mm2-ivh: simple and precise overlap detection in alpha satellite HORs with interval hashing

Hajime Suzuki[1,*], Masahiro Sugawa[1], Yoshitaka Sakamoto[1], and Yuichi Shiraishi[1,*]

[1]Division of Genome Analysis Platform Development, National Cancer Center Research Institute, Tokyo, Japan

[*]To whom correspondence should be addressed.

Abstract

Summary: We propose a new algorithm, "interval hashing," which distinguishes identical k-mers arising from different repeat sequences, particularly in complex repeat arrays such as alpha satellite HORs. We implement this algorithm as a fork of minimap2, named mm2-ivh. In local assembly of alpha satellite HORs, mm2-ivh accurately reconstructs more haplotypes than assemblers using standard minimizers.
Availability: mm2-ivh is available under the MIT license at https://github.com/ocxtal/mm2-ivh and runs on common Unix-compatible systems.

Introduction

Resolving repeat regions in genome assembly has been a persistent challenge. Recent advances in long-read sequencing technologies, including PacBio's HiFi sequencing (Wenger *et al*. 2019) and Oxford Nanopore Technologies' (ONT) ultra-long (UL) read protocol (Jain *et al*. 2018), have significantly improved the ability to resolve complex repeat regions. HiFi reads, which are approximately 10–20 kilobase pairs (kbp) long and have over 99.8% accuracy, enable the precise detection of minor variations between repeat elements (Dvorkina, Bzikadze and Pevzner 2020). UL reads, which exceed 100 kbp in length, enable the resolution of long and complex repeat element arrangements that were previously intractable (Miga *et al*. 2020; Logsdon *et al*. 2021; Nurk *et al*. 2022). Assemblers such as Hifiasm (Cheng *et al*. 2021) and Verkko (Rautiainen *et al*. 2023) use HiFi reads to build accurate assembly graphs and UL reads to untangle these graphs into linear contigs, enabling the construction of nearly complete telomere-to-telomere sequences for a considerable number of chromosomes (Yang *et al*. 2023; Logsdon *et al*. 2024).

Nevertheless, accurately assembling long and highly homologous repeat regions remains challenging. For instance, the centromeric alpha satellite consists of 171 bp monomers that repeat over several megabases in a higher-order repeat (HOR) structure (Wu and Manuelidis 1980; Willard and Waye 1987; Thakur, Packiaraj and Henikoff 2021; Logsdon and Eichler 2022). Several methods have been developed to assemble and validate alpha satellite HORs using "unique k-mers," observed only at specific locations within HOR regions, enabling the accurate reconstruction of HOR structures across several chromosomes (Bzikadze and Pevzner 2020; Miga *et al*. 2020; Logsdon *et al*. 2021; Bzikadze, Mikheenko and Pevzner 2022; Dishuck *et al*. 2023; Chakravarty, Logsdon and Lonardi 2025). In long-read mapping within HOR regions and comparisons between HOR regions, methods that adaptively adjust the length of seeds based on uniqueness have also proven effective (Jain *et al*. 2022; Bzikadze and Pevzner 2023; Zhang *et al*. 2025). However, a broadly applicable method for overlap detection in diverse alpha satellite HORs has yet to be developed. Even state-of-the-art assemblers often produce misassemblies or inconsistencies among tools in alpha satellite HORs (Logsdon *et al*. 2025).

To address the challenges of assembling alpha satellite HORs, we propose a simple yet effective algorithm to improve seeding in periodic repeats. This algorithm, interval hashing, encodes the distances between occurrences of identical k-mers or minimizers (Roberts *et al*. 2004) into hash values, enabling the distinction of identical k-mers based on their surrounding repeat contexts. We implemented the algorithm on top of minimap2 (Li 2018) and applied it in combination with miniasm (Li 2016) for the local assembly of alpha satellite HORs using high-quality human ONT UL reads. Minimap2 with interval hashing successfully reconstructed the full-length active HOR regions in 27 out of 46 haplotypes without macroscopic misassemblies, outperforming the original minimap2 and Hifiasm in ONT-only mode. The implemented tool, mm2-ivh, is freely available under the MIT license at https://github.com/ocxtal/mm2-ivh.

Algorithm

The difficulty of assembling alpha satellite HORs arises from the low specificity of seeds, which is caused by the presence of identical k-mers in multiple monomers. To improve seed specificity and make the overlap detection easier in alpha satellite HORs, we leverage an intrinsic property of the regions. In alpha satellite HORs, certain k-mers appear only in specific monomers and generate characteristic, sometimes non-equidistant patterns of k-mer occurrence intervals. The presence of truncated or rare HOR units, or the insertion of unrelated sequences such as transposable elements, also leads to distinct occurrence interval patterns compared to regions without such events. Our algorithm, interval hashing, encodes the k-mer occurrence intervals into hash values, serving as signatures to distinguish identical k-mers by their surrounding contexts (Fig. 1(a)).

The algorithm takes a vector of N ($N \geq 0$) intervals, derived from the $N + 1$ occurrences of identical k-mers, and produces a vector of $N + 1$ hash values corresponding to each occurrence. The core algorithm is defined by a wing length parameter W that indicates the number of k-mer occurrences to consider in each direction and consists of five steps (Fig. 1(b)). In the first step, two vectors of $N + 1$ slices are extracted from the input intervals, with slice lengths being $2W$ and $4W$. The central $2W$ span of i-th $4W$-long slice overlaps with i-th $2W$-long slice. The out-of-bound positions are assigned interval values of infinity. Next, the "unit interval," representing the length of a canonical repeat element, is calculated for each $4W$-long slice by taking the minimum interval value. These unit intervals are then used to normalize the interval values in the corresponding $2W$-long slices. The normalized intervals are rounded to the nearest integer after being multiplied by a resolution constant (fixed at 4.0 in our implementation) to ensure sufficient resolution for truncated repeat elements. Then, the integer intervals in each slice are individually mapped to partial hash values and then combined into a final hash value using a rolling hashing algorithm similar to ntHash (Mohamadi *et al*. 2016). Infinite interval values are mapped to a special partial hash value of zero to indicate the absence of interval information, ensuring that the hash value computed for an empty input interval vector derived from singleton k-mers is zero.

In practical implementations, it is necessary to match k-mers that are in a reverse-complement relationship. Tools such as minimap2 canonicalize k-mers by selecting the numerically smaller k-mer between the two strands. Interval hashing achieves consistent canonicalization by reversing the slices when the corresponding central k-mer is computed from the reverse-complement strand. Additionally, to prevent encoding intervals that span distant repeat contexts, it is practical to split the interval vector at points where the intervals exceed a certain threshold before applying interval hashing.

Implementation

We implemented the interval hashing algorithm in a fork of minimap2 2.28, which applies interval hashing to all minimizers. Further implementation details are provided in Supplementary Section S1. The executable is named mm2-ivh and is compatible with minimap2 in terms of options and input/output formats. Based on preliminary experiments described in Supplementary Section S3, we added a preset option, `-x ivh-ava-ont-ul`, tuned for the latest UL reads with a mean quality value of 20 or higher. The preset uses interval-hash-augmented minimizers with parameters $(k, w, W) = (19, 15, 3)$ and sets the maximum minimizer separation for splitting interval vectors to 20 kbp.

mm2-ivh is available at https://github.com/ocxtal/mm2-ivh under the MIT license, following the original minimap2. The program is implemented solely in C, requiring only a C compiler and make to build. It supports x86_64 and arm64 systems running Linux or macOS. The version used in our experiments is also available as an archive at 10.5281/zenodo.17143522.

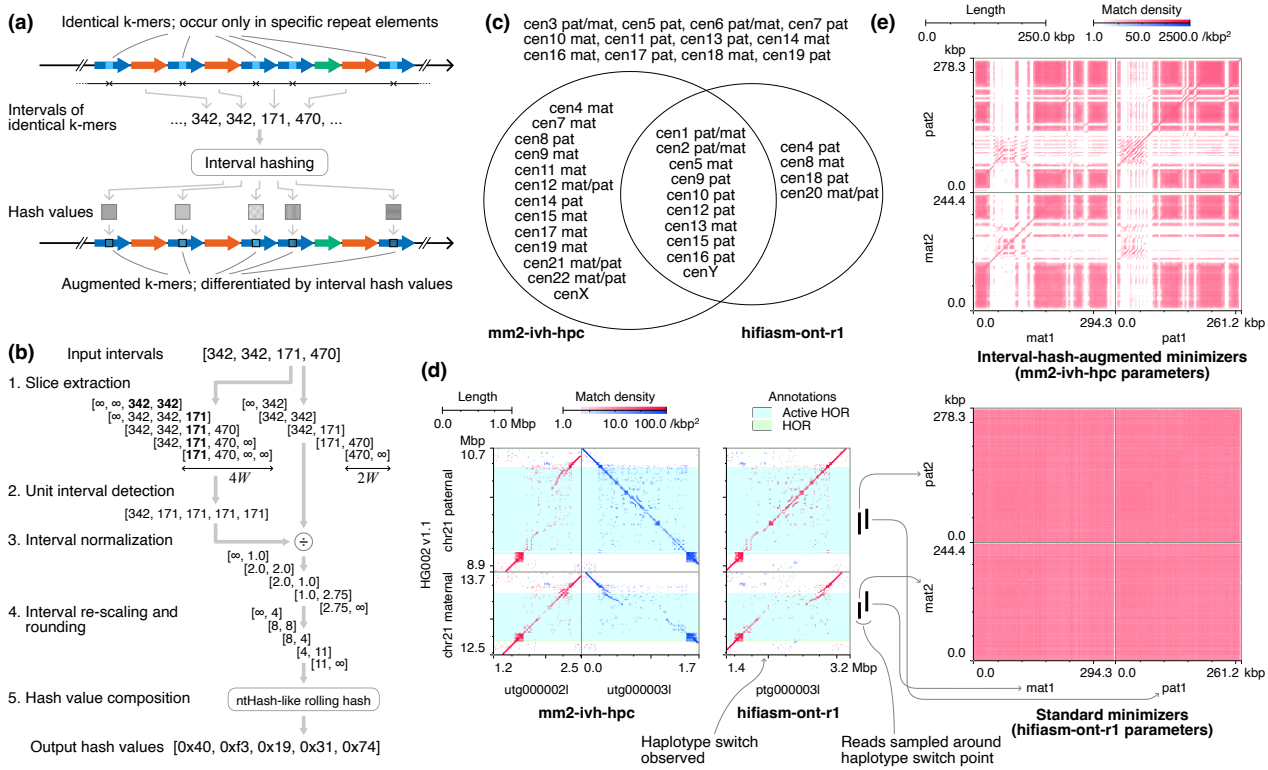

Figure 1. **(a)** Concept of interval hashing. **(b)** Flow of the five steps of the core of our algorithm (depicts $N = 4$ and $W = 1$). **(c)** Haplotypes in which the full length of active HOR regions is reconstructed into a single unitig. **(d)** Dotplots of the assembled unitigs of alpha satellite HORs of chromosome 21 against the corresponding regions of HG002 v1.1. Only minimizers with $(k, w) = (19, 15)$ occurring less than or equal to 10 times among all alpha satellite HOR regions in HG002 v1.1 are used. Red and blue dots are forward and reverse-complement matches, respectively. **(e)** Dotplots among four reads across the haplotype switch point in hifiasm-ont-r1 assembly. Reads mat1/2 and pat1/2 were primarily mapped across the corresponding position on the HG002 v1.1 chromosome 21 maternal and paternal, respectively. The colors are the same as in the panel (d).

Results

We evaluated mm2-ivh for the local assembly of human alpha satellite HOR regions using the HG002 high-accuracy UL data published by ONT (Supplementary Section S4.1). After quality control, we obtained 89.0 Gbp of reads with a mean length of 116.9 kbp and a mean quality value of 24.64. The reads were then aligned to CHM13 v2.0 (Nurk *et al*. 2022), which consists of the autosomes and chromosome X of CHM13 and chromosome Y derived from HG002, using minimap2 with the `-x map-ont` preset. Reads mapped to HOR regions with 500 kbp margins were extracted for each chromosome (Supplementary Table S3). The extracted reads were assembled using the following three assemblers (Supplementary Section S4.2). The first assembler computes overlaps using mm2-ivh and outputs a pairwise alignment format (PAF) file, which is then filtered to remove overlaps with low match rates and supplied to miniasm to generate assembly graphs (denoted as mm2-ivh-hpc). The parameters for mm2-ivh and miniasm were set as `-x ivh-ava-ont-ul -H` (additionally enabling homopolymer compression) and `-h5000 -c2` (setting the maximum overhang length to 5 kbp and minimum coverage to 2), respectively. The filtering step allows only overlaps with more than 20% match columns relative to the alignment length. The second assembler, based on mm2-ivh-hpc, disables interval hashing and instead increases the minimizer length to $k = 133$ (Supplementary Section S2 for details

on supporting long minimizers; denoted as mm2-k133-hpc). The third assembler is Hifiasm 0.25.0, run with `--ont -r1` options (one round of error correction; denoted as hifiasm-ont-r1). The `-r1` setting was chosen because it yielded the best results in our tests.

mm2-ivh-hpc reconstructed unitigs that fully covered the active HOR regions of the HG002 v1.1 reference assembly (Rautiainen *et al*. 2023) in 27 out of 46 haplotypes, while hifiasm-ont-r1 achieved this in 16 haplotypes (Supplementary Section S4.3, Supplementary Table S8). mm2-k133-hpc reconstructed the full length of active HOR regions only in chromosome Y. The number of active HOR regions that could be reconstructed into a single unitig by both mm2-ivh-hpc and hifiasm-ont-r1 was 11, while 16 were reconstructed only by mm2-ivh-hpc, 5 only by hifiasm-ont-r1, and 14 were not reconstructed by either (Fig. 1(c)). The effectiveness of interval hashing is clearly demonstrated in some HOR arrays. For instance, for the active HOR region in chromosome 21, mm2-ivh-hpc was able to assemble the full length for both haplotypes without haplotype mixing, whereas hifiasm-ont-r1 introduced a haplotype switch in the middle (Fig. 1(d), Supplementary Section S4.4). The dotplot of four reads mapped across the haplotype switch position (Supplementary Table S9) showed that interval hashing clearly distinguished haplotypes, while standard minimizers obscured them with dense matches (Fig. 1(e), Supplementary Fig. S5).

Execution time and memory consumption varied widely among chromosomes for mm2-ivh-hpc and mm2-k133-hpc, whereas hifiasm-ont-r1 exhibited stable performance (Supplementary Section S4.2, Supplementary Tables S4–S6). In both mm2-ivh-hpc and mm2-k133-hpc, the overlap detection step consumed the dominant proportion of execution time. The total CPU time was 11.8 hours for mm2-ivh-hpc, 339.7 hours for mm2-k133-hpc, and 3.5 hours for hifiasm-ont-r1. The geometric mean of memory usage per chromosome (worst case in parentheses) was 6.4 GiB (14.0 GiB), 40.5 GiB (181 GiB), and 16.5 GiB (16.8 GiB) in the same order. The total size of the overlap detection outputs (PAF files) was 77.0 GiB and 1.08 TiB for mm2-ivh-hpc and mm2-k133-hpc, respectively (Supplementary Table S7).

Discussion

The benchmark clearly demonstrated that interval hashing is effective for precise overlap detection in alpha satellite HORs. The substantial reductions in execution time, memory usage, and output file size for overlap detection compared to the control using standard minimizers (mm2-k133-hpc) also indirectly support that interval hashing reduces false-positive matches. These findings suggest that the intervals of k-mers within repeats are a critical feature for distinguishing alpha satellite HORs. We also noticed that the concept of k-mer intervals aligns with early studies of alpha satellites in which HORs are characterized by analyzing the lengths of DNA fragments after restriction enzyme digestion (Wu and Manuelidis 1980). This indicates that interval hashing essentially examines the same feature, renovating one of the most fundamental features of alpha satellite HORs using modern techniques.

Interval hashing resembles seed-and-extend alignment algorithms for optical mapping (Leung *et al*. 2017; Salmela *et al*. 2020) in that both use subsequences of interval values as seeds. Our contribution lies in demonstrating that subsequences of interval values are useful as signatures of repeats, which has been less explored in the field of optical mapping. The specific structure of interval hashing is also similar to RawHash, a hashing algorithm for matching on Nanopore raw signals, in that both construct hashes from coarsened numerical sequences (Firtina *et al*. 2023). In comparison with algorithms in the sequence space, interval hashing is similar to spaced seeds (Ma, Tromp and Li 2002) and strobemers (Sahlin 2021) in that these algorithms construct a single seed from multiple exact submatches separated by gaps. We found interval hashing and existing algorithms essentially differ in whether or not the submatches are synchronized with the underlying repeat structure. Interval hashing uses synchronized submatches, which effectively capture repeat contexts.

The weakness of interval hashing is its susceptibility to sensitivity degradation due to sequencing errors. This stems from the algorithm's reliance on the preservation of all k-mers within W occurrences before and after a match. In our experiments, the low error rates of ONT reads appeared to mitigate the sensitivity issue. Improvements to the algorithm will be necessary to apply interval hashing to data with higher error rates. In the field of optical mapping, algorithms for obtaining error-robust seeds on interval information have been explored (Salmela *et al*. 2020), which may help improve our algorithm.

Although our benchmark focused on alpha satellite HORs, interval hashing is likely to be compatible with unique genomic regions, segmental duplications, and isolated interspersed repeat elements. This compatibility stems from the property that singleton or isolated minimizers maintain the same values after applying interval hashing, thereby not affecting the behavior of the algorithms that use minimizers. Moreover, interval hashing potentially enhances specificity in less structured repeat regions, such as various satellite repeats and clusters of interspersed repeats, similar to its effect on alpha satellite HORs. These properties could make interval hashing applicable to pipelines relying on all-versus-all overlaps of reads (i.e., without probing or filtering, such as whole-genome assembly and read error correction (Espinosa *et al*. 2024; Stanojevic *et al*. 2024)), potentially improving accuracy in various repeat contexts. Our implementation, mm2-ivh, facilitates evaluation of how interval hashing improves the performance in these pipelines on diverse data.

Acknowledgments

The computational resources used in this study were provided by the Human Genome Center, the Institute of Medical Science, the University of Tokyo. We also thank Dr. Yoshihiko Suzuki for providing insightful comments on the algorithm and the manuscript.

Author contributions

Hajime Suzuki: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Visualization, Writing – original draft. Masahiro Sugawa: Validation, Resources, Writing – review & editing. Yoshitaka Sakamoto: Validation, Writing – review & editing. Yuichi Shiraishi: Conceptualization, Resources, Writing – review & editing, Supervision, Funding acquisition.

Funding

This work is supported by Grants-in-Aid from the Japan Agency for Medical Research and Development [Project for Cancer Research and Therapeutic Evolution: JP25ama221538], and National Cancer Center Research and Development Funds [2025-A-03].

References

Bzikadze AV, Mikheenko A, Pevzner PA. Fast and accurate mapping of long reads to complete genome assemblies with VerityMap. *Genome Res* 2022;**32**:2107–18.

Bzikadze AV, Pevzner PA. Automated assembly of centromeres from ultra-long error-prone reads. *Nature Biotechnology* 2020;**38**:1309–16.

Bzikadze AV, Pevzner PA. UniAligner: A parameter-free framework for fast sequence alignment. *Nat Methods* 2023;**20**:1346–54.

Chakravarty S, Logsdon G, Lonardi S. RAmbler resolves complex repeats in human chromosomes 8, 19, and X. *Genome Res* 2025;**35**:863–76.

Cheng H, Concepcion GT, Feng X *et al*. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nat Methods* 2021;**18**:170–5.

Dishuck PC, Rozanski AN, Logsdon GA *et al*. GAVISUNK: Genome assembly validation via inter-SUNK distances in Oxford Nanopore reads. Martelli PL (ed.). *Bioinformatics* 2023;**39**:btac714.

Dvorkina T, Bzikadze AV, Pevzner PA. The string decomposition problem and its applications to centromere analysis and assembly. *Bioinformatics* 2020;**36**:i93–101.

Espinosa E, Bautista R, Larrosa R *et al*. Advancements in long-read genome sequencing technologies and algorithms. *Genomics* 2024;**116**:110842.

Firtina C, Mansouri Ghiasi N, Lindegger J *et al*. RawHash: Enabling fast and accurate real-time analysis of raw nanopore signals for large genomes. *Bioinformatics* 2023;**39**:i297–307.

Jain C, Rhie A, Hansen NF *et al*. Long-read mapping to repetitive reference sequences using Winnowmap2. *Nat Methods* 2022;**19**:705–10.
Jain M, Koren S, Miga KH *et al*. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat Biotechnol* 2018;**36**:338–45.
Leung AK-Y, Kwok T-P, Wan R *et al*. OMBlast: Alignment tool for optical mapping using a seed-and-extend approach. Hancock J (ed.). *Bioinformatics* 2017;**33**:311–9.
Li H. Minimap and miniasm: Fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* 2016;**32**:2103–10.
Li H. Minimap2: Pairwise alignment for nucleotide sequences. Birol I (ed.). *Bioinformatics* 2018;**34**:3094–100.
Logsdon GA, Ebert P, Audano PA *et al*. Complex genetic variation in nearly complete human genomes. *Nature* 2025;**644**:430–41.
Logsdon GA, Eichler EE. The dynamic structure and rapid evolution of human centromeric satellite DNA. *Genes* 2022;**14**:92.
Logsdon GA, Rozanski AN, Ryabov F *et al*. The variation and evolution of complete human centromeres. *Nature* 2024;**629**:136–45.
Logsdon GA, Vollger MR, Hsieh P *et al*. The structure, function and evolution of a complete human chromosome 8. *Nature* 2021;**593**:101–7.
Ma B, Tromp J, Li M. PatternHunter: Faster and more sensitive homology search. *Bioinformatics* 2002;**18**:440–5.
Miga KH, Koren S, Rhie A *et al*. Telomere-to-telomere assembly of a complete human X chromosome. *Nature* 2020;**585**:79–84.
Mohamadi H, Chu J, Vandervalk BP *et al*. ntHash: Recursive nucleotide hashing. *Bioinformatics* 2016;**32**:3492–4.
Nurk S, Koren S, Rhie A *et al*. The complete sequence of a human genome. *Science* 2022;**376**:44–53.
Rautiainen M, Nurk S, Walenz BP *et al*. Telomere-to-telomere assembly of diploid chromosomes with Verkko. *Nat Biotechnol* 2023;**41**:1474–82.
Roberts M, Hayes W, Hunt BR *et al*. Reducing storage requirements for biological sequence comparison. *Bioinformatics* 2004;**20**:3363–9.
Sahlin K. Effective sequence similarity detection with strobemers. *Genome Res* 2021;**31**:2080–94.
Salmela L, Mukherjee K, Puglisi SJ *et al*. Fast and accurate correction of optical mapping data via spaced seeds. Hancock J (ed.). *Bioinformatics* 2020;**36**:682–9.
Stanojevic D, Lin D, Nurk S *et al*. Telomere-to-telomere phased genome assembly using HERRO-corrected simplex Nanopore reads. 2024.
Thakur J, Packiaraj J, Henikoff S. Sequence, chromatin and evolution of satellite DNA. *IJMS* 2021;**22**:4309.
Wenger AM, Peluso P, Rowell WJ *et al*. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat Biotechnol* 2019;**37**:1155–62.
Willard HF, Waye JS. Hierarchical order in chromosome-specific human alpha satellite DNA. *Trends in Genetics* 1987;**3**:192–8.
Wu JC, Manuelidis L. Sequence definition and organization of a human repeated DNA. *Journal of Molecular Biology* 1980;**142**:363–86.
Yang C, Zhou Y, Song Y *et al*. The complete and fully-phased diploid genome of a male Han Chinese. *Cell Res* 2023;**33**:745–61.
Zhang P, Wei Y, Tian Q *et al*. Fast sequence alignment for centromeres with RaMA. *Genome Res* 2025;**35**:1209–18.