNA data storage [49,59] that use short oligonucleotide sequences, also referred to as motifs, that are drawn from a fixed library as building blocks for assembling longer oligos. Using a motif library of M motifs, one can scale logical density by storing $log_2(M)$ data bits per synthesis cycle. The use of a fixed library of motifs similar to a typesetting press can also simplify miniaturization and automation. The composite motif framework builds on the benefits of motif-based DNA storage, and further improves logical density by exploiting sequencing multiplicity inherent in DNA synthesis by encoding data using a combination of motifs rather than individual motifs.

In this work, we show that a DNA storage system based on composite motifs can provide an order of magnitude improvement in logical density over state-of-the-art systems by implementing an end-to-end prototype system as shown in Figure 3.1. In doing so, we develop new encoding and enzymatic motif ligation techniques that can scale DNA synthesis in the DNA write pipeline, and assembly-free, Nanopore-based motif read out and alignment-based motif decoding techniques that can scale DNA sequencing in the DNA read pipeline.

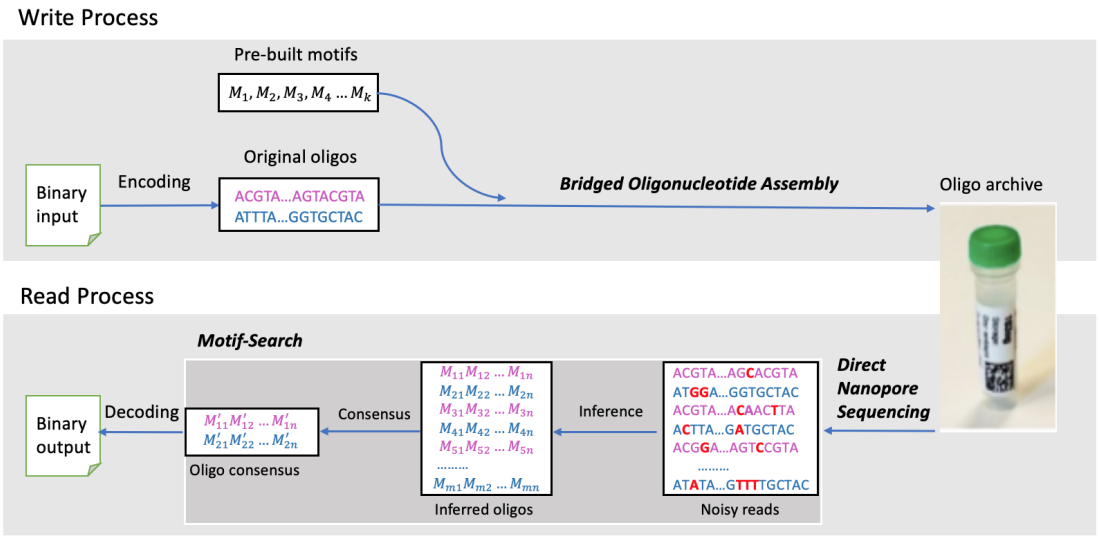


Figure 3.1: Data writing and reading pipeline of DNA storage.

3.2 Methods

3.2.1 Composite Motif-based Encoder

A composite motif is a representation of a position in an oligo sequence that uses a combination of motifs drawn from a fixed motif library to encode data as shown in Fig 3.2 and Fig 3.3. For example, assuming a library of 32 motifs, and a combination factor of four, there are $C(32,4) = 35960$ possible unique combinations with which we can encode 15 ($log_2 35960$) bits of data per composite motif. Composite motifs increase logical density by expanding the motif library using combinations of motifs without increasing the volume of motifs. As current synthesis platforms already use a high degree of sequence multiplicity (multiple copies of DNA molecules are synthesized per oligo), composite motifs can also be integrated into current platforms without any extra cost as they can exploit sequence multiplicity to scale logical density. Higher logical density also leads to a reduction in the length of DNA required to store the same amount of data, alleviating issues related to long oligo synthesis.

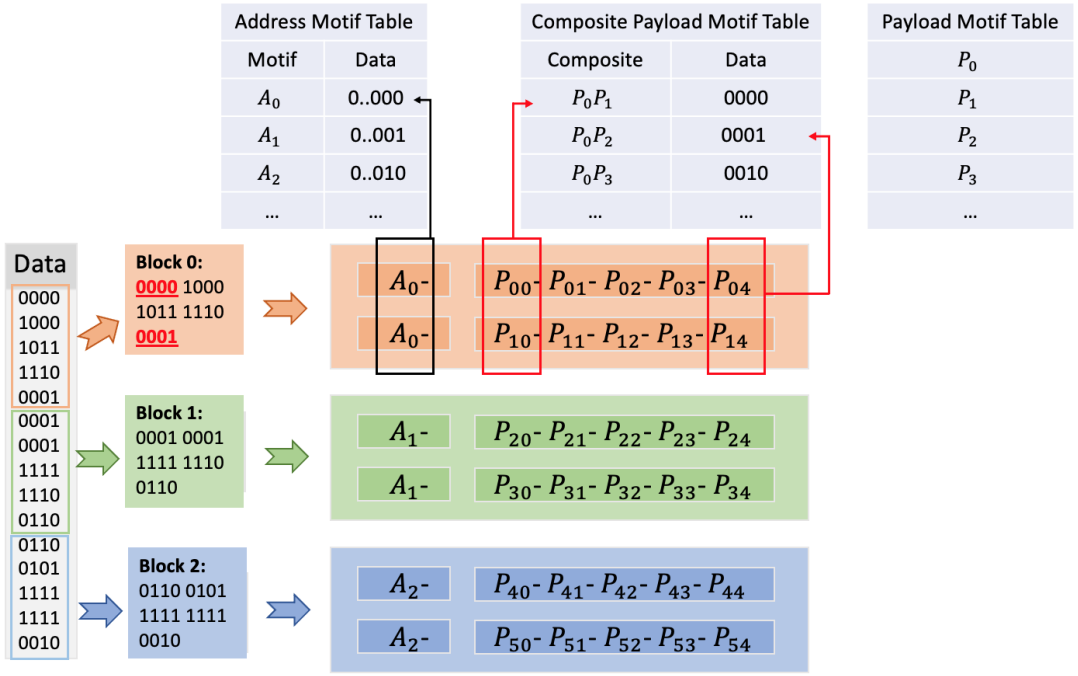


Figure 3.2: Composite motifs increases the logical density in DNA-based storage. A block of binary data is encoded to a sequence comprising a set of oligos with same address payload motifs. The composite of payload motifs from the same vertical position represents the binary data together.

In order to demonstrate the feasibility of using composite motifs, we developed a DNA storage system that uses composite motifs as building blocks. Fig 3.1 presents the read/write pipeline of our system. On the writing side, digital data is encoded into oligos containing composite

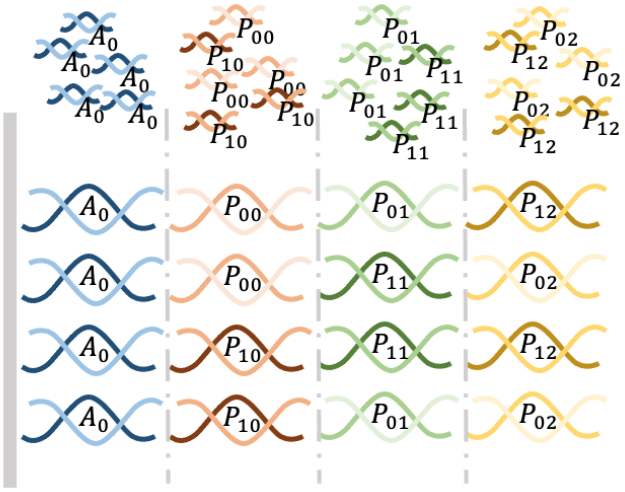


Figure 3.3: Composite motifs can be generated by mixing the motifs during each synthesis cycle. A: address motif, P: payload motif. Example: A_0–[P_{00}, P_{10}]–[P_{01}, P_{11}]–[P_{02}, P_{12}].

motifs using a motif encoder. Writing a composite motif at any given position of an oligo sequence is done by mixing multiple motifs during the synthesis procedure to synthesize multiple DNA molecules that contain the corresponding combinations of motifs using *Bridged Oligonucleotide Assembly* (BOA) (Sec.3.2.2). On the reading side, we read composite motifs by amplification-free sequencing of multiple DNA molecules using *Direct Oligonucleotide Sequencing* (DOS) (Sec.3.2.3), and then decode the data using our new motif-based consensus caller called *Motif-Search*(Sec.3.2.4).

3.2.2 Bridged Oligonucleotide Assembly

Oligo with a format of A_0–P_0 was realised with (i) a set of 8 ssDNA oligo sequences of 24-bases in length, representing A_0; and (ii) a set of 32 ssDNA oligo sequences of 50-bases in length, representing the common spacer motif and each P_0 motif. The sequences of motifs in these oligos were selected from 25mer DNA barcodes. A set of 8 ssDNA oligo sequences of 50-bases in length were designed to function as (i) a bridge between A_0 and P_0 for ligation; and (ii) an adenosine overhang on the 3' end to facilitate AMX sequencing adaptor ligation.

Phosphorylation. A pool of 32 oligos, representing the common spacer motif and each P_0 motif, were 5' phosphorylated using T4 PNK at a pool concentration of 300 pmol and reaction scale of 50 uL, as per the vendor guidelines at 37°C for 40 minutes. A denaturation step was performed to stop the phosphorylation at 65°C for 20 minutes.

Assembly. The 8 A_0 oligos and 8 Bridge oligos are pooled at equimolar concentrations and diluted to 25 uM final pool concentration. DNA assembly reaction was carried out by taking 2

ul of the P_0 phosphorylation mix whose component is shown in Table 3.1 and 0.5 ul of the A_0 + bridge pool and ligated using using Blunt/TA master mix as per vendor guidelines listed in Table 3.2.

The above reaction is incubated at 95°C for 3 minutes and gradually cooled to room temperature.

Table 3.1: Reaction components and volumes for P_0 phosphorylation

Component	Volume
T4 PNK Rx Buffer	5 ul
ATP (10 mM)	5 ul
T4 PNK	1 ul
NFW	36 ul
P_0 (300 pmol)	3 ul

Table 3.2: Volume composition for motif annealing reaction

Component	Volume
A_0 & Bridge (12 pmol)	0.5 ul
P_0 phosphorylation Rx	2 ul
Blunt/TA master mix	

3.2.3 Direct Oligonucleotide Sequencing

A key aspect of a DNA storage system, along with DNA writing performance, is the cost of DNA sequencing and time taken to read data from DNA molecules. Nanopore sequencing enables single molecule sensing capabilities and has the potential to create a low-cost, high-speed DNA storage read head. The yield of a Nanopore (ONT) flowcell is dependent on the size of the DNA to be sequenced. Small oligos result in a higher number of unoccupied pores over time. ONT estimates that the minimum DNA size to load in a R9.4 flowcell is 200 bases. Thus, prior work on DNA storage with Nanopore has relied on additional sample preparation steps for short oligos that are manual and time consuming [60]. In particular, DNA assembly methods were used to concatenate five or more DNA storage oligos into a longer fragment, and PCR amplification is used to sufficiently increase sequencing throughput and coverage for decoding.

We developed a method to enable direct sequencing of composite-motif-based oligos without amplification or second-strand synthesis. As mentioned earlier, our oligos have only two motifs concatenated by a spacer. Thus, we designed our eight bridge oligos to double in role as adapters that will include an adenosine overhang after annealing to address(A_0) and payload(P_0) oligos (Fig 3.6). The address motifs are 5' phosphorylated which results in all

oligos in our pool having their 5' end analogous to end-prepared dsDNA. Thus, these oligos can readily ligate with the AMX sequencing adapters from ONT's ligation sequencing kit (LSK-109). The AMX adapters were attached to the oligos in a 10 minute reaction. Sequencing was performed on a R9.4.1 flow cell for 4 hours. Basecalling was performed with both Guppy and Bonito basecallers. The sequencing run generated 27,198 reads with an N50 of 192bp.

Sequencing sample preparation was carried out using LSK-109 kit. AMX sequencing adaptors were ligated by mixing 2.5 ul of the assembly mix with 5 ul AMX and 5 ul Blunt/TA mastermix from NEB and incubated for 10 minutes. The sample was then loaded into a R9.4.1 MinIon flowcell and sequenced for 90 minutes. Basecalling was performed on the Guppy (v4.0.15).

3.2.4 Motif-Search Algorithm

Motif-Search works in two stages, *inference* and *consensus calling*. In the inference stage, it maps each read to an inferred oligo. During consensus calling, it uses all inferred oligos to produce a consensus set of inferred reference oligos.

Inference

The first task performed by Motif-Search is to extract one or more oligos from each read. There can be several oligos in one read because of the incorrect segmentation by MinKNOW during sequencing. Recall that an oligo is a set of motifs concatenated by spacers. Motif-Search infers oligos by first locating the spacer positions and then mapping the portions of the read between two spacers to the reference motifs to determine the payload and address motifs. Inference works in three steps: i) segmentation to locate spacer positions, ii) mapping to identify reference motifs between spacers, iii) overlap check to extract only oligos that do not overlap with each other.

Segmentation. Segmentation determines the spacer positions. Since all spacers are identical, their candidate positions can be located by k-mer seeding. We convert A, T, C and G into a two-bit equivalent representation and build the index of the spacer by extracting all k-mers of length four (found to be optimal experimentally). To process each read, we extract all 4-mers in the read, lookup the index, and collect positions with an index hit. The positions are adjusted by the offset of the k-mer to get normalized positions.

To eliminate candidate positions with low confidence, we filter out the positions having less than $spacer\_length/k$ k-mer votes. As reads are error prone, indels can cause candidate positions that should be identical to differ slightly by a few nucleotides. This could result in candidates receiving fewer votes and failing the filter. For instance, given a spacer "ATCG-TAGCAGT" (#1) and a read containing "ACGTAAAGCAG" (#2), among all the 4-mers of #2, only

the 4-mers "CGTA" and "GCAG" find the matched 4-mers in #1, and the normalized position will be -1 and 1 instead of 0. Hence, we merge neighboring positions and represent them by a centroid with a combined count. Regarding the previous example, we get the centroid position zero with two votes and it passes the vote filtering. At the end of this stage, we have all candidate positions for all spacers in a read.

In our experiment, each oligo has only one spacer. But in the general case, each oligo can contain multiple spacers. From the structure of the oligo, we know that each oligo with M motifs has $M-1$ spacers, with each spacer being spaced apart by a distance d equal to the sum of the motif length and spacer length. In order to accommodate synthesis and sequencing errors, these inter-spacer gaps can be slightly more or less than the motif length depending on indel errors. Thus, we identify all possible chains of $M-1$ positions which are within an expected distance threshold from each other.

As mentioned earlier, the candidate positions in these chains are approximate, as indel errors can result in observed starting position differing from actual starting position by a few nucleotides. We rectify and refine these positions to tolerate indel errors by using randomized embedding—a technique which has been demonstrated to be a scalable approach for mapping reads to references in genomic sequence alignment [50]. More specifically, for each candidate position, we extract a spacer-length portion of the read at that position and at several positions around that position. We embed each extracted read fragment using a randomized algorithm and compare with the embedded version of the original spacer motif using hamming distance. We select the shifted position with least embed distance as the final candidate position. As the number of candidate positions can be large, the use of embedding helps us to avoid expensive edit distance computations between the read and spacer motif, and use hamming distance between their embedded versions to rectify candidate positions.

Mapping. Given a chain of refined candidate positions, we can extract the potion of each read between two neighboring spacers. These portions correspond to address and payload motifs. The next step is to identify the original motif for each observed motif in the read. This can be translated to a sequence mapping problem by considering the original motif library as the reference and the observed motif in the read as the query. Therefore, we use the ksw-lib([61]) to select the optimal original motif with the highest mapping score for each observed motif. After this step, we have multiple chains of mapped motifs.

Overlap check. As we consider all possible chains, some chains might overlap each other. However, while each read can cover multiple oligos due to DOS, each nucleotide in a read should map to only one motif/oligo. Thus, the final step in the inference stage is to identify the optimal set of chains that do not overlap with each other. To do this, we traverse the chains to identify overlapping sets. For each overlapping set, we pick a chain with the highest mapping score such that no chain appears in two sets.

Consensus Calling

Each original encoded oligo can be synthesized with duplication. Library preparation steps, like PCR, also amplify the pool of oligos by creating multiple copies of each oligo to ensure successful sequencing. Thus, an original oligo can be covered by multiple reads. For each read, the inference stage identifies the optimal set of non-overlapping chains. As the final step, we apply consensus calling to group similar motif chains inferred from the *inference* stage, and obtain consensus to achieve higher confidence. We know that oligos do not have logical addresses and need an index to identify the serial information in DNA-based storage. This index information can not only order the oligos but also be used as the group key during the consensus calling. We do this by first clustering the inferred oligos using their address motifs. Then, we select the most frequent motifs at each position as the final consensus motif as shown in Fig 3.4.

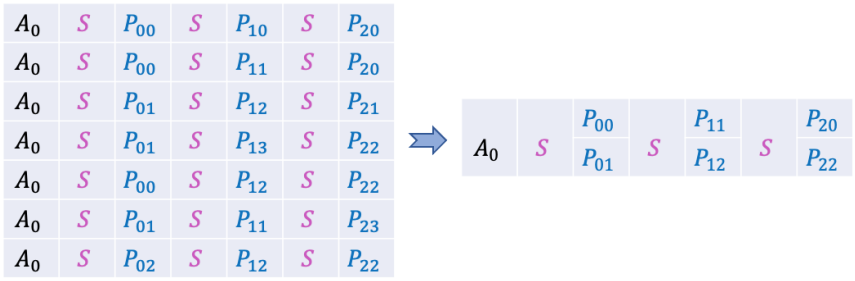


Figure 3.4: Example showing consensus calling with seven inferred oligos with the same address motif A_0 . The payload motifs are decoded as P_{00}, P_{01} at the first position, P_{11}, P_{12} at the second position and P_{20}, P_{22} at the third position which are the *topN* (N is the number of oligos in each sequence) frequent motifs in each column position.

3.3 Results

3.3.1 Encoding

In order to demonstrate the feasibility of composite motifs, we stored the text “HelloWorld” using our composite-motif-based DNA storage system. The sequence design rules for base motifs that are used to derive composite motifs are similar to those of DNA barcode design. Thus, we started with DNA sequences designed in prior work [62] to select 96 25nt base motifs. Using a combination factor of 32, we developed a composite motif set of 3×10^{25} composite motifs ($C(96,32)$). Thus, each composite motif, and hence, each synthesis cycle, can store 84-bits of data ($log_2 C(96,32)$). As our input text is 10 bytes, it can be stored using a single DNA sequence with one composite motif. However, in order to test precision and recall of methods in the read pipeline, we stored the same data eight times using eight sequences. We index each

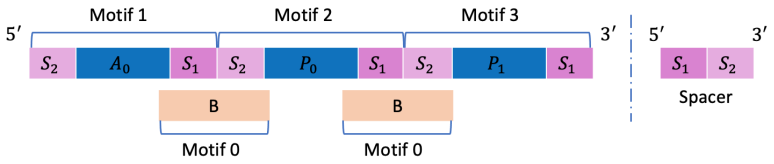


Figure 3.5: The general oligo structure design. A: address motif, P: payload motif, S: spacer, B: bridge.

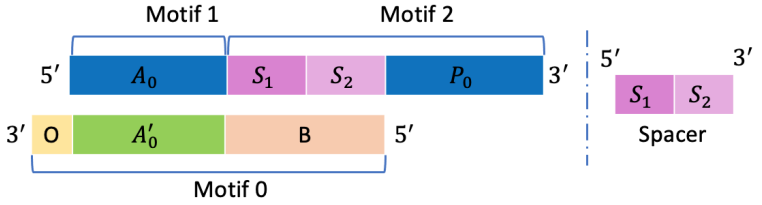


Figure 3.6: Bridged oligonucleotide assembly. (**a**) The general oligo structure design. (**b**) The experimental oligo structure design. A: address motif, A': reverse complement of A, P: payload motif, S: spacer, B: bridge, O: overhang.

sequence using eight unique 24nt address motifs that are separate from the 96 payload motifs.

3.3.2 Bridged Assembly of Composite Motifs

Each of the eight encoded sequences is then used to synthesize 32 oligos producing a total of 256 oligos. The address motif is repeated in each molecule, while the composite motif is expanded to generate a variant combination using 32 payload motifs. Oligos are synthesized using template-directed ligation. This method utilises single-strand sequences, referred to as *bridge* oligos, to facilitate the ligation of payload motifs to address motifs. In the general case, an oligo would contain one or more address and payload motifs as shown in Fig 3.2. As any motif can be ligated with any other, designing bridge oligos for each possibility is suboptimal and not scalable. We solve this problem by using a *spacer* motif. When the motif library is designed, each 25nt motif is extended on both 5' and 3' ends with 12nt and 13nt nucleotides from the 3' and 5' ends of the spacer motif (Fig 3.5). While this increases the length of each synthesized motif from 25nt to 50nt, it does not affect the number of motifs, and more importantly, it makes it possible to design the bridge oligo to be complementary to a single spacer. By doing so, the bridge oligos can hybridise to the spacer portions at the 3' and 5' ends of two payload motifs while the enzyme ligates them.

For the purpose of our experiment, as we have only 2 motifs per oligo, we modified this by (i) prepending the entire spacer sequence to the 5' end of each payload motif, and (ii) designing eight (instead of one) bridge oligos, each of which is complementary to both the spacer sequence and one of the eight address sequences (Fig 3.6). By doing so, the eight bridge motifs

also double in role as adapters during sequencing. The spacer-extended 32 payload motifs, eight address motifs, and eight bridge oligos were all synthesized base-by-base by Integrated DNA Technologies (IDT). The oligos were synthesized by selecting, annealing and ligating together the corresponding address–payload motif pairs. The inputs to the reaction comprise all motif oligos, bridge oligos, enzymes and ligation buffer. These reactions proceeded to produce ligated oligos through programmed temperature incubation and cycling, where each bridge oligo facilitates the ligation of a specific address motif with a payload motif via complementary annealing. We use the resulting oligo pool to test the feasibility of decoding the identity of motifs from an enzymatically-ligated, Nanopore-basecalled readout.

3.3.3 Error Characterization of Direct Nanopore Sequenced Reads

Despite having several reads, we found that the reads were low quality. From the read length distribution in Fig 3.7.a and Fig 3.7.b, we see that the median read length with Guppy and Bonito is 166nt and 110nt. Thus, more than half reads are 48% longer than original oligos as several reads were observed to contain multiple oligos in a single read. On further analysis, we identified wrong event detection by MinKNOW to be the root cause of the problem. When sequencing oligonucleotides on an ONT R9.4 flowcell, the movement of bases through the pore leads to a continual change in current, known as the "squiggle", that is recorded by MinKNOW. MinKNOW processes the squiggle into reads in real-time, and each read is supposed to correspond to a single strand of DNA. However, as our oligos were below 200 bases, we observed that sequencing our oligos generated low quality reads due to incorrect segmentation by MinKNOW which would earmark empty signals as valid reads, and created reads with merged squiggles for more than one strand of DNA.

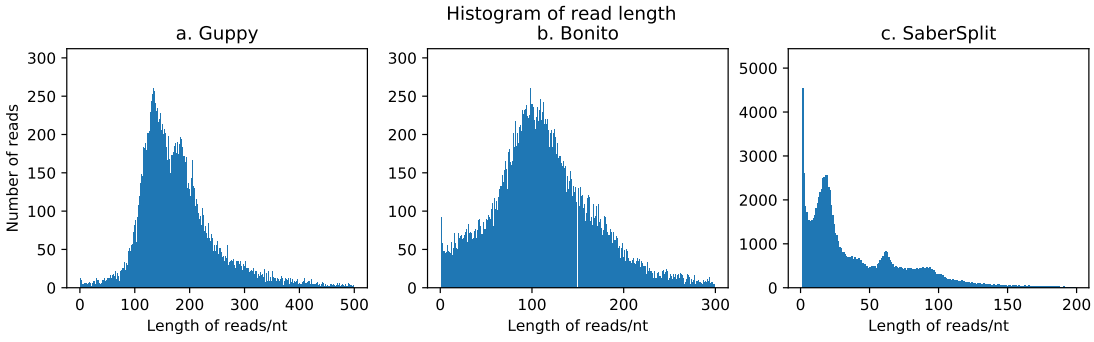


Figure 3.7: Distribution of read length with Guppy basecaller, Bonito basecaller and Bonito basecaller post-processed with SaberSplit.

Due to the presence of multiple oligos per read, we cannot directly align the reads to the reference oligos. So we did reverse alignment to study error characteristics and coverage distributions. We regard each read as a "reference" and build an index per read. Then, we treat

each oligo like a “read”, and align it to each reference. Thus, for each read, we get an alignment file that contains one record per oligo. To identify and retain only good alignments, we filter the alignments using the following criteria: (i) MAPQ > 10 (90% alignment confidence), ii) all alignments in a read should correspond to one orientation (no mixed forward and reverse alignments), and iii) there should not be any overlap when multiple oligos are mapped in a single read; only the alignment with the highest alignment score is kept if several alignments overlap each other. With this approach, we get the set of oligos that we can identify assuming we have full knowledge of the original oligos.

Using Minimap2 [30] for reverse alignment, we computed the substitution, insertion, deletion and soft-clipping rate per position (Fig 3.8). As can be seen, the rate of soft clipping is very high at the extremities (especially 3’ end)due to the very high error rate caused by BOA and DOS. In the middle portion of the read, the rates of error types vary, with no one error type being dominant over others. These results are in sharp contrast to error statistics published in prior work on DNA storage [7, 8, 35, 63, 64], where substitution errors have been shown to be more likely than indel errors, and overall error rates are at least 10× lower (Fig 3.9 and Table 3.3). The only exception is work on photolithographic synthesis [1], where the error rates reported were also high.

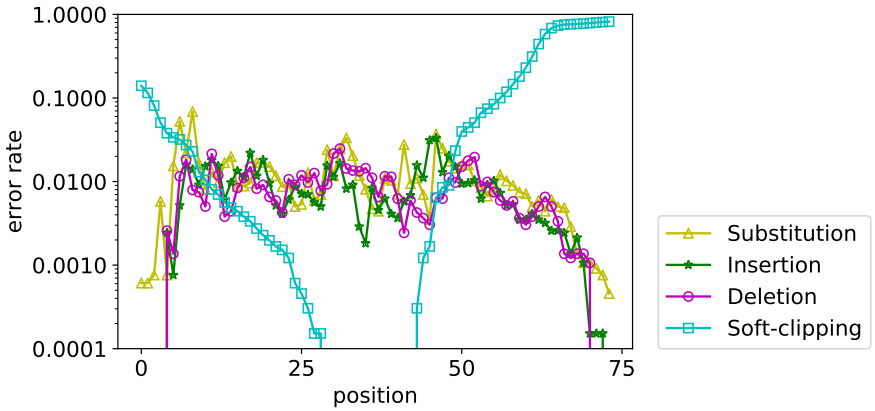


Figure 3.8: The substitution, insertion, deletion and soft-clipping rate per position of Guppy reads.

Table 3.3: The substitution (SUB), insertion (INS) and deletion (DEL) rate of SOTA work.

	Goldman et al.	Grass et al.	Erlich et Zielinski	Organick et al.	Antkowiak et al.	This Work
SUB	0.00088	0.005850	0.003870	0.005400	0.026000	0.011411
INS	0.00036	0.000230	0.000211	0.004500	0.057000	0.007817
DEL	0.00036	0.000230	0.000211	0.001500	0.062000	0.007485

The substitution (SUB), insertion (INS) and deletion (DEL) rate of SOTA work. Data for Goldman et al. [7], Grass et al. [8], Erlich & Zielinski [35], Organick et al. [63] and Antkowiak et al. [1] was taken from Antkowiak et al. [1].

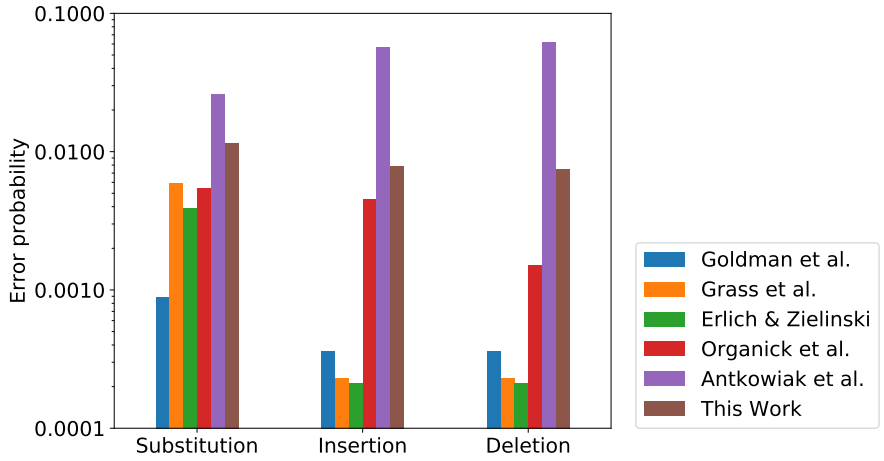


Figure 3.9: Comparison of errors in previous work.

3.3.4 Correcting Event Misdetection with SaberSplit

In the real DNA storage scenario, the original reference oligos must be inferred from erroneous reads automatically. Current read clustering and consensus callers used for this purpose assume that a read covers only a single oligo. To be able to use them, we developed SaberSplit, a tool that reduces the errors caused by incorrect segmentation by splicing squiggles to separate out reads belonging to different oligos. With SaberSplit, the original reads are chopped to 102,221 shorter reads of median length 25nt as shown in Fig 3.7. Then, we tried to use state-of-the-art clustering programs and position-wise consensus callers [65, 66] to infer the original oligos from both raw Bonito/Guppy reads, and SaberSplit processed reads. However, due to the high error rate, no oligos could be inferred in all cases.

To study SaberSplit further, we aligned the chopped reads to reference oligos with Minimap2. We compared the alignment statistics for raw Guppy, Bonito and Sabersplit processed reads in the Table 3.4. Guppy reads produced the highest number of alignments, with 102% more reads being aligned than Bonito. This could be explained by the fact Bonito is optimized to work better with longer reads, making it less suitable for short ones. Surprisingly, SaberSplit performed the worst with 9.5% fewer reads than even Bonito. This showed us that splitting reads amplifies the error rate and makes the case for a consensus caller that can directly work with raw reads covering multiple oligos.

Table 3.4: Statistic of the reads

	Nb. reads	Nb. aligned reads	Median read length
Guppy	27198	9960	166
Bonito	27198	4901	110
SaberSplit	102221	4434	25

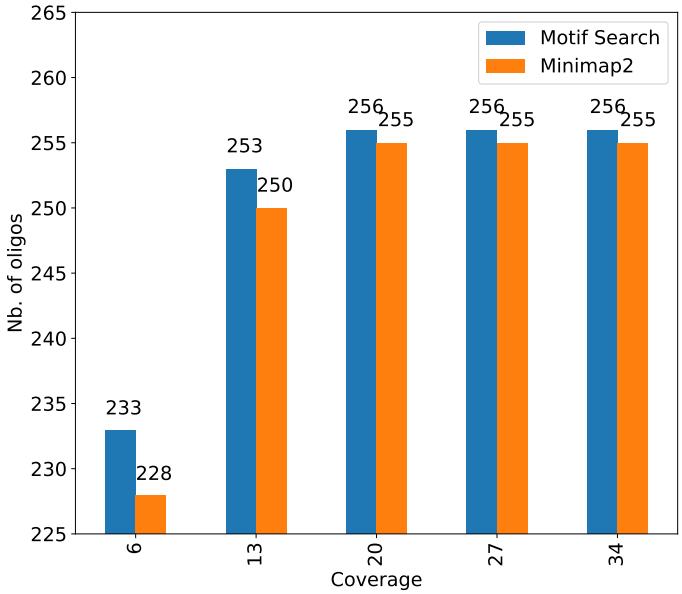


Figure 3.10: Number of oligos correctly reconstructed. Motif-Search fully recovers all oligos at 20× or higher coverage. Minimap2 misses one oligo even with 34× coverage.

3.3.5 Inference and Consensus with Motif-Search

To reconstruct the original data from noisy reads, we developed a new reconstruction algorithm called Motif-Search that meets two requirements:

- guarantee successful recovery despite high error rate,
- directly work with raw, basecalled, Nanopore reads that might contain multiple oligos per read.

Motif-Search differs from prior consensus callers that it is structure aware—while other callers view an oligo as a random collection of nucleotides, Motif-Search exploits the fact that our oligos are a collection of payload motifs separated by spacer motifs, with all motifs being drawn from a predefined, finite library. A detailed description of the Motif-Search algorithm is presented in Section 3.2.4. Here, we present our analysis results that demonstrate the ability of Motif-Search to accurately infer original oligos.

Fig 3.10 shows the true positive (TP) count (number of inferred oligos that are in the original set) of Motif-Search and Minimap2-based reverse alignment method at various coverage levels (lower sequencing coverage simulated via subsampling reads). It is important to note that

Table 3.5: Processing time (in second) for real dataset with 12 CPUs

	6×	13×	20×	27×	34×
Motif search exec. time	0.15	0.24	0.38	0.49	0.64
Minimap2 index time	13	27	41	55	68
Minimap2 align time	16	33	50	67	84
Reverse alignment exec. time	29	60	91	122	152

Minimap2 needs the original oligos which would not be available in the real DNA storage use case. Thus, Minimap2 results are used as a baseline for comparison rather than a real decoding solution. First, Motif-Search is able to fully recover all oligos at 20× coverage. Reverse alignment misses one oligo even with 34× coverage. Second, Motif-Search reconstructs more oligos than reverse alignment at all coverage levels. The under-performance of reverse alignment relative to Motif-Search is because all the reads covering the missing oligo had a very poor alignment and were filtered out.

Table 3.5 shows the execution time of Motif-Search and reverse alignment. Both support multi-threaded operation. On a 12-core Intel(R) Core(TM) i9-10920X CPU clocked at 3.50GHz, 128GB RAM with a 1TB SATA SSD, Motif-Search is 190–250× faster than Minimap2 due to the fact that Minimap2 needs to build an index for each read and align each oligo to each read while Motif-Search is custom-designed for the motif-based oligo reconstruction use case.

In order to investigate false positive (FP) behavior of Motif-Search and reverse alignment, we increase the motif library size. For a given set of address and payload motifs, we create oligos containing all possible combinations of motifs. For instance, if the motif set size is $64(address) \times 256(payload)$, we generate 16,384 possible oligos. We then use Minimap2 to align each oligo to each read. We use the same reads as before which were sequenced from 256 original oligos. As the motif set is expanded, Motif-Search can now report an inferred oligo which is not in the original set but from the expanded set, which would be labelled a FP.

Fig 3.11 shows the TP and FP counts for various expanded motif sets. First, note that Motif-Search is able to reconstruct all original oligos when sequence coverage reaches 27× for all motif set sizes. When the sequence coverage is low, Motif-Search is able to reconstruct more true positive oligos than reverse alignment even though it is unaware of the reference oligos. Second, as the motif set size increases, the number of FP for both approaches rise. Since the sequences are error-prone, both approaches make errors identifying the correct references from reads. However, the FP rate of Motif-Search is still lower than reverse alignment. While missing TP is an issue as it can lead to data loss, extra FP is not a problem as it can easily be discarded by using auxiliary metadata and/or error-control coding.

These results clearly demonstrate that (i) our motif-based, BOA method can successfully

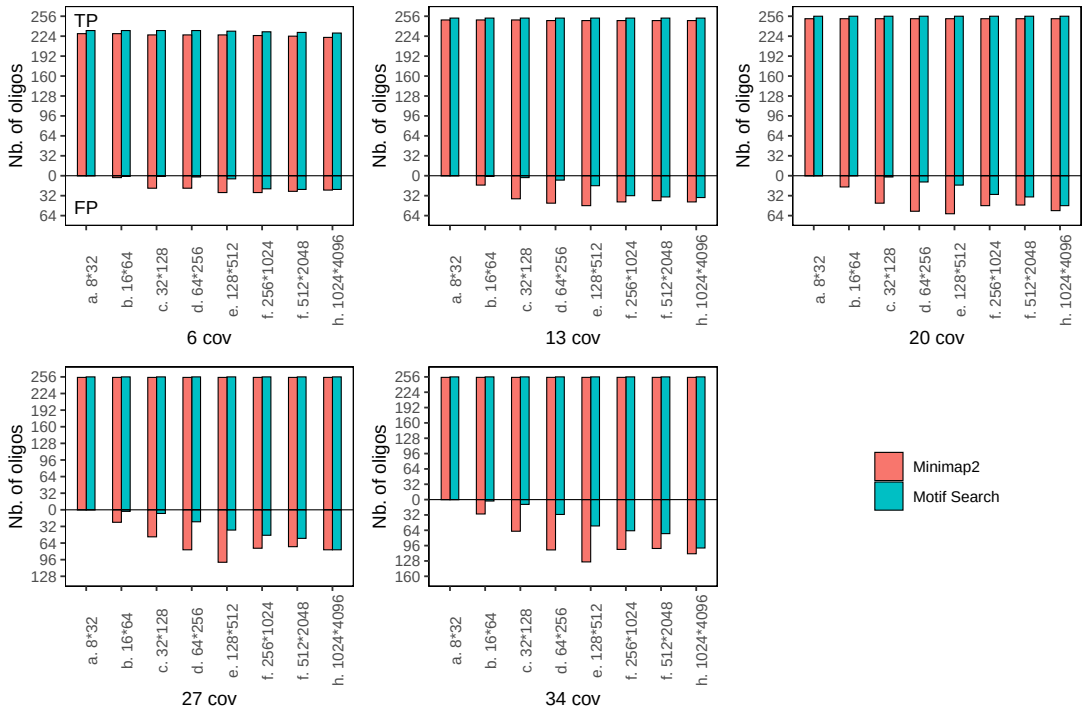


Figure 3.11: The number of true positive and false positive oligos reconstructed by MotifSearch and Minimap2 for different sequence coverages with expanded motif sets. i) MotifSearch reconstructs more true positive oligos than reverse alignment even without the knowledge of reference oligos. ii) False positive rises for both approaches when the motif set size increases.

encode information in DNA, and (ii) with sufficient coverage, Motif-Search is capable of reconstructing all original oligos, and thereby ensuring successful decoding, despite errors introduced by enzymatic BOA and DOS.

3.3.6 Read–Write Cost Comparison

The cost of storing data on DNA comes from two aspects, namely, the cost of sequencing for reading data and the cost of synthesis for writing data. Composite motifs has the potential to reduce the synthesis cost, thanks to the increase in logical density. For example, each synthesis cycle encodes 84 bits ($log_2 C(96,32)$) in our composite motif experiment. A native motif-by-motif approach, in contrast, can only encode 6 bits per cycle with the same 96 motifs, and the traditional phosphoramidite approach can encode 2–3.37 bits per cycle depending on whether standard or degenerate bases are used for encoding. This 14–42× increase in logical density will lead to a proportionate reduction in synthesis cost over conventional synthesis approaches, as fewer synthesis cycles and fewer oligos are required to encode the same digital data. Since current motif-based synthesis techniques already use a high degree of sequence multiplicity, composite motifs can be easily integrated by generating a variant motif mixture pool without much added costs. The physical density of our approach is 3.36bits/nt, which is also higher than the physical density of conventional base-by-base DNA storage solutions (2 bits/nt) and comparable to degenerate base approaches (3.37 bits/nt [57, 58]).

While our solution improves logical density and synthesis costs, it does so at the expense of higher read costs. Fig 3.12 presents a comparison of the cost to read 1MB of data stored in DNA of our approach and other related work [1, 7, 8, 35, 63, 67] based on the cost of DNA sequencing (0.006$ per megabase) reported by National Human Genome Research Institute (NHGRI) in August 2021 [5]. The detailed calculation is included in the Table 3.6. Clearly, our work increases read cost compared to prior work except Antowiak et al. This is expected, as these prior approaches to DNA storage are able to fully recover the data at much lower sequencing coverages of 5× and 10× due to (i) the use of low-error rate array synthesis and high-throughput sequencing with extensive library preparation, and (ii) the use of error-control coding. Our current work, in contrast, focuses on (i) tolerance to errors introduced by enzymatic ligation and direct sequencing without any library preparation , and (ii) complete recovery without additional error correction. As synthesis is approximately 80,000 times more expensive than sequencing [35] and the read cost continues to drop due to rapid advances in sequencing, we believe that it is more important to focus on reducing the write cost, which is a bottleneck today in DNA data storage.

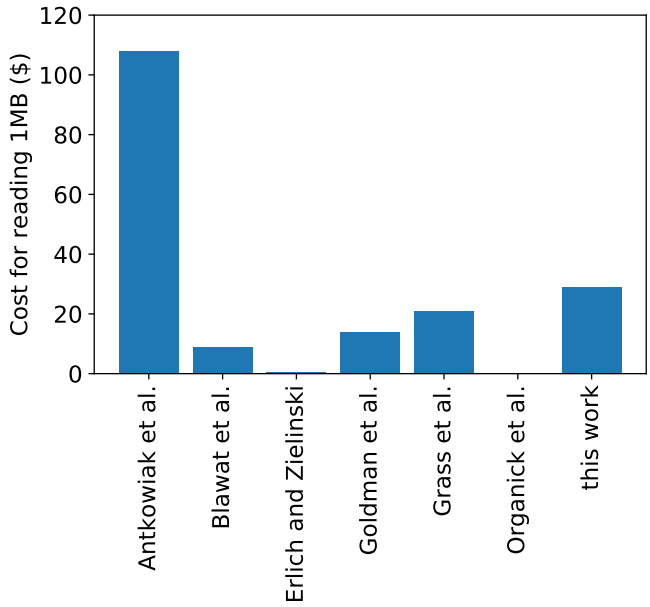


Figure 3.12: The cost of DNA sequencing to read 1 megabyte data. Our work increases read cost compared to prior work except Antowiak et al [1].

Table 3.6: Sequencing cost projection

	oligo length	nb of reads	data size	sequence cost ($/MB)
Antkowiak et al.	60	30000000	99103bit	108
this work	74	640	80 bit	29.8
Grass et al.	159	1858027	679000 bit	21.9
Goldman et al.	183	7960000	5200000 bit	14.1
Blawat et al.	230	144475005	22MB	9.06
Erlich and Zielinski	200	750000	2.11 MB	0.43
Organick et al.	150	67241860	200MB	0.3

- The sequencing cost to read 1 Megabyte is simulated as the Equation 3.2.

$$reading\_cost = oligo\_length * nb\_reads * sequencing\_cost\_per\_nt / stored\_data\_size \quad (3.2)$$

- The oligo_length includes the length of primers.
- The sequencing_cost_per_nt takes the value 0.006$ per megabase reported by National Human Genome Research Institute (NHGRI) in August 2021.
- Data for Goldmann et al. [7], Grass et al. [8], Erlich et al. [35] and Organick et al. [63] was taken from Organick et al. [63].

3.4 Discussion

In this work, we demonstrated the feasibility of using composite motifs to scale the logical density of DNA storage by an order of magnitude. We developed synthesis (BOA) and sequencing (DOS) methods customized for writing and reading oligos that regard composite motifs as building blocks, and showed that the error characteristics of these methods are different compared to state-of-the-art techniques. We developed a new motif-based consensus calling and oligo inference method (Motif-Search) that is able to recover all data at coverage as low as 20×. Our future work aims to scale up the methods presented in this paper on several fronts. First, to simplify the task of motif design, we built on an existing library of 25nt primers leading to a physical density of 3.36bits/nt. Future work will improve this further by optimizing the motif library. Second, we are working on reducing sequencing costs by adding error-control coding optimized to our DNA storage channel to enable data recovery at a lower sequencing coverage. Third, the short size of motif library, the library-preparation-free sequencing provided by DOS, and the error-tolerant nature of Motif-Search all simplify end-to-end automation. Thus, we are developing a fully automated DNA storage solution that can scale both oligo length and number of oligos beyond what was presented in this work.

Chapter 4

Sequence Analysis for Random Access in DNA-based Storage

4.1 Introduction

The end-to-end DNA storage workflow shown in Figure 3.1 demonstrates that the synthesized DNA oligos have to be sequenced and decoded back into digital data in order to extract the stored information. The prior work [7, 8, 35, 67, 68] has made an effort to sequence and decode the entire amount of stored data. However, in many real scenarios, not total information but only a fraction of information is expected to be read out. For example, clients request one single table from one unique database, or extract one image from a collection. To response these requests, it is redundant to sequence the entire amount of stored information. Also, sequencing more reads usually leads to the extra decoding time. To decode the information stored in the DNA-based storage, the decoder applies consensus calling to aggregate reads originated from the same oligo together. This step is computationally intensive due to the adoption of clustering and machine learning techniques. The redundancy introduced by sequencing and decoding makes the DNA-based storage impractical when the amount of data increases. Hence, random access is playing an important role to make large scale DNA-based storage viable.

One common approach to realize random access is based on Polymerase Chain Reaction (PCR). It is proved to be scalable enough to extract files of varying complexity and size reliably [63]. The PCR-based random access appends address sequences in each oligo such that PCR reactions can use these addresses as primer targets to selectively amplify the desired strands from a large pool of oligos. With this approach, the first step is to segment input data into chunks and encode as oligos with fixed lengths appropriate for the current DNA synthesis and sequencing platforms. Then, address sequences are appended to individual data-encoding oligos in the storage pool. During PCR reactions, these addresses are used as primer targets and only the desired strand are selectively amplify. Thus, by associating addresses, this approach can be used to randomly retrieve a fraction of data from a DNA pool.

Figure 4.1 shows a PCR-based random access example which selectively retrieves the strands started with the primer A'. Ideally, we desire to only have the strands starts with target primer A' in the final pool, such as the strand #1 and strand #2. Nevertheless, due to the improper designed primers or PCR procedure, sometimes the primers can bind to the improper location and make the copies of the wrong oligo. This phenomenon is called mispriming or improper binding. For instance, the primer B' in the strand $B' - B$ is attached to primer A' and it leads to extra strands (strand #3 and strand #4) in the final pool.

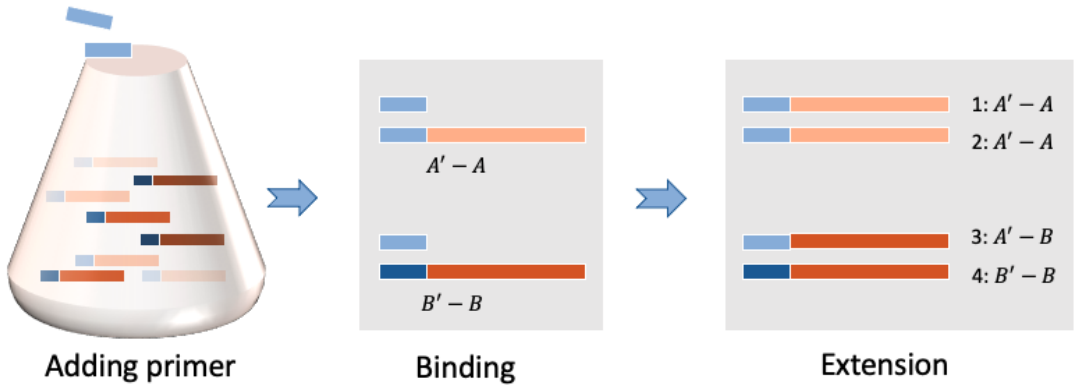


Figure 4.1: PCR-based random access example.

Most studies associate the address sequences for individual file which is an uni-dimensional one-primer-per-file approach enabling to randomly retrieve a single file from a DNA pool. However, such uni-dimensional approach is not scalable for large DNA databases due to several major limitations.

First, it is mandatory to take into account the biological restrictions during primer design. Given the primer length of N, it is possible to design 4^N primers theoretically. However, it is important to take into account following aspects to ensure the specificity of random access in large DNA databases.

- The GC content, which is the percentage of bases in a sequence that are either guanine or cytosine. Synthesis and sequencing errors are known to be exacerbated by either an excessive or inadequate GC content.
- Homopolymers can introduce a higher error rate during the synthesis and sequencing. The primers with single-nucleotide repetitions should be avoid [69, 70].
- The hairpin loop, which is the DNA molecule fold back on itself if several adjacent bases are complementary to each other. The hairpin structure should be eliminated in the primers.

- The primers are distinct from each other. The edit distance among the primers should be large enough to minimize the mispriming.

These biological restrictions further limit the number of possible primers within a particular length budget.

Second, one-primer-per-file approach is not applicable for an archive with many small files. For instance, assuming a DNA archive with 1 million small files of size 1MB, 1 million of distinct primers need to be designed. Given the biological limit, the primer should contain at least 10 ($log_4 1000,1000$) nucleotides to produce 1 million distinct primers. As we mentioned earlier, the oligo is limited to few hundred nucleotides in length due to the synthesis limitation. When the number of nucleotides in the primer increases, the number of nucleotides remaining available to store the payload data decreases. Massive small files lead to the increment of primer length dominating a non-significant portion of the oligo. As a result, it impacts the storage density.

Third, it does not allow the reuse of primers. Hence, it limits in terms of the type of access paths that can be supported, as only one file can be randomly accessed at a time. It requires multiple rounds of PCR with corresponding primers to retrieve a set of related files which is costly and inefficient.

4.2 Multi-dimensional Data Addressing

To overcome the limitation of uni-dimensional data addressing, we propose a multi-dimensional data addressing methodology inspired by prior work [71, 72, 73, 74]. In our design, a pool of data oligos is organized in a three-dimensional hierarchy **Collection – Object – Extent**. For instance, one can store a file system by mapping a directory to a collection, a file to an object, and a chunk of a file to an extent. Similarly, a relational database can be stored by mapping a database to a collection, a table to an object, and a column of a table to an extent.

Data is encoded to represent this hierarchy by having each oligo containing two components, namely, a data payload and a collection of access primers as shown in the Figure 4.2. The data payload refers to the quaternary-encoded input binary data that needs to be archived in DNA. It is similar to previous encoding design and do not provide any random access capability. The access primers, in contrast, are specifically added to achieve complex access paths.

Figure 4.2: Oligo structure.

Irrespective of how data is grouped in the Container–Object–Extent hierarchy, the encoding process must ensure that different primers are used to distinguish oligos at each level of the hierarchy. However, within any level of the hierarchy, it is possible to reuse primers. For example, let us consider a database archival use case where we map a database to a Container, a table to an Object, and a column to an Extent. Different databases archived together must use different Container Target Primers (CTP). Similarly, different tables within a database must use different Object Target Primers (OTP). But tables across databases can use the same Object primer. Similarly, different columns within a table must use different Extent Target Primers (ETP), but columns in different tables can reuse an Extent primer.

Given N primers, our design could encode $N \times (N-1) \times (n-2)$ random accessible units while the uni-dimensional data addressing could encode N random accessible units. This generic hierarchy scales the random access ability and is sufficient to express several higher-level access paths in the future. Arranging data this way enables several helpful access paths:

- All objects belonging to a collection can be retrieved using a single PCR round with the right CTP.
- All extents from a specific object in a specific collection can also be retrieved in a single PCR round by using the appropriate CTP–OTP pair.
- One specific extent can be retrieved using a nested PCR round by first using CTP–OTP pair to amplify extents belonging to an object, and then using the CTP–ETP pair to amplify a specific extent.

In this work, we want to study the effectiveness of multi-dimensional addressing. First, we start by considering file based addressing which attach the primers per file. We show that it suffers from mispriming and PCR bias. The data is either under or over represented after the PCR. Then, we propose the block-based addressing for DNA-based archives to solve these problems. The primers are elaborated designed and attached to each fixed size block. It is shown that the mispriming and PCR bias are eliminated in the block-based addressing experiment.

4.3 File-based Random Access for Databases

4.3.1 Database Design

In the file-based random access experiment, the primers are attached to each file. In order to test file-based addressing, we stored three databases, namely SSB, TPCH and SYN. The databases contain 5, 8 and 8 tables with 17 columns respectively. The resulting oligoarchive

has 5258 oligonucleotides of 110 bp as shown in Table 4.1. It is intentionally designed the SSB and TPCH databases with variable table size while SYN database with uniform table size.

Table 4.1: File-based random access databases

SSB tables	nb of oligos	TPCH tables	nb of oligos	SYN tables	nb of oligos
ssb-supplier	14	tpch-region	6	syn-t7	304
ssb-customer	16	tpch-part	18	syn-t5	312
ssb-part	42	tpch-orders	18	syn-t8	302
ssb-lineorder	2594	tpch-partsupp	10	syn-t3	302
ssb-date	34	tpch-nation	20	syn-t2	298
		tpch-supplier	14	syn-t6	300
		tpch-lineitem	34	syn-t1	298
		tpch-customer	16	syn-t4	306

The oligonucleotides is structured from the concatenation of 5'-Universal Forward Primer-Table Primer-Payload-Column Primer-Database Primer-Universal Reverse Primer-3'. The Universal Forward Primer (UFP) and Reverse Primer (URP) are used for sequencing. Rest of the primers are used as the address sequences to selectively amplify the desired strands. The rest of the oligo sequence corresponds to the payload.

The longer the primers are, the less nucleotides are left in the oligo for the payload. Therefore, we start to work with short primers to maximize the payload proportion. In our first experiment, we chose primers of length 5nt derived from the Illumina adapter sequences to define the database, table and column information. The primers are further selected to meet the two requirements.

- GC content is between 0.4 and 0.6.
- Absence of hairpin structure. The primers should not contain complementary nor inverse complementary sequence.

4.3.2 PCR Bias

As we mentioned earlier, both the synthesis and sequenceing steps introduce errors in DNA-based storage. Multiple reads covering each oligo enables to infer the original oligo during consensus calling despite errors. Suppose that PCR bias exists, especially when some information is underrepresented, the corresponding data might be absent after decoding. This brings recall of random access down. Hence, it is essential to study whether the selected data are equivalently present before and after PCR amplification during the multi-dimensional data addressing.

We begin the PCR bias study by defining the **population fraction** [75] as the individual oligo's share of the whole pool. The population fraction of a sequence i after k cycles of PCR among n sequences, p_i^k, is computed as

$$p_i^k = \frac{N_i^k}{\sum_{j=0}^{n} N_j^k} \tag{4.1}$$

where N_i^k represents the number of sequence i after kth PCR cycle. When k equals to 0, p_i^0 is the raw population fraction representing the initial oligos' proportion of each population. Then, we define the **population fraction change** as the quotient of population fraction after the PCR amplification divided by the raw population fraction as

$$c_i^k = \frac{p_i^k}{p_i^0}. \tag{4.2}$$

The PCR selection is considered as unbiased when the mathematical expectation of population fraction change for all sequences is one. That means all sequences are equivalently present after the PCR process.

PCR Bias at Database Level

First of all, we want to study PCR bias at the database level, i.e. whether the population fraction change uniform across databases. The Universal Forward Primer was used to pull out all the reads. Then we used Accel-Align to map reads to oligos to determine their original database. The number of reads extracted per database is shown in Table 4.2. We find that the population fraction of TPCH is only 1.2% after the PCR amplification while the original population fraction of TPCH is 2.6% over all. It indicates that TPCH database is insufficient represented after the PCR. On the contrary, SYN database's population fraction change is 1.3, meaning that it is 30% excessively represented. These results illustrate that PCR is biased, some sequences become overrepresented while some become underrepresented after the PCR amplification.

Table 4.2: Population fraction change

	nb of oligos	raw pop fraction	nb of reads	pop fraction	frac change
ssb	2700	0.514	654335	0.388	0.76
tpch	136	0.026	19576	0.012	**0.45**
syn	2422	0.461	1013152	0.601	**1.30**

PCR Bias at Table Level

We extend the PCR bias analysis further to the table level. The UFP was used together with a database primer to pull out a database. Then we used Accel-Align to map reads to oligos to determine their original table. Table 4.3 shows the population fraction change per table from three databases.

- TPCH is a small database of only 136 oligos in total. The population fraction change is different for different tables. For example, the part, orders, supplier and customer tables' proportions increase 32×, 4×, 3×, 5× respectively after PCR while region, nation and lineitem tables' proportions decrease 40%~60%.
- SSB is a bigger database with 20× more oligos than TPCH (2700 in total). The big table lineorder's fraction change is 1.01 which means it only has 1% excess. However, population fraction change of the small tables, such as supplier and date, is only 0.01 meaning their proportion decreases 100× after PCR. This indicates that even in a bigger database, if the data is not uniformly distributed through the tables, the small table would also be biased.
- On the contrary, SYN is also a bigger database with 2422 oligos while the data is uniformly distributed through tables. The mathematics expectation of population fraction change is 1.03 which is much more close to 1 compared to TPCH's (0.72) and SSB's (10.86).

To conclude, we learn that the big tables with uniform size will exhibit less variation in population fraction change. It indicates that file-based random access can suffer from PCR bias issues if file sizes are not uniform. To overcome this problem, we introduce a block-based random access in Sec 4.4.

Table 4.3: Population fraction change of each database

	nb of oligos	raw pop fraction	nb of reads	pop fraction	frac change
tpch-region	6	0.044	38408	0.06976	0.63
tpch-part	18	0.132	2238	0.00406	**32.56**
tpch-orders	18	0.132	15572	0.02828	**4.68**
tpch-partsupp	10	0.074	38385	0.06971	1.05
tpch-nation	20	0.147	194693	0.35360	0.42
tpch-supplier	14	0.103	14995	0.02723	**3.78**
tpch-lineitem	34	0.250	235239	0.42724	0.59
tpch-customer	16	0.118	11072	0.02011	**5.85**
ssb-supplier	14	0.005	26	0.00003	**0.01**
ssb-customer	16	0.006	4512	0.00483	0.81
ssb-part	42	0.016	25045	0.02680	1.72
ssb-lineorder	2594	0.961	904850	0.96818	1.01
ssb-date	34	0.013	155	0.00017	**0.01**
syn-t7	304	0.126	168749	0.12366	1.02
syn-t5	312	0.129	182311	0.13360	0.96
syn-t8	302	0.125	198241	0.14527	0.86
syn-t3	302	0.125	113461	0.08314	1.50
syn-t2	298	0.123	172734	0.12658	0.97
syn-t6	300	0.124	176060	0.12902	0.96
syn-t1	298	0.123	167900	0.12304	1.00
syn-t4	306	0.126	185190	0.13571	0.93

4.3.3 Extra Data and Improper Binding

PCR-based random access is based on the primer binding such that only the desired strands are selectively amplified. However, improper binding can lead to extra strands (strand #3 and strand #4) in the final pool as presented in Figure 4.1. During the decoding, it is still possible to get rid of strand #4 filtering the decoded strands by the decoded primer. However, it is hard to get rid of strand #3 because it does have the primer A' just with a wrong payload. It confuses the decoder since it introduces a new strand $A' - B$ that not existing in the original pool. If the decoder faithfully relies on the primer, this portion of data would be decoded into wrong database or table. Thus, the improper primer binding during PCR leads to **silent data corruption**, where data from database or table is silently classified as belonging to another.

To study the quality of selected reads, it should determine the payload's and the primer's origin. Since the payload part dominates the read in length, we could align a read to the designed reference oligos with Accel-Align to determine where the payload comes from. Consequently, the reads are classifies into three categories depending on the alignment result:

- Target payload reads. These are the reads that align to the expected reference oligos.
- Other payload reads. These are the reads that align to the other reference oligos rather than the expected ones.
- Bad quality reads. These are the reads that could not align to any reference oligo.

To comprehend the origin of primer, we study whether the sub-sequences in the two extremities of a read are as same as the target primers. Thus, we further classify the target payload reads and bad payload reads into three sub groups as following.

- Target primer reads. These are the reads whose primers are exactly as same as the target primers.
- Other primer reads. These are the reads whose primers are exactly the other primers rather than the target primers.
- No primer reads. These are the reads who do not have any primer.

Since the reads were sequenced by Illumina in paired-end mode, the two mates' lengths are 100 bp and 60 bp without guaranteeing the coverage of two extremities. So we merge two mates into a single longer strand through the bbmerge tool from BBMap [76] toolkit. The merged strands' median read length is 110 bp which is exact as same as the original oligo design. Hence, we could do the primer classification depending on the first and last five characters of the reads.

Error Analysis at Database Level

To begin with, we analyze the sequenced reads when we selectively retrieve data belonging to each database using the associated database primer as shown in Figure 4.3, Figure 4.4 and Figure 4.5. We have several findings:

- In the ideal case, we expect to have a high precision which means the data from other databases to be absent. We find that the databases are diverse in terms of their random access precision. Clearly, with the SYN database primer, PCR-based random access achives 96.6% of reads with target payload and only 0.9% reads belonging to other databases (TPCH or SYN)are retrieved. In contrast, with the TPCH primer, only 53.3% are belong TPCH database while 32.5% reads belonging to the other databases are retrieved.
- For the reads with target payload, 99.3%~99.4% reads have the target primer. 0.1%~0.2% reads contain primers of other databases and 0.5%~0.7% reads do not contain primer. This is because we use exact match strategy to determine the origin of primers rather than alignment since the primers are two short (5nt). However, the reads can contain errors which leads to no exact match or other primer matched for the corresponding portion of the reads. Even though, we still could find that the dominate target reads do select by the target primer.
- For the reads with other payload, however, 48.0%, 18.8% and 72.5% reads have the target primer surprisingly for SSB, TPCH and SYN database as shown in Figure 4.3.c, Figure 4.4.c and Figure 4.5.c. These extra reads are selected by the target primer because of improper binding and can lead to the extra data if the decoder faithfully relies on the primers.

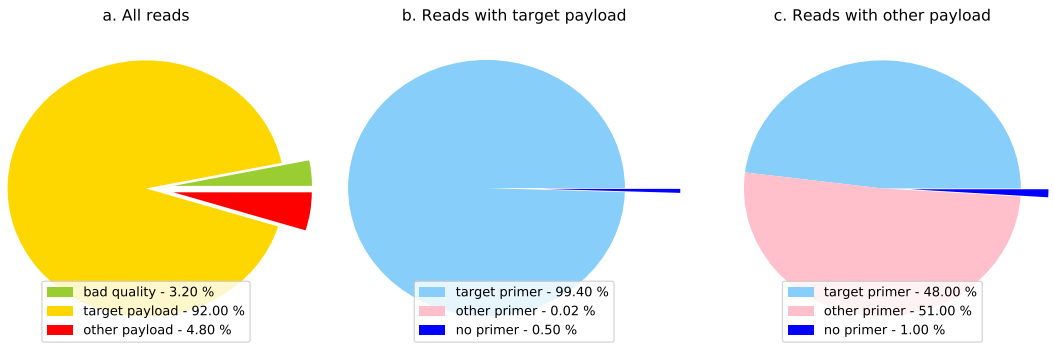


Figure 4.3: Reads selectively retrieved with SSB database primer.

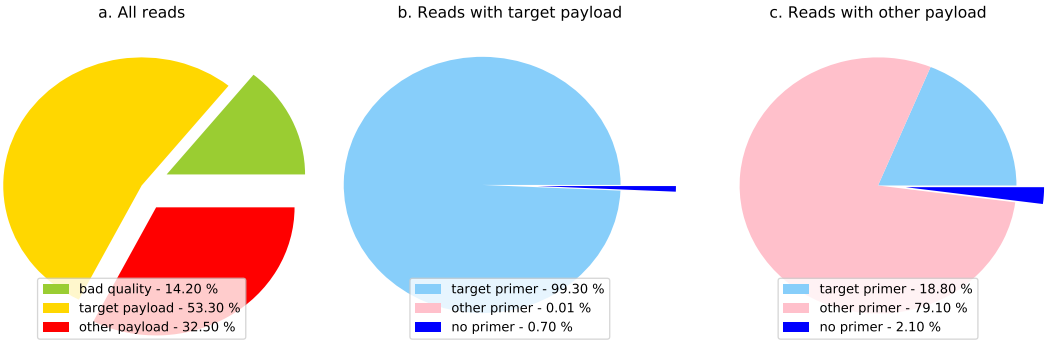


Figure 4.4: Reads selectively retrieved with TPCH database primer.

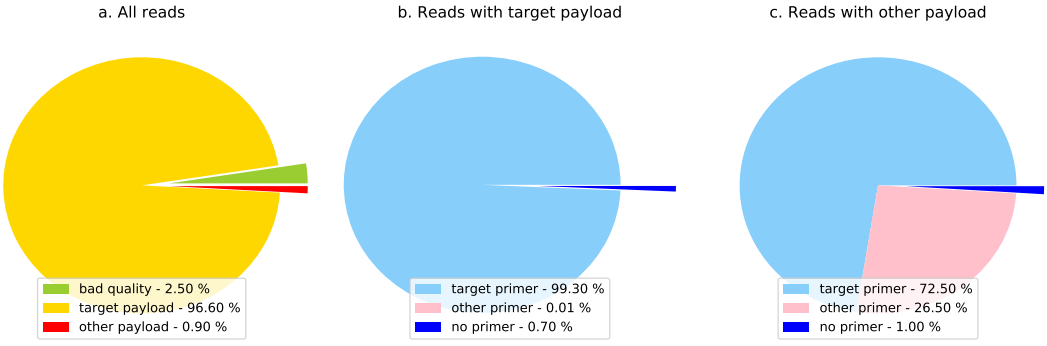


Figure 4.5: Reads selectively retrieved with SYN database primer.

Error Analysis at Table Level

We repeat the same analysis at table level. For the reads selected with UFP and an unique table primer, Table 4.4 shows the percent of each category depending on the payload's origin and primers' origin.

- Again, we find that the tables have different random access precision. We plot the random access precision over the number of oligos of each table and database in Figure 4.6. Usually, the bigger size is the table or database, the higher precision could be achieved.
- Similar to the result at database level, with short primers of 5 nt, extra data could be select by PCR at table level. Specifically, improper primer binding and low distance separation between primer can lead to poor precision. It indicates that it is important to not rely only on primers for addressing information. We suspect that higher-level type and data checkers, such as structural information and independent stored checksum, could help to avoid the silent data corruption. Although these invalid data could be tracked by the database schema and type information luckily, it is not general for file storage.

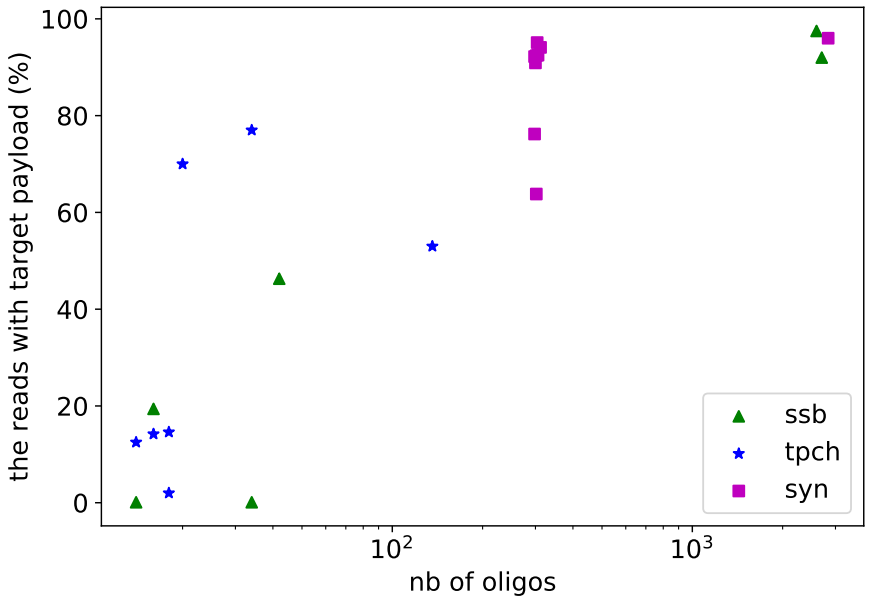


Figure 4.6: The relationship between random access precision and size.

Table 4.4: Reads selectively retrieved per database and table primer analysis (%)

	bad quality	target payload target primer	target payload other primer	target payload no primer	other payload target primer	other payload other primer	other payload no primer
ssb	3,2	91,5	0,02	0,5	2,3	2,5	0,05
tpch	14,2	53,0	0,01	0,4	6,1	25,7	0,69
syn	2,5	96,0	0,01	0,6	0,7	0,2	0,01
ssb-supplier	12,5	0,1	0,00	0,0	19,8	65,2	2,39
ssb-customer	11,6	18,9	0,00	0,6	13,7	51,7	3,53
ssb-part	5,7	44,6	0,01	1,7	6,2	39,1	2,70
ssb-lineorder	1,9	93,9	0,02	3,7	0,2	0,3	0,03
ssb-date	3,9	0,1	0,00	0,0	76,6	16,6	2,77
tpch-region	12,0	34,2	0,00	0,4	7,1	44,5	1,86
tpch-part	30,5	2,0	0,00	0,1	8,4	55,6	3,54
tpch-orders	19,6	14,1	0,00	0,5	7,5	55,2	3,14
tpch-partsupp	9,2	21,5	0,00	0,9	9,7	55,4	3,31
tpch-nation	3,6	68,0	0,02	2,0	4,3	20,9	1,25
tpch-supplier	14,2	12,4	0,00	0,1	5,7	65,4	2,18
tpch-lineitem	6,7	76,2	0,01	0,8	2,3	13,5	0,52
tpch-customer	21,2	14,1	0,00	0,1	9,6	52,6	2,36
syn-t7	2,1	93,8	0,01	1,3	0,1	2,7	0,08
syn-t5	2,3	91,0	0,01	3,1	0,1	3,3	0,24
syn-t8	2,0	89,6	0,01	2,8	0,2	5,1	0,28
syn-t3	3,9	61,0	0,03	2,8	4,7	26,4	1,19
syn-t2	2,2	74,0	0,02	2,2	8,3	11,1	2,12
syn-t6	2,0	89,8	0,01	1,1	0,1	6,9	0,22
syn-t1	2,1	91,2	0,01	1,1	0,1	5,3	0,16
syn-t4	2,2	91,2	0,01	1,3	0,1	4,6	0,61

4.4 Block-based Random Access for Databases

4.4.1 Database Design

In Sec. 4.3, we have introduced the limitation of file-based random access with the wet experiment. First, it demonstrates that the data corresponding to the small files suffer from population fraction variation after the PCR. And also they are exploited to a lower precision. To overcome the obstacle, we propose a block-based random access reminiscent to the blockwise storage in the hard disk. In the block-based design, the basic random access unit is a block which is an extent with fixed size in our multi-dimensional data addressing, rather than the file with variable size. Consequently, the unique primers are attached to the oligos belonging to each block rather than each file.

Primer Design

We start with a collection of 624 sequences as a concatenation of one 5'-primer of 20 nt, one payload of 256 nt, and one 3'-primer of 20 nt. The rationale is to select 5'- prime and 3'-prime 20-mers as divergent as possible in order to minimize the risk of cross-amplification. The initial primer sequences are from Organick et al [63]. It has already taken into account the GC-content (45%-55%), the absence of long sequence-complementarities, absence of long

stretches of homopolymers, and a minimum Hamming distance of 6.

We modify them such that the most extreme tri-nucleotide, in direct contact with the payload, are unique to each primer, in order to minimize the chance of non-specific annealing. The similarity between two primers is also minimized, by allowing no stretch of more than 8 identical nucleotides in common between the sequences. The selected primers are listed in the Table 4.5.

Table 4.5: Primer list (24*26)

left(5') primer list	right(3') primer list
L0:GTCCAGGCAAAGATACAGTC	R0:CAAGATTGAGGACGATTGGC
L1:TAGCCTCCAGAATGAGACAG	R1:TTCAAGCCAGACCGTGTGTA
L2:CATGGACGTTCCGCAATCAT	R2:GAACGGAGCGATGAGTTTGT
L3:AATGTCGAAGAAGGCCGGTT	R3:ATCCCGTTCCGAAGTTTCCA
L4:TGTATTTCCTTCGCAGCACC	R4:TAGAGAGCGTGGACAGACAA
L5:AGCCGACAAGTTCCAACACA	R5:TCCCGATAAGTCTTGCGGAA
L6:TGTCGGAAGCCGCTTTCTAT	R6:GGCAATGATTCCGTCGGTTT
L7:AGTCCAACAAGTCAATCCGC	R7:GCGGTAACGTAGTGAAGGTA
L8:CTGTCCATAGCCTTGTCCTT	R8:AGGTGAGTGCCGTAACGATT
L9:ACATGCCGTGCCATTGGATT	R9:TCGGCAGATCGTTCCACATA
L10:AAATCCTTTGTGCCTGCCAC	R10:CGGCACCGATTCGTAACAAT
L11:TACCGCATCCTTATTCGAGC	R11:TGTACCATCCGTTTGACTGG
L12:AATCATGGCCTTCAGACCGT	R12:CCGACCGTTAGTCTAAAGTG
L13:TGGCTCAGTTCACAATCGGT	R13:TCTGGTGCAACCCAATGAGA
L14:ACCGCGCTCGAAGCATTTAA	R14:AATCGAATGCTTGCTTGCCG
L15:CAAGTTACCGCCAACAACTG	R15:CCAATTAGTTGGCGCTTCCT
L16:CAATGGATGCCTTGTGCGAA	R16:CTAAATGACCTGCCGTGCAA
L17:AAGGCAAGTTGTTACCAGCG	R17:CATGTAGGCGGAAAGTGCAA
L18:TCCTGCTTGCGTTAGATGGA	R18:ACATAACAACCACCGCGAGA
L19:TTAATCGGTAACACCTGCGG	R19:ACCGTGCTTCACACCGAATT
L20:AGCCTTGTGTGCATCAATCC	R20:TTTCGACAAGGGTCTGGTCT
L21:GAAGAGTTTAGCCACCTGCT	R21:AGAGCCGTGGCAATGTAACT
L22:TCCATTGCGTCAACCGTGAT	R22:TTGTACGAAAGGTGCTCCGA
L23:TCCACGGTTCCTTGATTTCG	R23:ATGGCACTTATTGGGCGACT
	R24: AAGGCCAATTCGCGGTTAGT
	R25: ACGCAATCGGCCTGGTATTT

Further, We expect to select the primers with high significance. To study the significance, we do the following steps with Accel-Align:

- Align the reads to the payloads to determine the reference payload of each read.
- Conduct the substrings in the read that corresponding to the primers based on the CIGAR value computed from first step.

- Align the substrings to the left and right primers to determine the origin primers of each read.

Thus, the reads belonging to each primer pair could be further classified into two categories:

- Specific reads. These are the reads whose primers are as same as the target primers attached to the payloads in the reference oligos.
- Non specific reads. These are the reads whose primers are different from the target primers attached to the payloads.

The result is shown in Figure 4.8 and Figure 4.7. It shows that the number of non specific reads is no more than few hundred while the number of specific reads could be hundreds or tens of hundreds. To maximize the data to be recovered, the more specific reads the better. Thus, we select the L0, L1, L5, L9 and R1, R2, R8 and R24 as the validated primers to extend further storage collection. As highlighted in Figure 4.7, all the cross pairs have more than 17,000 specific reads and no more than 17 non specific reads. It guarantees that more than 99.94% reads are specific in the validated primer pairs.

	l0	l1	l2	l3	l4	l5	l6	l7	l8	l9	l10	l11	l12	l13	l14	l15	l16	l17	l18	l19	l20	l21	l22	l23
r0	928	5765	16641	14039	11246	7810	13461	14275	17540	67639	21983	19841	8688	1326	12616	15745	5470	5590	39544	14089	15447	5045	30961	7476
r1	***39288***	***21575***	2090	47829	18670	***34529***	22691	22221	24124	***30804***	8417	23049	3	2800	14741	28082	4715	10339	6426	12621	6626	10374	23443	16880
r2	***21493***	***36311***	35273	3853	39237	***65640***	27973	15479	16132	***17857***	20407	24860	20892	2539	14160	27026	7356	11528	6146	16755	4217	4966	18978	10093
r3	13379	31585	4826	11583	14337	26284	19253	12146	12337	18369	18636	8401	33340	1259	8357	13298	4850	3000	5735	2319	18464	14950	14349	13179
r4	8575	16724	1864	13899	3784	1239	20463	2991	2816	13719	1438	12183	13848	1142	6875	7273	6217	6366	5552	3126	5323	4873	30854	12655
r5	6325	2166	26855	13441	16480	18746	5804	2556	6619	7315	12689	8948	3842	684	11904	343	3496	12838	5333	1749	1805	6076	14279	7577
r6	5320	5247	20255	36779	27450	22628	11356	3789	13536	39215	16171	10144	21926	1530	24078	25768	11067	18957	11374	9524	7661	6969	26144	17965
r7	4201	3654	7269	15997	5926	6032	4530	3329	9293	6493	14274	7129	1123	571	10463	367	4875	6153	4745	3667	818	13331	7809	8273
r8	***24013***	***40572***	41504	9641	5289	***18741***	13325	9097	12415	***46686***	15984	3177	10331	5509	16836	11068	12284	12356	9443	533	10542	8362	41219	51943
r9	19294	6798	2881	14462	3955	833	10555	9248	11722	5928	10500	4932	7872	2356	10102	70	4494	5423	1931	3694	6493	1803	5980	14575
r10	8593	8433	9541	976	2991	16764	6654	5572	9586	6879	7810	9210	15268	1585	11520	9996	11802	6878	6120	8213	7498	1860	13077	2383
r11	9772	4706	6310	4300	1901	8851	1531	8235	4126	4566	13723	4722	29918	1004	13007	18038	4421	4187	7644	4739	4875	8472	11549	6804
r12	2754	4078	3127	14094	5051	4586	3493	3412	6272	3516	4710	4849	10071	2754	5142	4703	3485	5784	4934	3954	9631	7736	2739	18844
r13	5392	16507	3524	8741	2706	12637	3057	4915	3566	6293	23424	5656	12947	1497	5066	5336	8172	3502	10355	3791	9303	5686	10175	29020
r14	8482	16665	3397	18999	4277	23913	4328	6892	13983	7913	5372	4010	14899	1523	4267	12273	5169	4688	17284	3619	28695	19092	18538	13771
r15	74	38	107	380	34	108	54	158	95	170	41	139	105	38	81	76	100	80	117	67	99	87	160	173
r16	3737	9920	25622	6761	6678	8575	5020	7094	3303	8562	6636	11633	9629	12996	21514	19902	8019	45662	6818	16130	23913	20598	24694	8150
r17	8433	11216	11020	10672	9595	10935	3465	7212	6020	9960	4702	7455	5288	14364	21337	39788	10522	13075	8611	3295	23205	3452	15860	12746
r18	3149	2207	6064	4828	3675	6872	1088	5911	1748	1429	874	10659	3167	5264	12906	21506	5013	10411	4198	9115	11551	12153	3418	3055
r19	6801	4501	14923	14285	6064	17163	1444	1307	2697	6072	3974	3184	5560	6531	18545	19269	13343	13555	4512	41673	19173	16136	267	5596
r20	6462	6686	15559	8613	5734	9764	6587	5035	3062	1347	5359	6440	3556	7999	1755	8157	21783	15146	9624	55912	15512	25149	7423	4366
r21	6114	10652	8311	7260	11628	9422	4190	7161	6842	8411	9842	15790	8370	8849	5576	12814	30466	11872	10947	7289	4369	26189	9596	13016
r22	6441	4938	5676	4600	8053	10301	6599	7127	3367	10905	2353	6109	7335	5782	16180	6980	3713	6649	9456	5051	9556	23844	3882	7799
r23	9098	10417	41177	6186	21613	13456	27043	9160	4439	10527	3632	10629	3545	7801	16105	14254	13344	12539	11791	13817	15776	34278	7136	6795
r24	***27241***	***68253***	16293	56247	20988	***17090***	41048	12910	12353	***29269***	9049	3898	19175	12511	21029	37030	23916	28422	15674	7935	59409	24318	6543	25615
r25	12790	41049	556	18745	28286	42323	16432	10161	38097	18191	14235	1473	29640	1789	39684	13082	834	28053	38061	22467	13378	33832	11594	32942

Figure 4.7: Number of specific reads for each primer pair.

4.4.2 Experiment with Selected Primers

The validated primer L9, L5, L0, L1 and R24, R8, R2 and R1 are renamed to L0, L1, L2, L3 and R0, R1, R2, R3 to simplify the further illustration as shown in the Table 4.6. We design another experiment to valid the efficiency of the selected primers. We perform a proof of concept by storing the TPCH database compressed in a single archive file of 1.2MB. The

	l0	l1	l2	l3	l4	l5	l6	l7	l8	l9	l10	l11	l12	l13	l14	l15	l16	l17	l18	l19	l20	l21	l22	l23
r0	198	2	18	2	2	1	4	9	8	4	3	4	9	3	14	8	381	4	2	7	17	3	14	5
r1	9	4	35	21	38	5	9	12	6	17	5	9	5	8	5	10	35	10	5	8	19	10	31	13
r2	5	5	19	24	8	2	2	10	6	6	8	2	8	4	85	10	36	14	5	1	8	9	27	101
r3	2	6	12	14	10	7	6	19	21	8	3	5	15	2	16	6	8	4	1	6	10	21	15	8
r4	4	3	9	60	5	4	5	11	5	2	4	3	5	2	3	34	9	3	4	3	5	12	13	2
r5	4	4	8	2	3	5	3	5	4	7	2	2	3	1	10	3	10	6	1	3	3	7	14	5
r6	3	6	10	3	7	9	0	13	3	11	9	7	7	6	4	1	25	1	6	5	7	2	12	7
r7	1	4	8	8	9	6	3	2	2	7	38	2	2	0	15	10	19	4	1	9	3	7	16	5
r8	3	6	17	7	2	8	4	8	29	7	31	14	22	2	6	9	13	5	6	3	13	3	14	1
r9	1	3	31	7	3	3	6	18	3	7	4	4	10	1	9	2	12	7	2	1	9	2	13	8
r10	0	6	9	7	8	5	0	4	3	4	9	23	2	5	9	5	15	5	7	16	65	45	19	5
r11	1	12	7	13	2	2	0	3	1	0	6	3	6	1	1	2	11	3	2	4	71	10	5	25
r12	2	5	4	9	3	3	1	6	2	11	38	1	6	2	5	6	12	32	1	5	26	5	8	0
r13	1	4	6	2	3	10	1	5	2	6	3	3	3	1	0	3	8	1	5	2	9	16	8	0
r14	14	3	8	3	5	5	1	29	0	7	3	3	4	0	3	4	17	2	3	10	6	4	14	3
r15	4	3	1	2	1	6	1	0	1	3	0	0	1	1	3	3	1	1	8	3	3	0	2	2
r16	5	4	20	12	7	4	4	10	10	6	4	5	5	4	11	6	12	19	4	3	11	9	15	2
r17	3	9	10	4	0	5	0	11	2	3	3	6	4	3	3	6	17	18	8	5	9	1	18	3
r18	3	2	15	8	1	3	0	7	2	2	3	2	2	9	3	5	8	8	0	3	5	14	19	3
r19	8	3	51	8	6	2	5	66	3	8	6	2	6	3	11	17	14	9	11	10	9	11	34	13
r20	4	3	19	7	10	2	3	6	7	3	2	5	3	1	2	6	8	4	3	6	4	3	7	1
r21	5	14	15	43	11	10	4	17	6	15	5	4	14	4	8	12	24	5	8	12	10	14	25	17
r22	1	9	9	5	3	6	4	11	1	10	3	5	3	3	3	7	12	6	1	5	6	5	23	2
r23	2	1	23	10	6	2	2	3	5	3	2	3	4	8	9	5	16	6	9	7	11	4	11	3
r24	2	5	29	21	23	9	5	11	11	5	11	3	9	10	14	13	54	9	8	13	14	13	52	15
r25	11	25	14	13	14	3	10	31	10	12	9	3	22	4	16	10	35	7	7	19	18	10	38	12

Figure 4.8: Number of non specific reads for each primer pair.

database is encoded into 44376 oligos of 200 nt. The oligos are structured as one 5'-primer, one payload (out of a list of 44k distinct sequences), and one 3'-primer. Thus, each unique primer pair would be attached to 2770 oligos approximately. The 43.3 millions of reads are sequenced through Oxford Nanopore PromethION platform to validate whether the PCR bias and improper binding are eliminated.

Table 4.6: Primer list

left primer list	right primer list
L0: ACATGCCGTGCCATTGGATT	R0: AAGGCCAATTCGCGGTTAGT
L1: AGCCGACAAGTTCCAACACA	R1: AGGTGAGTGCCGTAACGATT
L2: GTCCAGGCAAAGATACAGTC	R2: GAACGGAGCGATGAGTTTGT
L3: TAGCCTCCAGAATGAGACAG	R3: TTCAAGCCAGACCGTGTGTA

PCR Bias

First of all, we check whether some primer pairs' oligos are over or underrepresented. We use Accel-Align to align the reads to reference oligos to determine their origin and achieved the copy number of each oligos. Table 4.7 shows the population fraction change per primer pair. We find that although the population fraction change does not equal to 1, the average of the population fraction change is 1. The standard deviation which is 0.13, has been greatly reduced compared to the TPCH database in the file-based random access design which is 10.8. It indicates that the oligos are more equally presented before and after PCR.

Table 4.7: Reads selectively retrieved per primer pair

	nb oligos	raw pop fraction	nb of read	pop fraction	frac change
L0_R0	2774	0.063	2112181	0.05	0.78
L0_R1	2770	0.062	2753811	0.06	1.02
L0_R2	2774	0.063	2710222	0.06	1.00
L0_R3	2774	0.063	3481596	0.08	1.28
L1_R0	2774	0.063	2992473	0.07	1.10
L1_R1	2770	0.062	2474146	0.06	0.91
L1_R2	2774	0.063	1954131	0.05	0.72
L1_R3	2774	0.063	2784902	0.06	1.03
L2_R0	2774	0.063	2649534	0.06	0.98
L2_R1	2774	0.063	2867796	0.07	1.06
L2_R2	2774	0.063	2560211	0.06	0.94
L2_R3	2774	0.063	2799222	0.06	1.03
L3_R0	2774	0.063	3103064	0.07	1.14
L3_R1	2774	0.063	2690192	0.06	0.99
L3_R2	2774	0.063	2487760	0.06	0.92
L3_R3	2774	0.063	2954518	0.07	1.09

Improper Binding

We analyze whether improper binding has been eliminated. To be noted, in this experiment, the oligos are sequenced by Oxford Nanopore PromethION platform. It produces long reads and leads to additional sub-sequences in two extremities of each read. Thus, we conduct the primer based on CIGAR to ignore the softclipping part. The CIGAR is computed by the alignment from a read to original oligos with Accel-Align. In the file-based random access experiment, we determine whether the primer is as same as the target primer, or other primer, or bad quality simply through a bit-by-bit comparison. This is because the primer is short with only 5 nt and also because Illuminia sequencer has high accuracy. However, in this case, the primer is longer (20nt) and Nanopore is less accurate than Illuminia. If we simply compare the extremities to reference primers, we find that only 80%~85% of the reads have the target primer while some other reads do have the right primer but containing substitutions or Indels. Hence, after we get the raw primers from the reads, we align raw primer to the reference primers to determine the origin, to determine whether the primer pairs belong to its oligo, or belong to another oligo, or have too bad quality. Table 4.8 shows the primer binding analysis. We find that all sets have more than 95% reads with target primers and improper binding has been decreased to 0.9%~3.5%.

Table 4.8: Primer binding analysis

	nb of read	target primer %	other primer %	no primer %
L0_R0	2112181	95.53	2.88	1.59
L0_R1	2753811	97.10	1.32	1.58
L0_R2	2710222	97.02	1.36	1.61
L0_R3	3481596	96.68	1.50	1.82
L1_R0	2992473	94.91	3.46	1.62
L1_R1	2474145	96.51	1.85	1.64
L1_R2	1954131	96.26	1.98	1.77
L1_R3	2784902	95.96	2.12	1.92
L2_R0	2649534	94.43	3.43	2.14
L2_R1	2867796	95.94	1.93	2.13
L2_R2	2560211	95.82	1.93	2.25
L2_R3	2799222	95.40	2.19	2.41
L3_R0	3103064	95.51	2.46	2.03
L3_R1	2690192	97.09	0.89	2.02
L3_R2	2487760	96.90	0.94	2.16
L3_R3	2954518	96.66	1.08	2.26

Data Recovery

Although the improper binding has been eliminated, there are still 0.9%~3.5% reads containing non specific primers. As an end-to-end workflow, we pass the reads through our decoder to validate whether the decoder is able to distinguish the improper binding and distribute the reads from same primer into the same bucket. Table 4.9 shows that all primer pair sets achieve high precision that more than 99% of reads selected do really belong to that group. We also find that there is no significance in the primer position since the fraction of left and right target primer are close to each other.

Table 4.9: Reads selectively retrieved per primer pair

	nb of reads	correct aligned (%)	with left target primer(%)	with right left primer(%)
L0_R0	2069808	99.97	0.9998	0.9998
L0_R1	2701444	99.97	0.9998	0.9998
L0_R2	2659990	99.97	0.9998	0.9997
L0_R3	3410425	99.98	0.9999	0.9998
L1_R0	2918999	99.96	0.9997	0.9997
L1_R1	2415844	99.95	0.9997	0.9996
L1_R2	1908381	99.87	0.9988	0.9997
L1_R3	2716409	99.88	0.9989	0.9997
L2_R0	2575053	99.98	0.9998	0.9998
L2_R1	2788303	99.98	0.9999	0.9998
L2_R2	2487260	99.98	0.9999	0.9998
L2_R3	2715068	99.98	0.9998	0.9998
L3_R0	3038095	99.97	0.9998	0.9998
L3_R1	2635874	99.97	0.9998	0.9998
L3_R2	2435858	99.97	0.9998	0.9998
L3_R3	2890926	99.97	0.9998	0.9998

4.5 Discussion

In this work, we have studied the PCR-based random access based on the sequence alignment tool Accel-Align. We expand the uni-dimensional data addressing to multi-dimensional data addressing hierarchy Collection – Object – Extent to support large scale storage.

Initially, we design the file-based random access with the primers of 5nt. During the experiment, we find that PCR bias impacts selection sensitivity of small files in a big pool. On the other hand, extra data are existing in the final pool due to the improper binding. This non-specific primer binding leads to silent data corruption if decoder depends on primers faithfully.

To overcome the bottlenecks, we propose the block-based random access such that each extent has fixed size and the small files are encoded together to avoid PCR bias. We elaborately design 24 left primers and 26 right primers of 20-mers and select 4 out of each them as the validated primers to test the storing of TPCH database. It shows that the new design eliminates the PCR bias and improves the proportion of reads with correctly primers. As a validation, more than 99.9% reads are correctly distributed by the decoder for each primer pair. As the supplementary, we are experimenting for a larger scale of database to store the Danish National Archive data.

Chapter 5

Conclusion

5.1 Conclusion

Sequence similarity analysis has been studied extensively in the literature as a fundamental problem in bioinformatics, data integration, collaborative filtering, and natural language processing. Given the sheer size of modern sequencing datasets, accurate and scalable edit similarity algorithms are essential to analyze data generated by genomics and DNA-based data storage.

5.1.1 Accel-Align: a Fast Sequence Aligner

Currently, most sequence alignment algorithms designed for the next-generation sequencing technology use seed–filter–extension strategy to speed up the alignment process. They try to find the candidate positions on the reference genome via a fast exact string matching lookup, and then eliminate sub-optimal positions through a filtering to save the time-consuming extension stage. However, filtering makes assumptions about error patterns and therefore has inherent performance–accuracy trade offs.

In this work, we propose a new methodology named seed–embed–extend (SEE) based on recent advances in randomized embedding. SEE transforms the read and reference strings from edit distance regime to the Hamming regime by embedding them using a randomized algorithm, and uses Hamming distance over the embedded set to identify optimal candidates. Instead of focusing on eliminating sub-optimal candidates, SEE focuses instead on identifying optimal candidates. However, we find that the initial randomized embedding might not able to identify the true optimal candidates in the presence of some pathological random sequences. Hence, we propose multiple embedding and chain embedding to improve further accuracy.

To show the efficiency of SEE in practice, we implemented a fast aligner and mapper – Accel-

Align in C++ with multi-threading support. It is publicly available on github via https://github.com/raja-appuswamy/accel-align-release. In our experiment, we show that Accel-Align is 2 ~ 12 faster than state-of-the-art aligners with both simulated and real genomic datasets. Accel-Align is also shown to be an efficient tool to determine the origin of reads such that we can study sequencing statistics and error characterization in DNA-based storage.

5.1.2 Edit Similarity-Based Decoder in DNA-based Storage

DNA is a promising storage medium due to its high density and long endurance. The key challenge in DNA storage today is the cost of synthesis. In this work, we propose the composite motifs framework to scale logical density (bits written per cycle) which results in the potential reduction of synthesis cost. In doing so, we developed a new structure-aware decoding algorithm Motif-Search based on the edit similarity analysis to recover the stored information despite errors created by the enzymatic ligation (Bridge Oligonucleotide Assembly) and Nanopore-based, direct oligonucleotide sequencing techniques. Using the proposed methods, we present an end-to-end experiment where we store the text "HelloWorld" at a logical density of 84 bits/cycle (14–42× improvement over state-of-the-art). The physical density of our approach is 3.36bits/nt, which is also higher than the physical density of conventional base-by-base DNA storage solutions (2 bits/nt) and comparable to degenerate base approaches (3.37 bits/nt). Finally, our decoder Motif-Search is able to fully recover stored data at a sequencing coverage 20×.

5.1.3 Sequence Analysis for Random Access in DNA-based Storage

Random access to a portion of data plays an important role to make large scale DNA-based storage viable. In this work, we focus on the PCR based random access. To study the precision of random access in DNA-based storage, the first step is to determine the origin payload and primer of the reads. Using Accel-Align, we performed a study of the characteristics of different random access design. We showed that file-based random access can suffer from PCR bias issues if file sizes are not uniform. Some information is over-presented while some is under-presented after the PCR. The under-presented information can be missed after decoding. We also showed that improper binding happens during PCR with the short primers. The extra data caused by improper binding can lead to silent data corruption. To overcome these issues, we proposed using a multi-dimensional addressing method using a block-based design, where each block has a uniform size. Using a large-scale experiment, we showed that the new design eliminates PCR bias and improves the proportion of reads with correct primers.

5.2 Future Work

Aiming at the approximate edit similarity problem, this thesis studied the scalability and accuracy of the algorithms in the domain of computational genomics and DNA data storage. In this section, we outline several perspectives of the research that require further investigation in the future.

5.2.1 Accel-Align: a Fast Sequence Aligner

In the area of computational genomics, we have demonstrated earlier in the thesis that low distortion embedding is capable of transforming the distance between two strings from the edit regime to Hamming regime to accelerate the sequence similarly measurement. We have implemented Accel-Align as a fast and accurate sequence aligner and mapper based on SEE methodology in practice. Nevertheless, future work can be performed in the following aspects in order to achieve further enhancements of the algorithm.

Accuracy improvement. We found that Accel-Align provides comparable accuracy with the simulated dataset. However, the variant calling pipeline based on NA12878 dataset with Accel-Align can miss 5 out of 23521 in the detection of true positive SNPs compared to the pipeline with BWA-MEM. The main reason behind this is the usage of a simple seeding method that relies on long seeds to identify candidate locations. As a result, reads with higher error rate might result in the seeding missing a few candidate locations. Minimizer [77], strobmer [78, 79] and learned-index [80] can be good candidates to replace our seeding module of simple k-mers.

Performance improvement. Figure 2.8 shows the breakdown of processing time for each Accel-Align stage. It illustrates that seed lookup is the dominant time-consuming step. One of the approaches to mitigate this problem is to accumulate the forward and reverse seeds together with a direction mark in one bit of the seed value during the indexing so that Accel-Align only needs to do the look up once. Consequently, it can reduce the list of candidates to be merged during the seeding step. We are implementing this optimization and we plan to release a new version in the near future with improved seeding time.

Hardware-acceleration techniques exploitation. Many bioinformatic algorithms have been improved using GPUs [81, 82, 83]. For instance, GPUs have been proved to improve the speed of the Smith-Waterman algorithm by order of magnitude [84]. In Accel-Align, seeding stage dominates overall execution time irrespective of the k-mer size. Fortunately, seeding is also a prime candidate for acceleration by GPUs due to its data-parallel nature as demonstrated by a recent microarchitectural study [85]. The embedding stage is also compatible to parallelization on GPU. Thus, in the future, we plan to develop a parallelized version of Accel-Align which

can work across CPUs and GPUs, and compare it with other GPU-based aligners such as SOAP3 [86], SOAP3-dp [87], Arioc [88, 89] and GASAL2 [90].

Bisulfite sequencing alignment adaptation. DNA methylation plays an important role in many biological applications. It is related to many diseases, especially cancer [91, 92, 93, 94]. The development of whole-genome methylation sequencing (WGBS) makes it possible to measure the methylation status of the entire genome. Although we designed Accel-Align initially for DNA sequence alignment problem, it can be adapted for bisulfite sequencing reads as well. Bisulfite sequencing applies a bisulfite treatment to genomic DNA to convert nonmethylated cytosines (C) to uracils(U), which can be sequenced as thymines (T) ; while methylated cytosines (C) cannot be converted to uracils(U) and are sequenced as cytosines (C). Hence, the index should be built with the sense strand twice, once with C converted to T and once with G converted to A. In the future work, we plan to add bisulfite support to Accel-Align and benchmark it with state-of-the-art aligners such as BS-Seeker2 [95], Bismark [96], BWA-Meth [97] and BiSulfite Bolt [98].

5.2.2 Edit Similarity-Based Decoder in DNA-based Storage

In the domain of DNA-based storage, we have proposed the new encoding logic – composite motifs to scale the logical density. We have designed the structure-aware decoder Motif-Search based on edit similarity and we have shown that Motif-Search is able to fully recover information even with the error-prone reads at a low coverage 20×. Nevertheless, future work can be performed in the following aspects to improve the scalablity and reduce cost.

Error-correction code utilization. State-of-the-art research, such as Reed Solomon codes [63], LDPC [99] and fountain codes [35], employs error correction block to improve the robustness. The similar techniques can also be applied in composite motifs encoding to allow a lower sequencing coverage to fully recover the information, and thus, to reduce the reading cost.

Physical density improvement. The physical density is defined as the number of bits that can be encoded by a nucleotide. In our experiment, we built upon an existing library of 25nt motifs to simplify the task of motif design. Thus, our composite motif approach provides a physical density of 3.36 bits/nt (encoding 84 bits with 25nt composite motifs). The physical density can be increased by selecting more compositions from the library. For instance, selecting 48 motifs from a 96-motif library can encode 90 bits per composite motifs and increase the physical density up to 3.6 bits/nt. The physical density can also be scaled up by reducing the motif length. In our experiment, we used a motif library with 96 unique motifs. Theoretically, the permutation of 4 nucleotides consists of 256 (4^4) unique motifs which is already sufficient to design 96 motifs. Hence, we can potentially use a shorter motif to improve the physical density.

Experiment scale improvement. As a prototype, we have stored 10 bytes data with short oligos of 74nt in this experiment. We plan to extend the oligo with multiple payload motifs and store large scale dataset with long oligos in order to demonstrate the efficiency and scalability brought by the composite motifs encoding schema and the Motif-Search decoding in the area of DNA-based storage.

Publications List

The research carried out during the reporting period has lead to the following scientific papers:

Journal

- **Yiqing Yan**, Nimisha Chaturvedi and Raja Appuswamy. **Accel-align: a fast sequence mapper and aligner based on the seed–embed–extend method[J].** BMC bioinformatics, 2021, 22(1): 1-20.

- Eugenio Marinelli, Eddy Ghabach, **Yiqing Yan**, Thomas Bolbroe, Omer Sella, Thomas Heinis, Raja Appuswamy. **Digital Preservation with Synthetic DNA[M].**Transactions on Large-Scale Data-and Knowledge-Centered Systems LI. Springer, Berlin, Heidelberg, 2022: 119-135.

Conference and Workshop

- **Yiqing Yan**, Nimisha Chaturvedi and Raja Appuswamy. **Accel-align: a fast sequence mapper and aligner based on the seed–embed–extend method.** 2021 RECOMB Satellite Workshop on Massively Parallel Sequencing (RECOMB-Seq).

- **Yiqing Yan**, Nimisha Chaturvedi and Raja Appuswamy. **Optimizing the Accuracy of Randomized Embedding for Sequence Alignment[C].** 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2022: 144-151.

- Giulio Franzese, **Yiqing Yan**, Giuseppe Serra, Ivan D'Onofrio, Raja Appuswamy, Pietro Michiardi. **Generative DNA: Representation Learning for DNA-based Approximate Image Storage[C].** 2021 International Conference on Visual Communications and Image Processing (VCIP). IEEE, 2021: 01-05.

Under Review

- **Yiqing Yan**, Nimesh Pinnamaneni, Sachin Chalapati, Connor Crosby, Raja Appuswamy. **Scaling Logical Density of DNA storage with Enzymatically-Ligated Composite Motifs.**
- Eugenio Marinelli, **Yiqing Yan**, Virginie Magnone, Marie-Charlotte Dumargne, Pascal Barbry, Thomas Heinis, Raja Appuswamy. **OligoArchive-DSM: Columnar Design for Error-Tolerant DatabaseArchival using Synthetic DNA.**

Declarations

The research was funded by the following support:

- European Union's Horizon research and innovation programme, project OligoArchive under Grant agreement No. 863320, from March 2020 to May 2022.
- European Union's Horizon research and innovation programme, project Molecular Storage System (MoSS) under Grant agreement No. 101058035, from June 2022 to February 2023.

Bibliography

[1] Philipp L Antkowiak, Jory Lietard, Mohammad Zalbagi Darestani, Mark M Somoza, Wendelin J Stark, Reinhard Heckel, and Robert N Grass. Low cost dna data storage using photolithographic synthesis and advanced information reconstruction and error correction. *Nature communications*, 11(1):1–10, 2020.

[2] Coulson AR Sanger F, Nicklen S. Dna sequencing with chain-terminating inhibitors. In *Natl. Acad. Sci. USA*, volume 74, pages 5463–5467, 1977.

[3] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. Big data: astronomical or genomical? *PLoS biology*, 13(7):e1002195, 2015.

[4] Kathryn A Phillips, Mark J Pletcher, and Uri Ladabaum. Is the "1000$genome$"$really$1000? understanding the full benefits and costs of genomic sequencing. *Technology and health care: official journal of the European Society for Engineering and Medicine*, 23(3):373, 2015.

[5] Kris A. Wetterstrand. Dna sequencing costs: Data from the nhgri genome sequencing program (gsp). https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data. Accessed: 2022-10-12.

[6] F Carrasco-Ramiro, R Peiró-Pastor, and B Aguado. Human genomics projects and precision medicine. *Gene therapy*, 24(9):551–561, 2017.

[7] Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M LeProust, Botond Sipos, and Ewan Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized dna. *nature*, 494(7435):77–80, 2013.

[8] Robert N Grass, Reinhard Heckel, Michela Puddu, Daniela Paunescu, and Wendelin J Stark. Robust chemical preservation of digital information on dna in silica with error-correcting codes. *Angewandte Chemie International Edition*, 54(8):2552–2555, 2015.

[9] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[10] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[11] AM Michelson and Alexander R Todd. Nucleotides part xxxii. synthesis of a dithymidine dinucleotide containing a 3: 5-internucleotidic linkage. *Journal of the Chemical Society (Resumed)*, pages 2632–2638, 1955.

[12] SL Beaucage and MH Caruthers. Deoxynucleoside phosphoramidites—a new class of key intermediates for deoxypolynucleotide synthesis. *Tetrahedron letters*, 22(20):1859–1862, 1981.

[13] FJ Bollum. Oligodeoxyribonucleotide-primed reactions catalyzed by calf thymus polymerase. *Journal of Biological Chemistry*, 237(6):1945–1949, 1962.

[14] Sebastian Palluk, Daniel H Arlow, Tristan De Rond, Sebastian Barthel, Justine S Kang, Rathin Bector, Hratch M Baghdassarian, Alisa N Truong, Peter W Kim, Anup K Singh, et al. De novo dna synthesis using polymerase-nucleotide conjugates. *Nature biotechnology*, 36(7):645–650, 2018.

[15] Michael A Jensen and Ronald W Davis. Template-independent enzymatic oligonucleotide synthesis (tieos): its history, prospects, and challenges. *Biochemistry*, 57(12):1821–1832, 2018.

[16] W Jou, G Haegeman, M Ysebaert, and W Fiers. Nucleotide sequence of the gene coding for the bacteriophage ms2 coat protein. *Nature*, 237(5350):82–88, 1972.

[17] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. Comparison of next-generation sequencing systems. *J Biomed Biotechnol*, 2012(251364):251364, 2012.

[18] Martin Kircher and Janet Kelso. High-throughput dna sequencing–concepts and limitations. *Bioessays*, 32(6):524–536, 2010.

[19] Sara Goodwin, John D McPherson, and W Richard McCombie. Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333–351, 2016.

[20] James Clarke, Hai-Chen Wu, Lakmal Jayasinghe, Alpesh Patel, Stuart Reid, and Hagan Bayley. Continuous base identification for single-molecule nanopore dna sequencing. *Nature nanotechnology*, 4(4):265–270, 2009.

[21] Michael J Levene, Jonas Korlach, Stephen W Turner, Mathieu Foquet, Harold G Craighead, and Watt W Webb. Zero-mode waveguides for single-molecule analysis at high concentrations. *science*, 299(5607):682–686, 2003.

[22] Kevin J Travers, Chen-Shan Chin, David R Rank, John S Eid, and Stephen W Turner. A flexible and efficient template format for circular consensus sequencing and snp detection. *Nucleic acids research*, 38(15):e159–e159, 2010.

[23] John F Thompson and Patrice M Milos. The properties and applications of single-molecule dna sequencing. *Genome biology*, 12(2):1–10, 2011.

[24] Aaron M Wenger, Paul Peluso, William J Rowell, Pi-Chuan Chang, Richard J Hall, Gregory T Concepcion, Jana Ebler, Arkarachai Fungtammasan, Alexey Kolesnikov, Nathan D Olson, et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature biotechnology*, 37(10):1155–1162, 2019.

[25] Franka J Rang, Wigard P Kloosterman, and Jeroen de Ridder. From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. *Genome biology*, 19(1):1–11, 2018.

[26] W John Wilbur and David J Lipman. Rapid similarity searches of nucleic acid and protein data banks. *Proceedings of the National Academy of Sciences*, 80(3):726–730, 1983.

[27] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[28] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[29] Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357, 2012.

[30] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

[31] Matei Zaharia, William J Bolosky, Kristal Curtis, Armando Fox, David Patterson, Scott Shenker, Ion Stoica, Richard M Karp, and Taylor Sittler. Faster and more accurate sequence alignment with snap. *arXiv preprint arXiv:1111.5572*, 2011.

[32] Daehwan Kim, Joseph M Paggi, Chanhee Park, Christopher Bennett, and Steven L Salzberg. Graph-based genome alignment and genotyping with hisat2 and hisat-genotype. *Nature biotechnology*, 37(8):907–915, 2019.

[33] Raja Appuswamy, Kevin Le Brigand, Pascal Barbry, Marc Antonini, Olivier Madderson, Paul Freemont, James McDonald, and Thomas Heinis. Oligoarchive: Using dna in the dbms storage hierarchy. In *CIDR*, 2019.

[34] Joe Davis. Microvenus. *Art Journal*, 55(1):70–74, 1996.

[35] Yaniv Erlich and Dina Zielinski. Dna fountain enables a robust and efficient storage architecture. *science*, 355(6328):950–954, 2017.

[36] Eugenio Marinelli Raja Appuswamy. Onejoin: Cross-architecture, scalable edit similarity join for dna data storage using oneapi. 2021.

[37] Md Vasimuddin, Sanchit Misra, Heng Li, and Srinivas Aluru. Efficient architecture-aware acceleration of bwa-mem for multicore systems. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 314–324. IEEE, 2019.

[38] Md Vasimuddin, Sanchit Misra, and Srinivas Aluru. Identification of significant computational building blocks through comprehensive investigation of ngs secondary analysis methods. *bioRxiv*, page 301903, 2018.

[39] Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V. Garimella, Jared R. Maguire, Christopher Hartl, Anthony A. Philippakis, Guillermo Del Angel, Manuel A Rivas, and Matt Hanna. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature genetics*, 43(5):491, 2011.

[40] Stefan Canzar and Steven L. Salzberg. Short read mapping: An algorithmic tour. *Proceedings of the IEEE*, 105(3):436–458, 2017.

[41] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, page 51–58, 2015.

[42] Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 712–725, 2016.

[43] Haoyu Zhang and Qin Zhang. Embedjoin: Efficient edit similarity joins via embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 585–594, 2017.

[44] Xiyuan Zhang, Yang Yuan, and Piotr Indyk. Neural embeddings for nearest neighbor search under edit distance. 2019.

[45] Hajime Suzuki and Masahiro Kasahara. Introducing difference recurrence relations for faster semi-global alignment of long sequences. *BMC bioinformatics*, 19(1):33–47, 2018.

[46] Yang Liao, Gordon K. Smyth, and Wei Shi. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Research*, 41(10), 2013.

[47] Manuel Holtgrewe. Mason: a read simulator for second generation sequencing data. 2010.

[48] Manojkumar Kumaran, Umadevi Subramanian, and Bharanidharan Devarajan. Performance assessment of variant calling pipelines using human whole exome sequencing and simulated data. *BMC bioinformatics*, 20(342), 2019.

[49] Eugenio Marinelli, Yiqing Yan, Virginie Magnone, Marie-Charlotte Dumargne, Pascal Barbry, Thomas Heinis, and Raja Appuswamy. Oligoarchive-dsm: Columnar design for error-tolerant database archival using synthetic dna. *bioRxiv*, 2022.

[50] Yiqing Yan, Nimisha Chaturvedi, and Raja Appuswamy. Accel-align: a fast sequence mapper and aligner based on the seed–embed–extend method. *BMC bioinformatics*, 22(1):1–20, 2021.

[51] Yiqing Yan, Nimisha Chaturvedi, and Raja Appuswamy. Optimizing the accuracy of randomized embedding for sequence alignment. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 144–151. IEEE, 2022.

[52] David Reinsel, John Gantz, and John Rydning. Data age 2025: The evolution of data to life-critical. *Don't Focus on Big Data*, 2, 2017.

[53] Victor Zhirnov, Reza M Zadegan, Gurtej S Sandhu, George M Church, and William L Hughes. Nucleic acid memory. *Nature materials*, 15(4):366–370, 2016.

[54] James Bornholt, Randolph Lopez, Douglas M Carmean, Luis Ceze, Georg Seelig, and Karin Strauss. A dna-based archival storage system. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 637–649, 2016.

[55] SM Tabatabaei Yazdi, Yongbo Yuan, Jian Ma, Huimin Zhao, and Olgica Milenkovic. A rewritable, random-access dna-based storage system. *Scientific reports*, 5(1):1–10, 2015.

[56] Henry H Lee, Reza Kalhor, Naveen Goela, Jean Bolot, and George M Church. Terminator-free template-independent enzymatic dna synthesis for digital information storage. *Nature communications*, 10(1):1–12, 2019.

[57] Leon Anavy, Inbal Vaknin, Orna Atar, Roee Amit, and Zohar Yakhini. Data storage in dna with fewer synthesis cycles using composite dna letters. *Nature biotechnology*, 37(10):1229–1236, 2019.

[58] Yeongjae Choi, Taehoon Ryu, Amos C Lee, Hansol Choi, Hansaem Lee, Jaejun Park, Suk-Heung Song, Seojoo Kim, Hyeli Kim, Wook Park, et al. High information capacity dna-based data storage with augmented encoding characters using degenerate bases. *Scientific reports*, 9(1):1–7, 2019.

[59] Nathaniel Roquet, Swapnil P Bhatia, Sarah A Flickinger, Sean Mihm, Michael W Norsworthy, Devin Leake, and Hyunjun Park. Dna-based data storage via combinatorial assembly. *bioRxiv*, 2021.

[60] Randolph Lopez, Yuan-Jyue Chen, Siena Dumas Ang, Sergey Yekhanin, Konstantin Makarychev, Miklos Z Racz, Georg Seelig, Karin Strauss, and Luis Ceze. Dna assembly for nanopore data storage readout. *Nature communications*, 10(1):1–9, 2019.

[61] Hajime Suzuki and Masahiro Kasahara. Introducing difference recurrence relations for faster semi-global alignment of long sequences. *BMC bioinformatics*, 19(45), 2018.

[62] Sachin Chalapati, Conor A Crosbie, Dixita Limbachiya, and Nimesh Pinnamaneni. Direct oligonucleotide sequencing with nanopores. *Open Research Europe*, 1(47):47, 2021.

[63] Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, et al. Random access in large-scale dna data storage. *Nature biotechnology*, 36(3):242–248, 2018.

[64] Reinhard Heckel, Gediminas Mikutis, and Robert N Grass. A characterization of the dna data storage channel. *Scientific reports*, 9(1):1–12, 2019.

[65] Eugenio Marinelli and Raja Appuswamy. Onejoin: Cross-architecture, scalable edit similarity join for dna data storage using oneapi. In *ADMS*, 2021.

[66] Eugenio Marinelli, Eddy Ghabach, Yiqing Yan, Thomas Bolbroe, Omer Sella, Thomas Heinis, and Raja Appuswamy. *Digital Preservation with Synthetic DNA*. 2022.

[67] Meinolf Blawat, Klaus Gaedke, Ingo Huetter, Xiao-Ming Chen, Brian Turczyk, Samuel Inverso, Benjamin W Pruitt, and George M Church. Forward error correction for dna data storage. *Procedia Computer Science*, 80:1011–1022, 2016.

[68] George M Church, Yuan Gao, and Sriram Kosuri. Next-generation digital information storage in dna. *Science*, 337(6102):1628–1628, 2012.

[69] Sander E Van der Verren, Nani Van Gerven, Wim Jonckheere, Richard Hambley, Pratik Singh, John Kilgour, Michael Jordan, E Jayne Wallace, Lakmal Jayasinghe, and Han Remaut. A dual-constriction biological nanopore resolves homonucleotide sequences with high fidelity. *Nature biotechnology*, 38(12):1415–1420, 2020.

[70] Thomas P Niedringhaus, Denitsa Milanova, Matthew B Kerby, Michael P Snyder, and Annelise E Barron. Landscape of next-generation sequencing technologies. *Analytical chemistry*, 83(12):4327–4341, 2011.

[71] Kyle J Tomek, Kevin Volkel, Alexander Simpson, Austin G Hass, Elaine W Indermaur, James M Tuck, and Albert J Keung. Driving the scalability of dna-based information storage systems. *ACS synthetic biology*, 8(6):1241–1248, 2019.

[72] Satoshi Kashiwamura, Masahito Yamamoto, Atsushi Kameda, Toshikazu Shiba, and Azuma Ohuchi. Hierarchical dna memory based on nested pcr. In *International Workshop on DNA-Based Computers*, pages 112–123. Springer, 2002.

[73] Billy Lau, Shubham Chandak, Sharmili Roy, Kedar Tatwawadi, Mary Wootters, Tsachy Weissman, and Hanlee P Ji. Magnetic dna random access memory with nanopore readouts and exponentially-scaled combinatorial addressing. *BiorXiv*, 2021.

[74] Claris Winston, Lee Organick, David Ward, Luis Ceze, Karin Strauss, and Yuan-Jyue Chen. Combinatorial pcr method for efficient, selective oligo retrieval from complex oligo pools. *ACS Synthetic Biology*, 11(5):1727–1734, 2022.

[75] Yuan-Jyue Chen, Christopher N Takahashi, Lee Organick, Callista Bee, Siena Dumas Ang, Patrick Weiss, Bill Peck, Georg Seelig, Luis Ceze, and Karin Strauss. Quantifying molecular bias in dna data storage. *Nature communications*, 11(1):1–9, 2020.

[76] Brian Bushnell. Bbmap: a fast, accurate, splice-aware aligner. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2014.

[77] Michael Roberts, Wayne Hayes, Brian R Hunt, Stephen M Mount, and James A Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, 2004.

[78] Kristoffer Sahlin. Flexible seed size enables ultra-fast and accurate read alignment. *bioRxiv*, pages 2021–06, 2022.

[79] Kristoffer Sahlin. Effective sequence similarity detection with strobemers. *Genome research*, 31(11):2080–2094, 2021.

[80] Saurabh Kalikar, Chirag Jain, Vasimuddin Md, and Sanchit Misra. Accelerating long-read analysis on modern cpus. *bioRxiv*, pages 2021–07, 2022.

[81] Xiangyuan Zhu, Kenli Li, Ahmad Salah, Lin Shi, and Keqin Li. Parallel implementation of mafft on cuda-enabled graphics hardware. *IEEE/ACM transactions on computational biology and bioinformatics*, 12(1):205–218, 2014.

[82] Alejandro Chacón, Santiago Marco-Sola, Antonio Espinosa, Paolo Ribeca, and Juan Carlos Moure. Boosting the fm-index on the gpu: Effective techniques to mitigate random memory access. *IEEE/ACM transactions on computational biology and bioinformatics*, 12(5):1048–1059, 2014.

[83] Edans Flavius de Oliveira Sandes, Guillermo Miranda, Xavier Martorell, Eduard Ayguade, George Teodoro, and Alba Cristina Magalhaes Melo. Cudalign 4.0: Incremental speculative traceback for exact chromosome-wide alignment in gpu clusters. *IEEE Transactions on Parallel and Distributed Systems*, 27(10):2838–2850, 2016.

[84] Yongchao Liu, Adrianto Wirawan, and Bertil Schmidt. Cudasw++ 3.0: accelerating smith-waterman protein database search by coupling cpu and gpu simd instructions. *BMC bioinformatics*, 14(1):1–10, 2013.

[85] Raja Appuswamy, Jacques Fellay, and Nimisha Chaturvedi. Sequence alignment through the looking glass. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018.

[86] Chi-Man Liu, Thomas Wong, Edward Wu, Ruibang Luo, Siu-Ming Yiu, Yingrui Li, Bingqiang Wang, Chang Yu, Xiaowen Chu, Kaiyong Zhao, et al. Soap3: ultra-fast gpu-based parallel alignment tool for short reads. *Bioinformatics*, 28(6):878–879, 2012.

[87] Ruibang Luo, Thomas Wong, Jianqiao Zhu, Chi-Man Liu, Xiaoqian Zhu, Edward Wu, Lap-Kei Lee, Haoxiang Lin, Wenjuan Zhu, David W Cheung, et al. Soap3-dp: fast, accurate and sensitive gpu-based short read aligner. *PloS one*, 8(5):e65632, 2013.

[88] Richard Wilton and Alexander S Szalay. Arioc: High-concurrency short-read alignment on multiple gpus. *PLoS computational biology*, 16(11):e1008383, 2020.

[89] Richard Wilton, Tamas Budavari, Ben Langmead, Sarah J Wheelan, Steven L Salzberg, and Alexander S Szalay. Arioc: high-throughput read alignment with gpu-accelerated exploration of the seed-and-extend search space. *PeerJ*, 3:e808, 2015.

[90] Nauman Ahmed, Jonathan Lévy, Shanshan Ren, Hamid Mushtaq, Koen Bertels, and Zaid Al-Ars. Gasal2: a gpu accelerated sequence alignment library for high-throughput ngs data. *BMC bioinformatics*, 20(1):1–20, 2019.

[91] Swarnaseetha Adusumalli, Mohd Feroz Mohd Omar, Richie Soong, and Touati Benoukraf. Methodological aspects of whole-genome bisulfite sequencing analysis. *Briefings in bioinformatics*, 16(3):369–379, 2015.

[92] David Capper, David TW Jones, Martin Sill, Volker Hovestadt, Daniel Schrimpf, Dominik Sturm, Christian Koelsche, Felix Sahm, Lukas Chavez, David E Reuss, et al. Dna methylation-based classification of central nervous system tumours. *Nature*, 555(7697):469–474, 2018.

[93] Javier Soto, Carlos Rodriguez-Antolin, Elena Vallespín, Javier de Castro Carpeño, and Inmaculada Ibanez De Caceres. The impact of next-generation sequencing on the dna methylation–based translational cancer research. *Translational Research*, 169:1–18, 2016.

[94] Rui-hua Xu, Wei Wei, Michal Krawczyk, Wenqiu Wang, Huiyan Luo, Ken Flagg, Shaohua Yi, William Shi, Qingli Quan, Kang Li, et al. Circulating tumour dna methylation markers for diagnosis and prognosis of hepatocellular carcinoma. *Nature materials*, 16(11):1155–1161, 2017.

[95] Weilong Guo, Petko Fiziev, Weihong Yan, Shawn Cokus, Xueguang Sun, Michael Q Zhang, Pao-Yang Chen, and Matteo Pellegrini. Bs-seeker2: a versatile aligning pipeline for bisulfite sequencing data. *BMC genomics*, 14(1):1–8, 2013.

[96] Felix Krueger and Simon R Andrews. Bismark: a flexible aligner and methylation caller for bisulfite-seq applications. *bioinformatics*, 27(11):1571–1572, 2011.

[97] Brent S Pedersen, Kenneth Eyring, Subhajyoti De, Ivana V Yang, and David A Schwartz. Fast and accurate alignment of long bisulfite-seq reads. *arXiv preprint arXiv:1401.1129*, 2014.

[98] Colin Farrell, Michael Thompson, Anela Tosevska, Adewale Oyetunde, and Matteo Pellegrini. Bisulfite bolt: A bisulfite sequencing analysis platform. *GigaScience*, 10(5):giab033, 2021.

[99] Shubham Chandak, Kedar Tatwawadi, Billy Lau, Jay Mardia, Matthew Kubit, Joachim Neu, Peter Griffin, Mary Wootters, Tsachy Weissman, and Hanlee Ji. Improved read/write cost tradeoff in dna-based data storage using ldpc codes. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 147–156. IEEE, 2019.