Accelerating k-mer-based sequence filtering

Igor Martayan[1 [0009−0008−3007−6168]], Léa Vandamme[1 [0009−0006−7585−7035]], Bede Constantinides[3 [0000−0002−3480−3819]], Bastien Cazaux[1 [0000−0002−1761−4354]], Charles Paperman[2 [0000−0002−6658−5238]], and Antoine Limasset[1 [0000−0002−0669−4141]]

[1] Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[2] Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, France
[3] University of Birmingham, Birmingham, United Kingdom

Abstract. The exponential growth of global sequencing data repositories presents both analytical challenges and opportunities. While k-mer-based indexing has improved scalability over traditional alignment for identifying potentially relevant documents, pinpointing the exact sequences matching a query remains a hurdle, especially when considering the false positives and false negatives caused by sparse k-mer sampling or probabilistic data structures. Furthermore, searching for numerous k-mers with a large query or multiple distinct ones strains existing exact matching tools, whose performance scales poorly with an increasing number of patterns. Indexing entire vast datasets for infrequent or ad-hoc searches is often resource-prohibitive. This paper addresses the renewed need for efficient streaming techniques to recover specific relevant sequences directly, without exhaustive pre-indexing. Having discussed the relevance of streaming search methods in bioinformatics, we propose diverse use cases as benchmarks to encourage further research into tailored solutions for these challenges, and demonstrate how sketching techniques like minimizers, alongside SIMD acceleration, can enhance the performance of such streaming searches. These contributions highlight practical approaches and identify avenues for future theoretical and empirical investigation in managing and querying vast genomic datasets. Availability: https://github.com/Malfoy/K2Rmini.

Keywords: High throughput sequencing · Pattern matching · SIMD.

1 Introduction

The immense volume of available sequencing data presents both significant opportunities and challenges in bioinformatics. Assembled datasets have reached the Petabase scale, as seen in the 2 Petabytes of unitigs from the Logan project [4], while even curated databases like GenBank are nearing 6 Terabases [28]. Although holistic analyses of these large-scale datasets, such as those on viral [9], soil [20], gut [23], and environmental metagenomes [24], are promising, they remain computationally expensive. Identifying matches to a query sequence in large collections, a fundamental task exemplified by BLAST [2], has evolved. Traditional alignment tools were overwhelmed by data growth, leading to the use

of k-mer based methods [36]. To improve scalability, indexing structures allowing false positives were adopted, making shared k-mer fingerprints a necessary, though not sufficient, condition for a match [21,22,18]. Some methods further enhance scalability by indexing only a fraction of k-mers, relying on detecting substantial matches from partial signals [8,26,1].

Large-scale indexes can thus efficiently identify a small, relevant subset of documents (e.g., read sets, unitigs, contigs). However, this initial filtering isn't final. Due to false positives from probabilistic indexes, or sparse/non-collinear matches, query k-mers may not be truly present or usable for mapping. Therefore, recovering specific sequences within identified documents that genuinely share k-mers with the query is crucial. This refinement, moving from relevant documents to relevant sequences suitable for alignment, mitigates downstream scalability issues and can reduce data volume by orders of magnitude, such as identifying a few relevant sequences from billions.

Although sequence-level indexes have been proposed [17,34], indexing and storing entire dataset collection at this resolution remains expensive and challenging at scale. This creates a pressing need for efficient streaming techniques to recover relevant sequences directly from unindexed data. Efficient multi-pattern matching is a highly studied topic from both the theoretical [6] and practical point of view [11]. Practical implementation can be found in classical data processing with grep-like efficient tools such as Ripgrep [12] or Hyperscan [35]. Some tools perform pattern matching adapted to genomic data, such as Seqkit2 [31]. However, these solutions scale poorly as the number of query k-mers increases and fail to leverage the uniform length of the search patterns. This is a critical issue, as querying numerous k-mers is often necessary for large queries (contigs, long reads, whole genomes), searching for many distinct sequences (variant or gene collections), or optimizing batched queries. The recent BackToSequences [3] tool, used in the Logan project, addresses these issues by simply indexing query k-mers in a hash table to search for target documents.

This paper has three main objectives. First, to emphasize the renewed relevance and practical importance of streaming search methods for numerous patterns. Second, to present benchmark use cases with distinct challenges that may require different approaches, highlighting areas for further research. Third, to demonstrate how sketching techniques like minimizers and acceleration with SIMD instructions can be effectively applied to this problem, offering practical and theoretical contributions. Based on these ideas, we present a new tool, K2Rmini, available at `https://github.com/Malfoy/K2Rmini`.

2 Methods

Classical multi-pattern matching algorithms, which generalize the Knuth-Morris-Pratt approach, construct deterministic finite automata to achieve linear time complexity relative to the pattern and input data sizes. While these methods offer strong theoretical guarantees, they often exhibit poor cache locality. Furthermore, they address the more general problem of matching variable-length

patterns, whereas our work focuses exclusively on fixed-size patterns. This constraint creates an opportunity for hash-based techniques, which are better suited for large pattern sets and more amenable to hardware acceleration through vector instructions.

In practice, Fasta-like sequence searching presents unique characteristics that allow specific optimizations. First, the objective is not merely to identify any sequence containing a query k-mer. Instead, we seek sequences that share a significant proportion of k-mers with the query set [32,8]. This proportion can be defined as a percentage of the total query k-mers or as a percentage of the k-mers within the sequence itself. This criterion is biologically relevant, as an isolated k-mer match is often insufficient to establish functional or structural significance. Second, these searches are essentially "needle in a haystack" operations. The k-mers are typically large (e.g., 21 or 31 nucleotides), making their unrelated occurrence highly improbable. Consequently, a specific k-mer is expected to appear only a limited number of times, a dozen or up to a hundred occurrences in a typical read dataset, depending on the sequencing depth. A vast fraction of the sequences in a dataset are therefore irrelevant to the query. Hence, the primary goal for accelerating such searches is to develop methods that can rapidly identify and discard these irrelevant sequences, efficiently skipping over large portions of data that do not meet the defined k-mer coverage threshold. Additionally, non-informative sections of sequence data, such as headers or quality scores, should be bypassed to further optimize processing.

One of the state-of-the-art solutions, BackToSequences [3], uses an efficient hash table containing all k-mers of interest and skips irrelevant fields. However, its main limitation is its necessity to perform a hash table lookup for every k-mer in the data stream. This process becomes computationally expensive, especially when the hash table is large and does not fit into low-level caches. To address this bottleneck, we propose K2Rmini (`https://github.com/Malfoy/K2Rmini`), a method that integrates minimizer-based filtering and Single Instruction, Multiple Data (SIMD) operations.

Minimizers [29,27] are a widely used k-mer sampling method that selects at least one out of every w consecutive m-mers. The set of sampled minimizers is typically much smaller (by a factor $2/(w+1)$ called the *density*) [25] than the whole sequence while preserving a lot of information, making it a key component of large-scale sequence analysis. Recent contributions have demonstrated that minimizer computation can be significantly accelerated using SIMD instructions [15], enabling extremely high throughput. An important property of minimizers is that they are *locally consistent*: two sequences sharing contexts of w m-mers will sample the same minimizers from these contexts. Thus, comparing minimizers can be used as a proxy to identify similar sequences [19,8,10].

The core idea of K2Rmini is to leverage this property to avoid exhaustive k-mer scanning. For each sequence, we first determine if it shares a threshold number of minimizers with the query set before proceeding to the full k-mer comparison. This approach offers a twofold benefit. First, it drastically reduces the number of hash table lookups, as checks are performed only for each min-

imizer rather than for every k-mer. Second, the resulting minimizer hash table is substantially smaller, leading to improved cache coherence. While this initial filtering step might seem like an overhead since the actual k-mer counting must still be performed to identify true hits, it effectively bypasses the costly k-mer analysis for the vast majority of sequences that lack significant similarity to the query. The overhead is minimal due to fast, SIMD-accelerated minimizer computation and the significantly reduced number of items to check.

Although the number of shared minimizers can be used to estimate the number of shared k-mers, our objective is to be exhaustive and not miss any true matches. By definition, every minimizer is shared by at most w consecutive k-mers. Thus, a necessary condition to share n k-mers is to have $\lceil n/w \rceil$ matching minimizers in the sequence. Hence, before counting the shared k-mers of a sequence, we first verify that the number of shared minimizers meets this threshold.

3 Results

3.1 State-of-the-art analysis

To evaluate state-of-the-art tools, we benchmarked a selection of specialized and general-purpose programs on an i9-13950HX $\times$ 32 with 64 GB of RAM running Ubuntu 24.04.2. The tools optimized for FASTA files included Seqkit [30,31], the grep-like implementations fqgrep [16] and grepq [7], and the hash-table-based BackToSequences [3]. For comparison, we also included widely-used general-purpose tools such as grep and its more efficient replacement, ripgrep [12]. Finally, we benchmarked Hyperscan [35], a state-of-the-art library for multiple pattern matching originally designed for deep packet inspection workloads. As shown in Figure 1, we benchmarked these tools on a real *E. coli* sequencing dataset with approximately 100X coverage while varying the number of indexed k-mers. We observed that the performance of most tools decreased when indexing a larger number of k-mers, with the exception of those based on a hash table. We also noted that while K2Rmini is almost an order of magnitude faster than BackToSequences, this improvement diminishes as the number of k-mers from the *E. coli* genome grows in the case of positive queries as more sequences pass the minimizer filter, reducing the performance gap with BackToSequences.

3.2 Influence of parameters

To explore the effect of different parameters on our minimizer filtering, we report the performance of BackToSequences and K2Rmini in Figure 2. We assessed the impact of a larger k-mer size on both methods, with a fixed minimizer size for K2Rmini. We observe that a larger k-mer size tends to increase the CPU time for BackToSequences while decreasing it for K2Rmini. This is expected, since a larger k-mer for a fixed minimizer size means a larger selection window, hence fewer selected minimizers, accelerating the filtering. In our multicore assessment, K2Rmini demonstrates high efficiency, achieving close to optimal performance

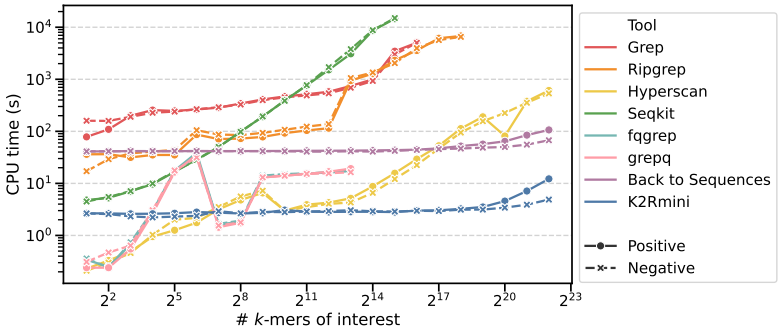

Fig. 1. Performance of state-of-the-art tools when parsing an *E. coli* sequencing dataset (100X coverage randomly extracted from SRR11434954) with an increasing number of queried k-mers. Positive k-mers are extracted from the genome while negative k-mers are random sequences.

with only four threads, reaching speeds approaching the theoretical lower bound determined by file decompression. In contrast, while BackToSequences can benefit from more cores, its performance remains several times slower.

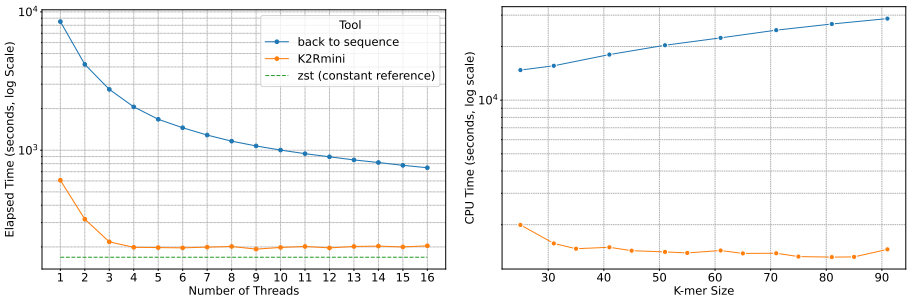

Fig. 2. Performance of BackToSequences and K2Rmini on a Human HiFi sequencing dataset with 50X coverage (SRX7897685-8). **Left:** K2Rmini and BackToSequences wall-clock time according to the number of cores used. For reference, a dry decompression (without producing an output file) of the file is performed. **Right:** Performance of BackToSequences and K2Rmini with varying k-mer sizes, where the minimizer size for K2Rmini is fixed to 21.

3.3 Real data

To provide a comprehensive comparison, we benchmarked the performance of BackToSequences and K2Rmini across a diverse set of datasets, summarized in Table 1. Our evaluation covers raw sequencing data with distinct characteristics, including a human Nanopore dataset with ultra-long reads, a HiFi

dataset with 10kb-20kb reads, and a standard Illumina 250bp dataset. We also assessed performance on assembled sequence on unitigs and contigs from Logan [4], the complete human T2T reference genome, for its very long contiguous sequences, and the entire RefSeq collection of bacterial genomes, for its large number of individual genomes. The results demonstrate K2Rmini is consistently and significantly faster than BackToSequences across all tested datasets and more memory-efficient in almost all cases.

	BackToSequences			**K2Rmini**		
Dataset	**CPU**	**Elapsed**	**Memory**	**CPU**	**Elapsed**	**Memory**
ONT (SRR23365080)	26,987	15m01	1,303	**1,465**	**6m20**	**103**
HiFi (SRX7897685-8)	13,363	7m26	235	**732**	**2m55**	**31**
Illumina (ERR3239454)	5,481	5m22	177	**465**	**1m36**	**34**
T2T (GCF_009914755.1)	152	2m33	**3,185**	**8**	**3**	4,112
Contigs (SRR7853572)	121	20	148	**16**	**5**	**32**
Unitigs (SRR7853572)	342	1m51	148	**94**	**38**	**36**
Refseq Bacteria	17,873	10m11	9,224	**862**	**5m5**	**1,777**

Table 1. Performance comparison of BackToSequences and K2Rmini tools. Times are expressed in minutes and seconds and memory in megabytes. Best results are highlighted.

4 Discussion

This work has demonstrated that a search strategy based on SIMD-accelerated minimizer filtering is fast and resource-efficient across a broad spectrum of applications. Similar tools are being developed for screening sequence repositories for antimicrobial resistance mutations, emerging pathogen surveillance, or sequencing contaminant filtration [5]. Future work could explore alternative search strategies, particularly for high-similarity contexts where dropping the search early would be possible. Developing high-performance methods to compute denser minimizer schemes also remains an open question [14,13,25], but a successful approach would reduce the memory footprint and the number of minimizers requiring verification even further. Although a hash table is an effective choice, other structures might yield greater performance. Small, cache-line-aligned bloom filters, for example, could prove to be more efficient. Conversely, for smaller sets of minimizers, a simple vector might outperform a hash table due to its structural simplicity. As the throughput of the core algorithm increases, performance bottlenecks shift to other components of the pipeline. The primary bottleneck is currently parsing the sequences to filter, which could be further optimized with vectorization. Moreover, reading input from FASTA/Q files is becoming a significant limitation, highlighting the need for more performant alternatives [33].

Acknowledgments. This work is funded by the French National Research Agency AGATE ANR-21-CE45-0012 and SxC ANR-24-CE25-2874-01. BC is supported by Wellcome Trust Award 313694/Z/24/Z.

Disclosure of Interests. No competing interests.

References

1. C. Agret, B. Cazaux, and A. Limasset. Toward optimal fingerprint indexing for large scale genomics. *bioRxiv*, pages 2021–11, 2021.
2. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.
3. A. Baire, P. Marijon, F. Andreace, and P. Peterlongo. Back to sequences: Find the origin of k-mers. *Journal of Open Source Software*, 9(101):7066, 2024.
4. R. Chikhi, B. Raffestin, A. Korobeynikov, R. Edgar, and A. Babaian. Logan: planetary-scale genome assembly surveys life's diversity. *bioRxiv*, pages 2024–07, 2024.
5. B. Constantinides. deacon. https://github.com/bede/deacon, 2025.
6. M. Crochemore, A. Czumaj, L. Gąsieniec, T. Lecroq, W. Plandowski, and W. Rytter. Fast practical multi-pattern matching. *Information Processing Letters*, 71(3):107–113, 1999.
7. N. D. Crosbie. grepq: A rust application that quickly filters fastq files by matching sequences to a set of regular expressions. *bioRxiv*, pages 2025–01, 2025.
8. M. Darvish, E. Seiler, S. Mehringer, R. Rahn, and K. Reinert. Needle: a fast and space-efficient prefilter for estimating the quantification of very large collections of expression experiments. *Bioinformatics*, 38(17):4100–4108, 2022.
9. R. C. Edgar, B. Taylor, V. Lin, T. Altman, P. Barbera, D. Meleshko, D. Lohr, G. Novakovsky, B. Buchfink, B. Al-Shayeb, et al. Petabase-scale sequence alignment catalyses viral discovery. *Nature*, 602(7895):142–147, 2022.
10. B. Ekim, B. Berger, and R. Chikhi. Minimizer-space de Bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer. *Cell systems*, 12(10):958–968, 2021.
11. S. Faro and T. Lecroq. The exact online string matching problem: A review of the most recent results. *ACM Computing Surveys (CSUR)*, 45(2):1–42, 2013.
12. A. Gallant. ripgrep. https://github.com/BurntSushi/ripgrep, 2024. Accessed: 2025-06-02.
13. S. Golan, I. Tziony, M. Kraus, Y. Orenstein, and A. Shur. GreedyMini: Generating low-density DNA minimizers. *bioRxiv*, pages 2024–10, 2024.
14. R. Groot Koerkamp, D. Liu, and G. E. Pibiri. The open-closed mod-minimizer algorithm. *Algorithms for Molecular Biology*, 20(1):4, 2025.
15. R. Groot Koerkamp and I. Martayan. SimdMinimizers: Computing random minimizers, fast. *bioRxiv*, pages 2025–01, 2025.
16. N. Homer, S. Stadick, S. Lambert, M. Stone, and T. Fennell. fqgrep. https://github.com/fulcrumgenomics/fqgrep, Mar. 2025.
17. M. Karasikov, H. Mustafa, G. Rätsch, and A. Kahles. Lossless indexing with counting de Bruijn graphs. *Genome Research*, 32(9):1754–1764, 2022.
18. T. Lemane, N. Lezzoche, J. Lecubin, E. Pelletier, M. Lescot, R. Chikhi, and P. Peterlongo. Indexing and real-time user-friendly queries in terabyte-sized complex genomic datasets with kmindex and ora. *Nature Computational Science*, 4(2):104–109, 2024.

19. H. Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.
20. B. Ma, C. Lu, Y. Wang, J. Yu, K. Zhao, R. Xue, H. Ren, X. Lv, R. Pan, J. Zhang, et al. A genomic catalogue of soil microbiomes boosts mining of biodiversity and genetic resources. *Nature communications*, 14(1):7318, 2023.
21. 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*, 31(1):1–12, 2021.
22. C. Marchet and A. Limasset. Scalable sequence database search using partitioned aggregated bloom comb trees. *Bioinformatics*, 39(Supplement_1):i252–i259, 2023.
23. S. Nayfach, Z. J. Shi, R. Seshadri, K. S. Pollard, and N. C. Kyrpides. New insights from uncultivated genomes of the global human gut microbiome. *Nature*, 568(7753):505–510, 2019.
24. D. H. Parks, C. Rinke, M. Chuvochina, P.-A. Chaumeil, B. J. Woodcroft, P. N. Evans, P. Hugenholtz, and G. W. Tyson. Recovery of nearly 8,000 metagenome-assembled genomes substantially expands the tree of life. *Nature microbiology*, 2(11):1533–1542, 2017.
25. D. Pellow, L. Pu, B. Ekim, L. Kotlar, B. Berger, R. Shamir, and Y. Orenstein. Efficient minimizer orders for large values of k using minimum decycling sets. *Genome Research*, 33(7):1154–1161, 2023.
26. M. Rahman Hera and D. Koslicki. Estimating similarity and distance using fracminhash. *Algorithms for Molecular Biology*, 20(1):1–13, 2025.
27. M. Roberts, W. Hayes, B. R. Hunt, S. M. Mount, and J. A. Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, July 2004.
28. E. W. Sayers, M. Cavanaugh, K. Clark, K. D. Pruitt, S. T. Sherry, L. Yankie, and I. Karsch-Mizrachi. Genbank 2024 update. *Nucleic Acids Research*, 52(D1):D134–D137, 2024.
29. S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 76–85, New York, NY, USA, June 2003. Association for Computing Machinery.
30. W. Shen, S. Le, Y. Li, and F. Hu. Seqkit: a cross-platform and ultrafast toolkit for fasta/q file manipulation. *PloS one*, 11(10):e0163962, 2016.
31. W. Shen, B. Sipos, and L. Zhao. Seqkit2: A swiss army knife for sequence and alignment processing. *Imeta*, 3(3):e191, 2024.
32. B. Solomon and C. Kingsford. Fast search of thousands of short-read sequencing experiments. *Nature biotechnology*, 34(3):300–302, 2016.
33. N. Teyssier and A. Dobin. Binseq: A family of high-performance binary formats for nucleotide sequences. *bioRxiv*, pages 2025–04, 2025.
34. L. Vandamme, B. Cazaux, and A. Limasset. K2R: Tinted de Bruijn Graphs implementation for efficient read extraction from sequencing datasets. *Bioinformatics Advances*, page vbaf111, 05 2025.
35. X. Wang, Y. Hong, H. Chang, K. Park, G. Langdale, J. Hu, and H. Zhu. Hyperscan: A Fast Multi-pattern Regex Matcher for Modern CPUs. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 631–648, 2019.
36. A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowski. Alignment-free sequence comparison: benefits, applications, and tools. *Genome biology*, 18:1–17, 2017.