# Minimap2: pairwise alignment for nucleotide sequences

Heng Li

Broad Institute, 415 Main Street, Cambridge, MA 02142, USA

**ABSTRACT**

**Motivation:** Recent advances in sequencing technologies promise ultra-long reads of ∼100 kilo bases (kb) in average, full-length mRNA or cDNA reads in high throughput and genomic contigs over 100 mega bases (Mb) in length. Existing alignment programs are unable or inefficient to process such data at scale, which presses for the development of new alignment algorithms.

**Results:** Minimap2 is a general-purpose alignment program to map DNA or long mRNA sequences against a large reference database. It works with accurate short reads of ≥100bp in length, ≥1kb genomic reads at error rate ∼15%, full-length noisy Direct RNA or cDNA reads, and assembly contigs or closely related full chromosomes of hundreds of megabases in length. Minimap2 does split-read alignment, employs concave gap cost for long insertions and deletions (INDELs) and introduces new heuristics to reduce spurious alignments. It is 3–4 times as fast as mainstream short-read mappers at comparable accuracy, and is ≥30 times faster than long-read genomic or cDNA mappers at higher accuracy, surpassing most aligners specialized in one type of alignment.

**Availability and implementation:** https://github.com/lh3/minimap2

**Contact:** hengli@broadinstitute.org

## 1 INTRODUCTION

Single Molecule Real-Time (SMRT) sequencing technology and Oxford Nanopore technologies (ONT) produce reads over 10kbp in length at an error rate ∼15%. Several aligners have been developed for such data (Chaisson and Tesler, 2012; Li, 2013; Liu et al., 2016; Sović et al., 2016; Liu et al., 2017; Lin and Hsu, 2017; Sedlazeck et al., 2017). Most of them were five times as slow as mainstream short-read aligners (Langmead and Salzberg, 2012; Li, 2013) in terms of the number of bases mapped per second. We speculated there could be substantial room for speedup on the thought that 10kb long sequences should be easier to map than 100bp reads because we can more effectively skip repetitive regions, which are often the bottleneck of short-read alignment. We confirmed our speculation by achieving approximate mapping 50 times faster than BWA-MEM (Li, 2016). Suzuki and Kasahara (2018) extended our work with a fast and novel algorithm on generating base-level alignment, which in turn inspired us to develop minimap2 with added functionality.

Both SMRT and ONT have been applied to the sequencing of spliced mRNAs (RNA-seq). While traditional mRNA aligners work (Wu and Watanabe, 2005; Iwata and Gotoh, 2012), they are not optimized for long noisy sequence reads and are tens of times slower than dedicated long-read aligners. When developing minimap2 initially for aligning genomic DNA only, we realized minor modifications could enable the base algorithm to map mRNAs as well. Minimap2 becomes a first RNA-seq aligner specifically designed for long noisy reads. We have also extended the original algorithm to map short reads at a speed faster than several mainstream short-read mappers.

In this article, we will describe the minimap2 algorithm and its applications to different types of input sequences. We will evaluate the performance and accuracy of minimap2 on several simulated and real data sets and demonstrate the versatility of minimap2.

## 2 METHODS

Minimap2 follows a typical seed-chain-align procedure as is used by most full-genome aligners. It collects minimizers (Roberts et al., 2004) of the reference sequences and indexes them in a hash table, with the key being the hash of a minimizer and the value being a list of locations of the minimizer copies. Then for each query sequence, minimap2 takes query minimizers as *seeds*, finds exact matches (i.e. *anchors*) to the reference, and identifies sets of colinear anchors as *chains*. If base-level alignment is requested, minimap2 applies dynamic programming (DP) to extend from the ends of chains and to close regions between adjacent anchors in chains.

Minimap2 uses indexing and seeding algorithms similar to minimap (Li, 2016), and furthers the predecessor with more accurate chaining, the ability to produce base-level alignment and the support of spliced alignment.

### 2.1 Chaining

#### *2.1.1 Chaining*

An *anchor* is a 3-tuple $(x, y, w)$, indicating interval $[x - w + 1, x]$ on the reference matching interval $[y - w + 1, y]$ on the query. Given a list of anchors sorted by ending reference position $x$, let $f(i)$ be the maximal chaining score up to the $i$-th anchor in the list. $f(i)$ can be calculated with dynamic programming:

$$f(i) = \max\Big\{ \max_{i>j\geq 1}\{f(j) + \alpha(j,i) - \beta(j,i)\}, w_i\Big\} \tag{1}$$

where $\alpha(j,i) = \min\big\{ \min\{y_i - y_j, x_i - x_j\}, w_i\big\}$ is the number of matching bases between the two anchors. $\beta(j,i) > 0$ is the gap cost. It equals $\infty$ if $y_j \geq y_i$ or $\max\{y_i - y_j, x_i - x_j\} > G$ (i.e. the distance between two anchors is too large); otherwise

$$\beta(j,i) = \gamma_c\big((y_i - y_j) - (x_i - x_j)\big) \tag{2}$$

In implementation, a gap of length $l$ costs

$$\gamma_c(l) = \begin{cases} 0.01 \cdot \bar{w} \cdot |l| + 0.5\log_2|l| & (l \neq 0) \\ 0 & (l = 0) \end{cases}$$

where $\bar{w}$ is the average seed length. For $N$ anchors, directly computing all $f(\cdot)$ with Eq. (1) takes $O(N^2)$ time. Although theoretically faster chaining algorithms exist (Abouelhoda and Ohlebusch, 2005), they are inapplicable to generic gap cost, complex to implement and usually associated with a large constant. We introduced a simple heuristic to accelerate chaining.

We note that if anchor $i$ is chained to $j$, chaining $i$ to a predecessor of $j$ is likely to yield a lower score. When evaluating Eq. (1), we start from anchor $i-1$ and stop the process if we cannot find a better score after up to $h$ iterations. This approach reduces the average time to $O(hN)$. In practice, we can almost always find the optimal chain with $h = 50$; even if the heuristic fails, the optimal chain is often close.

#### *2.1.2 Backtracking*

Let $P(i)$ be the index of the best predecessor of anchor $i$. It equals 0 if $f(i) = w_i$ or $\text{argmax}_j\{f(j) + \alpha(j,i) - \beta(j,i)\}$ otherwise. For each anchor $i$ in the descending order of $f(i)$, we apply $P(\cdot)$ repeatedly to find its predecessor and mark each visited $i$ as 'used', until $P(i) = 0$ or we reach an already 'used' $i$. This way we find all chains with no anchors used in more than one chains.

#### *2.1.3 Identifying primary chains*

In the absence of copy number changes, each query segment should not be mapped to two places in the reference. However, chains found at the

previous step may have significant or complete overlaps due to repeats in the reference (Li and Durbin, 2010). Minimap2 used the following procedure to identify *primary chains* that do not greatly overlap on the query.

Let $Q$ be an empty set initially. For each chain from the best to the worst according to their chaining scores: if on the query, the chain overlaps with a chain in $Q$ by 50% or higher percentage of the shorter chain, mark the chain as secondary to the chain in $Q$; otherwise, add the chain to $Q$. In the end, $Q$ contains all the primary chains. We did not choose a more sophisticated data structure (e.g. range tree or k-d tree) because this step is not the performance bottleneck.

For each primary chain, minimap2 estimates its mapping quality with an empirical formula:

$$\text{mapQ} = 40 \cdot (1 - f_2/f_1) \cdot \min\{1, m/10\} \cdot \log f_1$$

where log denotes natural logarithm, $m$ is the number of anchors on the primary chain, $f_1$ is the chaining score, and $f_2 \le f_1$ is the score of the best chain that is secondary to the primary chain. Intuitively, a chain is assigned to a higher mapping quality if it is long and its best secondary chain is weak.

#### 2.1.4 *Estimating per-base sequence divergence*

Suppose a query sequence harbors $n$ seeds of length $k$, $m$ of which are present in a chain. We want to estimate the sequence divergence $\epsilon$ between the query and the reference sequences in the chain. This is useful when base-level alignment is too expensive to perform.

If we model substitutions with a homogeneous Poisson process along the query sequence, the probablity of seeing $k$ consecutive bases without substitutions is $e^{-k\epsilon}$. On the assumption that all $k$-mers are independent of each other, the likelihood function of $\epsilon$ is

$$\mathcal{L}(\epsilon|n, m, k) = e^{-m\cdot k\epsilon}(1 - e^{-k\epsilon})^{n-m}$$

The maximum likelihood estimate of $\epsilon$ is

$$\hat{\epsilon} = \frac{1}{k} \log \frac{n}{m}$$

In reality, sequencing errors are sometimes clustered and $k$-mers are not independent of each other, especially when we take minimizers as seeds. These violate the assumptions in the derivation above. As a result, $\hat{\epsilon}$ is only approximate and can be biased. It also ignores long deletions from the reference sequence. In practice, fortunately, $\hat{\epsilon}$ is often close to and strongly correlated with the sequence divergence estimated from base-level alignments. On the several datasets used in Section 3.1, the Spearman correlation coefficient is around 0.9.

#### 2.1.5 *Indexing with homopolymer compressed k-mers*

SmartDenovo (https://github.com/ruanjue/smartdenovo; J. Ruan, personal communication) indexes reads with homopolymer-compressed (HPC) $k$-mers and finds the strategy improves overlap sensitivity for SMRT reads. Minimap2 adopts the same heuristic.

The HPC string of a string $s$, denoted by $\text{HPC}(s)$, is constructed by contracting homopolymers in $s$ to a single base. An HPC $k$-mer of $s$ is a $k$-long substring of $\text{HPC}(s)$. For example, suppose $s =$ `GGATTTTCCA`, $\text{HPC}(s) =$ `GATCA` and the first HPC 4-mer is `GATC`.

To demonstrate the effectiveness of HPC $k$-mers, we performed read overlapping for the example *E. coli* SMRT reads from PBcR (Berlin et al., 2015), using different types of $k$-mers. With normal 15bp minimizers per 5bp window, minimap2 finds 90.9% of $\geq$2kb overlaps inferred from the read-to-reference alignment. With HPC 19-mers per 5bp window, minimap2 finds 97.4% of overlaps. It achieves this higher sensitivity by indexing 1/3 fewer minimizers, which further helps performance. HPC-based indexing reduces the sensitivity for current ONT reads, though.

### 2.2 Aligning genomic DNA

#### 2.2.1 *Alignment with 2-piece affine gap cost*

Minimap2 performs DP-based global alignment between adjacent anchors in a chain. It uses a 2-piece affine gap cost (Gotoh, 1990):

$$\gamma_a(l) = \min\{q + |l| \cdot e, \tilde{q} + |l| \cdot \tilde{e}\} \tag{3}$$

Without losing generality, we always assume $q+e < \tilde{q}+\tilde{e}$. On the condition that $e > \tilde{e}$, it applies cost $q + |l| \cdot e$ to gaps shorter than $\lceil (\tilde{q} - q)/(e - \tilde{e}) \rceil$ and applies $\tilde{q} + |l| \cdot \tilde{e}$ to longer gaps. This scheme helps to recover longer insertions and deletions (INDELs).

The equation to compute the optimal alignment under $\gamma_a(\cdot)$ is

$$\begin{cases} H_{ij} = \max\{H_{i-1,j-1} + s(i,j), E_{ij}, F_{ij}, \tilde{E}_{ij}, \tilde{F}_{ij}\} \\ E_{i+1,j} = \max\{H_{ij} - q, E_{ij}\} - e \\ F_{i,j+1} = \max\{H_{ij} - q, F_{ij}\} - e \\ \tilde{E}_{i+1,j} = \max\{H_{ij} - \tilde{q}, \tilde{E}_{ij}\} - \tilde{e} \\ \tilde{F}_{i,j+1} = \max\{H_{ij} - \tilde{q}, \tilde{F}_{ij}\} - \tilde{e} \end{cases} \tag{4}$$

where $s(i, j)$ is the score between the $i$-th reference base and $j$-th query base. Eq. (4) is a natural extension to the equation under affine gap cost (Gotoh, 1982; Altschul and Erickson, 1986).

#### 2.2.2 *The Suzuki-Kasahara formulation*

When we allow gaps longer than several hundred base pairs, nucleotide-level alignment is much slower than chaining. SSE acceleration is critical to the performance of minimap2. Traditional SSE implementations (Farrar, 2007) based on Eq. (4) can achieve 16-way parallelization for short sequences, but only 4-way parallelization when the peak alignment score reaches 32767. Long sequence alignment may exceed this threshold. Inspired by Wu et al. (1996) and the following work, Suzuki and Kasahara (2018) proposed a difference-based formulation that lifted this limitation. In case of 2-piece gap cost, define

$$\begin{cases} u_{ij} \triangleq H_{ij} - H_{i-1,j} & v_{ij} \triangleq H_{ij} - H_{i,j-1} \\ x_{ij} \triangleq E_{i+1,j} - H_{ij} & \tilde{x}_{ij} \triangleq \tilde{E}_{i+1,j} - H_{ij} \\ y_{ij} \triangleq F_{i,j+1} - H_{ij} & \tilde{y}_{ij} \triangleq \tilde{F}_{i,j+1} - H_{ij} \end{cases}$$

We can transform Eq. (4) to

$$\begin{cases} z_{ij} &= \max\{s(i,j), x_{i-1,j} + v_{i-1,j}, y_{i,j-1} + u_{i,j-1}, \\ & \quad \tilde{x}_{i-1,j} + v_{i-1,j}, \tilde{y}_{i,j-1} + u_{i,j-1}\} \\ u_{ij} &= z_{ij} - v_{i-1,j} \\ v_{ij} &= z_{ij} - u_{i,j-1} \\ x_{ij} &= \max\{0, x_{i-1,j} + v_{i-1,j} - z_{ij} + q\} - q - e \\ y_{ij} &= \max\{0, y_{i,j-1} + u_{i,j-1} - z_{ij} + q\} - q - e \\ \tilde{x}_{ij} &= \max\{0, \tilde{x}_{i-1,j} + v_{i-1,j} - z_{ij} + \tilde{q}\} - \tilde{q} - \tilde{e} \\ \tilde{y}_{ij} &= \max\{0, \tilde{y}_{i,j-1} + u_{i,j-1} - z_{ij} + \tilde{q}\} - \tilde{q} - \tilde{e} \end{cases} \tag{5}$$

where $z_{ij}$ is a temporary variable that does not need to be stored.

An important property of Eq. (5) is that all values are bounded by scoring parameters. To see that,

$$x_{ij} = E_{i+1,j} - H_{ij} = \max\{-q, E_{ij} - H_{ij}\} - e$$

With $E_{ij} \le H_{ij}$, we have

$$-q - e \le x_{ij} \le \max\{-q, 0\} - e = -e$$

and similar inequations for $y_{ij}$, $\tilde{x}_{ij}$ and $\tilde{y}_{ij}$. In addition,

$$u_{ij} = z_{ij} - v_{i-1,j} \ge \max\{x_{i-1,j}, \tilde{x}_{i-1,j}\} \ge -q - e$$

As the maximum value of $z_{ij} = H_{ij} - H_{i-1,j-1}$ is $M$, the maximal matching score, we can derive

$$u_{ij} \le M - v_{i-1,j} \le M + q + e$$

In conclusion, in Eq. (5), $x$ and $y$ are bounded by $[-q - e, -e]$, $\tilde{x}$ and $\tilde{y}$ by $[-\tilde{q} - \tilde{e}, -\tilde{e}]$, and $u$ and $v$ by $[-q - e, M + q + e]$. When $-128 \le -q - e < M + q + e \le 127$, each of them can be stored as a 8-bit integer. This enables 16-way SSE vectorization regardless of the peak score of the alignment.

For a more efficient SSE implementation, we transform the row-column coordinate to the diagonal-antidiagonal coordinate by letting $r \leftarrow i + j$ and

$t \leftarrow i$. Eq. (5) becomes:

$$\begin{cases} z_{rt} &= \max\{s(t,r-t), x_{r-1,t-1}+v_{r-1,t-1}, y_{r-1,t} \\ & \quad +u_{r-1,t}, \tilde{x}_{r-1,t-1}+v_{r-1,t-1}, \tilde{y}_{r-1,t}+u_{r-1,t}\} \\ u_{rt} &= z_{rt}-v_{r-1,t-1} \\ v_{rt} &= z_{rt}-u_{r-1,t} \\ x_{rt} &= \max\{0, x_{r-1,t-1}+v_{r-1,t-1}-z_{rt}+q\}-q-e \\ y_{rt} &= \max\{0, y_{r-1,t}+u_{r-1,t}-z_{rt}+q\}-q-e \\ \tilde{x}_{rt} &= \max\{0, \tilde{x}_{r-1,t-1}+v_{r-1,t-1}-z_{rt}+\tilde{q}\}-\tilde{q}-\tilde{e} \\ \tilde{y}_{rt} &= \max\{0, \tilde{y}_{r-1,t}+u_{r-1,t}-z_{rt}+\tilde{q}\}-\tilde{q}-\tilde{e} \end{cases}$$

In this formulation, cells with the same diagonal index $r$ are independent of each other. This allows us to fully vectorize the computation of all cells on the same anti-diagonal in one inner loop. It also simplifies banded alignment (500bp band width by default), which would be difficult with striped vectorization (Farrar, 2007).

On the condition that $q+e < \tilde{q}+\tilde{e}$ and $e > \tilde{e}$, the initial values in the diagonal-antidiagonal formuation are

$$\begin{cases} x_{r-1,-1} = y_{r-1,r} = -q-e \\ \tilde{x}_{r-1,-1} = \tilde{y}_{r-1,r} = -\tilde{q}-\tilde{e} \\ u_{r-1,r} = v_{r-1,-1} = \eta(r) \end{cases}$$

where

$$\eta(r) = \begin{cases} -q-e & (r=0) \\ -e & (r < \lceil \frac{\tilde{q}-q}{e-\tilde{e}} - 1 \rceil) \\ r\cdot(e-\tilde{e})-(\tilde{q}-q)-\tilde{e} & (r = \lceil \frac{\tilde{q}-q}{e-\tilde{e}} - 1 \rceil) \\ -\tilde{e} & (r > \lceil \frac{\tilde{q}-q}{e-\tilde{e}} - 1 \rceil) \end{cases}$$

These can be derived from the initial values for Eq. (4).

When performing global alignment, we do not need to compute $H_{rt}$ in each cell. We use 16-way vectorization throughout the alignment process. When extending alignments from ends of chains, we need to find the cell $(r,t)$ where $H_{rt}$ reaches the maximum. We resort to 4-way vectorization to compute $H_{rt} = H_{r-1,t} + u_{rt}$. Because this computation is simple, Eq. (5) is still the dominant performance bottleneck.

In practice, our 16-way vectorized implementation of global alignment is three times as fast as Parasail's 4-way vectorization (Daily, 2016). Without banding, our implementation is slower than Edlib (Šošić and Šikic, 2017), but with a 1000bp band, it is considerably faster. When performing global alignment between anchors, we expect the alignment to stay close to the diagonal of the DP matrix. Banding is applicable most of the time.

#### *2.2.3 The Z-drop heuristic*

With global alignment, minimap2 may force to align unrelated sequences between two adjacent anchors. To avoid such an artifact, we compute accumulative alignment score along the alignment path and break the alignment where the score drops too fast in the diagonal direction. More precisely, let $S(i,j)$ be the alignment score along the alignment path ending at cell $(i,j)$ in the DP matrix. We break the alignment if there exist $(i',j')$ and $(i,j)$, $i'<i$ and $j'<j$, such that

$$S(i',j') - S(i,j) > Z + e\cdot|(i-i')-(j-j')|$$

where $e$ is the gap extension cost and $Z$ is an arbitrary threshold. This strategy is first used in BWA-MEM. It is similar to X-drop employed in BLAST (Altschul et al., 1997), but unlike X-drop, it would not break the alignment in the presence of a single long gap.

When minimap2 breaks a global alignment between two anchors, it performs local alignment between the two subsequences involved in the global alignment, but this time with the one subsequence reverse complemented. This additional alignment step may identify short inversions that are missed during chaining.

#### *2.2.4 Filtering out misplaced anchors*

Due to sequencing errors and local homology, some anchors in a chain may be wrong. If we blindly align regions between two misplaced anchors, we will produce a suboptimal alignment. To reduce this artifact, we filter out anchors that lead to a >10bp insertion and a >10bp deletion at the same time, and filter out terminal anchors that lead to a long gap towards the ends of a chain. These heuristics greatly alleviate the issues with misplaced anchors, but they are unable to fix all such errors. Local misalignment is a limitation of minimap2 which we hope to address in future.

### 2.3 Aligning spliced sequences

The algorithm described above can be adapted to spliced alignment. In this mode, the chaining gap cost distinguishes insertions to and deletions from the reference: $\gamma_c(l)$ in Eq. (2) takes the form of

$$\gamma_c(l) = \begin{cases} 0.01\cdot\bar{w}\cdot l + 0.5\log_2 l & (l>0) \\ \min\{0.01\cdot\bar{w}\cdot|l|, \log_2|l|\} & (l<0) \end{cases}$$

Similarly, the gap cost function used for DP-based alignment is changed to

$$\gamma_a(l) = \begin{cases} q+l\cdot e & (l>0) \\ \min\{q+|l|\cdot e, \tilde{q}\} & (l<0) \end{cases}$$

In alignment, a deletion no shorter than $\lceil(\tilde{q}-q)/e\rceil$ is regarded as an intron, which pays no cost to gap extensions.

To pinpoint precise splicing junctions, minimap2 introduces reference-dependent cost to penalize non-canonical splicing:

$$\begin{cases} H_{ij} = \max\{H_{i-1,j-1}+s(i,j), E_{ij}, F_{ij}, \tilde{E}_{ij}-a(i)\} \\ E_{i+1,j} = \max\{H_{ij}-q, E_{ij}\}-e \\ F_{i,j+1} = \max\{H_{ij}-q, F_{ij}\}-e \\ \tilde{E}_{i+1,j} = \max\{H_{ij}-d(i)-\tilde{q}, \tilde{E}_{ij}\} \end{cases} \quad (6)$$

Let $T$ be the reference sequence. $d(i)$ is computed as

$$d(i) = \begin{cases} 0 & \text{if } T[i+1,i+3] \text{ is } \texttt{GTA} \text{ or } \texttt{GTG} \\ p/2 & \text{if } T[i+1,i+3] \text{ is } \texttt{GTC} \text{ or } \texttt{GTT} \\ p & \text{otherwise} \end{cases}$$

where $T[i,j]$ extracts a substring of $T$ between $i$ and $j$ inclusively. $d(i)$ penalizes non-canonical donor sites with $p$ and less frequent Eukaryotic splicing signal `GT[C/T]` with $p/2$ (Irimia and Roy, 2008). Similarly,

$$a(i) = \begin{cases} 0 & \text{if } T[i-2,i] \text{ is } \texttt{CAG} \text{ or } \texttt{TAG} \\ p/2 & \text{if } T[i-2,i] \text{ is } \texttt{AAG} \text{ or } \texttt{GAG} \\ p & \text{otherwise} \end{cases}$$

models the acceptor signal. Eq. (6) is close to an equation in Zhang and Gish (2006) except that we allow insertions immediately followed by deletions and vice versa; in addition, we use the Suzuki-Kasahara diagonal formulation in actual implementation.

If RNA-seq reads are not sequenced from stranded libraries, the read strand relative to the underlying transcript is unknown. By default, minimap2 aligns each chain twice, first assuming `GT–AG` as the splicing signal and then assuming `CT–AC`, the reverse complement of `GT–AG`, as the splicing signal. The alignment with a higher score is taken as the final alignment. This procedure also infers the relative strand of reads that span canonical splicing sites.

In the spliced alignment mode, minimap2 further increases the density of minimizers and disables banded alignment. Together with the two-round DP-based alignment, spliced alignment is several times slower than genomic DNA alignment.

### 2.4 Aligning short paired-end reads

During chaining, minimap2 takes a pair of reads as one fragment with a gap of unknown length in the middle. It applies a normal gap cost between seeds on the same read but is a more permissive gap cost between seeds on different reads. More precisely, the gap cost during chaining is ($l \neq 0$):

$$\gamma_c(l) = \begin{cases} 0.01\cdot\bar{w}\cdot|l| + 0.5\log_2|l| & \text{if two seeds on the same read} \\ \min\{0.01\cdot\bar{w}\cdot|l|, \log_2|l|\} & \text{otherwise} \end{cases}$$

After identifying primary chains (Section 2.1.3), we split each fragment chain into two read chains and perform alignment for each read as in

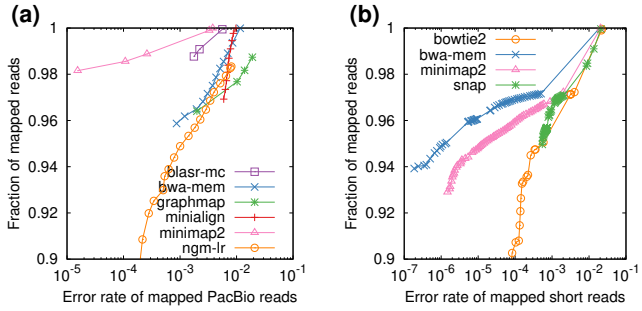



**Fig. 1.** Evaluation on aligning simulated reads. Simulated reads were mapped to the primary assembly of human genome GRCh38. A read is considered correctly mapped if its longest alignment overlaps with the true interval, and the overlap length is ≥10% of the true interval length. Read alignments are sorted by mapping quality in the descending order. For each mapping quality threshold, the fraction of alignments (out of the number of input reads) with mapping quality above the threshold and their error rate are plotted along the curve. (a) long-read alignment evaluation. 33,088 ≥1000bp reads were simulated using pbsim (Ono et al., 2013) with error profile sampled from file 'm131017_060208_42213_*.1.*' downloaded at http://bit.ly/chm1p5c3. The N50 read length is 11,628. Aligners were run under the default setting for SMRT reads. Kart outputted all alignments at mapping quality 60, so is not shown in the figure. It mapped nearly all reads with 4.1% of alignments being wrong, less accurate than others. (b) short-read alignment evaluation. 10 million pairs of 150bp reads were simulated using mason2 (Holtgrewe, 2010) with option '–illumina-prob-mismatch-scale 2.5'. Short-read aligners were run under the default setting except for changing the maximum fragment length to 800bp.

Section 2.2. Finally, we pair hits of each read end to find consistent paired-end alignments.

## 3 RESULTS

Minimap2 is implemented in the C programming language and comes with APIs in both C and Python. It is distributed under the MIT license, free to both commercial and academic uses. Minimap2 uses the same base algorithm for all applications, but it has to apply different sets of parameters depending on input data types. Similar to BWA-MEM, minimap2 introduces 'presets' that modify multiple parameters with a simple invocation. Detailed settings and command-line options can be found in the minimap2 manpage. In addition to the applications evaluated in the following sections, minimap2 also retains minimap's functionality to find overlaps between long reads and to search against large multi-species databases such as *nt* from NCBI.

### 3.1 Aligning long genomic reads

As a sanity check, we evaluated minimap2 on simulated human reads along with BLASR (v1.MC.rc64; Chaisson and Tesler, 2012), BWA-MEM (v0.7.15; Li, 2013), GraphMap (v0.5.2; Sović et al., 2016), Kart (v2.2.5; Lin and Hsu, 2017), minialign (v0.5.3; https://github.com/ocxtal/minialign) and NGMLR (v0.2.5; Sedlazeck et al., 2017). We excluded rHAT (Liu et al., 2016) and LAMSA (Liu et al., 2017) because they either crashed or produced malformatted output. In this evaluation, minimap2 has higher power to distinguish unique and repetitive hits, and achieves overall higher mapping accuracy (Fig. 1a). Minimap2 and NGMLR provide better mapping quality estimate: they rarely give repetitive hits high mapping quality. Apparently, other aligners may occasionally miss close suboptimal hits and be overconfident in wrong mappings. On run time, minimap2 took 200 CPU seconds, comparable to minialign and Kart, and is over 30 times faster than the rest. Minimap2 consumed 6.8GB memory at the peak, more than BWA-MEM (5.4GB), similar to NGMLR and less than others.

On real human SMRT reads, the relative performance and fraction of mapped reads reported by these aligners are broadly similar to the metrics on simulated data. We are unable to provide a good estimate of mapping error rate due to the lack of the truth. On ONT ∼100kb human reads (Jain et al., 2017), BWA-MEM failed. Kart, minialign and minimap2 are over 70 times faster than others. We have also examined tens of ≥100bp INDELs in IGV (Robinson et al., 2011) and can confirm the observation by Sedlazeck et al. (2017) that BWA-MEM often breaks them into shorter gaps. The issue is much alleviated with minimap2, thanks to the 2-piece affine gap cost.

**Table 1.** Evaluation of junction accuracy on 2D ONT reads

| | GMAP | minimap2 | SpAln | STAR |
|---|---|---|---|---|
| Run time (CPU min) | 631 | 15.9 | 2 076 | 33.9 |
| Peak RAM (GByte) | 8.9 | 14.5 | 3.2 | 29.2 |
| # aligned reads | 103 669 | 104 199 | 103 711 | 26 479 |
| # chimeric alignments | 1 904 | 1 488 | 0 | 0 |
| # non-spliced alignments | 15 854 | 14 798 | 17 033 | 10 545 |
| # aligned introns | 692 275 | 693 553 | 692 945 | 78 603 |
| # novel introns | 11 239 | 3 113 | 8 550 | 1 214 |
| % exact introns | 83.8% | 94.0% | 87.9% | 55.2% |
| % approx. introns | 91.8% | 96.9% | 92.5% | 82.4% |

Mouse cDNA reads (AC:SRR5286960; R9.4 chemistry) were mapped to the primary assembly of mouse genome GRCm38 with the following tools and command options: minimap2 ('-ax splice'); GMAP ('-n 0 –min-intronlength 30 –cross-species'); SpAln ('-Q7 -LS -S3'); STARlong (according to http://bit.ly/star-pb). The alignments were compared to the EnsEMBL gene annotation, release 89. A predicted intron is *novel* if it has no overlaps with any annotated introns. An intron is *exact* if it is identical to an annotated intron. An intron is *approximate* if both its 5'- and 3'-end are within 10bp around the ends of an annotated intron. Chimeric alignments are defined in the SAM spec (Li et al., 2009).

### 3.2 Aligning long spliced reads

We evaluated minimap2 on SIRV control data (AC:SRR5286959; Byrne et al., 2017) where the truth is known. Minimap2 predicted 59 918 introns from 11 018 reads. 93.8% of splice junctions are precise. We examined wrongly predicted junctions and found the majority were caused by clustered splicing signals (e.g. two adjacent `GT` sites). When INDEL sequencing errors are frequent, it is difficult to find precise splicing sites in this case. If we allow up to 10bp distance from true splicing sites, 98.4% of aligned introns are approximately correct. It is worth noting that for SIRV, we asked minimap2 to model the `GT..AG` splicing signal only without extra bases. This is because SIRV does not honor the evolutionarily prevalent signal `GT[A/G]..[C/T]AG` (Irimia and Roy, 2008).

We next aligned real mouse reads (Byrne et al., 2017) with GMAP (v2017-06-20; Wu and Watanabe, 2005), minimap2,

SpAln (v2.3.1; Iwata and Gotoh, 2012) and STAR (v2.5.3a; Dobin et al., 2013). In general, minimap2 is more consistent with existing annotations (Table 1): it finds more junctions with a higher percentage being exactly or approximately correct. Minimap2 is over 40 times faster than GMAP and SpAln. While STAR is close to minimap2 in speed, it does not work well with noisy reads.

We have also evaluated spliced aligners on a human Nanopore Direct RNA-seq dataset (http://bit.ly/na12878ont). Minimap2 aligned 10 million reads in $<$1 wall-clock hour using 16 CPU cores. 94.2% of aligned splice junctions consistent with gene annotations. In comparison, GMAP under option '-k 14 -n 0 –min-intronlength 30 –cross-species' is 160 times slower; 68.7% of GMAP junctions are found in known gene annotations. The percentage increases to 84.1% if an aligned junction within 10bp from an annotated junction is considered to be correct. On a public Iso-Seq dataset (human Alzheimer brain from http://bit.ly/isoseqpub), minimap2 is also faster at higher junction accuracy in comparison to other aligners in Table 1.

We noted that GMAP and SpAln have not been optimized for noisy reads. We are showing the best setting we have experimented, but their developers should be able to improve their accuracy further.

### 3.3 Aligning short genomic reads

We evaluated minimap2 along with Bowtie2 (v2.3.3; Langmead and Salzberg 2012), BWA-MEM and SNAP (v1.0beta23; Zaharia et al. 2011). Minimap2 is 3–4 times as fast as Bowtie2 and BWA-MEM, but is 1.3 times slower than SNAP. Minimap2 is more accurate on this simulated data set than Bowtie2 and SNAP but less accurate than BWA-MEM (Fig. 1b). Closer investigation reveals that BWA-MEM achieves a higher accuracy partly because it tries to locally align a read in a small region close to its mate. If we disable this feature, BWA-MEM becomes slightly less accurate than minimap2. We might implement a similar heuristic in minimap2 in future.

To evaluate the accuracy of minimap2 on real data, we aligned human reads (AC:ERR1341796) with BWA-MEM and minimap2, and called SNPs and small INDELs with GATK HaplotypeCaller v3.5 (Depristo et al., 2011). This run was sequenced from experimentally mixed CHM1 and CHM13 cell lines. Both of them are homozygous across the whole genome and have been *de novo* assembled with SMRT reads to high quality. This allowed us to construct an independent truth variant dataset (Li et al., 2017) for ERR1341796. In this evaluation, minimap2 has higher SNP false negative rate (FNR; 2.6% of minimap2 vs 2.3% of BWA-MEM), but fewer false positive SNPs per million bases (FPPM; 7.0 vs 8.8), similar INDEL FNR (11.2% vs 11.3%) and similar INDEL FPPM (6.4 vs 6.5). Minimap2 is broadly comparable to BWA-MEM in the context of small variant calling.

### 3.4 Aligning long-read assemblies

Minimap2 can align a SMRT assembly (AC:GCA_001297185.1) against GRCh38 in 7 minutes using 8 CPU cores, over 20 times faster than nucmer from MUMmer4 (Marçais et al., 2018). With the paftools.js script from the minimap2 package, we called 2.67 million single-base substitutions out of 2.78Gbp genomic regions. The transition-to-transversion ratio (ts/tv) is 2.01. In comparison, using MUMmer4's dnadiff pipeline, we called 2.86 million substitutions in 2.83Gbp at ts/tv=1.87. Given that ts/tv averaged across the human genome is about 2 but ts/tv averaged over random errors is 0.5, the minimap2 callset arguably has higher precision at lower sensitivity.

The sample being assembled is a female. Minimap2 still called 201 substitutions on the Y chromosome. These substitutions all come from one contig aligned at 96.8% sequence identity. The contig could be a segmental duplication absent from GRCh38. In constrast, dnadiff called 9070 substitutions on the Y chromosome across 73 SMRT contigs. This again implies our minimap2-based pipeline has higher precision.

## 4 DISCUSSIONS

Minimap2 is a versatile mapper and pairwise aligner for nucleotide sequences. It works with short reads, assembly contigs and long noisy genomic and RNA-seq reads, and can be used as a read mapper, long-read overlapper or a full-genome aligner. Minimap2 is also accurate and efficient, often outperforming other domain-specific alignment tools in terms of both speed and accuracy.

The capability of minimap2 comes from a fast base-level alignment algorithm and an accurate chaining algorithm. When aligning long query sequences, base-level alignment is often the performance bottleneck. The Suzuki-Kasahara algorithm greatly alleviates the bottleneck and enables DP-based splice alignment involving $>$100kb introns, which was impractically slow ten years ago. The minimap2 chaining algorithm is fast and highly accurate by itself. In fact, chaining alone is more accurate than all the other long-read mappers in Fig. 1a (data not shown). This accuracy helps to reduce downstream base-level alignment of candidate chains, which is still several times slower than chaining even with the Suzuki-Kasahara improvement. In addition, taking a general form, minimap2 chaining can be adapted to non-typical data types such as spliced reads and multiple reads per fragment. This gives us the opportunity to extend the same base algorithm to a variety of use cases.

Modern mainstream aligners often use a full-text index, such as suffix array or FM-index, to index reference sequences. An advantage of this approach is that we can use exact seeds of arbitrary lengths, which helps to increase seed uniqueness and reduce unsuccessful extensions. Minimap2 indexes reference k-mers with a hash table instead. Such fixed-length seeds are inferior to variable-length seeds in theory, but can be computed much more efficiently in practice. When a query sequence has multiple seed hits, we can afford to skip highly repetitive seeds without affecting the final accuracy. This further alleviates the concern with the seeding uniqueness. At the same time, at low sequence identity, it is rare to see long seeds anyway. Hash table is the ideal data structure for mapping long noisy sequences.

## ACKNOWLEDGEMENTS

We owe a debt of gratitude to H. Suzuki and M. Kasahara for releasing their masterpiece and insightful notes before formal publication. We thank M. Schatz, P. Rescheneder and F. Sedlazeck for pointing out the limitation of BWA-MEM. We are also grateful to minimap2 users who have greatly helped to suggest features and to fix various issues.

*Funding*: NHGRI 1R01HG010040-01

## REFERENCES


Abouelhoda, M. I. and Ohlebusch, E. (2005). Chaining algorithms for multiple genome comparison. *J. Discrete Algorithms*, 3:321–41.

Altschul, S. F. and Erickson, B. W. (1986). Optimal sequence alignment using affine gap costs. *Bull Math Biol*, 48:603–16.

Altschul, S. F. et al. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–402.

Berlin, K. et al. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nat Biotechnol*, 33:623–30.

Byrne, A. et al. (2017). Nanopore long-read RNAseq reveals widespread transcriptional variation among the surface receptors of individual B cells. *Nat Commun*, 8:16027.

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*, 13:238.

Daily, J. (2016). Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments. *BMC Bioinformatics*, 17:81.

Depristo, M. A. et al. (2011). A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat Genet*, 43:491–8.

Dobin, A. et al. (2013). STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29:15–21.

Farrar, M. (2007). Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, 23:156–61.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *J Mol Biol*, 162:705–8.

Gotoh, O. (1990). Optimal sequence alignment allowing for long gaps. *Bull Math Biol*, 52:359–73.

Holtgrewe, M. (2010). Mason – a read simulator for second generation sequencing data. Technical Report TR-B-10-06, Institut für Mathematik und Informatik, Freie Universität Berlin.

Irimia, M. and Roy, S. W. (2008). Evolutionary convergence on highly-conserved 3' intron structures in intron-poor eukaryotes and insights into the ancestral eukaryotic genome. *PLoS Genet*, 4:e1000148.

Iwata, H. and Gotoh, O. (2012). Benchmarking spliced alignment programs including Spaln2, an extended version of Spaln that incorporates additional species-specific features. *Nucleic Acids Res*, 40:e161.

Jain, M. et al. (2017). Nanopore sequencing and assembly of a human genome with ultra-long reads. *bioRxiv*. doi:10.1101/128835.

Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nat Methods*, 9:357–9.

Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv:1303.3997*.

Li, H. (2016). Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32:2103–10.

Li, H. and Durbin, R. (2010). Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, 26:589–95.

Li, H. et al. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25:2078–9.

Li, H. et al. (2017). New synthetic-diploid benchmark for accurate variant calling evaluation. *bioRxiv*. doi:10.1101/223297.

Lin, H.-N. and Hsu, W.-L. (2017). Kart: a divide-and-conquer algorithm for NGS read alignment. *Bioinformatics*.

Liu, B. et al. (2016). rHAT: fast alignment of noisy long reads with regional hashing. *Bioinformatics*, 32:1625–31.

Liu, B. et al. (2017). LAMSA: fast split read alignment with long approximate matches. *Bioinformatics*, 33:192–201.

Marçais, G. et al. (2018). MUMmer4: A fast and versatile genome alignment system. *PLoS Comput Biol*, 14:e1005944.

Ono, Y. et al. (2013). PBSIM: PacBio reads simulator–toward accurate genome assembly. *Bioinformatics*, 29:119–21.

Roberts, M. et al. (2004). Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20:3363–9.

Robinson, J. T. et al. (2011). Integrative genomics viewer. *Nat Biotechnol*, 29:24–6.

Sedlazeck, F. J. et al. (2017). Accurate detection of complex structural variations using single molecule sequencing. *bioRxiv*. doi:10.1101/169557.

Šošić, M. and Šikic, M. (2017). Edlib: a C/C++ library for fast, exact sequence alignment using edit distance. *Bioinformatics*, 33:1394–1395.

Sović, I. et al. (2016). Fast and sensitive mapping of nanopore sequencing reads with GraphMap. *Nat Commun*, 7:11307.

Suzuki, H. and Kasahara, M. (2018). Introducing difference recurrence relations for faster semi-global alignment of long sequences. *BMC Bioinformatics*, 19:45.

Wu, S. et al. (1996). A subquadratic algorithm for approximate limited expression matching. *Algorithmica*, 15:50–67.

Wu, T. D. and Watanabe, C. K. (2005). GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*, 21:1859–75.

Zaharia, M. et al. (2011). Faster and more accurate sequence alignment with SNAP. *arXiv:1111:5572*.

Zhang, M. and Gish, W. (2006). Improved spliced alignment from an information theoretic approach. *Bioinformatics*, 22:13–20.