Bioinformatics, 2025, **41(12)**, btaf652
https://doi.org/10.1093/bioinformatics/btaf652
Advance Access Publication Date: 4 December 2025
Original Paper

OXFORD

Sequence analysis

Columba: fast approximate pattern matching with optimized search schemes

Luca Renders[1,†], Lore Depuydt[1,†], Travis Gagie[2], Jan Fostier[1,*]

[1]Department of Information Technology—IDLab, Ghent University—imec, Ghent 9052, Belgium
[2]Faculty of Computer Science, Dalhousie University, Halifax, NS B3H 4R2, Canada

*Corresponding author. Jan Fostier, Department of Information Technology—IDLab, Ghent University—imec, Technologiepark 126, Ghent 9052, Belgium. E-mail: jan.fostier@ugent.be.

[†]= equal contribution.

Associate Editor: Peter Robinson

Abstract

Motivation: Aligning sequencing reads to reference genomes is a fundamental task in bioinformatics. Aligners can be classified as lossy or lossless: lossy aligners prioritize speed by reporting only one or a few high-scoring alignments, whereas lossless aligners output all optimal alignments, ensuring completeness and sensitivity.

Results: This paper introduces Columba, a high-performance lossless aligner tailored for Illumina sequencing data. Columba processes single or paired-end reads in FASTQ format and outputs alignments in SAM format. By utilizing advanced search schemes and bit-parallel alignment techniques, Columba achieves exceptional speed. Columba is available in two variants. The first, based on the bidirectional FM-index, prioritizes speed. The second, Columba RLC, uses run-length compression using a bidirectional move structure, significantly reducing memory usage for large, repetitive datasets like pan-genomes. Benchmarks on the human genome, as well as bacterial and human pan-genome datasets, demonstrate that Columba is much faster than existing lossless aligners and even competitive with lossy tools. We integrated Columba into the OptiType HLA genotyping pipeline, where it substantially reduced computational time while maintaining accuracy. These results position Columba as a versatile, state-of-the-art tool for high-sensitivity genomic analyses.

Availability and implementation: The source code of Columba is available at https://github.com/biointec/columba under AGPL license. Scripts to reproduce the benchmarks and analyses are available at https://doi.org/10.5281/zenodo.15849246.

1 Introduction

Sequence alignment is a fundamental task in bioinformatics, with applications in variant calling, genome assembly, transcriptome analysis, functional genomics, and more. Alignment tools can be broadly categorized as lossy or lossless (see Table 1).

Lossy aligners, such as Bowtie2 (Langmead and Salzberg 2012) and BWA-MEM(2) (Li 2013, Vasimuddin *et al.* 2019), use a seed-and-extend approach to quickly identify good alignments. However, since seeds—often maximal exact matches, or MEMs—can contain sequencing errors or variants, lossy tools do not guarantee optimal alignments. They typically report only one or a few good alignments per read, rather than providing a comprehensive overview of all possible optimal alignment positions. This limitation can reduce their utility in applications such as bacterial species identification, metagenomics read classification, or Human Leukocyte Antigen (HLA) typing, where reads are aligned against a pan-genome to determine which genome(s) or haplotype(s) best explain the read dataset. In these cases, the pan-genome often contains closely related reference sequences, and an individual read might align with many of them.

In contrast, lossless alignment (also known as *exact* alignment or approximate pattern matching) involves reporting *all* optimal alignments. Optimality is typically defined by the edit (Levenshtein) distance—i.e. the total number of substitutions and/or indels in the alignment. Notable examples of lossless aligners include RazerS3 (Weese *et al.* 2012), Yara (Siragusa 2015), Bwolo (Vroland *et al.* 2016), and DREAM-Yara (Dadi *et al.* 2018), which have their utility in specific applications. For instance, RazerS3 is used in the OptiType tool for HLA typing. Other tools, such as BWA-aln (Li and Durbin 2009), Bowtie (Langmead *et al.* 2009), and GEM (Marco-Sola *et al.* 2012), offer partial support for lossless read alignment, but they often do not report all optimal alignments (see Results). Similarly, Ropebwt3 (Li 2024), one of the first alignment tools designed for large-scale pan-genome references, reports a single alignment position along with the number of alternative alignment locations, but not their actual positions.

While lossless aligners offer stronger guarantees of optimality, lossy tools are often preferred in practical bioinformatics applications due to their speed. In this paper, we aim to bridge this performance gap by introducing Columba, a high-performance, lossless alignment tool, designed to operate under the edit distance metric. Given single or paired-end reads in FASTQ format and a reference (pan-)genome, Columba reports all optimal alignments in SAM format (Li *et al.* 2009). It supports two modes:

Received: 7 April 2025; **Revised:** 25 November 2025; **Accepted:** 26 November 2025
© The Author(s) 2025. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

Downloaded from https://academic.oup.com/bioinformatics/article/41/12/btaf652/8370107 by IMEC user on 01 April 2026

Table 1. Comparison of aligners.[a]

Aligner	Lossy/lossless	Max. error rate/n° errors	RLC	Index structure
RazerS3	Lossless	50%	No	No index
Yara	Lossless	10%	No	Unidirect.
Bwolo	Lossless	Unlimited	No	Unidirect.
BWA-aln	Config.	Unlimited	No	Unidirect.
Bowtie	Config.	3 errors	No	Unidirect.
GEM	Config.	Unlimited	No	Unidirect.
Bowtie2	Lossy	Unlimited	No	Unidirect.
BWA-MEM(2)	Lossy	Unlimited	No	Bidirect.
Ropebwt3	Lossy	Unlimited	Yes	Bidirect.
Columba	Lossless	13 errors	No	Bidirect.
Columba RLC	Lossless	13 errors	Yes	Bidirect.

[a] RLC: run-length compression; config.: tool is lossy, but can be configured to perform lossless alignment.

1) *All mode*: Reports every occurrence of a read within a user-specified edit distance of at most k errors.
2) *All-best mode*: Reports every optimal occurrence of a read [i.e. the one(s) with the smallest edit distance], using a strata-based approach within a user-specified edit distance of at most k errors.

Columba can identify alignments with up to $k = 13$ errors per read, making it particularly useful for Illumina sequencing data. However, it is less suited for spliced aligned of RNA-seq data, as the gap size may exceed the maximum threshold of $k = 13$ errors. For paired-end read input, Columba reports all optimal *proper pairs*. In this context, optimality is defined as the smallest edit distance across both reads. A proper pair is defined by a fragment size that does not deviate by more than six standard deviations from its expected value. The expected fragment size and standard deviation are inferred from the data.

Lossless alignment tools are particularly valuable for pan-genomes, where reads are aligned against a database of many related sequences. Columba can work with two indexes: the bidirectional FM-index and the recently proposed b-move structure (Depuydt *et al.* 2024). While the FM-index has a memory usage that scales linearly with the total sequence length of the pan-genome, the b-move structure leverages run-length compression (RLC) on the Burrows-Wheeler transform (BWT) to significantly reduce memory requirements. However, this reduction comes at a 30%–50% cost in performance. This principle of run-length compressing the BWT (Mäkinen and Navarro 2005) is also the foundation of several recent mapping tools, including MONI (Rossi *et al.* 2022), PHONI (Boucher *et al.* 2021), SPUMONI (Ahmed *et al.* 2021), SPUMONI 2 (Ahmed *et al.* 2023), and Movi (Zakeri *et al.* 2024). The key distinction between these tools and Columba is that they focus on MEM-finding, using (pseudo-)matching lengths and statistics, while Columba performs complete alignment.

We demonstrate that Columba (RLC) is significantly faster than existing lossless alignment tools for tasks such as alignment against the human reference genome as well as human and bacterial pan-genomes. Columba is written in C++ (2011 standard), supporting multi-threading. Columba's underlying algorithms have been previously presented at conferences. Version 2.0 of Columba (described in this paper) introduces several new features, including SAM output, support for run-length compressed indexes, multithreading, and support for all-best mode. Its source code is available at https://github.com/biointec/columba under AGPL license.

2 Methods

We briefly describe the key components of Columba: search schemes, the bidirectional index (either the FM-index or the run-length compressed move structure), and various performance enhancements. A more detailed discussion of the core ideas underlying Columba can be found in Section 1, available as supplementary data at *Bioinformatics* online.

2.1 Search schemes

Lossless approximate pattern matching (APM) aims to identify all approximate occurrences of a pattern P within a text T (one or more reference genomes) with up to k errors. In this paper, errors can be substitutions, insertions, or deletions, measured using the edit distance metric. By using a full-text index like the FM-index (Ferragina and Manzini 2000), a backtracking algorithm systematically explores all candidate occurrences character by character. However, the search tree induced by this method quickly becomes impractically large, even for moderate values of k.

Columba uses *search schemes* (Kucherov *et al.* 2016) and a bidirectional full-text index to significantly reduce the size of the search tree. The pattern P is divided into p potentially uneven-length parts P_i $(i = 1, \ldots, p)$. A search scheme $\mathcal{S}$ consists of multiple searches S, where each search is represented by the triplet $S = (\pi, L, U)$, where:

- π is a permutation of $\{1, 2, \ldots, p\}$ that specifies the order in which the parts of P are matched using the index, i.e. $P_{\pi[1]}, P_{\pi[2]}, \ldots, P_{\pi[p]}$. It must satisfy the *connectivity property*, ensuring that partial matches are extended contiguously, either to the left or to the right.
- L and U are arrays of size p, where $L[i]$ and $U[i]$ respectively denote the lower and upper bounds on the cumulative number of errors after part $P_{\pi[i]}$ has been processed. The values of L and U must increase monotonically and $L[i] \leq U[i]$, for all i.

The searches in a search scheme are designed to collectively cover all possible distributions of k or less errors across the parts of P. Therefore, each approximate occurrence of P is guaranteed to be identified by at least one search. The lower and upper bounds defined by the L and U arrays help reduce the search space and limit the redundant reporting of occurrences across different searches. Such redundant occurrences are filtered in a post-processing step.

Designing efficient search schemes is non-trivial, especially for higher values of k. The pigeonhole principle forms the

Downloaded from https://academic.oup.com/bioinformatics/article/41/12/btaf652/8370107 by IMEC user on 01 April 2026

basis for some of the simplest search schemes (Lam *et al.* 2009), where each search targets one part of P for exact matching, and allows up to k errors in the remaining parts. More efficient schemes, such as those proposed by Kucherov *et al.* (2016), Kianfar *et al.* (2017), and Pockrandt (2019), introduce tighter error bounds to make the overall process more efficient. In this work, we use the MinU search schemes recently proposed by us (Renders *et al.* 2024). The MinU schemes were designed using an Integer Linear Programming (ILP) approach for up to $k = 7$ errors and a greedy algorithm for $k = 8 \ldots 13$ errors. The MinU schemes were demonstrated to be very efficient for practical applications.

2.2 Bidirectional FM-index and move structure

The FM-index supports only backward search: given a partial match M, occurrences of cM (where c is a character) can be identified in $O(1)$ time (Ferragina and Manzini 2000). However, search schemes require *bidirectional* search functionality, as this allows for matching some middle part of P, followed by left and right extensions in an arbitrary order. The bidirectional FM-index, introduced by Lam *et al.* (2009), allows for extending a partial match M to either cM or Mc in $O(\sigma)$ time, with σ the size of the alphabet ($\sigma = 4$ for DNA sequences). Columba implements bidirectional character extensions in $O(1)$ time using a fast algorithm by Pockrandt *et al.* (2017).

For large pan-genomes, the bidirectional FM-index can have prohibitively large memory requirements. In such cases, the BWT, a key component of the FM-index, often contains long runs –consecutive occurrences of the same character– making it inherently compressible. To support lossless approximate pattern matching in run-length compressed space, Columba RLC replaces the bidirectional FM-index with the bidirectional *move structure* (b-move in short) (Depuydt *et al.* 2024). The r-index was the first index to support both counting and locating exact occurrences of patterns in $O(r)$ space, where r is the number of runs in the BWT (Gagie *et al.* 2018, 2020). The move structure, proposed by Nishimoto and Tabei (2021), forms an alternative to the r-index with significantly faster performance in practice. b-move extends the move structure to enable bidirectional character extensions in a conceptually similar manner to the bidirectional FM-index. Like the move structure, b-move supports fast, cache-efficient character extensions in run-length compressed space.

To construct the suffix array, Columba uses either `libsais` (https://github.com/IlyaGrebnov/libsais) or `libdivsufsort` (https://github.com/y-256/libdivsufsort), depending on the length of the text. By default, a $4\times$ subsampled suffix array is used. In the Columba RLC, there are two methods for index construction: an in-memory approach; which is fast but requires substantial RAM, and a prefix-free parsing method (Boucher *et al.* 2019), which is slower but significantly more memory-efficient.

2.3 Performance enhancements

Columba incorporates several performance enhancements. Recall that search schemes partition search patterns into p parts. Instead of using static partitioning (e.g. equally sized parts), we developed a *dynamic partitioning* method. This approach tailors the partition sizes for each pattern individually, ensuring a balanced computational workload across searches within the search scheme. The workload for each search is estimated by the number of exact matches of the part P_i that is matched first by that search. For example, if part P_i has a large number of exact matches in T, any search continuing from that part would likely generate an excessively large search tree to explore. By increasing the size of P_i, its number of exact matches decreases, thereby reducing its search tree and associated runtime. Dynamic partitioning of search patterns reduces runtime by roughly 25% (Renders *et al.* 2021).

To further optimize performance, we implemented a *dynamic selection* mechanism for search schemes. For a given number of errors k, multiple search schemes may be available, and their efficiency can vary depending on the specific search pattern. The dynamic selection mechanism uses a heuristic to identify the search scheme with the lowest estimated workload, again relying on the number of exact matches of each part P_i. This approach further reduces runtime by approximately 10% (Renders *et al.* 2024).

The use of the edit distance metric inherently involves some redundancy: a pattern P may be reported multiple times with slightly different start and/or end positions in T. For example, a leading or trailing gap can be exchanged for a substitution. This redundancy becomes particularly problematic in the context of search schemes, where parts are matched individually, and direction switches may occur. To address this issue and avoid redundant calculation, we applied a clustering technique to focus computations on representative alignment paths while maintaining the lossless nature of APM (Renders *et al.* 2021). Additionally, we implemented a bit-parallel approach, inspired by Myers (1998) and Hyyrö (2003), to compute all edit distance values in a row of a banded alignment matrix simultaneously.

Finally, while Columba primarily relies on a full-text index for APM, it dynamically switches to in-text verification once the number of candidate occurrences drops below a threshold (4 by default). In-text verification is performed in a bit-parallel, cache-efficient manner, avoiding the need for more costly character-by-character extensions using the index (Renders *et al.* 2022).

3 Material and experimental setup

The benchmarks on a single human genome, the bacterial pan-genome and for HLA-typing were obtained using a single thread of a 64-core Intel Xeon E5-2698 v3 CPU, operating at a base clock speed of 2.30 GHz, with 256 GiB of available RAM. For the benchmark on the human pan-genome, a single thread of a 64-core AMD EPYC 7773X (Milan-X @ 2.2 GHz) processor was used, with 940 GiB of RAM available. All experiments were run 10 times; the standard deviation was always lower than 1.7%. The lossless tools use the edit distance.

3.1 Comparing the output of different aligners

The aligners report results in SAM format (Li *et al.* 2009), except for Ropebwt3, which reports in PAF format (https://github.com/lh3/miniasm/blob/master/PAF.md). Ropebwt3 only reports 1 alignment per read, along with the number of alignments. BWA-aln outputs alignments in the intermediate SAI format and requires BWA-SAMSE to convert its output to SAM format.

Comparing the output of various aligners is challenging, even when using the SAM format. We examine all equally

Downloaded from https://academic.oup.com/bioinformatics/article/41/12/btaf652/8370107 by IMEC user on 01 April 2026

best alignments per read, however not all aligners restrict their output to the best alignment; e.g. BWA and Bowtie sometimes report additional alignments with a worse score. Additionally, supplementary co-optimal alignments are handled differently: Columba lists them as separate records, while BWA uses the XA tag. Yara also uses the XA tag but in a different format, though it can output separate records for supplementary alignments without CIGAR strings, which complicates SAMtools compatibility.

The interpretation of maximum error rate (or minimum identity score) can slightly vary across the different tools, resulting in tiny variations in the output. Finally, slight variations in alignment positions under an edit distance model necessitate some manual inspection, as aligners may produce subtly different results for the same region.

4 Results

4.1 Alignment of Illumina reads to the human reference genome

We aligned 1 million Illumina NovaSeq 6000 reads (151 bp), randomly sampled from a whole genome sequencing dataset (accession no. SRR9091899), to the hg38 human reference genome (Schneider *et al.* 2017). In this initial benchmark, the reads were treated as unpaired. We used both lossless and lossy alignment tools. For lossless alignment, we included Columba, Yara, BWA-aln with the -N flag, and Bowtie with the -a flag; these flags enable lossless search. For lossy alignment, we used BWA-MEM, BWA-MEM2, Bowtie2, and Bowtie2 in "very sensitive" (VS) mode. The lossless alignment tools were configured to report all optimal alignments in *all-best* mode for maximum error rates ranging from 0% to 8%. The commands used for each tool are listed in Section 2, available as supplementary data at *Bioinformatics* online. Table 2 shows the runtime, peak memory usage, and the percentage of reads that were aligned by each tool.

In terms of performance, Columba significantly outperforms existing lossless aligners. It is 1.4×, 1.9×, 8.1×, 25.8×, and 37.8× faster than Yara, its closest competitor, at error rates of 0%, 2%, 4%, 6%, and 8%, respectively. This indicates that optimized search schemes offer superior performance for lossless approximate pattern matching, particularly at higher error rates. The runtimes also indicate that BWA-aln was not primarily designed for lossless alignment, consistent with its manual, which notes that the -N flag may lead to significantly lower performance. Bowtie with the -a flag is slower than Yara and Columba at error rates 0% and 2%, and it does not support lossless alignment beyond 3 errors, which corresponds to a 2% error rate for this dataset.

When comparing Columba to lossy tools, we observe that Columba is faster than the lossy alternatives—except for BWA-MEM2—up to a maximum error rate of 6%, which is a reasonable threshold for Illumina data. Even at an error rate of 8%, Columba's runtime remains competitive with Bowtie2 in VS mode. Columba requires more RAM than most other tools because it uses a bidirectional index and a relatively densely sampled suffix array. However, at 11.06 GiB, it still fits comfortably on a modern laptop.

In principle, all lossless alignment tools should produce optimal –and therefore identical– results. Columba and Yara produce near-identical results, with minor differences arising from a different handling of non-ACGT characters and treatment of overlapping alignments. For example, one tool may list overlapping alignments as separate alignments, while the other may flag one as redundant. In contrast, despite being configured as lossless tools, BWA-aln and Bowtie do not always report the optimal alignment(s), as evidenced by slightly lower alignment percentages (Table 2). Additionally, when multiple optimal alignments exists, often only one or a few are reported (see Section 3, available as supplementary data at *Bioinformatics* online).

On the other hand, the lossy alignment tools align a higher fraction of reads. For example, at a 6% error rate threshold, Columba mapped 96.8% of the reads, while BWA-MEM(2) found alignments for nearly all (99.8%) reads. Upon inspection, these additional alignments contain error rates $>6\%$, often due to gapped alignment or read clipping. In Section 4, available as supplementary data at *Bioinformatics* online, we provide a detailed comparison of the SAM output between Columba and both Yara and BWA-MEM(2).

Additionally, in Section 5, available as supplementary data at *Bioinformatics* online, we repeated the same benchmark, this time aligning 1 million reads pairs (2 million reads in total). This task is significantly more challenging because optimal proper pairs are not necessarily obtained by combining optimal alignments of the individual reads. Nevertheless, Columba outperforms all lossless aligners by a large margin, and is competitive with lossy aligners until an error rate of 6%.

Table 2. Runtime and peak memory usage of lossless alignment tools (Columba, Yara, BWA-aln, and Bowtie) at varying maximum error rate thresholds, and lossy alignment tools [BWA-MEM and Bowtie2 in normal and very sensitive (VS) modes], for aligning 1 000 000 Illumina reads (length 151 bp) from a larger WGS dataset to the human reference genome using a single thread.[a]

Tool	Peak Memory	Maximum error rate 0%	2%	4%	6%	8%
Lossless alignment tools						
Columba (this paper)	11.06 GiB	**24 s** (66.6%)	**45 s** (91.1%)	**1 min 27 s** (95.2%)	**4 min 33 s** (96.8%)	**21 min 09 s** (97.7%)
Yara	4.94 GiB	33 s (66.6%)	1 min 24 s (91.1%)	11 min 42 s (95.2%)	1 h 57 min 10 s (96.8%)	11 h 11 min 42 s (97.7%)
BWA-aln	4.51 GiB	1 min 2 s (66.6%)	41 min 16 s (91.1%)	3 h 38 min 9s (95.0%)	8 h 41 min 28 s (96.5%)	13 h 55 min 54 s (97.2%)
Bowtie	2.29 GiB	37s (66.6%)	2 min 28 s (89.9%)	Not supported	Not supported	Not supported
Lossy alignment tools				**All Error Rates**		
BWA-MEM	5.19 GiB			6 min 46 s (99.8%)		
BWA-MEM2	16.5 GiB			3 min 31 s (99.8%)		
Bowtie2	3.22 GiB			7 min 53 s (98.8%)		
Bowtie2 (VS)	3.23 GiB			17 min 01 s (99.0%)		

[a] The alignment percentage of reads is noted alongside each runtime.
Boldface values show the lowest runtime.

Downloaded from https://academic.oup.com/bioinformatics/article/41/12/btaf652/8370107 by IMEC user on 01 April 2026

4.2 HLA typing from NGS data using OptiType

Lossless alignment tools are crucial for certain bioinformatics applications, including Human Leukocyte Antigen (HLA) typing from next-generation sequencing (NGS) data. Accurate, high-resolution HLA typing is essential for understanding immune system function and has significant applications in organ transplantation compatibility, autoimmune disease research, and cancer immunotherapy.

In a recent survey paper on HLA typing tools (Claeys *et al.* 2023), OptiType (Szolek *et al.* 2014) emerged as one of the most accurate tools for predicting HLA genotypes from NGS data. It focuses on Major Histocompatibility Complex (MHC) Class I alleles, including HLA-A, HLA-B, and HLA-C. For each read, OptiType requires a complete list of all possible candidate alleles from which the read could have originated. To achieve this, OptiType uses the lossless aligner RazerS3 (Weese *et al.* 2009, 2012) to map reads to a comprehensive collection of HLA allele sequences. Next, leveraging an integer linear programming (ILP) approach, OptiType identifies the most likely set of HLA alleles that explain the sequencing data.

While OptiType delivers precise genotyping, it was also found to be computationally intensive (Claeys *et al.* 2023), partly due to the alignment step. To address this, we use Columba as a drop-in replacement for RazerS3, which required only minimal changes to OptiType's script (see Section 6, available as supplementary data at *Bioinformatics* online). Using the dataset from the survey paper, we evaluated OptiType by aligning slices of 1012 CRAM files of whole-exome sequencing (WES) data from the 1000 Genomes on GRCh38 project (Zheng-Bradley *et al.* 2017), to an HLA-gene reference database (https://raw.githubusercontent.com/FRED-2/OptiType/refs/heads/master/data/hla_reference_dna.fasta) (Szolek *et al.* 2014).

Figure 1 shows the runtime distribution of the alignment phase for both Columba and RazerS3 across the 1012 samples. In both cases, 16 CPU cores were used. Columba achieved a median runtime of 4.83 s compared to 57.96 s for RazerS3, yielding a 12× speedup. Consequently, using Columba reduced the overall runtime of the OptiType pipeline (including the ILP step) by 44% (median value). As Columba and RazerS3 produce near-identical output, OptiType's accuracy remains unaffected. We further evaluated OptiType on 462 RNA-seq samples (see Section 6, available as supplementary data at *Bioinformatics* online). In that case, the median runtime of the alignment phase decreased from 4.82 min (RazerS3) to 19.52 s (Columba), demonstrating significant efficiency gains for RNA-seq data as well.

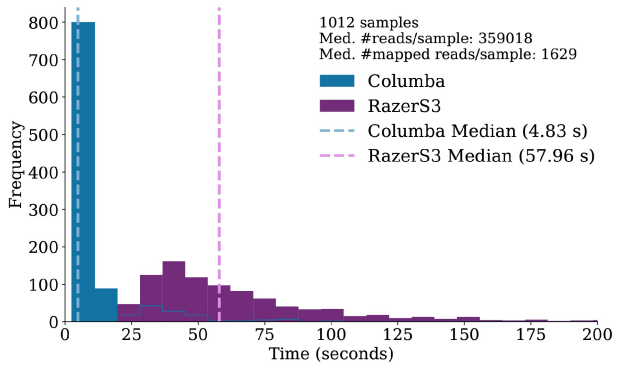

Figure 1. Runtime distribution of OptiType's alignment phase using Columba and RazerS3 across 1012 samples. The *x*-axis is capped at 200 s. RazerS3's distribution continues beyond this value.

4.3 Alignment to bacterial pan-genomes

We aligned 1 million *Escherichia coli* reads (150 bp, Illumina NovaSeq 6000), randomly sampled from a larger sequencing dataset (SRR19479103), to a pan-genome comprising 6115 bacterial strains: 2594 *Escherichia coli*, 1183 *Salmonella enterica*, 944 *Staphylococcus aureus*, 637 *Pseudomonas aeruginosa*, 254 *Bacillus subtilis*, 255 *Listeria monocytogenes*, 203 *Enterococcus faecalis*, and 45 *Limosilactobacillus fermentum* genomes (∼28 billion bp in total, accessions numbers in Section 7, available as supplementary data at *Bioinformatics* online) to evaluate runtime and peak memory usage. The lossless alignment tools were again run in *all-best* mode, a configuration suited for pan-genome analysis as it provides a comprehensive overview of all possible genomes a read could have originated from. This is particularly relevant for tasks such as bacterial species or strain identification and taxonomic sequence classification for metagenomics. The comparison includes Columba RLC, the run-length compressed variant of Columba that uses the bidirectional move structure instead of the bidirectional FM-index, as well as Ropebwt3, a mapper specifically designed for pan-genome references. Columba was run with a 16× sampled suffix array to reduce its memory use. We excluded BWA-MEM2 from this comparison due to its larger index size, which exceeded the 256 GiB of RAM available on our machine.

Table 3 shows that Columba is the fastest tool among the evaluated lossless aligners, outperforming Yara by up to 10.1×. Columba and Columba RLC generate identical alignments as they share the same alignment methodology, differing only in index structure. The outputs of Yara and Columba (RLC) are again nearly identical, with minor discrepancies caused by small implementation differences (see Section 4, available as supplementary data at *Bioinformatics* online for details). As before, BWA-aln, despite the use of the -N flag, does not always report an optimal alignment for each read. This is reflected in the slightly lower alignment percentage. Additionally, BWA-aln does not comprehensively report all optimal alignments, excluding nearly 312 million co-optimal alignments, over 99.7% of those reported by Columba and Yara. In the context of this pan-genome, many reads have hundreds of optimal alignments, and BWA-aln typically reports only one or a few (see Section 3, available as supplementary data at *Bioinformatics* online). This vastly reduced output volume explains BWA-aln's lower runtime on this dataset at a 0% error rate.

Compared to Columba, Columba RLC reduces memory usage by 5.6× to only 14.32 GiB, making it possible to analyze large bacterial pan-genomes on a modern laptop. Note that Columba RLC improves the memory complexity of Columba from $\mathcal{O}(n)$ (with n being the total size of the reference (pan-)genome) to $\mathcal{O}(r)$ (with r being the number of runs in the BWT). Therefore, the relative memory gain will only increase for larger (and more repetitive) pan-genomes. However, this memory efficiency comes at the cost of a 1.4× to 1.6× runtime increase compared to Columba. Even so, Columba RLC remains faster than other lossless and lossy alignment tools.

Ropebwt3 achieves superior compression compared to Columba RLC but at the expense of longer runtimes. Columba RLC is nearly two times faster for an error rate of

Downloaded from https://academic.oup.com/bioinformatics/article/41/12/btaf652/8370107 by IMEC user on 01 April 2026

Table 3. Runtime and peak memory usage of lossless alignment tools (Columba, Columba RLC, Yara, and BWA-aln) at varying maximum error rate thresholds, and lossy alignment tools (BWA-MEM and Ropebwt3), for aligning 1 million *E. coli* Illumina reads (length 150 bp) to a pan-genome comprising 6115 bacterial strains using a single thread.[a]

Tool	Peak	Maximum error rate				
	Memory	0%	2%	4%	6%	8%
Lossless alignment tools						
Columba	86.00 GiB	10 min 14 s (75.3%)	**12 min 9 s** (94.6%)	**12 min 26 s** (96.6%)	**13 min 46 s** (97.4%)	**18 min 12 s** (97.8%)
Columba RLC	14.32 GiB	15 min 18 s (75.3%)	18 min 40 s (94.6%)	19 min 32 s (96.6%)	21 min 11 s (97.4%)	25 min 27 s (97.8%)
Yara	48.27 GiB	11 min 55 s (75.3%)	19 min 10 s (94.6%)	26 min 24 s (96.6%)	38 min 54 s (97.4%)	3 h 4 min 43 s (97.8%)
BWA-aln	39.35 GiB	**3 min 06 s** (75.3%)	37 min 39 s (94.5%)	1 h 11 min 41 s (96.5%)	2 h 30 min 35 s (97.2%)	4 h 35 min 1 s (97.6%)
Lossy tools				**All Error Rates**		
BWA-MEM	46.92 GiB			30 min 13 s (99.2%)		
Ropebwt3	2.67 GiB			42 min 9 s (99.2%)		

[a] The alignment percentage of reads is noted alongside each runtime.
Boldface values show the lowest runtime.

6% and 40% faster for an error rate of 8%. Importantly, Ropebwt3 reports only a single alignment per read along with the count of equally optimal hits (without reporting their positions). For 95.9% of the reads, the alignment produced by Ropebwt3 matched one of the alignments reported by Columba (for an error rate of 8%) within at most 50 bp. For 1.8% of the reads, Ropebwt3 reported alignments beyond an error rate of 8% (corresponding to 12 errors), which Columba did not produce. However, for 9526 reads (0.95%), Columba reported more alignments (3 800 953 alignments not reported by Ropebwt3), while Ropebwt3 produced a higher number of alignments for 4527 reads (resulting in 279 042 additional alignments). As Columba and Yara are lossless and produce near-identical output, we infer that the additional alignments reported by Ropebwt3 correspond to overlapping occurrences in the reference genome (<20 bp apart), which are considered spurious by Columba and Yara.

Section 8, available as supplementary data at *Bioinformatics* online contains the runtimes for Columba (RLC), Yara, Bowtie2, BWA-MEM, and Ropebwt3 when utilizing multiple threads. The results demonstrate that all tools benefit from multi-threading. However, at higher thread counts, the performance of the lossless tools starts to be constrained by the speed at which output can be written to disk.

4.4 Alignment to human pan-genome reference

We aligned the same 1 million Illumina reads from the first benchmark, but now against pan-genomes containing up to 64 human haplotypes, sourced from the 1-year freeze of the Human Pangenome Reference Consortium, which provides a comprehensive representation of genetic diversity (Liao *et al.* 2023). Indexes were constructed for Columba, Columba RLC, and Ropebwt3. We could evaluate Columba for only up to 16 human haplotypes, as constructing the index for 32 haplotypes exceeded the available RAM on the machine (940 GiB). Columba and Columba RLC were executed in *all-best* mode with a maximum error rate of 6%, while Ropebwt3 was used with its default settings.

Figure 2 shows that Columba's memory usage scales linearly with the number of haplotypes, as shown by the trendline. In contrast, Columba RLC and Ropebwt3 demonstrate more favorable memory requirements. Both leverage run-length compression on the BWT and share the same $O(r)$ memory complexity, but Columba RLC's bidirectional move

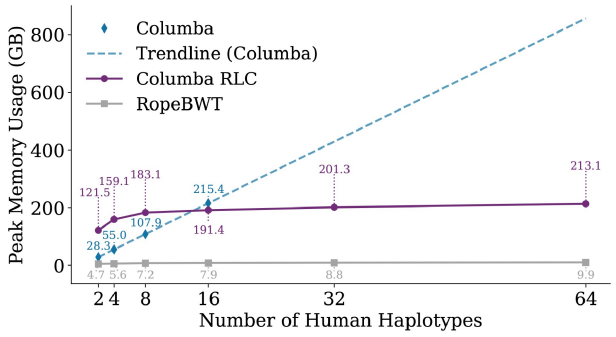

Figure 2. Comparison of peak memory usage among Columba, Columba RLC, and Ropebwt3 for aligning 1 000 000 reads to a pan-genome, with a varying number of human haplotypes.

structure implementation introduces a larger constant prefactor. As a result, Ropebwt3 has a better memory footprint than Columba RLC in practice. In comparison to Columba, Columba RLC exhibits lower memory usage only when processing more than 16 haplotypes.

Figure 3 illustrates the runtime for aligning 1 million reads to the human pan-genomes. Columba RLC has a performance penalty compared to Columba but remains significantly faster than Ropebwt3. The increase in runtime for Columba (RLC) with larger pan-genome sizes can be attributed to the greater volume of output that needs to be produced, i.e. a larger number of optimal alignments. In contrast, Ropebwt3 reports only a single alignment and the number of optimal alignments. When comparing the output of Columba (RLC) and Ropebtw3, a similar analysis can be made as for the bacterial pan-genome.

5 Discussion and conclusion

We introduce Columba, a fast and feature-rich lossless alignment tool. It accepts FASTQ or FASTA files as input and produces SAM records as output, leveraging multi-threading to enhance performance.

Unlike popular lossy aligners such as BWA-MEM(2) and Bowtie, Columba ensures both optimality and completeness in its output. Optimality is evaluated using edit distance, a standard metric for assessing differences between reads and reference sequences. However, some applications may benefit from alternative scoring schemes, like affine gap penalties, which can be more biologically relevant. In *all* mode,

Downloaded from https://academic.oup.com/bioinformatics/article/41/12/btaf652/8370107 by IMEC user on 01 April 2026

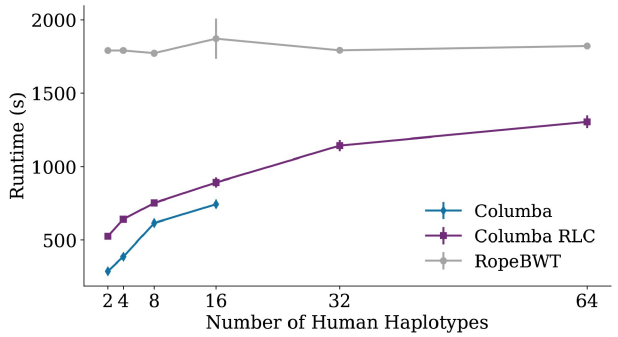

Figure 3. Comparison of runtime among Columba, Columba RLC, and Ropebwt3 for aligning 1 000 000 reads to a pan-genome, with a varying number of human haplotypes.

completeness means reporting all occurrences in the reference sequences with up to a predefined number of errors (k), with support for up to $k = 13$ errors. In *all-best* mode, completeness means reporting all optimal occurrences –those with the lowest edit distance to the query read. Columba supports paired-end read mapping, reporting only alignment positions that conform to the expected fragment size. If no such proper pair(s) can be found, the reads are treated as unpaired.

Our evaluation shows that Columba is significantly faster than other lossless alignment tools like RazerS3 and Yara, especially when aligning reads with more errors. This performance gain is due to optimized search schemes and an efficient bit-parallel implementation. However, because Columba's search schemes require a bidirectional index, its memory requirements are generally double those of tools using a unidirectional index.

In many cases, Columba is even faster than widely used lossy alignment tools while guaranteeing optimality and completeness of its output. However, Columba is limited to aligning reads with up to 13 errors, making it less suitable for spliced alignments. A potential future improvement could involve a two-pass solution: using search schemes in the first pass to ensure optimality, followed by a seed-and-extend approach for those reads that do not align initially. This hybrid approach could combine the strengths of both worlds.

Columba is available in two variants. The first, based on the bidirectional FM-index, prioritizes speed but requires more memory. The second, Columba RLC, leverages the bidirectional move structure, a run-length compressed index, making it ideally suited for repetitive pan-genomes. This approach significantly reduces memory usage when dealing with multiple similar genomes, albeit with a slight trade-off in speed. Other tools, such as Ropebwt3, offer a different time-space tradeoff: Ropebwt3 can better compress pan-genomes but is generally slower. Additionally, Ropebwt3 reports only one alignment and the alignment count, not all alignments. While Columba excels in providing complete and optimal alignments, the potentially large volume of output can be challenging for downstream tools to analyze. Further algorithmic development is likely needed to efficiently handle these large volumes of data in downstream analysis.

Columba is a robust and versatile alignment tool that can be used for a wide range of applications. Its successful integration into the OptiType pipeline for HLA genotyping highlights its practical utility and effectiveness. We believe Columba and Columba RLC can also be relevant for applications such as metagenomics read classification, bacterial species or strain identification, and pan-genome graph alignment and visualization (Depuydt *et al.* 2023). Additionally, Columba could be useful to identify seeds within long read sequencing data. Finally, Columba (RLC) can be used to benchmark the output of lossy alignment tools.

Author contributions

Luca Renders (Conceptualization [equal], Funding acquisition [supporting], Investigation [equal], Methodology [equal], Software [lead], Validation [lead], Visualization [lead], Writing—original draft [lead]), Lore Depuydt (Conceptualization [equal], Funding acquisition [lead], Investigation [equal], Methodology [equal], Software [supporting], Validation [supporting], Visualization [supporting], Writing—original draft [supporting]), Travis Gagie (Conceptualization [supporting], Supervision [supporting], Writing—review & editing [supporting]), and Jan Fostier (Conceptualization [lead], Funding acquisition [lead], Supervision [lead], Writing—review & editing [lead])

Supplementary data

Supplementary data is available at *Bioinformatics* online.

Conflict of interest: None declared.

Funding

This work was supported by the Research Foundation—Flanders (FWO) through a PhD Fellowship SB [1SE7822N to L.R.] and a PhD Fellowship FR [1117322N to L.D.].

Data availability

Columba is available at https://github.com/biointec/columba The code used to produce the benchmarks and analyses described in this manuscript is available at https://doi.org/10.5281/zenodo.15849246.

References

Ahmed O, Rossi M, Kovaka S *et al.* Pan-genomic matching statistics for targeted nanopore sequencing. *iScience* 2021;**24**:102696.

Ahmed OY, Rossi M, Gagie T *et al.* SPUMONI 2: improved classification using a pangenome index of minimizer digests. *Genome Biol* 2023;**24**:122.

Boucher C, Gagie T, Kuhnle A *et al.* Prefix-free parsing for building big BWTs. *Algorithms Mol Biol* 2019;**14**:13.

Boucher C, Gagie T, Tomohiro I *et al.* PHONI: Streamed matching statistics with multi-genome references. In: *IEEE Data Compression Conference (DCC)*, Snowbird, UT, USA. New York, USA: IEEE, 2021, 193–202.

Claeys A, Merseburger P, Staut J *et al.* Benchmark of tools for in silico prediction of MHC class I and class II genotypes from NGS data. *BMC Genomics* 2023;**24**:247.

Dadi TH, Siragusa E, Piro VC *et al.* DREAM-Yara: an exact read mapper for very large databases with short update time. *Bioinformatics* 2018;**34**:i766–i772.

Depuydt L, Renders L, Abeel T *et al.* Pan-genome de Bruijn graph using the bidirectional FM-index. *BMC Bioinformatics* 2023;**24**:400.

Depuydt L, Renders L, Van de Vyver S *et al.* b-move: Faster Bidirectional Character Extensions in a Run-Length Compressed Index. In: *24th International Workshop on Algorithms in Bioinformatics (WABI 2024)*, Leibniz International Proceedings in

Downloaded from https://academic.oup.com/bioinformatics/article/41/12/btaf652/8370107 by IMEC user on 01 April 2026

Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Vol. 312. 2024, **10**:1–10:18.
Ferragina P, Manzini G. Opportunistic data structures with applications. In: *Proceedings 41st IEEE Annual Symposium on Foundations of Computer Science, Redondo Beach, CA, USA*. New York, USA: IEEE, 2000, 390–8.
Gagie T, Navarro G, Prezza N. Optimal-time text indexing in BWT-runs bounded space. In: *29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*. New York, USA: ACM, 2018, 1459–77.
Gagie T, Navarro G, Prezza N *et al.* Fully functional suffix trees and optimal text searching in BWT-runs bounded space. *J ACM* 2020; **67**:1–54.
Hyyrö H. A bit-vector algorithm for computing Levenshtein and Damerau edit distances. *Nord J Comput* 2003;**10**:29–39.
Kianfar K, Pockrandt C, Torkamandi B *et al.* FAMOUS: fast approximate string matching using optimum search schemes. *CoRR* 2017.
Kucherov G, Salikhov K, Tsur D *et al.* Approximate string matching using a bidirectional index. *Theor Comput Sci* 2016;**638**:145–58.
Lam TW, Li R, Tam A *et al.* High throughput short read alignment via bi-directional BWT. In: *IEEE International Conference on Bioinformatics and Biomedicine*, Washington, DC, USA. New York, USA: IEEE, 2009, 31–6.
Langmead B, Salzberg SL. Fast gapped-read alignment with bowtie 2. *Nat Methods* 2012;**9**:357–9.
Langmead B, Trapnell C, Pop M *et al.* Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Gen Biol* 2009;**10**.
Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv, arXiv:1303.3997, 2013, preprint: not peer reviewed.
Li H. BWT construction and search at the terabase scale. *Bioinformatics* 2024;**40**:1–9.
Li H, Durbin R. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* 2009;**25**:1754–60.
Li H, Handsaker B, Wysoker A *et al.*; 1000 Genome Project Data Processing Subgroup. The sequence alignment/map format and SAMtools. *Bioinformatics* 2009;**25**:2078–9.
Liao W-W, Asri M, Ebler J *et al.* A draft human pangenome reference. *Nature* 2023;**617**:312–24.
Marco-Sola S, Navarro G. Succinct suffix arrays based on run-length encoding. In: *Combinatorial Pattern Matching*, Lecture Notes in Computer Science, vol 3537. Berlin, Heidelberg: Springer, 2005, 45–56.
Marco-Sola S, Sammeth M, Guigó R *et al.* The GEM mapper: fast, accurate and versatile alignment by filtration. *Nat Methods* 2012; **9**:1185–8.
Myers G. A fast bit-vector algorithm for approximate string matching based on dynamic programming. In: *Combinatorial Pattern Matching*, Lecture Notes in Computer Science, vol 1448. Berlin, Heidelberg: Springer, 1998, 1–13.
Nishimoto T, Tabei Y. Optimal-time queries on BWT-runs compressed indexes. In: *48th International Colloquium on Automata, Languages, and Programming, (ICALP 2021)*, Glasgow, Scotland (Virtual Conference), Vol. 198 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 101:1–101:15.
Pockrandt C, Ehrhardt M, Reinert K. EPR-dictionaries: a practical and fast data structure for constant time searches in unidirectional and bidirectional FM indices. In: *Research in Computational Molecular Biology*, Hong Kong, Cham: Springer, 2017, 190–206
Pockrandt C. Approximate string matching: improving data structures and algorithms. Dissertation, Freie Universität Berlin, 2019.
Renders L, Marchal K, Fostier J *et al.* Dynamic partitioning of search patterns for approximate pattern matching using search schemes. *iScience* 2021;**24**:102687.
Renders L, Depuydt L, Fostier J. Approximate pattern matching using search schemes and in-text verification. In: *Bioinformatics and Biomedical Engineering (IWBBIO 2022), Maspalomas, Gran Canaria, Spain*. Berlin, Heidelberg: Springer-Verlag, 2022, 419–35.
Renders L, Depuydt L, Rahmann S *et al.* Lossless approximate pattern matching: automated design of efficient search schemes. *J Comput Biol* 2024;**31**:975–89.
Rossi M, Oliva M, Langmead B *et al.* MONI: a pangenomic index for finding maximal exact matches. *J Comput Biol* 2022;**29**:169–87.
Schneider VA, Graves-Lindsay T, Howe K *et al.* Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Res* 2017;**27**:849–64.
Siragusa E. Approximate string matching for high-throughput sequencing. PhD thesis, 2015.
Szolek A, Schubert B, Mohr C *et al.* OptiType: precision HLA typing from next-generation sequencing data. *Bioinformatics* 2014; **30**:3310–6.
Vasimuddin M, Misra S, Li H *et al.* Efficient architecture-aware acceleration of BWA-MEM for multicore systems. In: *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil*. New York, USA: IEEE, 2019, 314–24. https://doi.org/10.1109/IPDPS.2019.00041
Vroland C, Salson M, Bini S *et al.* Approximate search of short patterns with high error rates using the 010 lossless seeds. *J Discret Algorithms* 2016;**37**:3–16.
Weese D, Emde A-K, Rausch T *et al.* RazerS - Fast read mapping with sensitivity control. *Genome Res* 2009;**19**:1646–54.
Weese D, Holtgrewe M, Reinert K *et al.* RazerS 3: faster, fully sensitive read mapping. *Bioinformatics* 2012;**28**:2592–9.
Zakeri M, Brown NK, Ahmed OY *et al.* Movi: a fast and cache-efficient full-text pangenome index. *iScience* 2024;**27**:111464.
Zheng-Bradley X, Streeter I, Fairley S *et al.*; 1000 Genomes Project Consortium. Alignment of 1000 genomes project reads to reference assembly GRCh38. *Gigascience* 2017;**6**:1–8.

Downloaded from https://academic.oup.com/bioinformatics/article/41/12/btaf652/8370107 by IMEC user on 01 April 2026

© The Author(s) 2025. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.
Bioinformatics, 2025, 41, 1–8
https://doi.org/10.1093/bioinformatics/btaf652
Original Paper