Label-guided seed-chain-extend alignment on annotated De Bruijn graphs

Harun Mustafa[1,2,3,*], Mikhail Karasikov[1,2,3], Nika Mansouri Ghiasi[4], Gunnar Rätsch[1,2,3,5,6,7], and André Kahles[1,2,3,7,*]

[1] Department of Computer Science, ETH Zurich, Zurich Switzerland
[2] University Hospital Zurich, Zurich, Switzerland
[3] Swiss Institute of Bioinformatics, Zurich, Switzerland
[4] Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich, Switzerland
[5] ETH AI Center, Zurich, Switzerland
[6] Department of Biology, ETH Zurich, Zurich, Switzerland
[7] The LOOP Zurich – Medical Research Center, Zurich, Switzerland

Abstract. Exponential growth in sequencing databases has motivated scalable De Bruijn graph-based (DBG) indexing for searching these data, using annotations to label nodes with sample IDs. Low-depth sequencing samples correspond to fragmented subgraphs, complicating finding the long contiguous walks required for alignment queries. Aligners that target single-labelled subgraphs reduce alignment lengths due to fragmentation, leading to low recall for long reads. While some (e.g., label-free) aligners partially overcome fragmentation by combining information from multiple samples, biologically-irrelevant combinations in such approaches can inflate the search space or reduce accuracy.

We introduce a new scoring model, *multi-label alignment* (MLA), for annotated DBGs. MLA leverages two new operations: To promote biologically-relevant sample combinations, *Label Change* incorporates more informative global sample similarity into local scores. To improve connectivity, *Node Length Change* dynamically adjusts the DBG node length during traversal. Our fast, approximate, yet accurate MLA implementation has two key steps: a single-label seed-chain-extend aligner (*SCA*) and a multi-label chainer (*MLC*). *SCA* uses a traditional scoring model adapting recent chaining improvements to assembly graphs and provides a curated pool of alignments. *MLC* extracts seed anchors from *SCA*'s alignments, produces multi-label chains using MLA scoring, then finally forms multi-label alignments. We show via substantial improvements in taxonomic classification accuracy that MLA produces biologically-relevant alignments, decreasing average weighted UniFrac errors by 63.1–66.8% and covering 45.5–47.4% (median) more long-read query characters than state-of-the-art aligners. MLA's runtimes are competitive with label-combining alignment and substantially faster than single-label alignment.

1 Introduction

Sequencing databases are growing exponentially in size [1]. In recent years, *sequence graphs*, in particular *De Bruijn graphs* (DBGs), have become increasingly prominent models for representing and indexing large collections of sequencing data [2], enabling improvements in both the scale and accuracy of many biological analysis tasks, including genotyping [3, 4], variant calling [5, 6], and sequence search [3, 7].

Established search methods are designed for databases of assembled genomes [8]. However, a large fraction of sequencing data deposited in archives like the Sequence Read Archive (SRA) or the European Nucleotide Archive (ENA) have not yet been assembled [9]. This is because assembly requires expensive compute and human labour resources, done by first representing the overlaps between reads as an *assembly graph* (e.g., a DBG), then extensively cleaning the graph (often requiring manual intervention), and finally assembling contiguous sequences (*contigs*) by graph traversal [10]. Since genome assembly strives for long, high-quality contigs [11], the cleaning may discard a significant amount of signal from the sample [12, 13] with no guarantee that there will be no misassemblies among the final contigs [11].

To avoid these signal reduction and misassembly issues, a common way to compress and index a collection of unassembled read sets for search queries is to first construct and only lightly clean an assembly graph for each read set, then merge the graphs into a *joint assembly graph* [7, 14]. For indexing diverse sequencing data sets, light cleaning is still crucial to reduce the accumulation of noise when indexing diverse collections containing hundreds of thousands of samples [7]. DBG-based indexing tools

encode metadata as *graph annotations*, a key-value store associating each node with one or more metadata tracks, such as sample *labels* [7, 15–23]. Similar to these previous works, we use accession IDs as labels to associate nodes back to their original database entries.

A key search task on these indexes is *sequence-to-graph alignment*, a generalisation of pairwise sequence-to-sequence alignment (i.e., computing the maximum similarity *score* between a *query* and a *target* sequence). In this setting, the target sequences are the spellings of contiguous walks (§2.1) on the sequence graph [24–34]. Many of these tools follow a three-step *seed-chain-extend* search paradigm (§2.2). This involves extracting and *anchoring* query *seeds* to the graph, using a *co-linear chaining* algorithm to construct anchor chains, and then *extending* the chains via a search in the graph to form alignments.

1.1 Challenges when aligning to annotated De Bruijn graphs

A large proportion of read sets in the SRA are sequenced at low depth[8], producing heavily fragmented assembly graphs [13]. For metagenomics samples in particular, the constituent organisms are often sequenced at or below 1× coverage [35]. Although the light cleaning applied to assembly graphs ensures that exact seed matches can be found [7], the long contiguous walks required for high-scoring (i.e., high-precision) alignments are often not present because of high graph fragmentation. This results in low recall, particularly for long reads, because the short alignments that can be found are not reported to maintain search precision.

Current alignment approaches for sequence graphs have limited support for fragmented graphs. The first approach is *label-free* alignment, which ignores annotations during alignment. When applied to an annotated DBG representing a diverse cohort, these tools [21, 25, 26, 28, 29, 36, 37] can meander search through a large search space inflated by biologically-irrelevant sample combinations [3, 15]. For this reason, these tools primarily target single-species pangenomes that often satisfy the assumption that all walks are biologically relevant. If this assumption holds, then these methods can overcome fragmentation in an individual sample's assembly graph by combining sequence information from multiple samples since such contiguous walks are present in the joint graph. A second approach aligns queries to walks where all nodes in the walk share one (*single-label*) or more (*label-consistent*) common label(s) [24, 38]. These tools suffer from low recall on fragmented assembly graphs, and so, this property also applies to joint annotated graphs because sample combinations cannot compensate for fragmentation. However, these tools do not suffer from search space inflation because all walks are biologically relevant. This is why these tools are applied to high-quality contiguous graphs indexing reference genomes or high-depth sequencing samples [24, 38]. A third, recently-emerging intermediate approach is *haplotype-aware* alignment, which either aligns in a label-free fashion and scores recombinations afterwards [39] or combines samples to a restricted degree by penalising each combination during alignment [40, 41]. These alignment strategies do not consider similarity (hence, biological relevance) when scoring a sample change and have so far only been applied to single-species pangenomes.

A property shared by all of these approaches is that a discontinuity in an individual assembly graph that does not overlap with another assembly graph will propagate to the joint DBG, limiting the joint graph's contiguity. This stems from the approaches' shared definition of alignment: the alignment target must be a contiguous walk.

1.2 Contribution: Label-guided sequence-to-DBG alignment

Our goal in this work is to develop an alignment approach that can produce long accurate alignments to collections of low-depth sequencing samples represented by fragmented annotated DBGs. To this end, we propose a new alignment strategy called *multi-label alignment* (MLA) designed for annotated DBGs. The strategy extends alignment scoring models with two key new operations: *(i) Label Change* and *(ii) Node Length Change*. The label change operation penalizes traversals that change from one sample to a dissimilar sample, thus enhancing local single-character similarity scores with more informative global similarity. The node length change operation dynamically adjusts the node length, thus using shorter-length nodes as proxies for missing nodes to locally improve graph connectivity (i.e., reducing fragmentation).

[8] The fungi SRA samples indexed by [7] have a median (mean) k-mer multiplicity of 9 (10.5) (Supp. Fig. 2).

Efficiently implementing these operations must overcome several computational challenges. First, efficient anchor chaining relies on having a small number of anchors per query [42], an assumption that is easily violated in a multi-label setting. Second, sequence-to-graph alignment decision problems (i.e., does there exist an alignment) that allow for DBG edits is shown to be $\mathcal{NP}$-complete [43], necessitating heuristics to prevent excessive use of node length change operations.

To address these challenges, we implement MLA in a fast, approximate, yet accurate two-step approach: The first step is a new single-label seed-chain-extend aligner for annotated DBGs called *SCA* (Fig 1.1-3), meant to reduce the computational burden of multi-label chaining by first performing single-label chaining with a traditional scoring model and extending the top chains among all labels to provide a preliminary alignment pool. *SCA*, adapts recent improvements in chaining to a DBG setting. The second step is a multi-label chainer called *MLC* (Fig 1.4) that incorporates our novel MLA scoring operations into its chain scoring. It extracts anchors from the alignments provided by *SCA*, resulting in a much smaller curated anchor set on which we apply our more expensive operations. It then performs multi-label chaining on these anchors and stitches the multi-label chains into alignments using fragments from *SCA*'s alignments.

Overall, we show in this work how our fast approximate implementation of MLA produces substantially longer and more accurate alignments compared to state-of-the-art aligners.

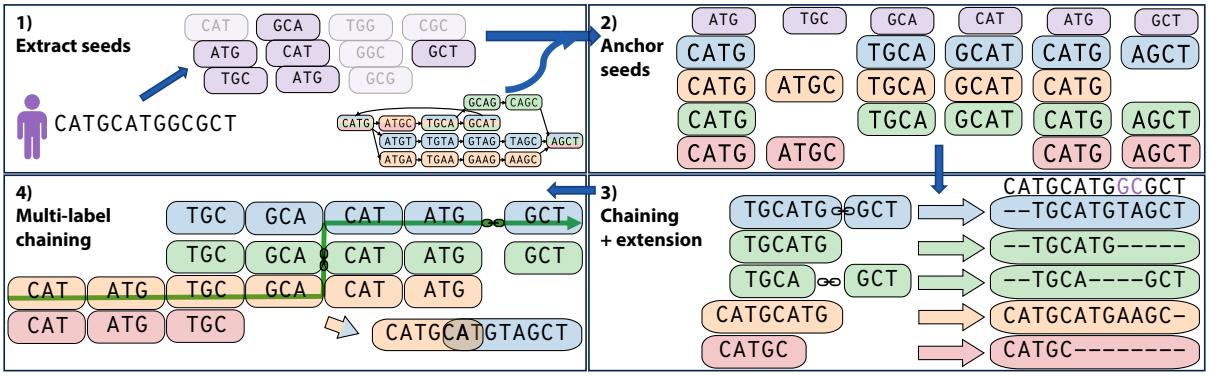

Fig. 1. Computing multi-label alignments (MLAs) of a query sequence to an annotated De Bruijn graph. Each colour represents a label in the graph annotation, with some nodes having multiple labels. We first *1)* extract seeds (shown in purple, shaded seeds have no match) of length $l \leq k$ from the query sequence (in this example, $k = 4$, $l = 3$, and the query is `CATGCATGGCGCT`) and *2)* anchor the seeds to the graph, where each anchor matches a seed to a node and a label (each column represents the node-label pairs to which a seed matches). Then, we *3)* construct single-label chains and extend them along single-label walks into alignments using *SCA* (the purple characters in the query indicate mismatched characters). We then *4)* extract anchors from this alignment pool and form multi-label chains using *MLC* (indicated by the green line). We connect anchors using single-label alignment segments to form MLAs.

2 Preliminaries and Background

2.1 Notation and Definitions

A *string* is a finite sequence of characters drawn from an alphabet Σ. Σ^k denotes the set of all strings of length k (*k-mers*). For a string $s = s[1]s[2]\cdots s[l]$ of length $|s| = l$, with indices $1 \leq i \leq j \leq l$, we denote a substring by $s[i:j] := s[i]\cdots s[j]$. The set of all k-mers extracted from a string set S is denoted by $\mathcal{K}(S,k) := \bigcup_{s\in S}\bigcup_{i=1}^{|s|-k+1}\{s[i:i+k-1]\}$.

A *node-centric De Bruijn graph* (DBG) of order k representing S has the nodes $V := \mathcal{K}(S,k)$ and implicit edges $E(V) := \{(v_1,v_2) \in V^2 : v_1[2:k] = v_2[1:k-1]\}$ [44]. The *spelling* of a walk $W = (v_1,\ldots,v_m)$ on a DBG is the string $T = v_1 v_2[k]\cdots v_m[k]$. An *annotated DBG* has an auxiliary set of string labels $\mathcal{L}$ and an annotation $\mathcal{A} : V \to 2^{\mathcal{L}}$ associating each node with a label set. W is *label consistent* if $\bigcap_{i=1}^{m}\mathcal{A}(v_i) \neq \emptyset$. $\mathcal{V} : \mathcal{L} \to 2^V$ fetches all nodes with a given label.

We denote a query string by $Q \in \Sigma^{|Q|}$. A *sequence-to-graph alignment* of Q to the *target string* T along W is a tuple $a = (Q_a, T_a, E_a, W)$, where Q_a is a substring of Q, T_a is a substring of T, and E_a is a sequence of *edit operations* transforming T_a to Q_a. These operations are in {match, mismatch, insertion open, insertion extension, deletion open, deletion extension}. Each operation has a *score*, denoted by $\Delta_=$, $\Delta_\neq$, Δ_{IO}, Δ_{IE}, Δ_{DO}, and Δ_{DE}, respectively. Only $\Delta_=$ is positive, all other scores are negative. The score of a, denoted by $\Delta_S(a)$, is the sum of all edit operation scores, where a higher score indicates greater similarity. a is label consistent if W is label consistent. Given two alignments a_1, a_2 with respective substrings $Q[i_1 : i_1+l_1-1], Q[i_2 : i_2+l_2-1]$ s.t. $i_1 < i_2$, we define $\text{overlap}(a_1, a_2) := \min\{l_2, i_1 + l_1 - i_2\}$. a_1 and a_2 *overlap* if $\text{overlap}(a_1, a_2) > 0$ and are *disjoined* by a *gap* of length $-\text{overlap}(a_1, a_2)$ otherwise. For a gap length $l \in \mathbb{N}^+$, we denote the scoring model's *gap penalty* by $\Delta_G(l)$. In this work, we assume affine scoring (e.g., $\Delta_G(l) := \Delta_{IO} + (l-1)\Delta_{IE}$ for an insertion).

2.2 Sequence-to-graph alignment with seed-extend-chain

Modern approximate aligners predominantly use the *seed-chain-extend* paradigm involving *(i) seed anchoring*, *(ii) co-linear chaining*, and *(iii) anchor extension* [45]. A *seed* is a query substring while an *anchor* is a tuple of a seed and a graph walk spelling a superstring of the seed. A *chain* is a sequence of anchors s.t. any two consecutive anchors are in order in both the query and target, meaning that there exists a walk connecting their nodes [46]. Some works determine a traversal distance between nodes on-the-fly by traversing local neighbourhoods around nodes [29, 30]. More efficient strategies require a decomposition of the graph, typically into subgraphs [28, 47] or a path/walk cover [31–33, 46]. After chaining, anchor extension searches the graph forwards and backwards from the ends of each anchor to find high-scoring walks.

3 Methods

3.1 General alignment workflow

Given a query Q, we find and anchor seeds, compute label-consistent alignments using *SCA*, then chain these alignments together into MLAs using *MLC*. First, given a user-set seed length $l \leq k$, we extract l-mer seeds from Q (Fig. 1.1) and anchor them by fetching all graph nodes with matching l-length suffixes and their associated labels (Fig. 1.2). An anchor $\alpha_{ilv\ell}$ is a tuple in $A := \mathbb{N} \times \mathbb{N} \times V \times \mathcal{L}$ anchoring the seed $Q[i : i + l - 1]$ to a node v, with an associated label $\ell \in \mathcal{A}(v)$. Afterwards, we find the top-scoring single-label chains (Fig. 1.3, §3.4). We extend each chain into a single-label alignment by using global alignment to connect consecutive anchors and ends-free extension from the first and last anchor in the chain. Given the resulting alignment pool, we extract all l-mer anchors and compute multi-label chains (Fig. 1.4, §3.5), incorporating label change (Δ_{LC}) and node length change operations (Δ_L). We construct MLAs from the top multi-label chains using segments from the label-consistent alignments.

3.2 Deriving a Scoring Model for Novel Alignment Operations

We now detail the scoring model for our new operations for sequence-to-graph alignment: Δ_{LC} and Δ_L. For all constants defined in this section, refer to §4.3 for the values we use in our experiments. We define and extend two probabilistic graphical models for sequence-to-sequence alignment: a *target model* and a *null model*, based on the probabilistic models presented by [48]. Briefly, a model represents all possible mutations of an underlying target sequence, where a walk in a model emits edit operations that generate a query sequence (detailed in Supp. §2.2). The score of any edit operation is the log probability ratio of the operation's corresponding transition probabilities in the target and the null model. The alignment score is the sum of these log ratios (i.e., the query's log-likelihood ratio). These sequence-to-sequence alignment models trivially induce analogous models for sequence-to-graph alignment by treating the spelling of each walk in a sequence graph as a separate target sequence and, hence, a pair of target and null graphical models.

Deriving and Computing Label-change Scores For simplicity, suppose that we traverse along an edge (v_1, v_2) s.t. $\mathcal{A}(v_1) = \{\ell_1\}$ and $\mathcal{A}(v_2) = \{\ell_2\}$, where $\ell_1 \neq \ell_2$. We denote this *label change* from ℓ_1 to ℓ_2 as $\ell_1 \rightarrow \ell_2$ and score this event using the probability that v_2 has label ℓ_1 conditioned on v_2 having the label ℓ_2. Intuitively, this is the probability that the k-mer v_2 observed in the sample with label ℓ_2 is also present in the sample with label ℓ_1, but was not observed (Fig. 2). To formulate this precisely, we extend our alignment models so that each graph-traversing operation emits a label change with transition probability $\Pr(\ell_1 \rightarrow \ell_2)$. Thus, the *label-change score* is

$$\Delta_{LC}(\ell_1 \rightarrow \ell_2) := \lambda_{LC} \left\lfloor \log_2 \frac{\Pr(\ell_1 \rightarrow \ell_2)}{\Pr_0(\ell_1 \rightarrow \ell_2)} \right\rfloor, \tag{1}$$

where λ_{LC} is a user-set scaling constant. For the null model, we assume no relationship between ℓ_1 and ℓ_2, so $\Pr_0(\ell_1 \rightarrow \ell_2) := \Pr(\ell_2)$, where $\Pr(\ell)$ is the empirical probability of a label ℓ: $\Pr(\ell) := \frac{|\mathcal{V}(\ell)|}{|V|}$. For this new model to be reducible to a label-free setting, we require that $\Delta_{LC}(\ell \rightarrow \ell) = 0$ (i.e., only score if the label changes) and that $\Delta_{LC}(\ell_1 \rightarrow \ell_2) \leq 0$ (i.e., no label change increases the score). We satisfy these requirements if $\Pr(\ell_1 \rightarrow \ell_2) := \Pr(\mathcal{V}(\ell_1) \cap \mathcal{V}(\ell_2))$, which simplifies Equation 1 to

$$\Delta_{LC}(\ell_1 \rightarrow \ell_2) = \lambda_{LC} \lfloor \log_2 \Pr(\ell_1 | \ell_2) \rfloor . \tag{2}$$

Due to the log in the definition of Δ_{LC} and the use of integers for alignment scores, we only require order-of-magnitude accuracy when computing $\Pr(\ell_1|\ell_2)$. So, we approximate $\Pr(\ell_1|\ell_2)$ using HyperLogLog++ counters [49] of $\mathcal{V}(\ell_1)$ and $\mathcal{V}(\ell_2)$, respectively, and the precomputed values $|\mathcal{V}(\ell_1)|$ and $|\mathcal{V}(\ell_2)|$.

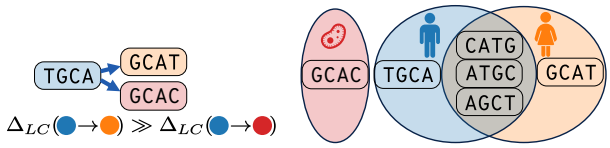

Fig. 2. Label change scores measure sample similarity. Δ_{LC} measures the probability that a node with an orange label is also present in the blue sample but was not observed in that sample. We score a change to the orange label much higher because of the large overlap between the orange and blue k-mer sets.

Deriving Penalties for Node Length Changes Since low sequencing depth can produce disconnected graphs, one way to compensate for this is to allow for node insertions into the graph during the search. In this section, we describe how we use dynamic changes of the underlying DBG's order k during alignment as a more tractable proxy for inserting nodes.

Although *MLC* only utilises node length change operations during chaining, we nonetheless incorporate this operation into our scoring model to derive its scoring function. For this, we switch our graph from a DBG of fixed order k to a variable-order DBG of maximum order k [50].

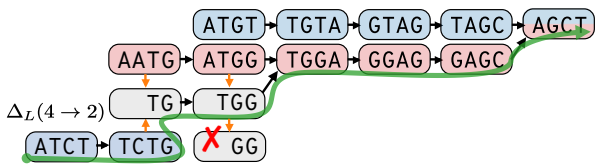

Fig. 3. Traversal in a variable-order DBG overcomes graph disconnects to spell the sequence `ATCTGGAGCT`. Traversing from `TCTG` to its suffix `TG` allows for traversal to `TGGA`, so the red nodes compensate for the disconnect in the blue subgraph. In MLA, nodes of length $l < k$ act as proxies for k-mers. In this graph, `TGG` stands in for `CTGG`. In our model, nodes of length $< k$ may only traverse to longer nodes (e.g., `TGG` cannot traverse to `GG`). The green line represents the path taken to spell the sequence above and the orange arrows represent node length-changing traversals.

Given a node v of length k with spelling s and a suffix length $1 \leq l < k$, a node length change is the traversal from v to the node with spelling $s[k-l+1:k]$. In our model, nodes with length l are proxies for missing k-mers (Fig. 3). However, we need to define penalties for changing the node length to prevent degenerate cases, such as searches along walks where every node has a short length (e.g., 1 or 2) that exist for every sequence over the alphabet.

To avoid this case, and to ensure that the model reduces to standard sequence-to-graph alignment on fixed-k DBGs, our scoring model does not penalise traversals that increase the node length by 1 or maintain the node length at k. Otherwise, we penalise traversal from a node of length k to one of length $1 \leq l < k$ with a score $\Delta_L(k \to l)$ and disallow all other node length-changing traversals. Consolidating these rules,

$$\Delta_L(l_1 \to l_2) := \begin{cases} 0 & \text{if } l_1 = l_2 = k \text{ or } l_1 + 1 = l_2 \\ (k - l_2)\Delta_J & \text{if } l_1 = k \text{ and } l_2 < k \\ -\infty & \text{if } l_1 < k \text{ and } l_1 + 1 \neq l_2 \end{cases}, \tag{3}$$

where $\Delta_J < 0$ is a user-set constant.

3.3 Local co-linear chaining on assembly DBGs

We now describe the modular chaining algorithm used by both *SCA* and *MLC*. Given anchors sorted by increasing end position (i.e., $i + l$), we perform local chaining using Algorithm 1, based on the more practical alternative algorithm implementation by [42][9]. Their algorithm minimises a non-negative chain cost rather than maximising an integer score, so we use the equations by [52] to convert scores into costs when evaluating our termination condition. This *forward pass* computes optimal chaining scores. We reconstruct chains by *backtracking*, ensuring that we incorporate each anchor into, at most, one chain.

The helper function $\mathsf{connect} : A \times A \to \mathbb{Z}$ approximates the score of a global alignment connecting anchor α_1 to α_2, with different implementations for *SCA* and *MLC*. Note that this score does not include α_1 since its score is already included in the score vector Δ_S when computing updates.

Algorithm 1 Computing local co-linear chaining scores.

Input: A sequence of anchors $(\alpha_{i_1 l_1 v_1 \ell_1}, \ldots, \alpha_{i_n l_n v_n \ell_n})$ s.t. $x < y \implies i_x + l_x \leq i_y + l_y$. An initial guess of the maximum distance between anchors $b \in \mathbb{N}^+$ and a scaling factor $b_2 > 1$.

Output: $\Delta_S \in \mathbb{Z}^n$ s.t. $\Delta_S[j]$ is the best chaining score from any subsequence of $(\alpha_{i_1 l_1 v_1 \ell_1}, \ldots, \alpha_{i_j l_j v_j \ell_j})$ ending with $\alpha_{i_j l_j v_j \ell_j}$.

```
Δ_S ← [Δ_= · l_1, ..., Δ_= · l_n]
repeat
    F ← 1
    for L ← 1 to n do
        F ← min {j ∈ {F, ..., L} : i_L + l_L − (i_j + l_j) ≤ b}
        for j ← F to L s.t. i_j < i_L and i_j + l_j < i_L + l_L do
            s ← Δ_S[j] + connect(α_{i_j l_j v_j ℓ_j}, α_{i_L l_L v_L ℓ_L})
            Δ_S[L] ← max {Δ_S[L], s}
        end for
    end for
    b_last ← b
    b ← b · b_2                                   ▷ Increase max. distance between anchors
    s_best ← max {Δ_S[1], ..., Δ_S[n]}            ▷ Find the best score.
    c_best ← ⌈2 (|Q| − s_best/Δ_=)⌉                ▷ Convert score into a cost.
until c_best > b_last
return Δ_S
```

[9] We fall back to the chaining algorithm from minimap2 [51] with affine gap scoring if there are more than $|Q|$ anchors (§3.6).

3.4 *SCA*: Single-label seed-chain-extend alignment

SCA implements co-linear chaining and anchor extension algorithms for label-consistent alignment. After seed anchoring, we merge the anchors into maximal unique matches (MUMs). We then group the anchors by label and perform separate forward passes of the chaining algorithm for each group. Afterwards, we select the top $\rho|Q|$ chains (with ties) among all labels and backtrack to construct these chains. ρ is a user-set chain density (§4.3).

To compute the connection score between two anchors α_1 and α_2, we sum a match score for the additional characters introduced by α_2 to a gap penalty based on the absolute difference between *(i)* the difference of α_1 and α_2's seed end positions in the query, and *(ii)* the traversal distance between their nodes in the graph (Supp. Algorithm 1). To quickly estimate a traversal distance between two nodes along a label-consistent walk, we use *walk covers* of the subgraphs representing each label. A walk cover is a set of walks s.t. each node is visited at least once. So, a traversal distance is known if there exists a walk from α_1 to α_2 in the cover.

For each chain, we connect consecutive anchors using global alignment, then extend the first anchor backwards and the last anchor forwards using ends-free extension. For global alignment, we use *TCG-Aligner* [15] to ensure that each connecting walk is represented by the cover. For ends-free extension, we modify *TCG-Aligner*'s extender to restrict traversal to label-consistent walks. To further reduce the number of extensions, we discard chains that overlap with already-completed alignments. The result is a pool of label-consistent alignments.

3.5 *MLC*: Multi-label co-linear chaining

Given a pool of label-consistent alignments, we extract anchors from these alignments and construct multi-label chains (Fig. 1.4). Unlike the co-linear chaining method in *SCA*, we have access to global alignment scores for connecting any in-order pair of anchors extracted from the same label-consistent alignment. We leverage this to define an anchor connection score that ensures that *MLC*'s chaining scores equal the final MLA scores.

One property of *MLC*'s scoring is that, given a chain of anchors from the same alignment and label, we only need the anchors at the beginning and end of the chain to compute the chain score. So, as a preprocessing step, we discard any anchor $\alpha_{i_j l v_j \ell_j}$ if it is the only anchor at position i_j and if another anchor $\alpha_{i_{j+1} l v_{j+1} \ell_j}$ exists from the same alignment.

After the forward pass, we reconstruct the highest-scoring chains that cover each label from the pool of label-consistent alignments. We construct an MLA from each chain by connecting consecutive anchors using alignment segments from the pool.

Multi-label anchor connection scoring We now detail *MLC*'s anchor connection scoring (Supp. Algorithm 2). Suppose we have anchors $\alpha_{i_j l v_j \ell_j}$ and $\alpha_{i_L l v_L \ell_L}$ from alignments a_x and a_y, respectively. We consider three cases for the update score: *(i)* extending along the same alignment (i.e., $x = y$), *(ii)* connecting disjoint alignments (i.e., $\text{overlap}(a_x, a_y) \leq 0$), and *(iii)* connecting overlapping alignments (i.e., $\text{overlap}(a_x, a_y) > 0$).

(i) If $x = y$, then the anchor connection score is the score of a_y's segment up to the longer query substring ending at $i_L + l$ (i.e., $\Delta_S(a_y[: i_L + l]) - \Delta_S(a_y[: i_j + l])$).

(ii) If a_x and a_y cover disjoint regions of Q, then the connection score includes the sum of a_x's remaining score, the score of a_y's segment up until $i_L + l$, the label change score, and the gap penalty for inserting extra query characters:

$$\begin{aligned} &\Delta_S(a_x) - \Delta_S(a_x[: i_j + l]) + \Delta_S(a_y[: i_L + l]) \\ &\quad + \Delta_{LC}(\ell_j \to \ell_L) + \Delta_{DO} + \Delta_G(-\text{overlap}(a_x, a_y)). \end{aligned} \tag{4}$$

To differentiate this case from *(iii)*, we insert a sentinel \$ character into the target spelling with an incurred Δ_{DO} score.

(iii) If a_x and a_y cover overlapping regions of Q, assume that the current alignment segment of a_x up until $i_j + l$ is of length $\geq k$ and that $|a_y[i_L :]| \geq k$ (meant to avoid ambiguity when spelling a walk). We also only consider overlaps of length $< k - 1$ (i.e., one cannot traverse from a_x to a_y with a single step without modifying the graph) since we assume that longer overlaps would have been discovered

otherwise via normal graph traversal during anchor extension. We use a two-step procedure to try to connect the two anchors. First, we try to find an intermediate anchor $\alpha_{i_{j'} l v_{j'} \ell_L}$ from a_y s.t. $i_j = i_{j'}$ and hop to that node. Then we continue the traversal along a_y towards $v_{j'}$ to complete the connection. If such a node exists, the connection score is

$$\begin{aligned} &\Delta_S(a_y[: i_L + l]) - \Delta_S(a_y[: i_j + l]) + \Delta_{LC}(\ell_j \to \ell_L) \\ &\quad + \Delta_L(k \to l) \cdot \mathbf{1}_{v_j \neq v_{j'}} \end{aligned} \tag{5}$$

Using Fig. 3 as an example, one such intermediate anchor is the node `AATG` sharing a 2-mer suffix with the node `TCTG`.

Note that this score does not depend on the length of v_j and $v_{j'}$'s longest common suffix if $v_j \neq v_{j'}$. We motivate this design choice with the following observation: If there is a single character mismatch between position $k - l' + 1$ (where l' need not equal the seed length l) in a node v's spelling and position $k - l'$ in another node v''s spelling that prevents the nodes from being adjacent (e.g., v spells `GATGC` and v' spells `ACGCT` for $k = 5$ and $l' = 3$), then we require l' node insertions to create a path to v' from a predecessor of v, despite the ground truth that it originated from a single substitution (for our example, given appropriate characters $c, d \in \Sigma$, these new nodes would spell cd`GAC`, d`GACG`, and `GACGC`). Thus, defining the connection score as a function of $k - l'$ instead of $k - l$ would induce unequal scores for each value of l', even though all of these edit events are equally likely.

3.6 Time and Space Complexity

Suppose we have n sorted anchors (incurring a worst-case complexity of $\mathcal{O}(n \log n)$ if the seeder does not produce sorted anchors). Let C denote the time to execute a `connect` function. If $n \leq |Q|$, then Algorithm 1 has an average-case time complexity in $\mathcal{O}(c^*|Q|C)$, where c^* is the minimum chain cost [42]. If $n > |Q|$, then we fall back to the chaining algorithm from minimap2 [51] with a worst-case time complexity in $\mathcal{O}(bnC)$ for the forward pass, where b is a user-set bandwidth parameter. Although we perform $\rho|Q|$ backtracking procedures, since each anchor is incorporated into at most one chain, we terminate a procedure as soon as we reach an already-chained anchor. So, backtracking takes worst-case $\mathcal{O}(n)$ time. Alongside the n anchors, we store $\Theta(1)$ information per anchor for the chaining scores and backtracking information, so the total space complexity is in $\Theta(n)$.

Consider *SCA*'s `connect`. Suppose that Q maps to L labels, with corresponding anchor counts $n_1, \ldots, n_L$, walk cover sizes $W_1, \ldots, W_L$, and optimal chaining costs $c_1^*, \ldots, c_L^*$. Since MUMs generally satisfy $n_i \leq |Q|$ [42], the average-case time complexity of *SCA*'s forward passes is $\mathcal{O}\left(|Q| \sum_{i=1}^{L} c_i^* W_i\right)$. After extension, we extract m anchors from the resulting alignments.

Considering *MLC*, it is clear that $C_{\text{MLA}} \in \mathcal{O}(1)$. Since we chain anchors from multiple labels in a single forward pass, we can no longer assume that $m \leq |Q|$ will generally hold. Thus, MLA chaining has an average-case time complexity in $\mathcal{O}(c^*|Q|)$ when $m \leq |Q|$ and worst-case time complexity in $\mathcal{O}(bm)$ otherwise. Splicing alignment segments to convert a multi-label chain into an MLA spelling T runs in worst-case $\mathcal{O}(|T|)$ time.

4 Evaluation Methodology

We compare our methods to *GraphAligner* [28], a state-of-the-art tool for traditional sequence-to-graph alignment to DBGs, and *PLAST* [24], a BLAST-like tool for label-consistent alignment to annotated DBGs. We implement two additional baselines: *SCA+FixedMLC100* implements multi-label chaining with $\Delta_{LC} = -100$, while *SCA+MLC(no NLC)* implements our Δ_{LC} but disallows node length changes. We evaluate all tools on a simulated joint assembly graph indexing 3042 fungi genomes from GenBank [53].

4.1 Simulating assembly graphs and query sequences

From each genome, we simulate a 10×-coverage Illumina HiSeq-type read set using *ART* v2.5.8 [54] and construct a cleaned assembly DBG of order $k = 31$ using the procedure in *MetaGraph* [7]. We then

merge these graphs into an annotated DBG with nodes labelled by their source accession IDs. To merge the graphs, we compute walk covers for each graph and construct the joint graph from these sequences.

We pre-process the graph and generate walks using the procedure described by [33]. To reduce the final representation size, we refrain from maintaining a matrix of traversal distances from each node to each stored walk. Instead, we augment each cover with additional walks spanning the edges discarded during pre-processing. We losslessly encode the walks as coordinate graph annotations [15]. We use the GFA representation of the joint graph to interface with *GraphAligner* and the walk covers to construct the *PLAST* index.

We simulate query reads from the same genomes with a different random seed, using *ART* for Illumina HiSeq-type reads and *pbsim3* [55] for PacBio Sequel CLR-type, PacBio Sequel HiFi-type, and ONT-type reads. We generate HiFi reads from simulated 10-pass Sequel subreads using the PacBio `ccs` tool. We form a query set for each read type by drawing 100 random reads from the pool of reads aggregated from all genomes.

4.2 Recall-, coverage- and taxonomy-based measures

Alongside read mapping quality, we measure the accuracy of the retrieved labels w.r.t. the ground-truth genomes using *taxonomic profiles* since our query reads are simulated from known genomes.

Taxonomic classification accuracy is highly dependent on which alignments are reported in a read mapping. So, we first find an appropriate score cut-off to determine which alignments to report for each read. Since our test reads are of different lengths, we divide each score by the length of its corresponding query to get a *relative score*. Given the alignments of a read sorted by decreasing relative score and a cut-off, we select alignments by greedily picking a disjoint subset whose relative scores are above the cut-off. We vary the cut-off from 0.0 to 1.0 in steps of 0.02. For each cut-off, we evaluate the *(i)* recall and the *(ii)* mean taxonomic profile error. We measure taxonomic profile error using the WGSUniFrac error of the profile relative to the ground-truth profile [56], a measure of the fraction of the taxonomic tree traversal distance that differs between the two profiles. For easier interpretation, we define the *UniFrac accuracy* as $1.0 -$ WGSUniFrac error. Since taxonomic IDs are not available for all strains, we generate a custom taxonomic tree by augmenting the NCBI Taxonomy with a new leaf node for each GenBank accession ID. The ground-truth profile of a read states that its ground-truth accession is 100% abundant, whereas the profiles computed from alignments weight each accession by the fraction of query characters covered by alignments to that accession.

4.3 Experimental setup and code availability

We performed all experiments on an AMD EPYC-Rome processor from ETH Zurich's high-performance compute systems using a single thread and a seed size of $l = 19$, with default parameters for all tools except for the following: For scoring, we set $\Delta_{=} = 1$ and $\Delta_{\neq} = \Delta_{IO} = \Delta_{IE} = \Delta_{DO} = \Delta_{DE} = -1$. For *SCA* and *MLC*, we use a chain density of $\rho = 0.01$ and chaining parameters $b = 400$ and $b_2 = 4$. In *MLC*, we set the label-change score scaling factor to $\lambda_{LC} = \Delta_{=}$ and the node length change scaling factor to $\Delta_J = \Delta_{DE}$. We implemented our methods within the GPLv3-licensed *MetaGraph* framework [7] hosted at `https://github.com/ratschlab/metagraph`. The data and scripts for reproducing our results are available from the `biorxiv` branch at `https://github.com/ratschlab/mla/tree/biorxiv`.

5 Results and Discussion

5.1 Assembly graphs are much wider than pangenomes

Our simulated assembly graphs have a median (mean) graph size of 48,660 (122,826) k-mers (Supp. Fig. 1). After merging these assembly graphs, the annotated DBG contains 187,662,586 k-mers. We observe that unlike the pangenome graphs explored in previous chaining works with widths ranging from 1 to 60 [31–33], our assembly graphs are much wider (i.e., their minimal walk covers are large), with a median (mean) width of 221 (542.8). These observations corroborate our choice to refrain from encoding a full node-to-walk distance matrix for *SCA*.

5.2 Multi-label alignments are substantially longer, computed at competitive execution times

For all read types, we observe that full MLA (*SCA+MLC*) produces the longest alignments (Fig. 4), with median read coverage increases of 47.4% for HiFi reads, 46.7% for CLR reads, and 45.5% for ONT reads, respectively, relative to the next-best state-of-the-art aligner. All methods have median coverages of 100% for Illumina reads, with the smallest inter-quartile range observed for MLA. Our baselines *SCA+FixedMLC100* and *SCA+MLC(no NLC)* achieve similar or slightly improved performance relative to *SCA*, but far below MLA. *PLAST* has the highest third quartile coverage for PacBio reads among all tools and higher median coverage on HiFi reads than the most comparable tool *SCA*. However, its execution time is ∼280–1200× slower than *SCA* (Fig. 7).

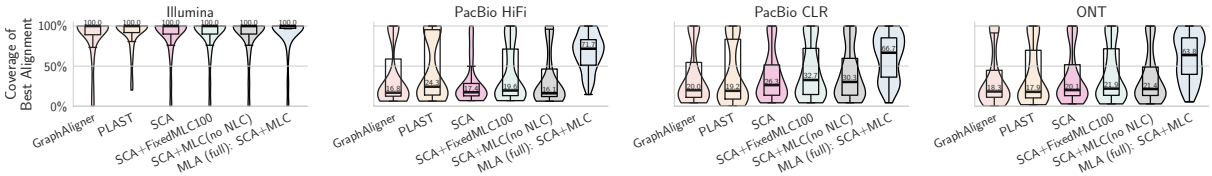

Fig. 4. Coverage of each read's best alignment. Coverage is the percentage of query characters covered by an alignment. *SCA+FixedMLC100* sets $\Delta_{LC} = -100$ for label changes. *SCA+MLC(no NLC)* uses our Δ_{LC}, but not does allow node length changes.

5.3 MLA maintains the best classification accuracy at most read mapping recall levels

Sweeping through different relative score cut-offs, we observe that full MLA has the greatest recall on error-prone reads for all cut-offs and the greatest recall for HiFi reads at cut-offs below 0.5 (Fig. 5). All tools perform similarly on Illumina reads. When comparing taxonomic profiles, MLA has the greatest UniFrac accuracy for long reads at recall values above 20% for HiFi and ONT reads and above 50% for CLR reads (Fig. 6). All tools perform similarly on Illumina reads. For a score cut-off of 0.0, all methods have notable decreases in UniFrac accuracy. The areas under the mean UniFrac Accuracy-Recall curves (AUARCs) for Illumina reads are 0.90 for all methods. For all long-read types, MLA has the highest AUARCs, ranging from 0.85–0.88, compared to *SCA* (0.80–0.83), *SCA+FixedMLC100* (0.82–0.84), *SCA+MLC(no NLC)* (0.82–0.85), and *PLAST* (0.55–0.68). We infer from these results that MLA improves accuracy by chaining together alignments to related samples.

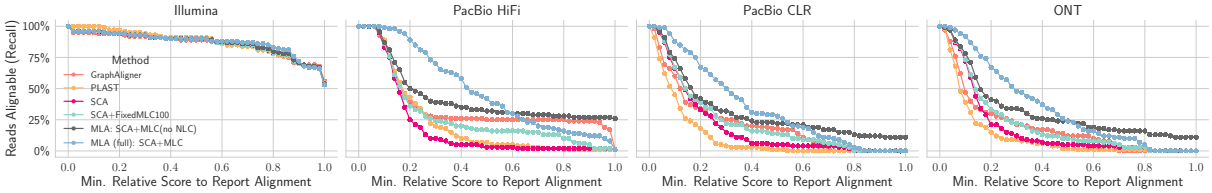

Fig. 5. Read mapping recall for different relative score cut-offs. For each relative score (i.e., score scaled by query length) cut-off, the recall is the fraction of reads with an alignment scoring at least at that cut-off.

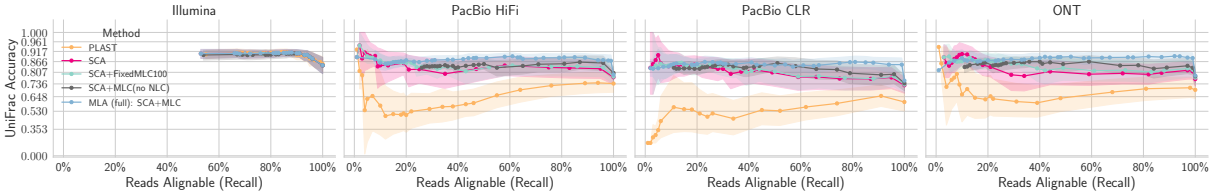

Fig. 6. Mean UniFrac accuracy of all mapped reads at different read mapping recall levels. UniFrac accuracy measures the similarity between a read's taxonomic profile (induced from the labels of all reported alignments) and its ground-truth profile, similar to precision. We estimate means from 1000 bootstrap samples, with shading representing the 95% CI of the mean. Based on our interpretation of WGSUniFrac error values (detailed in Supp. §2.4), each y-axis grid line corresponds to the midpoint value of a taxonomic rank (accession, strain, species, genus, etc.). The y-axis is scaled to better emphasise the upper range of UniFrac accuracy values, corresponding to accuracy at lower taxonomic ranks. We exclude *GraphAligner* since it does not consider labels.

5.4 Overall performance

Overall, MLAs are substantially longer than alignments produced by other tools (Fig. 7). We achieve these results while maintaining competitive execution times and the lowest RAM usage. For Illumina reads, *SCA* and *MLC* are substantially faster than all other tools. *GraphAligner* is the fastest tool for long reads, with *SCA* and MLA having comparable execution times.

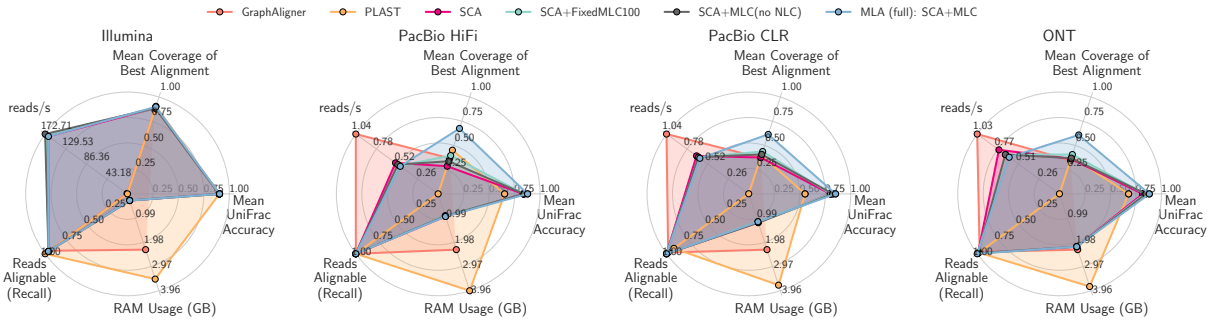

Fig. 7. Comparison of performance measures for each alignment tool. See Supp. Tab. 2 for the numbers plotted here.

6 Conclusions

In this work, we presented the multi-label alignment (MLA) strategy, a novel alignment scoring model that compensates for disconnects in low-depth assembly graphs by combining sequence information from multiple related samples and improving graph connectivity during alignment. We implement MLA on annotated De Bruijn graphs (DBGs) with a two-step process: *SCA* computes label-consistent alignments and *MLC* computes multi-label alignments from alignments produced by *SCA*.

Since *SCA* does not encode a traversal distance matrix from each node to each walk encoded by the graph annotation, providing such a matrix can potentially increase label-consistent alignment lengths and consequently provide a stronger basis for *MLC*. Despite the large widths of assembly graphs, we expect this matrix to be sparse and, hence, easily compressible.

Another possible extension is merging *SCA* and *MLC* into a single holistic seed-chain-extend procedure. We maintained these two steps in this work to provide a small number of anchors to each chaining run. One can explore how well our current approach approximates this unified approach and how to approximate an ends-free extension incorporating label and node length changes. In this context, there are a few possibilities for implementing the node length change operation into chain scoring. These include using the current procedure of finding intermediate suffix-sharing nodes and a more sensitive, but daunting approach of representing walk covers for all desirable node length values $l < k$.

Although our algorithms are implemented within the *MetaGraph* framework, the concepts from our methods can readily be applied in other pangenome graph frameworks and potentially see more widespread use. These methods make unassembled read sets a more powerful resource for bioinformatics research.

Acknowledgements

We thank Ragnar Groot Koerkamp, Maximilian Mordig, Mohammed Alser, Mario Stanke, Inanc Birol, and anonymous reviewers for their helpful discussions and feedback. H.M. and M.K. were partially funded as part of Swiss National Research Programme (NRP) 75 “Big Data” by the SNSF grant #407540_167331. A.K. is partially funded by The LOOP Zurich and the Monique Dornonville de la Cour Foundation to G.R. H.M., M.K., and A.K. were also partially funded by ETH core funding (to G.R.). H.M. is also partially funded by the Personalized Health and Related Technologies (PHRT) Transition Postdoc Fellowship Project #2021/453. We declare no conflicts of interest.

References

1. K. Katz, O. Shutov, R. Lapoint, M. Kimelman, J. Brister, and C. O'Sullivan, "The Sequence Read Archive: a decade more of explosive growth," *Nucleic Acids Research*, vol. 50, no. D1, pp. D387–D390, 11 2021.
2. C. Marchet, C. Boucher, S. J. Puglisi, P. Medvedev, M. Salson, and R. Chikhi, "Data structures based on k-mers for querying large collections of sequencing data sets," *Genome Research*, vol. 31, no. 1, pp. 1–12, 2021.
3. J. Sirén, J. Monlong, X. Chang, A. M. Novak, J. M. Eizenga, C. Markello, J. A. Sibbesen, G. Hickey, P.-C. Chang, A. Carroll, N. Gupta, S. Gabriel, T. W. Blackwell, A. Ratan, K. D. Taylor, S. S. Rich, J. I. Rotter, D. Haussler, E. Garrison, and B. Paten, "Pangenomics enables genotyping of known structural variants in 5202 diverse genomes," *Science*, vol. 374, no. 6574, p. abg8871, 2021.
4. G. Hickey, D. Heller, J. Monlong, J. A. Sibbesen, J. Sirén, J. Eizenga, E. T. Dawson, E. Garrison, A. M. Novak, and B. Paten, "Genotyping structural variants in pangenome graphs using the vg toolkit," *Genome Biology*, vol. 21, no. 1, p. 35, 02 2020.
5. T. Krannich, W. T. J. White, S. Niehus, G. Holley, B. V. Halldórsson, and B. Kehr, "Population-scale detection of non-reference sequence variants using colored de Bruijn graphs," *Bioinformatics*, vol. 38, no. 3, pp. 604–611, 11 2021.
6. R. M. Colquhoun, M. B. Hall, L. Lima, L. W. Roberts, K. M. Malone, M. Hunt, B. Letcher, J. Hawkey, S. George, L. Pankhurst, and Z. Iqbal, "Pandora: nucleotide-resolution bacterial pan-genomics with reference graphs," *Genome Biology*, vol. 22, no. 1, p. 267, 09 2021.
7. M. Karasikov, H. Mustafa, D. Danciu, M. Zimmermann, C. Barber, G. Rätsch, and A. Kahles, "Metagraph: Indexing and analysing nucleotide archives at petabase-scale," *bioRxiv*, 2020.
8. A. Morgulis, E. M. Gertz, A. A. Schäffer, and R. Agarwala, "A fast and symmetric dust implementation to mask low-complexity dna sequences," *Journal of Computational Biology*, vol. 13, no. 5, pp. 1028–1040, 2006.
9. P. W. Harrison, A. Ahamed, R. Aslam, B. T. F. Alako, J. Burgin, N. Buso, M. Courtot, J. Fan, D. Gupta, M. Haseeb, S. Holt, T. Ibrahim, E. Ivanov, S. Jayathilaka, V. Balavenkataraman Kadhirvelu, M. Kumar, R. Lopez, S. Kay, R. Leinonen, X. Liu, C. O'Cathail, A. Pakseresht, Y. Park, S. Pesant, N. Rahman, J. Rajan, A. Sokolov, S. Vijayaraja, Z. Waheed, A. Zyoud, T. Burdett, and G. Cochrane, "The European Nucleotide Archive in 2020," *Nucleic Acids Research*, vol. 49, no. D1, pp. D82–D85, 11 2020.
10. A. Bankevich, A. V. Bzikadze, M. Kolmogorov, D. Antipov, and P. A. Pevzner, "Multiplex de bruijn graphs enable genome assembly from long, high-fidelity reads," *Nature Biotechnology*, vol. 40, no. 7, pp. 1075–1081, 07 2022.
11. A. Rahman and P. Medvedev, "Assembler artifacts include misassembly because of unsafe unitigs and underassembly because of bidirected graphs," *Genome Research*, vol. 32, no. 9, pp. 1746–1753, 2022.
12. J. Sirén, "Indexing variation graphs," in *Proceedings of the Meeting on Algorithm Engineering and Experiments*, 2017, pp. 13–27.
13. A. Rhie, S. A. McCarthy, O. Fedrigo, J. Damas, G. Formenti, S. Koren, M. Uliano-Silva, W. Chow, A. Fungtammasan, J. Kim, C. Lee, B. J. Ko, M. Chaisson, G. L. Gedman, L. J. Cantin, F. Thibaud-Nissen, L. Haggerty, I. Bista, M. Smith, B. Haase, J. Mountcastle, S. Winkler, S. Paez, J. Howard, S. C. Vernes, T. M. Lama, F. Grutzner, W. C. Warren, C. N. Balakrishnan, D. Burt, J. M. George, M. T. Biegler, D. Iorns, A. Digby, D. Eason, B. Robertson, T. Edwards, M. Wilkinson, G. Turner, A. Meyer, A. F. Kautt, P. Franchini, H. W. Detrich, H. Svardal, M. Wagner, G. J. P. Naylor, M. Pippel, M. Malinsky, M. Mooney, M. Simbirsky, B. T. Hannigan, T. Pesout, M. Houck, A. Misuraca, S. B. Kingan, R. Hall, Z. Kronenberg, I. Sović, C. Dunn, Z. Ning, A. Hastie, J. Lee, S. Selvaraj, R. E. Green, N. H. Putnam, I. Gut, J. Ghurye, E. Garrison, Y. Sims, J. Collins, S. Pelan, J. Torrance, A. Tracey, J. Wood, R. E. Dagnew, D. Guan, S. E. London, D. F. Clayton, C. V. Mello, S. R. Friedrich, P. V. Lovell, E. Osipova, F. O. Al-Ajli, S. Secomandi, H. Kim, C. Theofanopoulou, M. Hiller, Y. Zhou, R. S. Harris, K. D. Makova, P. Medvedev, J. Hoffman, P. Masterson, K. Clark, F. Martin, K. Howe, P. Flicek, B. P. Walenz, W. Kwak, H. Clawson, M. Diekhans, L. Nassar, B. Paten, R. H. S. Kraus, A. J. Crawford, M. T. P. Gilbert, G. Zhang, B. Venkatesh, R. W. Murphy, K.-P. Koepfli, B. Shapiro, W. E. Johnson, F. Di Palma, T. Marques-Bonet, E. C. Teeling, T. Warnow, J. M. Graves, O. A. Ryder, D. Haussler, S. J. O'Brien, J. Korlach, H. A. Lewin, K. Howe, E. W. Myers, R. Durbin, A. M. Phillippy, and E. D. Jarvis, "Towards complete and error-free genome assemblies of all vertebrate species," *Nature*, vol. 592, no. 7856, pp. 737–746, 04 2021.
14. B. Solomon and C. Kingsford, "Improved search of large transcriptomic sequencing databases using split sequence bloom trees," *Journal of Computational Biology*, vol. 25, no. 7, pp. 755–765, 2018.
15. M. Karasikov, H. Mustafa, G. Rätsch, and A. Kahles, "Lossless indexing with counting de bruijn graphs," *Genome Research*, vol. 32, no. 9, pp. 1754–1764, 2022.
16. P. Pandey, F. Almodaresi, M. A. Bender, M. Ferdman, R. Johnson, and R. Patro, "Mantis: A fast, small, and exact large-scale sequence-search index," *Cell Systems*, vol. 7, no. 2, pp. 201–207.e4, 08 2018.
17. Z. Iqbal, M. Caccamo, I. Turner, P. Flicek, and G. McVean, "De novo assembly and genotyping of variants using colored de bruijn graphs," *Nature Genetics*, vol. 44, no. 2, pp. 226–232, 02 2012.

18. G. Holley and P. Melsted, "Bifrost: highly parallel construction and indexing of colored and compacted de bruijn graphs," *Genome Biology*, vol. 21, no. 1, p. 249, 09 2020.
19. C. Marchet, Z. Iqbal, D. Gautheret, M. Salson, and R. Chikhi, "REINDEER: efficient indexing of k-mer presence and abundance in sequencing datasets," *Bioinformatics*, vol. 36, no. Supplement_1, pp. i177–i185, 07 2020.
20. F. Almodaresi, H. Sarkar, A. Srivastava, and R. Patro, "A space and time-efficient index for the compacted colored de Bruijn graph," *Bioinformatics*, vol. 34, no. 13, pp. i169–i177, 06 2018.
21. E. Garrison, J. Sirén, A. M. Novak, G. Hickey, J. M. Eizenga, E. T. Dawson, W. Jones, S. Garg, C. Markello, M. F. Lin, B. Paten, and R. Durbin, "Variation graph toolkit improves read mapping by representing genetic variation in the reference," *Nature Biotechnology*, vol. 36, no. 9, pp. 875–879, 10 2018.
22. I. Turner, K. V. Garimella, Z. Iqbal, and G. McVean, "Integrating long-range connectivity information into de Bruijn graphs," *Bioinformatics*, vol. 34, no. 15, pp. 2556–2565, 03 2018.
23. J. Fan, J. Khan, N. P. Singh, G. E. Pibiri, and R. Patro, "Fulgor: a fast and compact k-mer index for large-scale matching and color queries," *Algorithms for Molecular Biology*, vol. 19, no. 1, p. 3, 01 2024.
24. T. Schulz, R. Wittler, S. Rahmann, F. Hach, and J. Stoye, "Detecting high-scoring local alignments in pangenome graphs," *Bioinformatics*, vol. 37, no. 16, pp. 2266–2274, 02 2021.
25. P. Ivanov, B. Bichsel, H. Mustafa, A. Kahles, G. Rätsch, and M. Vechev, "Astarix: Fast and optimal sequence-to-graph alignment," in *Research in Computational Molecular Biology*, 2020, pp. 104–119.
26. P. Ivanov, B. Bichsel, and M. Vechev, "Fast and optimal sequence-to-graph alignment guided by seeds," in *Research in Computational Molecular Biology*, 2022, pp. 306–325.
27. C. Lee, C. Grasso, and M. F. Sharlow, "Multiple sequence alignment using partial order graphs," *Bioinformatics*, vol. 18, no. 3, pp. 452–464, 03 2002.
28. M. Rautiainen and T. Marschall, "Graphaligner: rapid and versatile sequence-to-graph alignment," *Genome Biology*, vol. 21, no. 1, p. 253, 09 2020.
29. T. Dvorkina, D. Antipov, A. Korobeynikov, and S. Nurk, "Spaligner: alignment of long diverged molecular sequences to assembly graphs," *BMC Bioinformatics*, vol. 21, no. 12, p. 306, 07 2020.
30. H. Li, X. Feng, and C. Chu, "The design and construction of reference pangenome graphs with minigraph," *Genome Biology*, vol. 21, no. 1, p. 265, 10 2020.
31. J. Ma, M. Cáceres, L. Salmela, V. Mäkinen, and A. I. Tomescu, "Chaining for accurate alignment of erroneous long reads to acyclic variation graphs," *Bioinformatics*, vol. 39, no. 8, p. btad460, 07 2023.
32. G. Chandra and C. Jain, "Gap-sensitive colinear chaining algorithms for acyclic pangenome graphs," *Journal of Computational Biology*, vol. 30, no. 11, pp. 1182–1197, 2023.
33. J. Rajput, G. Chandra, and C. Jain, "Co-Linear Chaining on Pangenome Graphs," in *23rd International Workshop on Algorithms in Bioinformatics*, ser. Leibniz International Proceedings in Informatics, vol. 273, 2023, pp. 12:1–12:18.
34. A. Joudaki, A. Meterez, H. Mustafa, R. Groot Koerkamp, A. Kahles, and G. Rätsch, "Aligning distant sequences to graphs using long seed sketches," *Genome Research*, 2023.
35. D. Danko, D. Bezdan, E. E. Afshin, S. Ahsanuddin, C. Bhattacharya, D. J. Butler, K. R. Chng, D. Donnellan, J. Hecht, K. Jackson, K. Kuchin, M. Karasikov, A. Lyons, L. Mak, D. Meleshko, H. Mustafa, B. Mutai, R. Y. Neches, A. Ng, O. Nikolayeva, T. Nikolayeva, E. Png, K. A. Ryon, J. L. Sanchez, H. Shaaban, M. A. Sierra, D. Thomas, B. Young, O. O. Abudayyeh, J. Alicea, M. Bhattacharyya, R. Blekhman, E. Castro-Nallar, A. M. Cañas, A. D. Chatziefthimiou, R. W. Crawford, F. De Filippis, Y. Deng, C. Desnues, E. Dias-Neto, M. Dybwad, E. Elhaik, D. Ercolini, A. Frolova, D. Gankin, J. S. Gootenberg, A. B. Graf, D. C. Green, I. Hajirasouliha, J. J. Hastings, M. Hernandez, G. Iraola, S. Jang, A. Kahles, F. J. Kelly, K. Knights, N. C. Kyrpides, P. P. Łabaj, P. K. Lee, M. H. Leung, P. O. Ljungdahl, G. Mason-Buck, K. McGrath, C. Meydan, E. F. Mongodin, M. O. Moraes, N. Nagarajan, M. Nieto-Caballero, H. Noushmehr, M. Oliveira, S. Ossowski, O. O. Osuolale, O. Özcan, D. Paez-Espino, N. Rascovan, H. Richard, G. Rätsch, L. M. Schriml, T. Semmler, O. U. Sezerman, L. Shi, T. Shi, R. Siam, L. H. Song, H. Suzuki, D. S. Court, S. W. Tighe, X. Tong, K. I. Udekwu, J. A. Ugalde, B. Valentine, D. I. Vassilev, E. M. Vayndorf, T. P. Velavan, J. Wu, M. M. Zambrano, J. Zhu, S. Zhu, C. E. Mason, N. Abdullah, M. Abraao, A. hamlat Adel, M. Afaq, F. S. Al-Quaddoomi, I. Alam, G. E. Albuquerque, A. Alexiev, K. Ali, L. E. Alvarado-Arnez, S. Aly, J. Amachee, M. G. Amorim, M. Ampadu, M. A.-F. Amran, N. An, W. Andrew, H. Andrianjakarivony, M. Angelov, V. Antelo, C. Aquino, Álvaro Aranguren, L. F. Araujo, H. F. Vasquez Arevalo, J. Arevalo, C. Arnan, L. E. Alvarado Arnez, F. Arredondo, M. Arthur, F. Asenjo, T. S. Aung, J. Auvinet, N. Aventin, S. Ayaz, S. Baburyan, A.-M. Bakere, K. Bakhl, T. F. Bartelli, E. Batdelger, F. Baudon, K. Becher, C. Bello, M. Benchouaia, H. Benisty, A.-S. Benoiston, J. Benson, D. Benítez, J. Bernardes, D. Bertrand, S. Beurmann, T. Bitard-Feildel, L. Bittner, C. Black, G. Blanc, B. Blyther, T. Bode, J. Boeri, B. Boldgiv, K. Bolzli, A. Bordigoni, C. Borrelli, S. Bouchard, J.-P. Bouly, A. Boyd, G. P. Branco, A. Breschi, B. Brindefalk, C. Brion, A. Briones, P. Buczansla, C. M. Burke, A. Burrell, A. Butova, I. Buttar, J. Bynoe, S. Bönigk, K. O. Bøifot, H. Caballero, X. W. Cai, D. Calderon, A. Cantillo, M. Carbajo, A. Carbone, A. Cardenas, K. Carrillo, L. Casalot, S. Castro, A. V. Castro, A. Castro, A. V. B. Castro, S. Cawthorne, J. Cedillo, S. Chaker, J. Chalangal, A. Chan, A. I. Chasapi, S. Chatziefthimiou,

S. R. Chaudhuri, A. K. Chavan, F. Chavez, G. Chem, X. Chen, M. Chen, J.-W. Chen, A. Chernomoretz, A. Chettouh, D. Cheung, D. Chicas, S. Chiu, H. Choudhry, C. Chrispin, K. Ciaramella, E. Cifuentes, J. Cohen, D. A. Coil, S. Collin, C. Conger, R. Conte, F. Corsi, C. N. Cossio, A. F. Costa, D. Cuebas, B. D'Alessandro, K. E. Dahlhausen, A. E. Darling, P. Das, L. B. Davenport, L. David, N. R. Davidson, G. Dayama, S. Delmas, C. K. Deng, C. Dequeker, A. Desert, M. Devi, F. S. Dezem, C. N. Dias, T. R. Donahoe, S. Dorado, L. Dorsey, V. Dotsenko, S. Du, A. Dutan, N. Eady, J. A. Eisen, M. Elaskandrany, L. Epping, J. P. Escalera-Antezana, C. L. Ettinger, I. Faiz, L. Fan, N. Farhat, E. Faure, F. Fauzi, C. Feigin, S. Felice, L. P. Ferreira, G. Figueroa, A. Fleiss, D. Flores, J. L. Velasco Flores, M. A. Fonseca, J. Foox, J. C. Forero, A. Francis, K. French, P. Fresia, J. Friedman, J. J. Fuentes, J. Galipon, M. Garcia, L. Garcia, C. García, A. Geiger, S. M. Gerner, S. L. Ghose, D. P. Giang, M. Giménez, D. Giovannelli, D. Githae, S. Gkotzis, L. Godoy, S. Goldman, G. H. Gonnet, J. Gonzalez, A. Gonzalez, C. Gonzalez-Poblete, A. Gray, T. Gregory, C. Greselle, S. Guasco, J. Guerra, N. Gurianova, W. Haehr, S. Halary, F. Hartkopf, J. J. Hastings, A. Hawkins-Zafarnia, N. H. Hazrin-Chong, E. Helfrich, E. Hell, T. Henry, S. Hernandez, P. L. Hernandez, D. Hess-Homeier, L. E. Hittle, N. X. Hoan, A. Holik, C. Homma, I. Hoxie, M. Huber, E. Humphries, S. Hyland, A. Hässig, R. Häusler, N. Hüsser, R. A. Petit, B. Iderzorig, M. Igarashi, S. B. Iqbal, S. Ishikawa, S. Ishizuka, S. Islam, R. Islam, K. Ito, S. Ito, T. Ito, T. Ivankovic, T. Iwashiro, S. Jackson, J. Jacobs, M. James, M. Jaubert, M.-L. Jerier, E. Jiminez, A. Jinfessa, Y. De Jong, H. W. Joo, G. Jospin, T. Kajita, A. S. Ahmad Kassim, N. Kato, A. Kaur, I. Kaur, F. de Souza Gomes Kehdy, V. S. Khadka, S. Khan, M. Khavari, M. Ki, G. Kim, H. J. Kim, S. Kim, R. J. King, K. Knights, G. KoLoMonaco, E. Koag, N. Kobko-Litskevitch, M. Korshevniuk, M. Kozhar, J. Krebs, N. Kubota, A. Kuklin, S. S. Kumar, R. Kwong, L. Kwong, I. Lafontaine, J. Lago, T. Y. Lai, E. Laine, M. Laiola, O. Lakhneko, I. Lamba, G. de Lamotte, R. Lannes, E. De Lazzari, M. Leahy, H. Lee, Y. Lee, L. Lee, V. Lemaire, E. Leong, M. H. Leung, D. Lewandowska, C. Li, W. Liang, M. Lin, P. Lisboa, A. Litskevitch, E. M. Liu, T. Liu, M. A. Livia, Y. H. Lo, S. Losim, M. Loubens, J. Lu, O. Lykhenko, S. Lysakova, S. Mahmoud, S. A. Majid, N. Makogon, D. Maldonado, K. Mallari, T. M. Malta, M. Mamun, D. Manoir, G. Marchandon, N. Marciniak, S. Marinovic, B. Marques, N. Mathews, Y. Matsuzaki, V. Matthys, M. May, E. McComb, A. Meagher, A. Melamed, W. Menary, K. N. Mendez, A. Mendez, I. M. Mendy, I. Meng, A. Menon, M. Menor, R. Meoded, N. Merino, C. Meydan, K. Miah, M. Mignotte, T. Miketic, W. Miranda, A. Mitsios, R. Miura, K. Miyake, M. D. Moccia, N. Mohan, M. Mohsin, K. Moitra, M. Moldes, L. Molina, J. Molinet, O.-E. Molomjamts, E. Moniruzzaman, S. Moon, I. de Oliveira Moraes, M. Moreno, M. S. Mosella, J. W. Moser, C. Mozsary, A. L. Muehlbauer, O. Muner, M. Munia, N. Munim, M. Muscat, T. Mustac, C. Muñoz, F. Nadalin, A. Naeem, D. Nagy-Szakal, M. Nakagawa, A. Narce, M. Nasu, I. G. Navarrete, H. Naveed, B. Nazario, N. R. Nedunuri, T. Neff, A. Nesimi, W. C. Ng, S. Ng, G. Nguyen, E. Ngwa, A. Nicolas, P. Nicolas, A. Nika, H. Noorzi, A. Nosrati, H. Noushmehr, D. N. Nunes, K. O'Brien, N. B. O'Hara, G. Oken, R. A. Olawoyin, J. Q. Oliete, K. Olmeda, T. Oluwadare, I. A. Oluwadare, N. Ordioni, J. Orpilla, J. Orrego, M. Ortega, P. Osma, I. O. Osuolale, O. M. Osuolale, M. Ota, F. Oteri, Y. Oto, R. Ounit, C. A. Ouzounis, S. Pakrashi, R. Paras, C. Pardo-Este, Y.-J. Park, P. Pastuszek, S. Patel, J. Pathmanathan, A. Patrignani, M. Perez, A. Peros, S. Persaud, A. Peters, A. Phillips, L. Pineda, M. P. Pizzi, A. Plaku, A. Plaku, B. Pompa-Hogan, M. G. Portilla, L. Posada, M. Priestman, B. Prithiviraj, S. Priya, P. Pugdeethosal, C. E. Pugh, B. Pulatov, A. Pupiec, K. Pyrshev, T. Qing, S. Rahiel, S. Rahmatulloev, K. Rajendran, A. Ramcharan, A. Ramirez-Rojas, S. Rana, P. Ratnanandan, T. D. Read, H. Rehrauer, R. Richer, A. Rivera, M. Rivera, A. Robertiello, C. Robinson, P. Rodríguez, N. A. Rojas, P. Roldán, A. Rosario, S. Roth, M. Ruiz, S. E. Boja Ruiz, K. Russell, M. Rybak, T. S. Sabedot, M. Sabina, I. Saito, Y. Saito, G. A. Malca Salas, C. Salazar, K. M. San, J. Sanchez, K. Sanchir, R. Sankar, P. T. de Souza Santos, Z. Saravi, K. Sasaki, Y. Sato, M. Sato, S. Sato, R. Sato, K. Sato, N. Sayara, S. Schaaf, O. Schacher, A.-L. M. Schinke, R. Schlapbach, C. Schori, J. R. Schriml, F. Segato, F. Sepulveda, M. S. Serpa, P. F. De Sessions, J. C. Severyn, H. Shaaban, M. Shakil, S. Shalaby, A. Shari, H. Shim, H. Shirahata, Y. Shiwa, R. Siam, O. Da Silva, J. M. Silva, G. Simon, S. K. Singh, K. Sluzek, R. Smith, E. So, N. Andreu Somavilla, Y. Sonohara, N. Rufino de Sousa, C. Souza, J. Sperry, N. Sprinsky, S. G. Stark, A. La Storia, K. Suganuma, H. Suliman, J. Sullivan, A. A. M. Supie, C. Suzuki, S. Takagi, F. Takahara, N. Takahashi, K. Takahashi, T. Takeda, I. K. Takenaka, S. Tanaka, A. Tang, Y. Man Tang, E. Tarcitano, A. Tassinari, M. Taye, A. Terrero, E. Thambiraja, A. Thiébaut, S. Thomas, A. M. Thomas, Y. Togashi, T. Togashi, A. Tomaselli, M. Tomita, I. Tomita, X. Tong, O. Toth, N. C. Toussaint, J. M. Tran, C. Truong, S. I. Tsonev, K. Tsuda, T. Tsurumaki, M. Tuz, Y. Tymoshenko, C. Urgiles, M. Usui, S. Vacant, B. Valentine, L. E. Vann, F. Velter, V. Ventorino, P. Vera-Wolf, R. Vicedomini, M. A. Suarez-Villamil, S. Vincent, R. Vivancos-Koopman, A. Wan, C. Wang, T. Warashina, A. Watanabe, S. Weekes, J. Werner, D. Westfall, L. H. Wieler, M. Williams, S. A. Wolf, B. Wong, Y. L. Wong, T. Wong, R. Wright, T. Wunderlin, R. Yamanaka, J. Yang, H. Yano, G. C. Yeh, O. Yemets, T. Yeskova, S. Yoshikawa, L. Zafar, Y. Zhang, S. Zhang, A. Zhang, Y. Zheng, and S. Zubenko, "A global metagenomic map of urban microbiomes and antimicrobial resistance," *Cell*, vol. 184, no. 13, pp. 3376–3393.e17, 2021.

36. M. Heydari, G. Miclotte, Y. Van de Peer, and J. Fostier, "Browniealigner: accurate alignment of illumina sequencing data to de bruijn graphs," *BMC Bioinformatics*, vol. 19, no. 1, p. 311, 09 2018.

37. A. Limasset, B. Cazaux, E. Rivals, and P. Peterlongo, "Read mapping on de bruijn graphs," *BMC Bioinformatics*, vol. 17, no. 1, p. 237, 06 2016.
38. N. Luhmann, G. Holley, and M. Achtman, "Blastfrost: fast querying of 100,000s of bacterial genomes in bifrost graphs," *Genome Biology*, vol. 22, no. 1, p. 30, 01 2021.
39. Y. Rosen, J. Eizenga, and B. Paten, "Modelling haplotypes with respect to reference cohort variation graphs," *Bioinformatics*, vol. 33, no. 14, pp. i118–i123, 07 2017.
40. J. Avila, P. Bonizzoni, S. Ciccolella, G. D. Vedova, L. Denti, D. Monti, Y. Pirola, and F. Porto, "Recgraph: adding recombinations to sequence-to-graph alignments," *bioRxiv*, 2022.
41. G. Chandra and C. Jain, "Haplotype-aware sequence-to-graph alignment," *bioRxiv*, 2023.
42. C. Jain, D. Gibney, and S. V. Thankachan, "Algorithms for colinear chaining with overlaps and gap costs," *Journal of Computational Biology*, vol. 29, no. 11, pp. 1237–1251, 2022.
43. D. Gibney, S. V. Thankachan, and S. Aluru, "On the hardness of sequence alignment on de bruijn graphs," *Journal of Computational Biology*, vol. 29, no. 12, pp. 1377–1396, 2022.
44. R. Chikhi and G. Rizk, "Space-efficient and exact de bruijn graph representation based on a bloom filter," *Algorithms for Molecular Biology*, vol. 8, no. 1, p. 22, 09 2013.
45. J. Shaw and Y. W. Yu, "Proving sequence aligners can guarantee accuracy in almost o(m log n) time through an average-case analysis of the seed-chain-extend heuristic," *Genome Research*, vol. 33, no. 7, pp. 1175–1187, 2023.
46. V. Mäkinen, A. I. Tomescu, A. Kuosmanen, T. Paavilainen, T. Gagie, and R. Chikhi, "Sparse dynamic programming on dags with small width," *ACM Trans. Algorithms*, vol. 15, no. 2, 02 2019.
47. X. Chang, J. Eizenga, A. M. Novak, J. Sirén, and B. Paten, "Distance indexing and seed clustering in sequence graphs," *Bioinformatics*, vol. 36, no. Supplement_1, pp. i146–i153, 07 2020.
48. M. C. Frith, "How sequence alignment scores correspond to probability models," *Bioinformatics*, vol. 36, no. 2, pp. 408–415, 07 2019.
49. S. Heule, M. Nunkesser, and A. Hall, "Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm," in *Proceedings of the 16th International Conference on Extending Database Technology*, 2013, p. 683–692.
50. C. Boucher, A. Bowe, T. Gagie, S. J. Puglisi, and K. Sadakane, "Variable-order de bruijn graphs," in *2015 Data Compression Conference*, 2015, pp. 383–392.
51. H. Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinformatics*, vol. 34, no. 18, pp. 3094–3100, 05 2018.
52. J. M. Eizenga and B. Paten, "Improving the time and space complexity of the wfa algorithm and generalizing its scoring," *bioRxiv*, 2022.
53. E. W. Sayers, E. E. Bolton, J. Brister, K. Canese, J. Chan, D. Comeau, C. Farrell, M. Feldgarden, A. M. Fine, K. Funk, E. Hatcher, S. Kannan, C. Kelly, S. Kim, W. Klimke, M. Landrum, S. Lathrop, Z. Lu, T. Madden, A. Malheiro, A. Marchler-Bauer, T. Murphy, L. Phan, S. Pujar, S. Rangwala, V. Schneider, T. Tse, J. Wang, J. Ye, B. Trawick, K. Pruitt, and S. Sherry, "Database resources of the National Center for Biotechnology Information in 2023," *Nucleic Acids Research*, vol. 51, no. D1, pp. D29–D38, 11 2022.
54. W. Huang, L. Li, J. R. Myers, and G. T. Marth, "ART: a next-generation sequencing read simulator," *Bioinformatics*, vol. 28, no. 4, pp. 593–594, 12 2011.
55. Y. Ono, M. Hamada, and K. Asai, "PBSIM3: a simulator for all types of PacBio and ONT long reads," *NAR Genomics and Bioinformatics*, vol. 4, no. 4, p. lqac092, 12 2022.
56. W. Wei and D. Koslicki, "WGSUniFrac: Applying UniFrac Metric to Whole Genome Shotgun Data," in *22nd International Workshop on Algorithms in Bioinformatics*, ser. Leibniz International Proceedings in Informatics, vol. 242, 2022, pp. 15:1–15:22.