Accelerating Genomic Alignment: Integrating Mod-Minimizers into Minimap2

Marcel Pokorski

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

BIOMEDICAL INFORMATICS

29.01.2025

Table of Contents

Introduction

Sequence Alignment

- similarities & differences between DNA/mRNA sequences
- understanding of genetic variations

Why Minimap2?

- aligner & mapper
- *minimizers*
- efficient, popular

Challenges

- long sequences
- objective-based balancing

Background: Minimap2

Seed-Based Indexing

- anchor points
- alignment candidates
- hash-based index

Sliding-Window

- smallest kmers
- reset at base $\notin$ {A,C,G,T}
- no deduplication

Background: Minimap2 cont'd

Backward Strand

- reverse complement
- no symmetric kmers

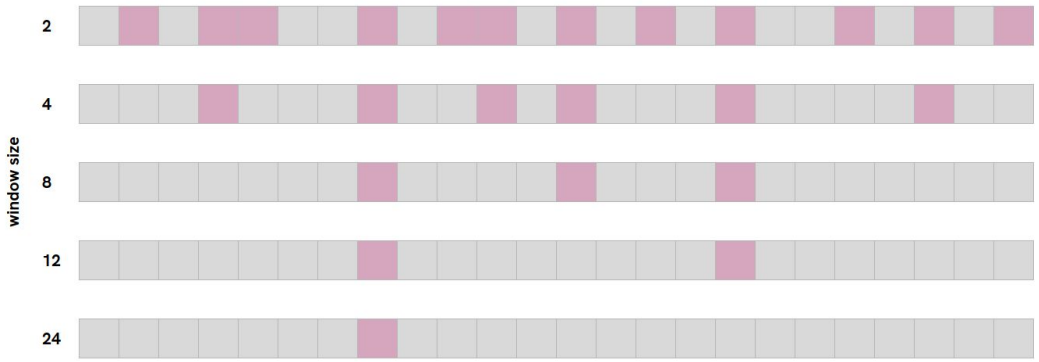
ACTG → TGAC → CAGT

Homopolymers

- lossy compression

GTAAAAAAAAAC → GTAC

Sensitive Balancing



Background: Mod-Minimizers

Change in Selection

1. minimal tmer at position x
2. select kmer starting at $(x \bmod w)$

- forward property
- lower, improved density

Formulae

$$\left.\begin{aligned} t &= r + ((k - r) \bmod w) \\ r &= \lceil \log_4(w + k - 1) \rceil + 1 \end{aligned}\right\} \quad \begin{aligned} k &= 15 \\ w &= 10 \end{aligned} \longrightarrow \begin{aligned} r &= 4 \\ t &= 5 \end{aligned}$$

Implementation: Approach

Minimal Changes

Runtime over Memory

```c
void mm_sketch(void *km, const char *str, int len, int w, int k, uint32_t rid, int is_hpc, mm128_v *p)
{
    int r = (int) ceil(log2(w + k - 1) / 2) + 1; //default: w = 10, k = 15, r = 4
    int t = r + ((k - r) % w);
    mm128_t min = { UINT64_MAX, UINT64_MAX };
    int min_pos = 0;

    uint64_t shift1 = 2 * (t - 1);
    uint64_t mask = (1ULL << 2 * t) - 1;
    uint64_t tmer[2] = { 0, 0 };
    int buf_pos = 0;
    mm128_t buf[256];

    // variables needed for simultaneous kmer creation
    uint64_t k_shift1 = 2 * (k - 1);
    uint64_t k_mask = (1ULL << 2 * k) - 1;
    uint64_t kmer[2] = { 0, 0 };
    int k_buf_pos = 0;
    mm128_t k_buf[256];

    memset(buf, 0xff, (w + k - t) * 16);
    memset(k_buf, 0xff, w * 16);
```

Implementation: Approach

Minimal Changes

Runtime over Memory

```
mm128_t info = { UINT64_MAX, UINT64_MAX };
mm128_t k_info = { UINT64_MAX, UINT64_MAX };

tmer[0] = (tmer[0] << 2 | c) & mask;           // forward t-mer
tmer[1] = (tmer[1] >> 2) | (3ULL^c) << shift1; // reverse t-mer

kmer[0] = (kmer[0] << 2 | c) & k_mask;           // forward k-mer
kmer[1] = (kmer[1] >> 2) | (3ULL^c) << k_shift1; // reverse k-mer


if (tmer[0] == tmer[1]) {
    continue; // symmetric tmer, strand unknown
}
int z = tmer[0] < tmer[1]? 0 : 1; // strand
if (l ≥ t && tmer_span < 256) {
    info.x = hash64(tmer[z], mask) << 8 | tmer_span;
    // tmer[z] hashed & trimmed, tmer_span
    info.y = i - tmer_span + 1;
    // global starting position
}


if (kmer[0] == kmer[1]) {
    continue; // symmetric tmer, strand unknown
}
int k_z = kmer[0] < kmer[1]? 0 : 1; // strand
if (l ≥ k && kmer_span < 256) {
    k_info.x = hash64(kmer[k_z], k_mask) << 8 | kmer_span;
    // kmer[k_z] hashed & trimmed, kmer_span
    k_info.y = (uint64_t)rid << 32 | (uint32_t)i << 1 | k_z;
    // sequence ref. ID, last pos., strand
}
```

Implementation: Approach

Minimal Changes

Runtime over Memory

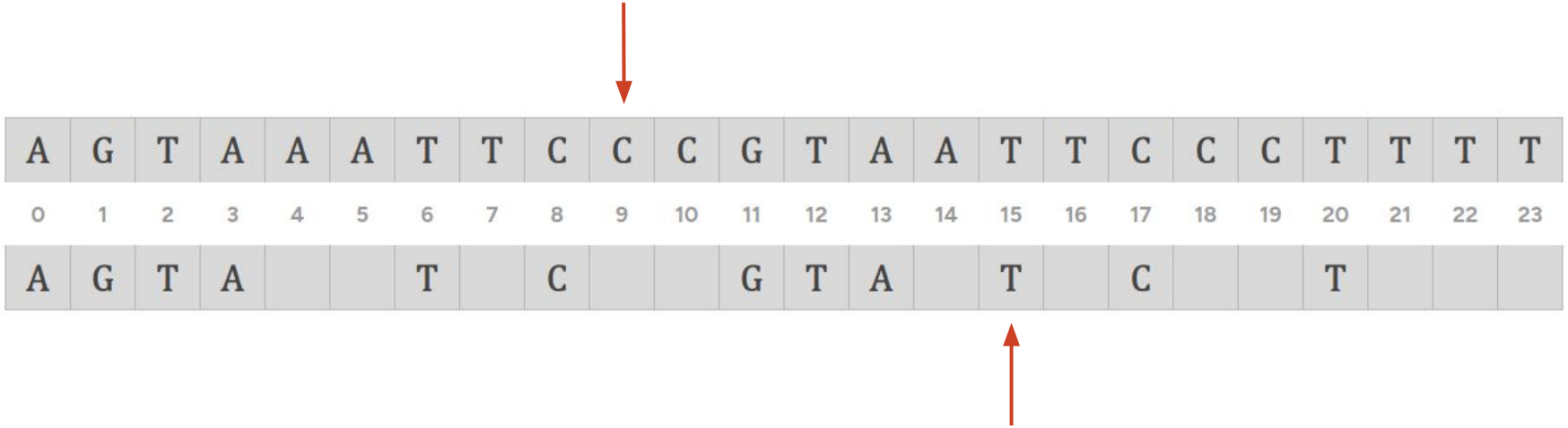
w	elements	$\longrightarrow$	w	$+$	$(w + k - t)$	elements
$w \times 16$	[Bytes]	$\longrightarrow$	$w \times 16$	$+$	$(w + k - t) \times 16$	[Bytes]
160	Bytes	$\longrightarrow$	160	$+$	320	Bytes

Implementation: Hurdles

HPC vs Indexing

Buffer Index Translation

Duplicates



Implementation: Hurdles

HPC vs Indexing

Buffer Index Translation

Duplicates

```c
int static mm_push_kmer(int w, int k, int t, kmer_info *kmer_vars) {
    int buf_size = w + k - t;
    int k_buf_size = w;
    // 1. calculate the minimal element's positon within the window
    int window_pos = (*kmer_vars->min_pos - *kmer_vars->buf_pos + (buf_size - 1)) % buf_size;
    // 2. mod-minimizer: modulo operation
    window_pos %= w;
    // 3. calculate the kmer buffer index of the window position
    int k_buf_idx = (*kmer_vars->k_buf_pos + 1 + window_pos) % k_buf_size;
    // 4. retrieve the corresponding kmer
    mm128_t k_target = kmer_vars->k_buf[k_buf_idx];

    // retrieve the target's end position within the sequence
    uint32_t k_target_glbl_pos = k_target.y;
    k_target_glbl_pos >>= 1;
    assert(k_target_glbl_pos >= 0 && k_target_glbl_pos < kmer_vars->len);

    // Push the k-mer if not already pushed
    if (*kmer_vars->prev_push != k_target_glbl_pos) {
        kv_push(mm128_t, kmer_vars->km, *kmer_vars->p, k_target);
    }

    return k_target_glbl_pos;
}
```

Implementation: Hurdles

HPC vs Indexing

Buffer Index Translation

Duplicates

```
int prev_push = -1; // last pushed kmer's global position

// creating the necessary structs to call mm_push_kmer
kmer_info kmer_vars = {
    .km = km,
    .p = p,
    .k_buf = k_buf,
    .min_pos = &min_pos,
    .buf_pos = &buf_pos,
    .k_buf_pos = &k_buf_pos,
    .prev_push = &prev_push,
    .len = len
};
```

Implementation: Hurdles

HPC vs Indexing

Buffer Index Translation

Duplicates

```
if (buf_pos == min_pos) {
    min.x = UINT64_MAX;

// scan for the leftmost minimal element
    int w_relative;
    for (w_relative = 0; w_relative < w + k - t; ++w_relative) {
      int wrapped_index = (buf_pos + 1 + w_relative) % (w + k - t);

      if (buf[wrapped_index].x < min.x) {
          min = buf[wrapped_index];
          min_pos = wrapped_index;
      }
  }

  prev_push = mm_push_kmer(w, k, t, &kmer_vars);
  }
```

Results: Approach

Minimap2	(default, $k=15, w=10$),	(SR, $k=21, w=11$)
Mod-Min	(default, $k=15, w=10, t=5$)	(SR, $k = 21, w = 11, t = 10$)
Mod-Min	(density adaptation, $k=15, w=8, t=5$)	(SR, $k = 21, w = 8, t = 5$)

↓

Default	`./minimap2 -a`
HPC	`./minimap2 -a -H`
SR	`./minimap2 -a -x sr`

Results: Metrics

Performance

real Time(s) | peak RSS (GB)

Minimizers

distinct minimizer | singleton rate | average spacing

Accuracy

total reads | mapped reads rate | unmapped reads rate | high-quality reads rate

Results: Dataset Aliivibrio fischeri

Reference Genome	~4.3Mb, well-characterized, annotated
6 Samples	~870Mb, short reads (10-76bp), Illumina HiSeq 2500

Why?

- interesting benchmark
- publicly available, well-researched
- balanced size

Results: Keys

Default		HPC		Density Adaptation	
-14%	**runtime**	-6%	runtime	±0%	runtime
-24%	distinct minimizers	-25%	distinct minimizers	**-5% to -8%**	**distinct minimizers**
+30%	avg spacing	+27%	avg spacing	**+4%**	**avg spacing**
-15%	**unmapped reads**	**+65%**	**unmapped reads**	**-14% to -21%**	**unmapped reads**
-1%	high-quality reads	**-19%**	**high-quality reads**	**+1% to +12%**	**high-quality reads**

Conclusion

Mod-Minimizer

- 14% quicker
- 15% less unmapped reads
- HPC & SR penalty

Density Adaptation

- levels benefits
- up to 21% less unmapped reads
- up to 12% more high-quality reads

Conclusion cont'd

Challenges

- sensitivity
- minimizer count-placement synergy

Future Work

- adaptive strategies
- dynamic adjustments
  - parameters
  - data characteristics
  - objectives

Complete Experiment Data

	Default		HPC		Short Reads	
	Default	Adapted	Default	Adapted	Default	Adapted
Performance Metrics						
Real Time (s)	-14.06%	+0.64%	-6.09%	-1.39%	-4.88%	-6.56%
Peak RSS (GB)	-0.08%	+0.09%	-1.19%	+0.31%	-0.18%	+0.03%
Minimizer Metrics						
Distinct Minimizers	-23.51%	-4.87%	-24.55%	-8.36%	-19.82%	-3.68%
Singleton Rate	-0.32%	-0.38%	-4.26%	-4.71%	-0.01%	-0.01%
Average Spacing	+30.32%	+4.71%	+27.06%	+4.19%	+24.72%	+3.85%
Accuracy Metrics						
Total Reads	+0.02%	+0.16%	-1.26%	+0.41%	±0.00%	±0.00%
Mapped Reads Rate	+3.40%	+3.42%	-0.24%	+0.05%	-0.13%	±0.00%
Unmapped Reads Rate	-15.25%	-21.35%	+65.41%	-13.72%	+14.16%	-0.05%
High-Quality Reads Rate	-1.05%	+1.05%	-18.76%	+12.00%	-2.03%	-0.81%

References

Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*

Holmberg, T. J. (2024). 3.7.3: Mapping Genomes. *BioLibreTexts*

Tamang, S. (2024). Sequence alignment—definition, types, methods, uses. *Microbe Notes*

Li, H. (2021). New strategies to improve minimap2 alignment accuracy. *Bioinformatics*

Smith, M. A., Gamaarachchi, H., & Parameswaran, S. (2019). Featherweight long read alignment using partitioned reference indexes. *Scientific Reports*

Xue, D., Feng, W., & Zhao, S. (2016). Improving alignment accuracy on homopolymer regions for semiconductor-based sequencing technologies. *BMC Genomics*

Chikhi, R., Blassel, L., & Medvedev, P. (2022). Mapping-friendly sequence reductions: going beyond homopolymer compression. *iScience*

Koerkamp, R. G., & Pibiri, G. E. (2024). The mod-minimizer: a simple and efficient sampling algorithm for long k-mers. *bioRxiv*

Pokorski, M. (2024). Minimap2: mod-mini, the modified version of minimap2. *GitHub repository*

Li, H. (2018). Minimap2: the original version of minimap2. *GitHub repository*

Banović Deri, B., Nešić, S., Vićić, I., Samardžić, J., & Nikolić, D. (2024). Benchmarking of five NGS mapping tools for the reference alignment of bacterial outer membrane vesicles-associated small RNAs. *Frontiers in Microbiology*

NCBI BioProject. (2024). PRJNA12986: Whole genome sequencing project of Aliivibrio fischeri.

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., … & Li, H. (2021). Twelve years of samtools and bcftools. *GigaScience*

Pokorski, M. (2024). mod-mini eval, benchmarking suite for evaluating the modified version of minimap2. *GitHub repository*

Bradnam, K. (2014). Understanding mapq scores in sam files: does 37 = 42? *Microbe Notes*

QnA

Accelerating Genomic Alignment: Integrating Mod-Minimizers into Minimap2

Marcel Pokorski,

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

BIOMEDICAL INFORMATICS

29.01.2025

marcikque/minimap2_mod-mini/

marcikque/mod-mini_eval/

lh3/minimap2