Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Subject Section

Real-time mapping of nanopore raw signals

Haowen Zhang [1,†], Haoran Li [2,†], Chirag Jain [3], Haoyu Cheng [4,5], Kin Fai Au [2,*], Heng Li [4,5,*] and Srinivas Aluru [1,6,*]

[1]School of Computational Science and Engineering, Georgia Institute of Technology,
[2]Department of Biomedical Informatics, Ohio State University,
[3]Department of Computational and Data Sciences, Indian Institute of Science,
[4]Department of Data Sciences, Dana-Farber Cancer Institute,
[5]Department of Biomedical Informatics, Harvard Medical School and
[6]Institute for Data Engineering and Science, Georgia Institute of Technology.

*To whom correspondence should be addressed.
†Should be regarded as joint first-authors.

Abstract

Motivation: Oxford Nanopore Technologies sequencing devices support adaptive sequencing, in which undesired reads can be ejected from a pore in real time. This feature allows targeted sequencing aided by computational methods for mapping partial reads, rather than complex library preparation protocols. However, existing mapping methods either require a computationally expensive base calling procedure before using aligners to map partial reads, or work well only on small genomes.
Results: In this work, we present a new streaming method that can map nanopore raw signals for real-time selective sequencing. Rather than converting read signals to bases, we propose to convert reference genomes to signals and fully operate in the signal space. Our method features a new way to index reference genomes using k-d trees, a novel seed selection strategy and a seed chaining algorithm tailored towards the current signal characteristics. We implemented the method as a tool Sigmap. Then we evaluated it on both simulated and real data, and compared it to the state-of-the-art nanopore raw signal mapper Uncalled. Our results show that Sigmap yields comparable performance on mapping yeast simulated raw signals, and better mapping accuracy on mapping yeast real raw signals with a 4.4x speedup. Moreover, our method performed well on mapping raw signals to genomes of size $>$100Mbp and correctly mapped 11.49% more real raw signals of green algae, which leads to a significantly higher F_1-score (0.9354 vs. 0.8660).
Availability: Sigmap code is accessible at `https://github.com/haowenz/sigmap`
Contact: kinfai.au@osumc.edu, hli@jimmy.harvard.edu, and aluru@cc.gatech.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Oxford Nanopore Technologies (ONT) sequencers produce millions of long reads with $>$10kbp N50 in a single 48 to 72 hour run. These long reads can span repetitive regions of a genome that are hard to resolve using short reads, thus enabling assemblies with high continuity (Miga *et al.*, 2020). Direct RNA sequencing through nanopores can sequence full-length RNA transcripts without amplification, which can greatly aid in *de novo* transcriptome analysis (Garalde *et al.*, 2018). Without the need for additional library preparation, amplification-free nanopore sequencing also enables detection of nucleotide modifications (Simpson *et al.*, 2017).

Nanopore sequencers work by measuring ionic current as a molecule passes through a pore. Since different molecules in the pore modulate the current in specific ways, individual nucleotides can be inferred by base calling of the raw current signal. For various ONT pore versions (e.g., R7, R9), the current signal is mainly affected by five or six nucleotides (i.e., k-mers where $k = 5$ or 6) occupying the pore at a given time point. These current readings usually have a low signal-to-noise ratio, which makes it hard to identify the corresponding k-mers. To tackle this problem, many base callers have been developed to “translate" the raw signals to nucleotide sequences (Rang *et al.*, 2018). State-of-the-art base callers (e.g., ONT official base caller Guppy) can achieve around 90% accuracy. However, base calling is computationally expensive and can last days on a high-end central processing unit (CPU) or hours on a graphical processing unit (GPU) even for a relatively low throughput run with only ∼20Gbp data.

The ONT MinION is a portable device which typically yields up to 30Gbp sequencing data using a single flow cell at a low cost. Portability of the MinION sequencer allows sequencing to be performed in the field or the clinic, for example, surveillance for Ebola virus in West Africa (Quick *et al.*, 2016) and fast detection of SARS-CoV-2 with high sensitivity (Wang *et al.*, 2020). The MinION device is compatible with recently released Flongle flow cells with even lower prices while reducing the sequencing throughput to ∼2Gbp for smaller analyses and tests. However, this throughput is usually too low for many applications that require high sequencing depth, which makes targeted sequencing necessary.

Targeted sequencing allows for enriched coverage of desired genomic regions, which reduces sequencing costs and labor to achieve high coverage at regions of interest. Typical targeted sequencing approaches do not work well with nanopore sequencing due to loss of nucleotide modifications, high input requirements, low throughput or long protocols (Gilpatrick *et al.*, 2020). On the other hand, the targeted sequencing protocol designed specifically for nanopore sequencing (Gilpatrick *et al.*, 2020) addressed some of these issues, but still requires additional preparation time and is limited by the maximum size and number of targeted regions.

Alternatively, Loose *et al.* took advantage of the selective sequencing feature of the MinION sequencer and performed real-time targeted sequencing for amplicon enrichment. This is achieved by temporarily reversing the voltage across a nanopore, thereby rejecting an undesired molecule and making the pore available for other molecules. Thus if there is a sufficiently fast computational method that can identify whether reads come from regions of interest, one can quickly eject undesired reads and leave the pores for reads of interest so that undesired genomic regions are not sampled and regions of interest are enriched. In their work, they use dynamic time warping (DTW) to align raw signals to reference genomes to decide whether reads are of interest. Since the time complexity of DTW is quadratic in terms of sequence length, it only works on small genomes that are kilobase pairs long. To address this issue, methods based on base calling followed by read mapping were proposed (Edwards *et al.*, 2019; Payne *et al.*, 2020). However, base callers are not optimized to work on small chunks of reads; thus, they may generate sub-optimal read sequences, which makes mapping challenging (Kovaka *et al.*, 2020). As base calling is a computationally intensive process, enough compute power (e.g., sufficiently powerful GPUs) to achieve real-time base calling may not always be available outside laboratories.

To avoid these drawbacks, Uncalled (Kovaka *et al.*, 2020) was developed to map raw signals in real time without base calling. It builds an FM-index (Ferragina and Manzini, 2005) for reference genomes, segments the raw signals into events (collapsed current readings for each k-mer), and converts the events into possible k-mers using the ONT pore model. High-probability k-mers are used to query the index and extended. Since raw signals are noisy, Uncalled keeps track of all possible positions of each k-mer as the mapping proceeds. After removing false positive locations by a seed clustering method, the final mapping is reported if one of the locations is sufficiently better than the others. The authors demonstrated successful use of Uncalled on targeted sequencing of small genomes (<30 Mbp) and reported that it cannot work properly on mapping raw signals to large genomes that have high repeat content.

In this work, we present a new streaming method to map raw signals for real-time adaptive sequencing. In contrast to previous scalable methods which convert signals to sequences and then leverage existing methods or data structures to map sequences, we convert reference genomes to signals and present a novel streaming method and tool Sigmap to map raw signals to the reference. We evaluated the performance of Sigmap and Uncalled on simulated and real data. Compared with Uncalled, while achieving comparable performance on mapping yeast simulated raw signals, Sigmap mapped slightly more yeast real raw signals accurately and provided 4.4× speedup. Moreover, Sigmap correctly mapped 11.49% more green algae raw signals with significantly higher F_1-scores (0.9354 vs 0.8660). This indicates that our method can map raw signals to genomes of size >100 Mbp, an important advancement over previous base-calling-free methods.

2 Methods

Seed-and-extend is a widely applied strategy to map erroneous long reads (Chaisson and Tesler, 2012; Sović *et al.*, 2016; Sedlazeck *et al.*, 2018; Li, 2018). Typically, exact or approximate word matches between reads and reference genomes are extracted and then co-linear matches (a sequence of matches that occur in ascending order in both reads and reference genomes) are identified to generate final alignments. Our algorithm also follows the seed-and-extend strategy (see Figure 1 for an overview) but is specifically designed to handle noisy raw signal data. Prior to mapping, the reference genome is converted to events and an index of the reference is built once (Section 2.1). In the mapping step, raw current signals are first segmented into events and normalized (Section 2.2). Then seeds that are less likely to contain segmentation errors are selected from the processed raw signal and used to query the index (Section 2.3). After collecting the seed hits (anchors) on the reference, we designed and implemented a chaining algorithm tailored towards the current signal characteristics to find co-linear anchors as chains (Section 2.4). The chains are filtered by their scores to ignore sub-optimal mappings. To do real-time selective sequencing, we presented a streaming version of the proposed algorithm (Section 2.5). The details of each step are as follows.

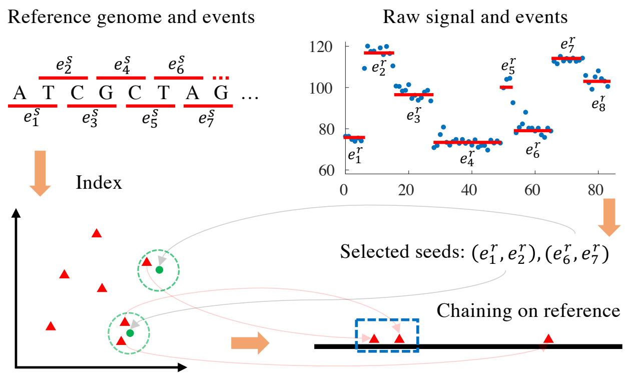
Reference genome and events
e_2^s e_4^s e_6^s
A T C G C T A G ...
e_1^s e_3^s e_5^s e_7^s
Index
Raw signal and events
120
100
80
60
0 20 40 60 80
e_1^r e_2^r e_3^r e_4^r e_5^r e_6^r e_7^r e_8^r
Selected seeds: $(e_1^r, e_2^r), (e_6^r, e_7^r)$
Chaining on reference

Fig. 1. Overview of the proposed algorithm. The reference genome is first converted to a sequence of events $e_1^s, e_2^s, \ldots$ (red lines) using the expected current value of each k-mer in the pore model. For simplicity of illustration, we use 2-mers in this example. Now every pair of consecutive events (e_i^s, e_{i+1}^s) is a point in two-dimensional space, thus a spatial index for these points (red triangles) can be created. For visualization purpose we set dimension to 2, but higher dimensions may be used. In the mapping stage, raw signals (blue dots) are first segmented into events $e_1^r, e_2^r, \ldots$ (red lines). Then seeds are selected to query the index with range search and hits on the reference are chained to get the mapping (in the blue rectangle).

2.1 Indexing

Different pore models are provided by ONT for various pore versions since current readings are affected by different number of nucleotides occupying the pore at each sequencing time point. In this probabilistic model, current readings for each k-mer are assumed to follow a Gaussian distribution with known parameters. Thus using the pore model, one can estimate the probability of a given event being any of the k-mers, or convert a nucleotide sequence to an event sequence by simply substituting k-mers with their expected current readings.

Uncalled uses the prior strategy to generate high-probability k-mers from read events, while our method leverages the latter to convert the reference to events. Note that in the first case a full iteration on all the distributions is usually required to identify high-probability k-mers that an event may correspond to, which can be slow when many events in the read are processed simultaneously. But converting a k-mer to its expected current reading is a direct translation once a hash table is built for the pore model using k-mers as keys and expected current as values. Since the conversion is only done once for reference genomes, we can save the overhead of applying pore models to read events to find high-probability k-mers during the mapping stage.

Formally, let $s = s_1 s_2 \dots s_n$ be a nucleotide sequence of length n over alphabet Σ and its corresponding sequence of k-mers be $K(s) = k_1 k_2 \dots k_{n-k+1}$, where $k_i = s_i s_{i+1} \dots s_{i+k-1}$. The pore model is defined as $f : \Sigma^k \rightarrow \mathbb{R}$, which gives the expected current corresponding to a k-mer. We create the corresponding event sequence as $E(s) = e_1^s e_2^s \dots e_{n-k+1}^s$, where $e_i^s = f(k_i)$. This is translated to a set of points $P(s) = \{p_i^s = (e_i^s, e_{i+1}^s, \dots, e_{i+d-1}^s), 1 \le i \le n-k-d+2\}$ in d-dimensional space. Similarly, for each raw signal sequence r, we generate its events $E(r) = e_1^r e_2^r \dots e_m^r$ (described in Section 2.2). The reads are also translated to points $P(r) = \{p_i^r = (e_i^r, e_{i+1}^r, \dots, e_{i+d-1}^r), 1 \le i \le m-d+1\}$ in d-dimensional space, some of which are used as seeds in the mapping step. Therefore we need a data structure to organize points of the reference sequence in d-dimensional space so that given a query point p^r of the read, we can efficiently retrieve points $p_{i_1}^s, p_{i_2}^s, \dots$ of the sequence near p^r, i.e., $\left\| p^r - p_{i_j}^s \right\|_2 \le \epsilon$ where ϵ is the threshold for this range search.

The k-d tree (Bentley, 1975) is a data structure designed for partitioning space and organizing points with a binary tree. The leaf nodes of the tree are points while every non-leaf node implicitly divides a subspace into two parts by a hyperplane within that subspace. The points on either side of this hyperplane are associated with the left/right subtrees, respectively. In a balanced k-d tree, the time complexity of range search is $O(dn^{1-\frac{1}{d}})$ in worst case for a fixed range size (Lee and Wong, 1977). But in practice, this typically takes $O(\log n + 2^d)$ time, where logarithmic time is spent in finding the nodes "near" the query point and $O(2^d)$ time is spent to explore their neighborhoods. Therefore we use the k-d tree to organize points generated from the reference to handle large number of queries efficiently during mapping process. Note that construction of the index requires $O(n \log n)$ time when using an $O(n)$ median of medians algorithm (Cormen *et al.*, 2009), and the index only needs to be built once prior to mapping. In the implementation, we used the highly-optimized k-d tree package nanoflann (`https://github.com/jlblancoc/nanoflann`), which supports k-d tree construction and queries.

2.2 Signal pre-processing

There are two signal pre-processing steps: signal segmentation and normalization. For R9.4 pore, the DNA molecule transits through the pore with an average speed of 450bp/s and the electric current is sampled at 4kHz, which means on average each k-mer has around 8 current samples. The purpose of signal segmentation is to collapse the current readings of the same k-mer into an event. However, speed of the molecule passing through the pore varies significantly. As a result, some k-mers may stay longer in the pore and generate more current readings (*stay errors*) while some k-mers may have no recorded current as the time they reside in the pore is too short (*skip errors*), which makes it hard to segment signals accurately. Moreover, to process signals in real time, we need a fast segmentation method.

Scrappie (`https://github.com/nanoporetech/scrappie`) is a base caller from ONT, which has a segmentation step prior to fine-grained base calling. It uses t-test over rolling window on the raw signal to detect where the current changes significantly, thereby segmenting the signal. Similar to this method, we also use the Welch's t-test to segment the signal. We choose a fixed window size w and for raw current samples in every two adjacent windows we compute the t-statistics $t = (\bar{x}_1 - \bar{x}_2)/\sqrt{(y_1^2 + y_2^2)/w}$ where $\bar{x}_i$ is the current sample mean and y_i is the current sample standard deviation in the window. Then all the local maxima and minima are identified among the computed t-statistics along the sequence. When a local extremum passes a significance threshold, its position is selected to segment the signal. Due to the various molecule transiting speeds, t-statistics should be computed using multiple window sizes. Local extrema are chosen as segmentation positions using the smallest possible window size if the local extrema reach the significance threshold of that window size. After the signal is segmented, the detected events are normalized to account for the shift or drift during sequencing.

2.3 Seeding

After reference genomes and raw signals are converted into events, the mapping problem is as follows: given read events $E(r)$ and reference events $E(s)$, find consecutive events $E_{i,j}(s) = e_i^s e_{i+1}^s \dots e_j^s$ in $E(s)$ such that $E(r)$ can be aligned to $E_{i,j}(s)$ with high confidence. Note that the mapping can be found by using subsequence dynamic time warping (sDTW) (Han *et al.*, 2020). But the time to compute DTW distance is quadratic in the length of events sequences, which is too slow to compute for long reads in real time. Since the reads are long, though they are erroneous, there are still many subsequences shared in a high confidence mapping region of the read and the reference. Taking advantage of this fact, long read aligners such as minimap2 (Li, 2018) can efficiently map reads using the seed-and-extend strategy and so does our method.

As the reference points are indexed for fast queries, we can use read points $P(r) = \{p_i^r = (e_i^r, e_{i+1}^r, \dots, e_{i+d-1}^r), 1 \le i \le m-d+1\}$ as the seeds. Note that the number of seeds (or points) needed to query the index is roughly the length of the event sequence. For real-time mapping, the reads have to be mapped within their first few hundreds of base pairs (events). Thankfully, searching for all the seeds can be completed in reasonable time. However, more seeds also lead to more hits on the reference, thereby potentially increasing the time spent in chaining the hits. For organisms like yeast, the number of hits is limited by the small genome size and fewer repetitive regions. But for larger genomes with more repetitive structures, the number of hits can increase significantly, which makes the chaining step time consuming.

To address this problem, one can select seeds with a fixed step size l and only use a subset of all the read points $P(r)$ as seeds, $P_l(r) = \{p_i^r = (e_i^r, e_{i+1}^r, \dots, e_{i+d-1}^r), 1 \le i \le m-d+1, i \mod l = 0\}$. However, raw signals are noisy, which also makes the events erroneous. Simply picking seeds with a fixed step size could miss some "error free" seeds (query points that have true hits in the index within a certain range) and reduce mapping accuracy. This problem is even more serious when mapping reads in a streaming manner, where the read is supposed to be mapped with only its first few hundreds of base pairs sequenced.

As an alternative, if the quality of the seed can be measured by a score, then error-free seeds can be preferred during seed selection procedure. Formally, we define a scoring function $g : \mathbb{R}^d \rightarrow \mathbb{R}$ which computes the score for a given point in d-dimensional space. Note that during sequencing, stay errors happen more frequently than skip errors. Affected by the noise during sequencing, stay errors result in many current samples for the same k-mer with large variance, which leads to over segmentation of the raw signal. If a seed contains stay errors, range search can fail to find true hits of the seed.

We present a method to avoid seeds that are likely to contain stay errors. For a seed (query point) $p_i^r = (e_i^r, e_{i+1}^r, \ldots, e_{i+d-1}^r)$, we define the seed scoring function as $g(p_i^r) = \sum_{j=i+1}^{i+d-1} |e_j^r - e_{j-1}^r|$, which is the sum of the differences between every pair of consecutive events in the seed. Then with step size l, top $\lceil (m-d+1)/l \rceil$ seeds are selected based on their scores. Note that seeds with more abrupt changes in their events are considered better since the segmentation is more reliable in that case.

2.4 Chaining

The time for computing an optimal alignment between two sequences is quadratic in the length of the sequences. To avoid this computational bottleneck for aligning long sequences, chaining approaches (Li, 2018) have been proposed and used to efficiently find mapping positions of long reads in large reference genomes.

Inspired by the chaining method of minimap2, we present a dynamic programming algorithm to identify a set of co-linear anchoring point matches. Formally, each seed hit (anchor) is a triple (u, v, h), which represents a read point p_u^r matching a reference point p_v^s with distance h, i.e., $\|p_u^r - p_v^s\|_2 = h$. Given a list of anchors sorted by their position on the reference, the best chaining score up to the ith anchor can be computed using the recurrence $D_i = \max\left\{\max_{1 \le j < i} \{D_j + \alpha_{ji} - \beta_{ji}\}, (1 - h_i/\epsilon)d\right\}$, where $\alpha_{ji} = (1 - h_i/\epsilon) * \min\{u_i - u_j, v_i - v_j, d\}$ is the bonus for the seed hit and β_{ji} is the gap penalty. Let $a_{ji} = |(u_i - u_j) - (v_i - v_j)|$ denote the gap length and $b_{ji} = |(u_i - u_j)/(v_i - v_j)|$ denote the gap scale. The gap penalty β_{ji} is set to ∞ when $v_i < v_j$ (ith anchor is not co-linear with the jth anchor), or gap length a_{ji} or gap scale b_{ji} is too large. Due to stay and skip errors, the gap length and scale are usually unpredictable. Hence, we do not penalize the gap as long as its length and scale are below certain thresholds. Instead, when computing the bonus α_{ji} for seed hits, we scale it down by the factor $(1 - h_i/\epsilon)$.

Note that the time of the chaining algorithm is quadratic in the number of anchors, which is slow. In practice, we use similar heuristics as in minimap2 chaining to reduce the number of anchors to examine. When computing D_i, we start the iteration from $j = i - 1$ and stop when no better chaining score is found after c iterations. For n_a anchors, this heuristic reduces the average time to $O(cn_a)$. The default c is set to the same value used in minimap2 since it led to reasonable speed and accuracy on mapping reads to various genomes empirically. There are theoretically faster chaining algorithms (Abouelhoda and Ohlebusch, 2005) but they are usually not adapted to generic gap functions, or have large hidden constants in their time complexity.

2.5 Streaming mapping

In nanopore real-time sequencing, the signal is returned in chunks, and each chunk by default is one second's worth of signal and contains 4000 current samples or roughly 450bp. We developed a streaming method to map raw signals by chunks. The signal preprocessing and seeding are performed on each chunk individually. As for chaining, the anchors in the good chains (chaining scores are at least half of the best score) generated using previous chunks are kept and used in the chaining together with the anchors in the current chunk. Each time after a chunk is processed, we compute the ratio between the best chaining score and the second best chaining score. If the ratio exceeds a certain threshold, we stop mapping more chunks and report the best chain as the mapping. By default, we set this ratio to 1.4. If this ratio cannot exceed this threshold after mapping the first 30 chunks of the read, the mapping process of this read will be stopped and the read will be reported as unmapped. These parameters can be adjusted by users to increase mapping speed or lower false positive rate based on the applications if necessary.

3 Experimental Results

We demonstrate empirically the advantages of our method on both simulated and real data sets on two different genomes. The implementation of our proposed method is termed *Sigmap*, which is available at `https://github.com/haowenz/sigmap`. We compare Sigmap with Uncalled (v2.1).

3.1 Experimental setup

3.1.1 Benchmarking data sets

We used one simulated and two real data sets to test the methods. The number of reads, N50 values, genome sizes and average coverage for these data sets are shown in Table 1. Simulated raw signals of *Saccharomyces cerevisiae* (yeast) were generated using DeepSimulator (Li *et al.*, 2020) with its context-dependent model (-M 0) and sequencing coverage set to 20x (-K 20). For real data sets, 100,000 raw reads were randomly selected from nanopore sequencing of *S. cerevisiae* using ONT R9.4 chemistry (available at NCBI under the study PRJNA510813). The first run of *Chlamydomonas reinhardtii* (green algae) nanopore sequencing using ONT R9.4 chemistry was also used (under study PRJEB31789 on EMBL-EBI) in the evaluation. Note that in real-time targeted sequencing applications, the regions of interest are usually from ∼10 Mbp to ∼100 Mbp and the coverage of target regions is around 20x (Kovaka *et al.*, 2020; Miller *et al.*, 2020). Thus in the evaluation, the yeast and green algae sequencing data were used as their genome sizes are appropriate and their whole genome sequencing data are subsampled to the proper coverage for real-time targeted sequencing applications. Besides, since Uncalled only supports R9.4 chemistry so far, we used R9.4 data in our evaluation. But with some parameter tuning for both methods, they might also be able to work on R10 data with the R10 pore model (`https://github.com/jts/nanopolish/tree/r10/etc/r10-models`) trained using Nanopolish (Simpson *et al.*, 2017).

3.1.2 Hardware and software

For all experiments, we used a compute node with dual Intel Xeon Gold 6226 CPU (2.70GHz) processors equipped with a total of 24 cores and 128GB main memory. We run Sigmap and Uncalled with all the available cores.

The k-d tree index constructed by Sigmap has two important parameters: dimension d and the maximum number of points associated with a leaf node, n_p. The empirical performance of k-d trees is usually good in low-dimensional spaces (e.g., 2D or 3D) but degrades in high-dimensional spaces as more tree branches need to be visited for each query. For this application a low d such as 2 or 3 cannot be chosen, as querying points in low-dimensional spaces usually results in too many hits, which can slow down mapping. Thus we set d to 6 by default. Since the ONT R9.4 pore model lists the expected current reading for each 6-mer, a point in the 6-dimensional space is analogous to an 11-mer, which is also a reasonable k-mer size for read mapping on genomes from tens of Mbp to several hundred Mbp. As for the other parameter, n_p controls the maximum number of points associated with a leaf node (points are stored in leaf nodes of k-d trees). A larger n_p can make the tree smaller but may cause more explorations of points during the search process and increase the query time. On the other hand, a smaller n_p may reduce the number of points to inspect for a query but increase the tree size. By default, we set n_p to 20 and studied how it can affect memory usage and mapping time on D2. Moreover, to study the effect of seeding step size on mapping time, we evaluated Sigmap with various seeding step sizes l from 2 to 6 on D3 while other parameters are set to the default. We set the maximum

Table 1. List of benchmarking data sets.

Data set	Type	Number of reads	N50 (bp)	Reference genome	Genome size (Mbp)	Avg. coverage
D1	Simulated	30,385	11,984	S. cerevisiae S288c	12.2	20x
D2	Real	100,000	8,348	S. cerevisiae S288c	12.2	58x
D3	Real	63,215	32,025	C. reinhardtii v5.5	111.1	12x

amount of chunks to use for mapping a raw signal as 30 and the search radius ϵ to 0.08 by default since they led to proper mapping accuracy and time. These parameters can be adjusted by users according to their data and applications in practice.

To test Uncalled, we used default parameters for indexing reference genomes and mapping raw signals. Kovaka *et al.* (2020) showed that masking repeats in genomes improved the mapping speed and accuracy of Uncalled. In the evaluation, we used recommended parameters and procedures stated in the Uncalled's user documentation for *C. reinhardtii* genome repeat masking.

3.1.3 Evaluation criteria

We followed a similar evaluation criteria previously used by Kovaka *et al.* (2020). Raw reads that are mapped to their true mapping locations are true positives (TP). Reads that are mapped by their raw signals but not to the correct locations are false positives (FP). Reads that have true mapping locations but are not mapped by their raw signals are false negatives (FN). Precision equals $TP/(TP+FP)$, recall equals $TP/(TP+FN)$, and F_1-score is calculated by $2*precision*recall/(precision+recall)$. The percent of correctly mapped reads is the portion of reads that are mapped to their true mapping locations.

For simulated data set D1, we evaluated the mapping accuracy against the ground truth output by the simulator. For real data sets, we mapped the base-called read sequences with the well-established long read aligner minimap2 (Li, 2018) and used the read alignments as ground truth to validate Sigmap and Uncalled. We excluded reads that are not mapped by minimap2 in the evaluation.

Moreover, we measured the mean mapping time of each read and the number of chunks used to map a read. In practical applications, mapping results are needed in real time to decide whether to eject a pore. Therefore, instead of cumulative mapping time, time spent on individual reads is an important metric to show whether most of the reads can be mapped fast enough for real-time decisions. To accurately measure the mapping time for individual reads, the mapping start time and end time of each read were recorded and the wall time for mapping each read was computed as the difference between these two values and then reported. This way of timing the mapping process for individual reads avoids the effect of loading index or the scalability of multi-thread implementation on measuring mapping time, which is a fair way to compare the two methods.

3.2 Comparison with Uncalled

We evaluated the performance of Sigmap and Uncalled on data sets D1-D3. The results on yeast genome are shown in Table 2. On the simulated data set D1, Sigmap achieved higher percentage of correctly mapped reads, precision and F_1-score while Uncalled has higher recall and faster speed. Since simulated data might not be as noisy as real data, the events were likely to be detected and converted to corresponding k-mers more reliably, which reduced the number of high-probability k-mers to explore in Uncalled and made it faster. On yeast real data set D2, 93,544 of the 100,000 reads were mapped by minimap2 and used in the evaluation. Sigmap achieved higher percent of correctly mapped reads, precision, recall and F_1-score. Notably, its speed of mapping a raw signal on average was 4.4 times faster than Uncalled.

Next, we tested Sigmap and Uncalled on the green algae real data set D3, where minimap2 mapped 60,313 out of 63,215 reads. Table 3 shows the evaluation results. We denote Sigmap run with seeding step size 3 by Sigmap (l3), etc. Since the green algae genome is much larger than the yeast genome and has more repetitive regions, genome repeat masking was performed as suggested when using Uncalled to map raw signals. After repeat masking, both mapping accuracy and mean time to map a read improved. But Sigmap significantly outperformed Uncalled with or without repeat masking on the percentage of correctly mapped reads, recall, and F_1-score, while achieving comparable precision. Moreover, compared with Uncalled with and without masking respectively, Sigmap using default parameters was 1.3 and 1.2 times faster on mapping reads, and Sigmap using seeding step size 6 was 2.6 and 2.3 times faster. Though the mapping accuracy of Sigmap degraded when increasing the seeding step size, it was overall better compared to Uncalled. The reason for this observation is that using larger seeding step size reduces the number of picked seeds that go into chaining, which would reduce chaining time and thereby reducing mapping time. But picking fewer seeds also reduced the mapping accuracy since the true mapping location would have fewer supported seeds making it harder to distinguish from other false mapping locations.

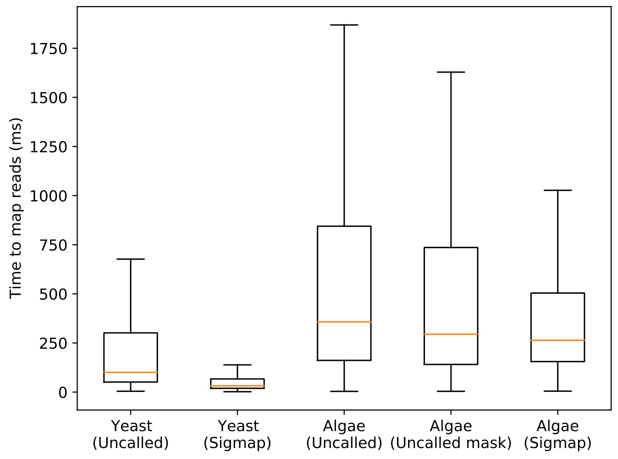

Fig. 2. Boxplots showing the mapping time distributions of Uncalled and Sigmap on mapping real reads in D2 and D3. Center lines denote the median, box limits are the quartiles and the whiskers extended from the boxes represent 5% and 95% confidence intervals.

The mapping time distributions of Uncalled and Sigmap on D2 and D3 are shown in Figure 2. We observed that overall Sigmap achieved much shorter mapping time on mapping yeast real raw reads compared with Uncalled. We noticed the speedup of mapping reads on green algae genome is not as significant as the speedup of mapping yeast reads. One

Table 2. Performance comparison between Sigmap and Uncalled on yeast genome.

Data set	Method	Correctly mapped reads (%)	TP	FP	FN	Precision (%)	Recall (%)	F1-score	Mean time per read (ms)
D1	Sigmap	**97.66**	**29675**	**7**	661	**99.98**	97.82	**0.9889**	59
	Uncalled	97.47	29615	722	**47**	97.62	**99.84**	0.9872	**18.3**
D2	Sigmap	**87.54**	**81892**	**964**	**10683**	**98.84**	**88.46**	**0.9336**	**68.3**
	Uncalled	87.37	81725	1054	10765	98.73	88.36	0.9326	303.1

Table 3. Performance comparison between Sigmap and Uncalled on green algae genome.

Data set	Method	Correctly mapped reads (%)	TP	FP	FN	Precision (%)	Recall (%)	F1-score	Mean time per read (ms)
D3	Sigmap	**87.86**	**52989**	1694	**5628**	96.90	**90.40**	**0.9354**	509.1
	Sigmap (l3)	86.21	51998	1973	6338	96.34	89.14	0.9260	373
	Sigmap (l4)	83.51	50370	2542	7397	95.20	87.20	0.9102	314.8
	Sigmap (l5)	80.69	48669	3107	8532	94.00	85.08	0.8932	279.6
	Sigmap (l6)	77.20	46564	3781	9962	92.49	82.38	0.8714	**261.2**
	Uncalled	72.18	43534	883	15896	98.01	73.25	0.8384	677
	Uncalled (mask)	76.37	46060	**881**	13372	**98.12**	77.50	0.8660	596.5

reason is that the size of green algae genome is as around 9 times larger as the size of yeast genome. Given the fact that in practice the time of k-d tree queries is usually logarithmic in the number of points (explained in section 2.1), which is roughly the size of the genome, the query time is supposed to increase accordingly. In addition, the green algae genome has more repetitive regions than the yeast genome and thus the number of signal chunks needed to map algae reads confidently on average is expected to be greater than that to map yeast reads. In the evaluation, we studied the number of chunks needed for Sigmap to map yeast and green algae reads correctly and present the results in Figure 3. We observe that using the same number of chunks, a smaller fraction of green algae reads were correctly mapped compared with yeast reads. This also indicates overall more chunks were needed to map green algae reads confidently, which increased the mapping time.

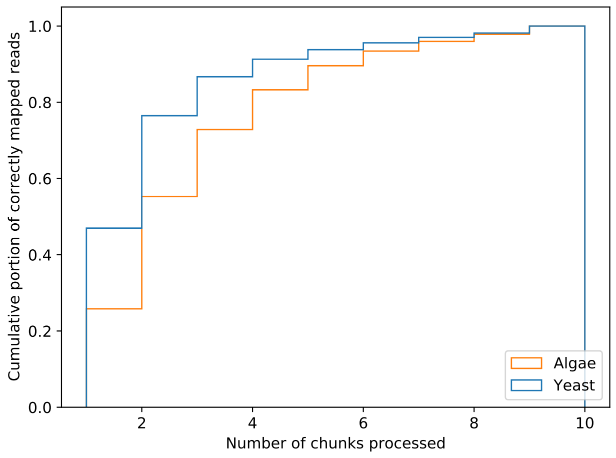

Fig. 3. Number of chunks processed by Sigmap to correctly map reals read in D2 and D3. Most of the reads were mapped using $<= 10$ chunks.

Besides mapping speed, we investigated the index size of Sigmap and Uncalled, which contributes to most of the memory usage in real-time signal mapping. Note that Uncalled mainly relies on an FM-index of the reference sequence which is a compressed full text index, hence expected to be space-efficient. The index size of the yeast genome and the green algae genome built by Uncalled is 21MB and 186MB respectively. Using default parameters, Sigmap built a 417MB index for the yeast genome and a 3.2GB index for the green algae genome, which are larger than the indices built by Uncalled but can still be accommodated on typically used computing systems.

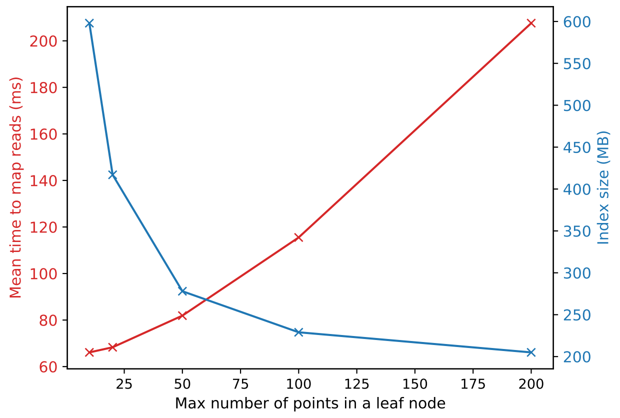

Fig. 4. Index size and mean read mapping time with respect to the maximum number of points allowed in a leaf node of the k-d tree.

As discussed in section 3.1.2, increasing the maximum number of points associated with a leaf node, n_p, can trade off mapping speed for smaller index. We studied how mean time to map reads and index size vary with different $n_p = 10, 20, 50, 100, 200$ on D2 and showed the results in Figure 4. We observed that the mean mapping time increased and the index size decreased as n_p increased and when $n_p = 200$, the index size can be reduced by a half while the average time to map a read increased by about two times. Similarly, the index size of green algae genome can be reduced to 1.8GB when setting $n_p = 200$.

Note that the space complexity of the k-d tree is linear in the number of points and the reason that Sigmap index size is large can be partly attributed to the implementation. Therefore, another possible way to reduce the index size without sacrificing mapping speed is to implement a memory efficient k-d tree customized for this application rather than using a generic k-d tree library, which is a useful direction for future work.

4 Conclusions

Mapping nanopore raw signals in real time is challenging under limited computing resources. Most mapping methods require base calling, which is computationally expensive. Uncalled is an efficient method that does not require base calling, but hits performance limitations on large genomes with higher repeat content. In this work, we introduced a new nanopore raw signal mapping method and implemented it as a tool Sigmap. On small genomes like yeast, while Sigmap has comparable performance with Uncalled on mapping simulated data, Sigmap is $4.4\times$ faster than Uncalled on mapping yeast real raw signals and has the potential to support real-time signal mapping for high-yield run ONT sequencing devices with more pores (e.g., GridION), which previous mapping methods without base calling might not be able to achieve. Sigmap also has good performance on genomes of size $>$100Mbp such as green algae, where Uncalled could not identify many correct mappings. The method avoids any conversion of signals to sequences and fully works in signal space, which holds promise for completely base-calling-free nanopore sequencing data analysis.

We envision two directions for future research. First, we intend to accelerate Sigmap by utilizing CPU SIMD instruction sets or GPUs so that it can scale to support real-time sequencing on GridION or PromethION which has even more pores. Second, we plan to study whether Sigmap can be adapted to map RNA nanopore raw signals. This may require the development of new seeding and chaining methods that are suitable to the characteristics of direct RNA-sequencing.

Funding

This work is supported in part by the US National Science Foundation under CCF-1816027, the National Human Genome Research Institute grant R01HG010040, and the National Institutes of Health grant R01HG008759. It is also supported in part by an institutional fund of the Department of Biomedical Informatics, The Ohio State University, and an institutional fund of the Department of Internal Medicine, University of Iowa.

References

Abouelhoda, M. I. and Ohlebusch, E. (2005). Chaining algorithms for multiple genome comparison. *Journal of Discrete Algorithms*, **3**(2-4), 321–341.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, **18**(9), 509–517.

Chaisson, M. J. and Tesler, G. (2012). Mapping single molecule sequencing reads using basic local alignment with successive refinement (blasr): application and theory. *BMC bioinformatics*, **13**(1), 238.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

Edwards, H. S., Krishnakumar, R., Sinha, A., Bird, S. W., Patel, K. D., and Bartsch, M. S. (2019). Real-time selective sequencing with rubric: Read until with basecall and reference-informed criteria. *Scientific Reports*, **9**(1), 1–11.

Ferragina, P. and Manzini, G. (2005). Indexing compressed text. *Journal of the ACM (JACM)*, **52**(4), 552–581.

Garalde, D. R., Snell, E. A., Jachimowicz, D., Sipos, B., Lloyd, J. H., Bruce, M., Pantic, N., Admassu, T., James, P., Warland, A., *et al.* (2018). Highly parallel direct rna sequencing on an array of nanopores. *Nature methods*, **15**(3), 201.

Gilpatrick, T., Lee, I., Graham, J. E., Raimondeau, E., Bowen, R., Heron, A., Downs, B., Sukumar, S., Sedlazeck, F. J., and Timp, W. (2020). Targeted nanopore sequencing with cas9-guided adapter ligation. *Nature biotechnology*, **38**(4), 433–438.

Han, R., Wang, S., and Gao, X. (2020). Novel algorithms for efficient subsequence searching and mapping in nanopore raw signals towards targeted sequencing. *Bioinformatics*, **36**(5), 1333–1343.

Kovaka, S., Fan, Y., Ni, B., Timp, W., and Schatz, M. C. (2020). Targeted nanopore sequencing by real-time mapping of raw electrical signal with uncalled. *Nature Biotechnology*, pages 1–11.

Lee, D.-T. and Wong, C. (1977). Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Informatica*, **9**(1), 23–29.

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

Li, Y., Wang, S., Bi, C., Qiu, Z., Li, M., and Gao, X. (2020). Deepsimulator1.5: a more powerful, quicker and lighter simulator for nanopore sequencing. *Bioinformatics*, **36**(8), 2578–2580.

Loose, M., Malla, S., and Stout, M. (2016). Real-time selective sequencing using nanopore technology. *Nature methods*, **13**(9), 751–754.

Miga, K. H., Koren, S., Rhie, A., Vollger, M. R., Gershman, A., Bzikadze, A., Brooks, S., Howe, E., Porubsky, D., Logsdon, G. A., *et al.* (2020). Telomere-to-telomere assembly of a complete human x chromosome. *Nature*, **585**(7823), 79–84.

Miller, D. E., Sulovari, A., Wang, T., Loucks, H., Hoekzema, K., Munson, K. M., Lewis, A. P., Fuerte, E. P. A., Paschal, C. R., Thies, J., *et al.* (2020). Targeted long-read sequencing resolves complex structural variants and identifies missing disease-causing variants. *bioRxiv*.

Payne, A., Holmes, N., Clarke, T., Munro, R., Debebe, B. J., and Loose, M. (2020). Readfish enables targeted nanopore sequencing of gigabase-sized genomes. *Nature Biotechnology*, pages 1–9.

Quick, J., Loman, N. J., Duraffour, S., Simpson, J. T., Severi, E., Cowley, L., Bore, J. A., Koundouno, R., Dudas, G., Mikhail, A., *et al.* (2016). Real-time, portable genome sequencing for ebola surveillance. *Nature*, **530**(7589), 228–232.

Rang, F. J., Kloosterman, W. P., and de Ridder, J. (2018). From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. *Genome biology*, **19**(1), 90.

Sedlazeck, F. J., Rescheneder, P., Smolka, M., Fang, H., Nattestad, M., Von Haeseler, A., and Schatz, M. C. (2018). Accurate detection of complex structural variations using single-molecule sequencing. *Nature methods*, **15**(6), 461–468.

Simpson, J. T., Workman, R. E., Zuzarte, P., David, M., Dursi, L., and Timp, W. (2017). Detecting dna cytosine methylation using nanopore sequencing. *Nature methods*, **14**(4), 407–410.

Sović, I., Šikić, M., Wilm, A., Fenlon, S. N., Chen, S., and Nagarajan, N. (2016). Fast and sensitive mapping of nanopore sequencing reads with graphmap. *Nature communications*, **7**(1), 1–11.

Wang, M., Fu, A., Hu, B., Tong, Y., Liu, R., Liu, Z., Gu, J., Xiang, B., Liu, J., Jiang, W., *et al.* (2020). Nanopore targeted sequencing for the accurate and comprehensive detection of sars-cov-2 and other respiratory viruses. *Small*, **16**(32), 2002169.