PNAS NEXUS

PNAS Nexus, 2026, **5**, pgag001

https://doi.org/10.1093/pnasnexus/pgag001
Advance access publication 6 January 2026
Research Report

Fast barcode calling based on *k*-mer distances

Riko Corwin Uphoff[a], Steffen Schüler[a], Ivo Grosse[a] and Matthias Müller-Hannemann[a,*]

[a]Institute of Computer Science, Martin Luther University Halle-Wittenberg, Von-Seckendorff-Platz 1, D 06099 Halle, Germany
*To whom correspondence should be addressed: Email: muellerh@informatik.uni-halle.de
Edited By Christopher Dupont

Abstract

DNA barcodes, which are short DNA strings, are regularly used as tags in pooled sequencing experiments to enable the identification of reads originating from the same sample. A crucial task in the subsequent analysis of pooled sequences is barcode calling, where one must identify the corresponding barcode for each read. This task is computationally challenging when the probability of synthesis and sequencing errors is high, like in photolithographic microarray synthesis. Identifying the most similar barcode for each read is a theoretically attractive solution for barcode calling. However, an all-to-all exact similarity calculation is practically infeasible for applications with millions of barcodes and billions of reads. Hence, several computational approaches for barcode calling have been proposed, but the challenge of developing an efficient and precise computational approach remains. Here, we propose a simple, yet highly effective new barcode calling approach that uses a filtering technique based on precomputed *k*-mer lists. We find that this approach has a slightly higher accuracy than the state-of-the-art approach, is more than 500 times faster than that, and allows barcode calling for one million barcodes and one billion reads per day on a server GPU. The same throughput can even be realized using a CPU-parallel implementation.

Keywords: DNA barcode calling, *k*-mer filtering, Sequence Levenshtein distance, algorithm engineering, spatial transcriptomics

Significance Statement

Barcode calling is the computational process of assigning sequencing reads to barcodes that uniquely identify biomolecules in pooled sequencing experiments. Typical single-cell experiments require barcode calling for billions of reads and millions of barcodes with error rates exceeding 10% per nucleotide, but current methods can only assign several million reads to one million barcodes per day per GPU in such cases. Here, we describe a method that is several hundred times faster, enabling the analysis of one billion reads per day on a CPU. This advance makes large-scale applications in single-cell and spatial transcriptomics, lineage tracing, or synthetic biology computationally feasible and could facilitate the acceleration of biomedical discoveries and reduce computational costs in a wide range of scientific disciplines.

Introduction

DNA barcodes are used for identifying individual biomolecules in large populations. Applications include studying gene expression at the single-cell level (1), lineage tracing and screening (2), spatial transcriptomics (3, 4), DNA data storage (5), and the exploration of developmental trajectories and cancer progression (6, 7).

In a typical experiment, hundreds of millions of reads are sequenced, each containing one barcode modified by synthesis or sequencing errors that include substitutions, insertions, and deletions (8). Correctly assigning the reads containing these modified barcodes to the barcodes they originate from is referred to as barcode identification or barcode calling (9). If the expected error rate is very low, one can simply look for exact barcode matches, which is very fast and often used in practice. Error-correcting codes such as those from Hawkins et al. (10) allow also fast barcode calling, but impose severe limitations on the number of barcodes. Modern synthesis techniques such as photolithographic microarray synthesis allow the synthesis of millions of barcodes needed for spatial transcriptomics studies at a reasonable cost (11), but their error rates may exceed 20% per base (11), leading to the computational challenge of barcode calling for large barcode sets with high error rates. In a recent breakthrough result, Press (8) has shown that *random* barcodes of sufficient length are accurately decodable even for applications with such high error rates.

The central idea of many barcode calling approaches is to determine for each read a *most similar* barcode with respect to a given distance measure. However, the naive approach of identifying the most similar barcode for each read by computing the distance of each read and each barcode is prohibitively time-consuming, so several alternative approaches for barcode calling have been developed in the last decade: Zorita et al. (12) propose the calculation

Competing Interest: The authors declare no competing interests.
Received: May 18, 2025. **Accepted:** December 19, 2025

of all pairwise edit distances using prefix trees and several optimizations of the classical Needleman-Wunsch algorithm as well as clustering ("starcode"). Tambe and Pachter (9) develop the barcode-calling approach "sircel" and demonstrate its efficacy for small barcode sets and short barcodes. Press (8) proposes a filtering approach (triage) based on trimer similarities and thus accelerates barcode calling by orders of magnitude, thereby allowing barcode calling for one million barcodes and ten million reads per day on a server GPU. Yet, Press' filtering approach is still computationally expensive as it requires a linear scan over the entire barcode set for each read, with a non-negligible workload per iteration.

Here, we propose a k-mer filtering approach that enables barcode calling for one million barcodes and one *billion* reads per day on a server GPU or on a 64-core CPU.

This speed-up is accomplished by a simple auxiliary data structure that links given barcodes to their sub-sequences. Setting up this data structure requires only linear additional space and negligible running time in a preprocessing phase. We find by theoretical and experimental studies that the proposed k-mer filtering approach allows reducing the computational work by more than two orders of magnitude compared to Press (8). An implementation of this barcode calling approach named "Quik" can be accessed freely on GitHub.

The rest of this article is structured as follows: In Methods section, we provide a high-level description of the k-mer filtering approach and additional material used for the subsequent analyses and experiments. In Results section, we present results on the accuracy of barcode calling using the k-mer filtering approach, results of a theoretical analysis of its running time as a function of k, and results of an empirical study of its running time. Finally, we conclude this article in Conclusions section.

Methods

In this section, we present a high-level description of the barcode calling approach used in Quik, details of the k-mer pseudo-distance used to filter candidate barcodes, several distance measures for quantifying sequence similarities, and combinations of different variants of the barcode calling approach, resulting in a two-step barcode calling scheme.

Barcode calling

The task of barcode calling is to identify for each read the barcode from which it originated. The Quik approach assumes that all barcodes have a fixed length ℓ. Meanwhile, in a real experiment, reads are generally much longer DNA strings, each containing a barcode section of the same length ℓ at a well-defined position. To identify the barcode of a read, one can extract the barcode section of the longer DNA string. For simplicity, we will refer only to this extracted barcode section when referring to a specific read. On an abstract level, the Quik approach can be divided into three phases.

- **Phase 0: Preprocessing:** For each possible k-mer $m \in \{A, C, G, T\}^k$ and each possible starting position $j \in \{0, \ldots, \ell - k\}$ within a barcode, compile a list $L(m, j)$ of *all barcodes* in which m occurs as a sub-sequence starting at position j.

While phase 0 only needs to be executed once for a fixed barcode set, the other phases need to be executed *for all reads* r.

- **Phase 1: Barcode filtering:** Evaluate for each barcode b some efficiently computable *pseudo-distance* $q(b, r)$ (k-mer pseudo-distance section). Compile a small candidate set C of at most c barcodes with *smallest* pseudo-distance to r.
- **Phase 2: Final assessment:** For each candidate barcode b in C, evaluate some distance $d(b, r)$ (see Distance measures section). Return a candidate barcode with minimum distance to r if this distance is smaller than a given threshold D.

Our approach has some similarities to that of Press (8). However, the main difference is the filtering step in phase 1. While Press evaluates several criteria that are ultimately combined into a final barcode ranking, we construct a candidate set with much less computational effort. As we will show in Results section, this accelerates the filtering step by a large factor without any loss of accuracy. In the remainder of this section, we describe some details of the Quik barcode calling approach.

k-mer pseudo-distance

The original barcode will be included within the candidate set with a high probability if the pseudo-distance q and the exact distance d have a high correlation, so the challenge of devising a suitable filtering approach is to find a pseudo-distance (i) that can be computed sufficiently fast and (ii) that is sufficiently similar to the distance d.

The central idea of our k-mer pseudo-distance $q(b, r)$ is to consider each k-mer in the read r and to search for its nearest occurrence on the barcode b. The absolute difference between the starting positions of the k-mer in r and b quantifies how much this k-mer has been shifted away from its original position. We define the pseudo-distance as the sum of the absolute positional differences over all k-mers that occur in the read r. If some k-mer of r does not exist in b or is shifted away by more than $d_{\max}$ positions, we add ℓ as a penalty to the pseudo-distance instead.

The above description should be sufficient to explain the main idea and give some intuitions but is not enough to evaluate the pseudo-distances efficiently. To achieve this goal, we had to make some adjustments.

1. Instead of searching for the *closest* occurrence of a given k-mer in the barcode, we assume that each k-mer occurs *only once* within the threshold difference $d_{\max}$, and speed up the calculation by accounting for *all* occurrences of the k-mer within the threshold distance.
2. We make use of the precomputed lists $L(m, j)$ of barcodes in which a k-mer m occurs at position j. This allows us to evaluate the pseudo-distances between a fixed read r and *all* barcodes in one run without having to search for k-mers on the barcodes.
3. We avoid explicitly adding the penalty term ℓ for k-mers that *do not* occur in the barcode or that are shifted by more than $d_{\max}$ positions. To do this, we subtract from each pseudo-distance $q(b, r)$ a fixed value of ℓ for *each* k-mer in the read r, and thus $\ell \cdot (\ell - k + 1)$ in total. This makes the penalty terms zero and they no longer need to be treated explicitly. As a consequence, the pseudo distance is always nonpositive, with zero being the largest possible (and therefore worst) distance.

Algorithm 1 implements these ideas and gives some additional details.

The algorithm constructs a candidate set C of barcodes that share at least one close k-mer with the read r. In the end, the array

Algorithm 1 k-mer pseudo-distance

Require: Read r; barcode count n; list of barcodes $L(m, j)$ for each $m \in \{A, C, G, T\}^k$ and each $j \in \{0, \ldots, \ell - k\}$
$D \leftarrow \text{int}[n]$ ▷ pseudo-distances, initialized with zero
$C \leftarrow$ empty set ▷ candidate set of barcodes

for each starting position i from 0 up to $\ell - k$ **do**
  $m \leftarrow k$-mer starting at position i in r
  for $j = \max(0, i - d_{max})$ up to $\min(\ell - k, i + d_{max})$ **do**
    $B \leftarrow L(m, j)$ ▷ barcodes containing m at position j
    for each barcode $b \in B$ **do**
      if $D[b] = 0$ **then**
        add b to C
      end if
      $D[b] \leftarrow D[b] + |i - j| - \ell$
    end for
  end for
end for

return C and D

D contains the pseudo-distance of each barcode. The candidate set C can then be reduced to the c barcodes with the smallest pseudo-distance.

Distance measures

After compiling the candidate set, we compute a precise distance measure $d(b, r)$ for each candidate barcode. Several distance measures have been proposed to quantify the distance between two given sequences, including the Hamming distance (13, 14), the Levenshtein distance (8, 15, 16), or the Sequence-Levenshtein distance (17). While the Hamming distance is not suited to account for insertions or deletions ("indels") in the read, the Levenshtein distance and the Sequence-Levenshtein distance do so. However, the Levenshtein distance requires the user to know the lengths of both sequences. As deletions and insertion can alter the length of the relevant segment in the read, the actual length of the substring corresponding to its barcode is unknown. To account for this, Buschmann and Bystrykh have proposed the Sequence-Levenshtein distance to allow truncation of both sequences at their ends at no cost (17). Hawkins et al. (10) have noted that the Sequence-Levenshtein distance is not a distance as it can violate the triangle inequality. Still, they have used it for generating error-correcting barcodes, and we use it in this work as well.

Several ways to compute the (Sequence-)Levenshtein distance have been discussed. For barcodes of length ℓ, the classical dynamic programming algorithm computes the (Sequence-)Levenshtein distance in $O(\ell^2)$. Under the Strong Exponential Time Hypothesis (SETH) this is optimal up to subpolynomial factors (18). However, for a given threshold D, one can check in time $O(D\ell)$ whether two strings of length ℓ have a (Sequence-)Levensthein distance of at most D (19, 20). As we are only interested in the (Sequence-)Levenshtein distance up to $D < \ell$, we might also adapt the algorithm proposed by Zorita et al. (12) to accelerate the computation.

However, the sequence length ℓ is typically a small constant in practical situations (in our case $\ell = 34$). In such situations, an optimized variant of the classical dynamic program is usually faster than the other, more advanced methods. Thus, we use the classical $O(\ell^2)$ algorithm in our implementation.

Two-step barcode calling

We find from the results of Results section that increasing the length k of the subsequences trades better performance for a loss in precision and recall. Furthermore, one can raise the precision at the cost of diminished recall by decreasing the maximum accepted Sequence-Levenshtein distance D. Hence, we can tune Quik to attain high throughput and high precision but low recall by choosing a large k and a small D.

We can exploit this observation to quickly identify the "easy" reads with few mutations and leave the rest of the missed reads for a slower but more accurate variant. We call the resulting scheme *two-step calling*. This calling scheme has two parameters for the threshold distances D_1 and D_2 and two parameters for the lengths of the subsequences k_1 and k_2 (with $k_1 > k_2$). In a first step, we run the barcode calling approach with the parameters k_1 and D_1 to quickly identify barcodes that are close to the associated read. For the reads that are not yet assigned, we run a second step with the parameters k_2 and D_2.

Parallelization strategies

To gain a high throughput of reads, it is essential to run a large number of calculations in parallel. Since phases 1 and 2 are executed independently for each read, there are two obvious parallelization strategies for the Quik calling scheme.

1. Press (8) describes how to parallelize the search for the most similar barcode for a fixed read. Following this idea, one would sequentially process one read after the other and do the calculations of phases 1 and 2 in parallel.
2. We propose an orthogonal parallelization strategy in which we process the reads in parallel and, for each read, sequentially search for the most similar barcode. We thus launch a large number of long-running parallel threads, each responsible for one read, and each running phases 1 and 2 sequentially.

While the first strategy is highly efficient when the number of reads is small, the resulting parallelization overhead makes it unattractive when the number of reads grows into the millions. In such situations, the second approach seems to be more promising. For this reason, we implemented the second strategy.

Error simulation

To assess the accuracy of barcode calling approaches, one needs labeled data sets where the original barcode is known for each read. Such data sets cannot be obtained experimentally, but they can be obtained by simulation (8), and we follow this approach here. As we employ a different error simulation strategy than Press (8), we give a brief overview of how we generate a read r from a barcode b.

We view the synthesis of a read as sending the barcode sequence through a noisy channel. Thus, we incrementally build up the read from the barcode with a certain probability of mutation p in each iteration. A mutation can be either a substitution (false interpretation of the signal), a deletion (missing a signal), or an insertion (false detection of a signal). To simplify the simulation, we assume that all three types of mutation are equally likely, each with a probability of $p/3$, and that the errors are independent across iterations. We also assume identical substitution probabilities for the 12 substitutions, identical insertion probabilities for the four insertions, and identical deletion probabilities for the four deletions. We note in passing that these assumptions are consistent with choosing the Levenshtein distance or the Sequence-Levenshtein distance for asserting the similarity

between the read and the barcode. See Supplementary material S1 for further detail on the simulation of reads.

While the assumptions typically do not hold in practice, the details can vary greatly between different experimental settings. Thus, we avoid making overly detailed assumptions about error distributions in order to maintain generalizability. One could also easily extend the procedure to more accurately capture the mutation probabilities of the experimental setting.

Results and discussion

In this section, we present the results of several theoretical and empirical studies on our barcode calling approach. In particular, we discuss the accuracy of the k-mer pseudo-distance q and the k-mer filtering approach as well as a theoretical and empirical analysis of its running time.

Data set

In all of the following experiments and analyses, we used a fixed set of $n = 1{,}000{,}000$ barcodes of length $\ell = 34$, drawn uniformly at random from the set of all barcodes of this length. To each of these barcodes, we applied the error model described in Error simulation section with an error probability of $p = 0.2$ per base. In doing so, we generated a fixed set of 1,000,000 reads. Supplementary material S2 contains the results for the same experiments with error probabilities of 0.1 and 0.3, experiments with different error probabilities for insertions, deletions, and substitutions, namely using error rates for experiments with barcodes constructed by photolithography (11), and experiments with designed barcodes that satisfy constraints on CG content, homopolymer avoidance, and pairwise mutual Sequence-Levenshtein distance.

Parameters

Both the accuracy and the running time of the Quik barcode calling approach depend on the parameter choice, in particular, on the k-mer length k and the threshold distance D. In the following experiments, we tested several combinations of $k \in \{4, 5, 6\}$ and $D \in [0, 15]$. For a clean and concise presentation, we fixed the parameters of the two-step variants to $k_1 = 7$ and $D_1 = 10$ and let the parameters k_2 and D_2 vary in the ranges mentioned above. In all experiments, we used a maximum shift size of $d_{\max} = 5$ for the evaluation of the pseudo-distance (see k-mer pseudo-distance section). Finally, we used a candidate set of size $c = 100$ for all experiments unless explicitly stated otherwise.

Accuracy of the k-mer pseudo-distance q

In a first experiment, we studied the accuracy of the k-mer pseudo-distance q in comparison to the triage suggested by Press (8) in phase 1. For this purpose, we applied each approach on our benchmark data set and determined a posteriori how often the original barcode occurred within the candidate set of size $c \in [1, 10^4]$. We call the relative frequency of this number the *hit rate* for size c. Thus, the larger the hit rate, the more often was the original barcode within the candidate set of size c. Consequently, a large hit rate is necessary for an accurate barcode calling. Figure 1 shows the results of this experiment. We want to highlight some essential observations.

- For all approaches, the hit rate grows monotonically and approximately linearly with $\log c$ for $c \in [1, 10^3]$.

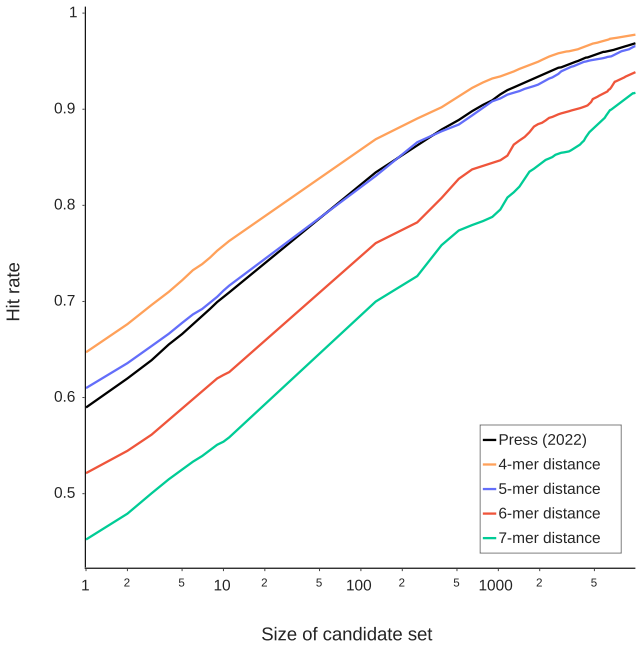

Fig. 1. Relative frequency of the original barcode occurring in the candidate set of size $c \leq 10^4$. Note that the x-axis is scaled logarithmically.

- A candidate set of size $c = 100$ contains the correct barcode candidates in about 80% of the cases for $k = 4$.
- Increasing the value of k by one reduces the hit rate by a constant for a fixed size c.
- The hit rate for $k = 5$ is similar to that of the approach by Press (8).

We found qualitatively similar results by analogous studies with error probabilities of $P = 0.1$ and $P = 0.3$ (see Supplementary material S2). We conclude that the pseudo-distance is well-suited for effectively prefiltering the relevant barcodes. In particular, the hit rate is with $k = 5$ similarly high as with the triage rules suggested by Press.

Accuracy of the k-mer filtering approach

To study the accuracy of the *complete* barcode calling process, we apply the different approaches to the benchmark data set and calculate the *precision* defined as the ratio of correctly called reads and all accepted reads (8) and the *recall* defined as the ratio of accepted reads and all 1,000,000 reads (8). Precision and recall highly depend on the choice of the exact distance measure and on the maximum acceptable distance parameter D. In this experiment, we calculated precision and recall for various values of $D \in [0, 15]$ and for both the Levenshtein and Sequence-Levenshtein distance. Figures 2 and 3 show the results. The following observations are essential:

- At lower recall, all approaches yield a precision of ~100%.
- The Sequence-Levenshtein distance gives higher precision and recall than the Levenshtein distance.
- Using the Sequence-Levenshtein distance, the k-mer distance approaches for $k \in [4, 6]$ yield higher precision and recall than Press' approach.
- We find that the precision and recall of the two-step variants is almost identical to the associated one-step approach.

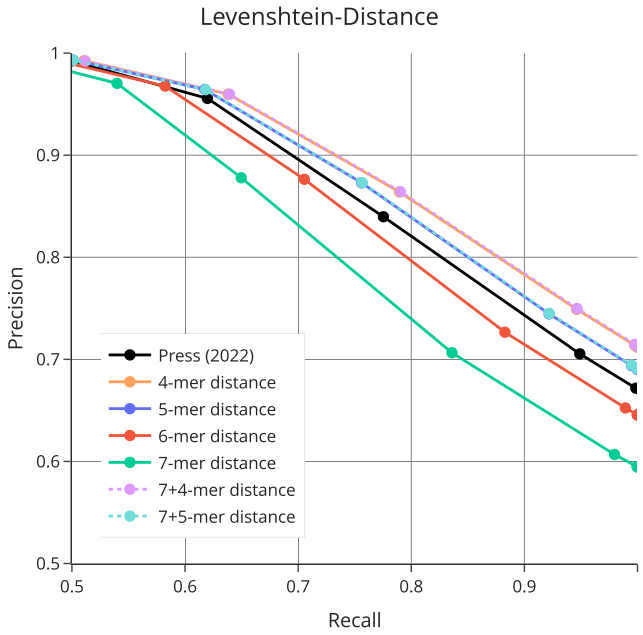

Fig. 2. Precision-recall curves for different values of the maximum acceptable distance D to the read. Each integer value of $D \in [0, 15]$ yields one point, forming a curve of 16 connected points for each approach. All approaches use the Levenshtein distance as an exact distance measure. The curves for the 4-mer and 7 + 4-mer distance are almost identical. The same is true for the 5-mer and 7 + 5-mer curves.

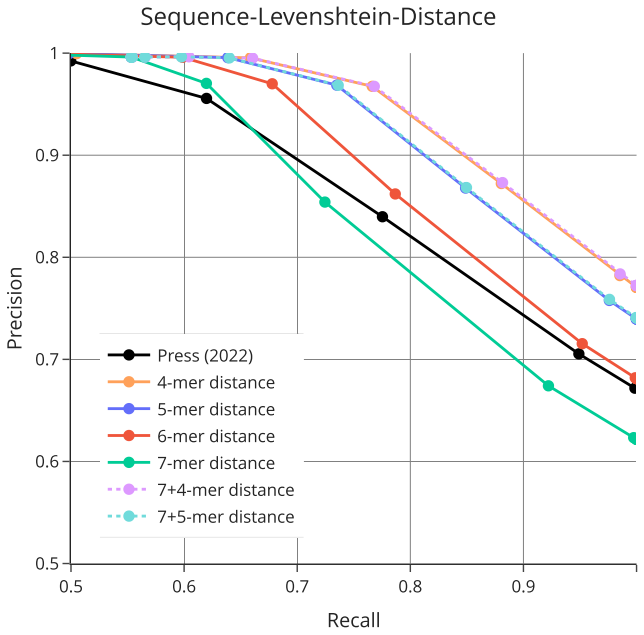

Fig. 3. Precision-recall curves for different values of the maximum acceptable distance D to the read. Each integer value of $D \in [0, 15]$ yields one point, forming a curve of 16 connected points for each approach. The k-mer distance approaches used the Sequence-Levenshtein distance while the approach by Press used the Levenshtein distance. The curves for the 4-mer and 7 + 4-mer distance are almost identical. The same is true for the 5-mer and 7 + 5-mer curves. Note that Press' program does not support the Sequence-Levenshtein distance, so the black curve is the same as in Fig. 2.

Based on the results of this experiment, we used a maximum acceptable distance of $D = 8$ and the Sequence-Levenshtein distance in the following experiments. In doing so, we gain a barcode calling approach with similar or higher accuracy than Press. In Supplementary material S2, these results are confirmed when unequal error probabilities or designed barcodes are employed.

Theoretical analysis of running time

In this subsection, we present a theoretical analysis of the expected running time required to calculate the pseudo-distances between a single read and *all* barcodes as described in Algorithm 1. For this purpose, we concentrate on the expected number of executions of line 12 in Algorithm 1, which we call the number of *inner iterations*.

This analysis is based on the two simplifying assumptions that all k-mers of a sequence be independent and all k-mers be equally likely. In general, assuming independence will lead to an overestimation of the number of operations performed. The main result of this analysis is the following.

THEOREM 1 Phase 1 is expected to take at most

$$n\frac{(\ell - k + 1)\cdot(2d_{\max} + 1)}{4^k}$$

inner iterations for computing the pseudo-distances for n barcodes of length ℓ.

Proof. The probability that a random k-mer resembles a queried sub-sequence is about $\frac{1}{4^k}$, since there are 4^k different k-mers. Therefore, the list of barcodes containing a given k-mer at a fixed position has an average length of at most $\frac{n}{4^k}$. Hence, due to the assumed independence of the sub-sequences and since we consider at most $2d_{\max} + 1$ positions for each of the $\ell + k - 1$ sub-sequences in the read, the number of inner iterations is at most

$$n\frac{(\ell - k + 1)(2d_{\max} + 1)}{4^k}.$$

Theorem 1 implies that, under the assumption that $d_{\max}$ is a constant and that $k \geq \frac{1}{2}\log_2 \ell$ (a viable choice according to previous results), we obtain a time complexity of

$$\mathcal{O}\left(n\frac{\ell}{4^{\log_2 \ell/2}}\right) = \mathcal{O}\left(n\frac{\ell}{\ell}\right) = \mathcal{O}(n).$$

Theorem 1 also implies that increasing the parameter k by one reduces the expected number of iterations per read by a factor of four. Theorem 1 further implies that, for a parameter set of $\ell = 34$, $k = 4$, and $d_{\max} = 5$, we obtain on average at most $1.33n$ iterations per read. Using similar arguments, one can show that the approach by Press (8) takes $262n$ iterations per read (see Supplementary material S3).

We conclude that the k-mer filtering step can *in theory* be executed about $262/1.33 \approx 197$ times faster than the triage rule filtering proposed by Press.

Empirical analysis of running time

To test whether the theoretical results from the previous section hold in practice, we next performed an *empirical* analysis of the number of inner iterations of Algorithm 1. To this end, we applied the approaches for several values of k to our benchmark data set and simply counted how often line 12 is executed.

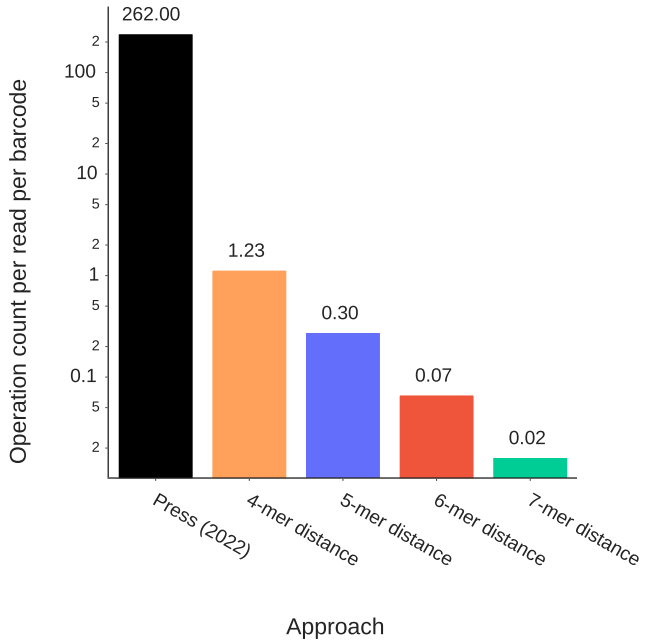

Fig. 4. Normalized operation count. The ordinate shows the log-scaled mean numbers of inner iterations per read and barcode.

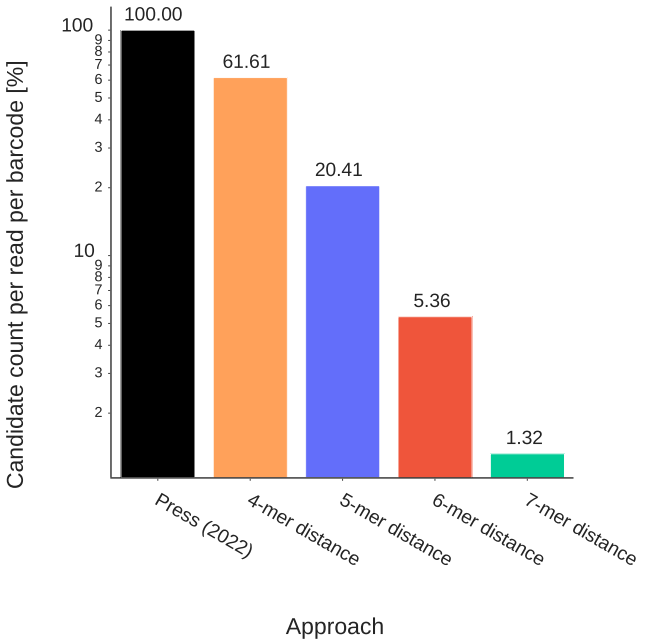

Fig. 5. Fraction of barcodes that need to be scanned to find the best candidates. Note the log-scaled y-axis.

The following observations from Figs. 4 and 5 are essential:

- The number of inner iterations decreases by a factor of 4 when increasing k by one. That is consistent with the theoretical analysis of Theoretical analysis of running time section.
- The computation of the 4-mer pseudo-distance requires about $1.23n$ inner iterations and thus about $262/1.23 \approx 213$ times fewer iterations than that of the approach by Press.
- When increasing k by one, the number of barcodes that share at least one k-mer with the read r reduces by a factor of about $\frac{2}{3}$.
- For a given read, the approach by Press has to sort all barcodes by pseudo-distance to obtain the final candidate set. More accurately, the approach sorts five different candidate sets, each containing 100% of the barcodes, to combine them into the final pseudo-distance and sort all barcodes again. In contrast, the Quik approach only once probes a much smaller fraction of all barcodes, namely those for which a k-mer of the read appears in the precomputed lists.

In summary, our experiment confirms the previous theoretical analysis.

Empirical analysis of running time on GPUs

Next, we examined the *empirical* running time of the complete barcode calling approach as a function of k in comparison to Press' approach. For this purpose, we implemented the k-mer filtering approach as a GPU program in CUDA and compared it to the GPU program developed by Press. We applied each program to our benchmark data set and measured the running time of the calling process (phases 0 to 2, but excluding the time for loading barcodes and reads from disk.) All computations have been executed on a Linux system with an AMD EPYC 7662 64-Core processor, 1 TB of main memory and an NVIDIA A100-SXM4 GPU having 6912 CUDA cores and 40 GB GPU memory. Figure 6 shows the results of this experiment.

- We observe that all k-mer approaches are much faster than Press' program. In particular, the 6-mer calling approach (which has higher accuracy than Press' program) is faster by a factor of $31.82/0.06 \approx 530$. This implies a theoretical throughput of about $24 \cdot 60 \cdot 60 \cdot 1000/0.06 = 1.44 \cdot 10^9$ reads per day.
- This large speed-up is induced by our efficient filtering step (phase 1) and the different parallelization strategy. Our strategy utilizes 100% of the computing resources. Instead, Press's program utilizes at best 33% of our GPU.
- We further observe that the two-stage filtering approaches (the two rightmost columns) can slightly reduce the running times. The combined $7+4$-mer distance improves on the 4-mer distance, and the $7+5$-mer distance improves on the 5-mer distance. We also find that the speed-up due to two-stage filtering depends on the error probability P. The advantage decreases as P increases (Supplementary material S2). This observation is due to the fact that the 7-mer distance approach becomes less effective as p increases.

We conclude that the k-mer distance approach is not only more accurate but also *much* faster than that of Press' program on a server GPU.

Empirical analysis of running time on CPUs

Finally, we investigate which running times of the k-mer distance approach could be accomplished by a classical CPU-parallel implementation using OpenMP. We find from Fig. 7 that the running times of the CPU implementation are even slightly lower than those of the GPU implementation. This finding is surprising and indicates that GPUs do not give an advantage to CPUs for the barcode calling process using the k-mer distance approach. This is good news for many applications where GPUs are still rare. The relatively poor performance of the GPU seems to be due to the fact that calculating pseudo-distances and running through individual candidate lists are tasks that involve an unstructured and therefore GPU-unfriendly memory access pattern. This means

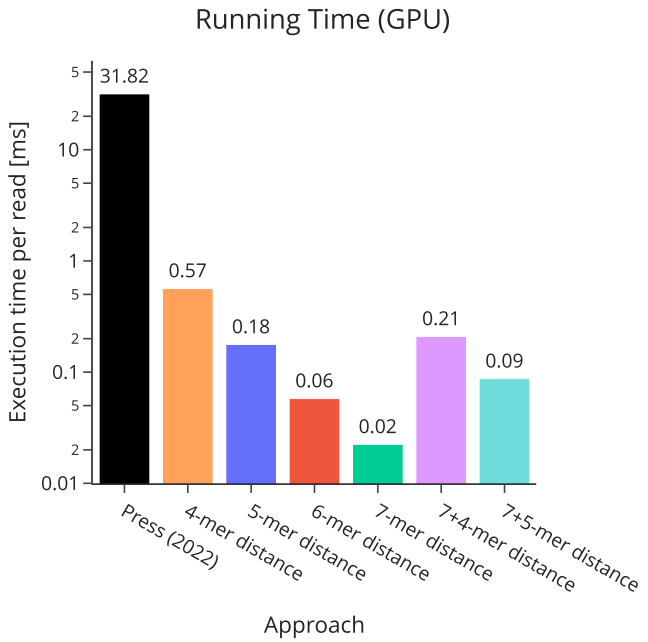

Fig. 6. Average running times per read of our GPU implementations and Press' GPU program. The ordinate shows the log-scaled running time in milliseconds per read.

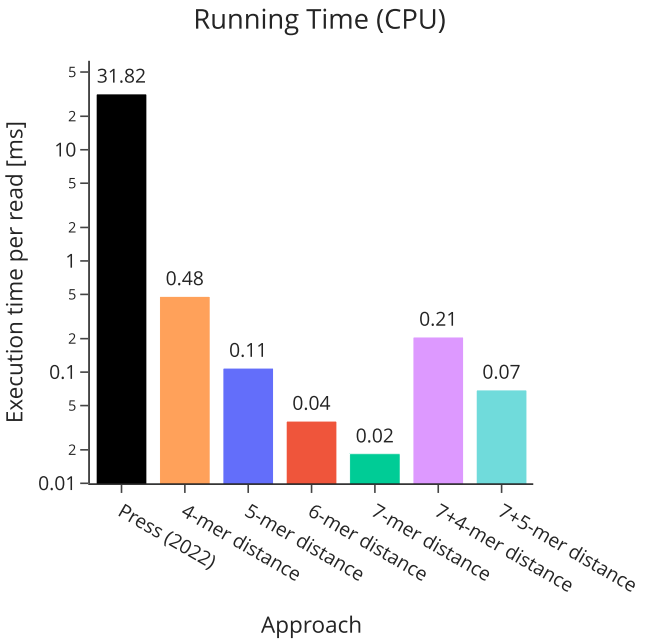

Fig. 7. Average running times per read of our CPU-parallel implementations and Press' GPU program. The ordinate shows the log-scaled running time in milliseconds per read.

that the memory transfer rate of the GPU limits the throughput during barcode calling. In contrast, the CPU variant does not have these limitations. Due to the various cache levels in modern multicore processors, memory accesses are much faster. In addition, the CPU cores can work independently of each other due to the MIMD principle, whereas thread divergence (i.e. the GPU threads wait to execute until their neighboring threads are finished) is likely to occur to a high degree when executing the GPU variant.

Conclusions

The capability of performing barcode calling for billions of reads and millions of barcodes is essential in the modern life sciences in diverse areas including single-cell and spatial transcriptomics, lineage tracing, or synthetic biology. This computational problem is particularly challenging when the error rate exceeds 10% or even 20% per nucleotide as in modern synthesis techniques such as photolithographic microarray synthesis. A recent breakthrough by Press (8) enables barcode calling in such cases for several million reads and one million barcodes per day per GPU. Here, we have developed a method called Quik based on k-mer filtering, which allows the user to tune the parameter k for choosing the trade-off between a desired level of accuracy and a low running time. This method is based on a precomputed auxiliary data structure that allows an efficient filtering of relevant barcodes. In a theoretical analysis, we have found that Quik's filtering step requires about 197 times fewer inner loop iterations than that of Press on average. In a first experimental analysis based on one million barcodes and simulated sets of reads with an error rate of 20% per nucleotide, we have found that the Quik approach yields an accuracy slightly higher than that of Press and that it reduces the running time on a GPU by more than two orders of magnitude, enabling the analysis of one billion reads per day per GPU. In a second experimental analysis, we have found that the running time of a parallel CPU implementation of the Quik approach using OpenMP is even slightly faster than that of the corresponding GPU implementation. While large-scale analyses of single cell or spatial transcriptomics data with millions of barcodes would currently require a high-performance cluster for barcode calling on billions of reads with error rates above 10%, the increased throughput of the Quik approach now enables these analyses on much cheaper hardware widely available in research labs worldwide.

Acknowledgments

We thank Lothar Altschmied, Jan Grau, Rene Malsch, Paride Rizzo, and Antonia Schmidt for valuable discussions.

Supplementary Material

Supplementary material is available at *PNAS Nexus* online.

Funding

This study has been performed as part of the DiP network "Digitalization of plant value chains" within project "DiP-HyperSpace" funded by the German Federal Ministry of Research, Technology and Space (BMBFTR FKZ 031B1448C).

Author Contributions

R.C.U. and M.M.-H. developed the approach, R.C.U. and S.S. implemented the algorithms and performed the studies, and all authors analyzed the results and wrote the article.

Preprints

This manuscript was posted on a preprint: https://doi.org/10.1101/2025.05.12.653416.

Data Availability

The CUDA implementation of the Quik barcode calling approach, data files, and auxiliary scripts to reproduce our experiments are available on Github https://github.com/uni-halle/quik (21).

References

1 Klein AM, *et al.* 2015. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*. 161(5):1187–1201.
2 Kebschull JM, Zador AM. 2018. Cellular barcoding: lineage tracing, screening and beyond. *Nat Methods*. 15(11):871–879.
3 Liu Y, *et al.* 2020. High-spatial-resolution multi-omics sequencing via deterministic barcoding in tissue. *Cell*. 183(6):1665–1681.e18.
4 Wirth J, *et al.* 2023. Spatial transcriptomics using multiplexed deterministic barcoding in tissue. *Nat Commun*. 14(1):1523.
5 Banal JL, *et al.* 2021. Random access DNA memory using Boolean search in an archival file storage system. *Nat Mater*. 20(9):1272–1280.
6 Bhang H-C, *et al.* 2015. Studying clonal dynamics in response to cancer therapy using high-complexity barcoding. *Nat Med*. 21(5):440–448.
7 Wang Y, Zhang X, Wang Z. 2023. Cellular barcoding: from developmental tracing to anti-tumor drug discovery. *Cancer Lett*. 567(8):216281.
8 Press WH. 2022. Fast trimer statistics facilitate accurate decoding of large random DNA barcode sets even at large sequencing error rates. *PNAS Nexus*. 1(5):pgac252.
9 Tambe A, Pachter L. 2019. Barcode identification for single cell genomics. *BMC Bioinformatics*. 20(1):32.
10 Hawkins JA, Jones SK, Finkelstein IJ, Press WH. 2018. Indel-correcting DNA barcodes for high-throughput sequencing. *Proc Natl Acad Sci U S A*. 115(27):E6217–E6226.
11 Lietard J, *et al.* 2021. Chemical and photochemical error rates in light-directed synthesis of complex DNA libraries. *Nucleic Acids Res*. 49(12):6687–6701.
12 Zorita E, Cuscó P, Filion GJ. 2015. Starcode: sequence clustering based on all-pairs search. *Bioinformatics*. 31(12):1913–1919.
13 Bystrykh LV. 2012. Generalized DNA barcode design based on Hamming codes. *PLoS One*. 7(5):e36852.
14 Lyons E, Sheridan P, Tremmel G, Miyano S, Sugano S. 2017. Large-scale DNA barcode library generation for biomolecule identification in high-throughput screens. *Sci Rep*. 7(1):1–7.
15 Ashlock D, Houghten SKDNA error correcting codes: no crossover. In: *2009 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*. IEEE, 2009. p. 38–45.
16 Faircloth BC, Glenn TC. 2012. Not all sequence tags are created equal: designing and validating sequence identification tags robust to indels. *PLoS One*. 7(8):e42543.
17 Buschmann T, Bystrykh LV. 2013. Levenshtein error-correcting barcodes for multiplexed DNA sequencing. *BMC Bioinformatics*. 14(1):272.
18 Backurs A, Indyk P. 2018. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM Journal on Computing*. 47(3):1087–1097.
19 Landau GM, Vishkin U. 1988. Fast string matching with k differences. *J Comput Syst Sci*. 37(1):63–78.
20 Myers EW. 1986. An $O(ND)$ difference algorithm and its variations. *Algorithmica*. 1(1-4):251–266.
21 Uphoff R, Schüler S. Quik barcode calling software: PNAS Nexus submission [dataset]. May 2025, https://doi.org/10.5281/zenodo.15342483.