# Partitioned multi-MUM finding for scalable pangenomics with MumemtoM

Vikram S. Shivakumar and Ben Langmead

*Genome Res.* published online November 7, 2025
Access the most recent version at doi:10.1101/gr.280940.125

Method

# Partitioned multi-MUM finding for scalable pangenomics with MumemtoM

Vikram S. Shivakumar and Ben Langmead

*Department of Computer Science, Johns Hopkins University, Baltimore, Maryland 21218, USA*

**Pangenome collections are growing to hundreds of high-quality genomes. This necessitates scalable methods for constructing pangenome alignments that can incorporate newly sequenced assemblies. We previously developed Mumemto, which computes maximal unique matches (multi-MUMs) across pangenomes using compressed indexing. In this work, we introduce MumemtoM (Mumemto Merge), comprising two new partitioning and merging strategies. Both strategies enable highly parallel, memory-efficient, and updateable computation of multi-MUMs. One of the strategies, called string-based merging, is also capable of conducting the merges in a way that follows the shape of a phylogenetic tree, naturally yielding the multi-MUM for the tree's internal nodes as well as the root. With these strategies, Mumemto now scales to 474 human haplotypes, the only multi-MUM method able to do so. It also introduces a time–memory tradeoff that allows Mumemto to be tailored to more scenarios, including in resource-limited settings.**

**[Supplemental material is available for this article.]**

Pangenome assembly collections continue to proliferate and grow. Recently published pangenomes have covered crops (Garg et al. 2024), model organisms (Hufford et al. 2021; Lian et al. 2024), and the expanded version 2 (v2) of the Human Pangenome Reference Consortium (HPRC) data set spanning 474 human haplotypes (Liao et al. 2023; Li 2025). Other recent pangenomes span distinct but related species (Yoo et al. 2025), as well as geographically and genetically distinct accessions of the same species (Lian et al. 2024). Because of the growing size and complexity of these collections, it is increasingly important that our methods for studying and indexing pangenomes be scalable. Specifically, we wish to be able to find conserved elements across pangenomes in a way that scales to large pangenomes and that can be updated incrementally when the pangenome grows further. Such methods will allow us build the common coordinate systems that will underlie future studies (Taylor et al. 2024).

A multiple maximal unique match (multi-MUM) is an exact sequence match that is present exactly once in all the pangenome sequences. Because they are both unique in each sequence and common to all the sequences, multi-MUMs serve as particularly strong guideposts for establishing common pangenome coordinates. They have been used as anchors for multiple alignment (Deogun et al. 2004; Darling et al. 2010; Kille et al. 2024; Olbrich et al. 2025) and as markers for chaining of pangenome read alignments (Brown et al. 2025). We previously proposed a framework and tool called Mumemto for rapidly identifying multi-MUMs during construction of a compressed pangenome index (Shivakumar and Langmead 2025). Using prefix-free parsing (PFP) (Boucher et al. 2019), a compressed-space method for computing full-text indexes, Mumemto outperforms existing methods for identifying sequence-level multi-MUMs. Another method, PANAMA (Olbrich et al. 2025), uses a similar PFP-based approach to compute larger syntenic blocks for multiple alignment. However, although both of these tools were capable of operating over pangenomes of moderate size, PFP cannot currently be scaled to the larger collections available today. For instance, running Mumemto on the 474 human haplotypes of the HPRC v2 release (Liao et al. 2023; Li 2025) requires over 2TB of memory.

To address this, we developed a partition-merging approach to compute multi-MUMs with Mumemto, which we call MumemtoM (Mumemto merge). MumemtoM separates the input genomes into partitions, then uses PFP to compute per-partition intermediate results, comprising multi-MUMs and additional match information. The method then merges results from the partitions to yield a global set of multi-MUMs across the union of the partitions. Whereas past methods like Parsnp have proposed ways to iteratively merge pairwise MUMs into multi-MUMs (Treangen 2008; Treangen et al. 2014), our method generalizes to merging across arbitrary initial collections of sequences, including multi-way merging. This yields a highly parallelizable and easily updateable multi-MUM-finding workflow that scales to hundreds of human genomes.

MumemtoM comprises two methods for merging multi-MUMs. "Anchor-based merging" uses a common reference sequence in each partition to anchor matches and identify overlaps between subsets for subsequent merging. "String-based merging" finds overlaps between multi-MUMs from the sequence directly, merging multi-MUMs across disjoint sequence collections. When applying string-based merging to partitions that correspond to evolutionarily related clades, the intermediate results computed at each step are the multi-MUMs for some internal node of the phylogenetic tree. As a result, we can establish a set of multi-MUMs, and therefore a common coordinate system, at multiple levels of the phylogeny at once.

By implementing a partition-merging scheme for computing multi-MUMs, we also introduce a tradeoff space between highly parallelizable multi-MUM computation and peak memory usage. This allows Mumemto to scale to hundreds of human genomes by splitting the genomes into partitions and processing each

**Corresponding authors: vshivak1@jhu.edu, langmea@cs.jhu.edu**
Article published online before print. Article, supplemental material, and publication date are at https://www.genome.org/cgi/doi/10.1101/gr.280940.125.

partition serially. It also allows Mumemto to accelerate multi-MUM finding by processing the partitions in parallel, at the expense of a higher total memory footprint. Using MumemtoM, Mumemto can also now easily integrate newly assembled genomes without recomputing from scratch, making it well-suited to scale as pangenome collections continue to grow in the future and as a core algorithm for pangenome construction and analysis.

## Methods

### Preliminaries

Consider text $T$, a concatenation of $N$ sequences $[T_0 \cdots T_{N-1}]$, with total length $n$ over alphabet $\Sigma$. We use closed, $[i, j]$, zero-indexed interval notation for arrays and string indexing in the following preliminaries. The $i$th suffix is defined as the substring $T[i \cdots n-1]$. The suffix array SA of $T$ contains the offsets $[0 \cdots n-1]$ of $T$, ordered by lexicographic rank of the corresponding suffix. The Burrows–Wheeler transform $\mathrm{BWT}_T$ is a sequence consisting of characters that precede each of $T$'s suffixes in the order given by SA, that is, $\mathrm{BWT}_T[i] = T[\mathrm{SA}[i]-1]$ (assuming the text is circular). The longest common prefix array at offset $i$, LCP[$i$], contains the longest common prefix between suffixes $T[\mathrm{SA}[i-1] \cdots n-1]$ and $T[\mathrm{SA}[i] \cdots n-1]$ (with $\mathrm{LCP}[0] = 0$). The document array (DA) contains the text of origin for each suffix in suffix array order; for example, for $\mathrm{DA}[i] = k$, $T[\mathrm{SA}[i] \cdots n-1]$ begins in $T_k$ in the concatenation of sequences. In practice, text $T$ contains a delimiter between each sequence that is lexicographically smaller than any character in $\Sigma$.

### Mumemto

Mumemto (Shivakumar and Langmead 2025) computes various types of exact matches. These might be constrained to be unique in each sequence (i.e., a maximal unique match or MUM) or not (i.e., a maximal exact match or MEM). They might also be required to appear in all sequences (a multi-MUM or multi-MEM) or only in some fraction of them (a partial multi-MUM or partial multi-MEM). In this work, we focus specifically on multi-MUMs. We may refer to them as simply MUMs for short when the "multi" is clear from context. Multi-MUMs are represented by an interval $[i, j]$ of size $N$ in the SA, BWT, LCP array, and DA. This window has the following properties:

1. (Exact match): With $l = \min \mathrm{LCP}[i+1 \cdots j]$ representing the length of the shared common prefix among suffixes in the interval, $\mathrm{LCP}[i+1] \geq l > \mathrm{LCP}[i]$ and $\mathrm{LCP}[j] \geq l > \mathrm{LCP}[j+1]$.
2. (Uniqueness): $\mathrm{DA}[i \cdots j]$ contains pairwise distinct values.
3. (Left-maximal): The characters in $\mathrm{BWT}[i \cdots j]$ are not all identical.

Mumemto adapts an algorithm (Shivakumar and Langmead 2025, Algorithm 1) proposed by Abouelhoda et al. (2004) to find multi-MUMs by identifying intervals in the various arrays where these properties hold.

### Prefix-free parsing

Prefix-free parsing (Boucher et al. 2019) is a compressed-space algorithm for computing the BWT, SA, and LCP arrays. It begins by parsing the input text into distinct phrases (the dictionary $D$) and storing the sequence of phrases that make up the original text (the parse $P$). By computing the SA, LCP, and BWT over both $D$ and $P$, PFP can reconstruct the original arrays, requiring only $O(|D| + |P|)$ space. The space requirement tends to grow sublinearly for repetitive texts and represents the amount of unique sequence present in the pangenome (Lucà et al. 2025). Crucially, PFP computes and reports these arrays in order and in tandem, allowing Mumemto to scan for windows satisfying the properties mentioned above (exact match, uniqueness, and left-maximality) in a streaming fashion and without writing the arrays to disk.

### Anchor-based merging

We propose a pair of related algorithms for merging multi-MUMs computed across $k$ pangenome partitions. In both cases, the final set of merged multi-MUMs is the same as if they had been computed over the union of the merged partitions. We propose two methods: one that uses a common sequence present in each partition to serve as a coordinate system and "anchor" the merging process ("anchor-based merging") and a second that does not require a common sequence across partitions and merges purely based on the sequences of the multi-MUMs themselves ("string-based merging"). The choice of anchor sequence is arbitrary, due to the fact that a multi-MUM appears in all sequences by definition, and so appears at some offset in any particular sequence.

Merging multi-MUMs across partitions requires some care. A simple case is one where the two partitions each have a corresponding multi-MUM that, once the partitions are merged, is still a multi-MUM in the merged collection. But subtler cases arise when (a) a multi-MUM in one partition is no longer a multi-MUM in the merged collection due to genetic variation or differences in repetitive sequence or (b) multi-MUMs in the different partitions combine to form a merged multi-MUM that is shorter than one or both of the originals, due to genetic differences that are unique to one partition or the other. To handle all such cases, we store extra information for each partition that indicates how short a match can be truncated before it is no longer unique in the given partition. To do so, we consider unique matches (UMs), which contain properties (1) and (2) of multi-MUMs, but not property (3); that is, they may be further extendable to the left. While traversing the arrays, for every multi-MUM or UM encountered, we store $\max(\mathrm{LCP}[i], \mathrm{LCP}[j+1])$, which represents the longest prefix of the MUM or UM that appears elsewhere in the text. In other words, if the MUM or UM is truncated below this length from the right, it will no longer be unique in at least one sequence in the partition. We store this length at the offset where the MUM or UM occurs in the anchor reference sequence.

We store such a value for each offset of the anchor reference, yielding an array called `unique_threshold`. Each value in `unique_threshold` is either 0, indicating no UM or MUM starts at that offset, or $l > 0$, indicating the match occurring at that offset will become nonunique at length $l$ or shorter. Algorithm 1 describes and Figure 1A–D illustrates merging two partitions using this array. After merging, an updated `unique_threshold` array for the merged output is computed using an element-wise max operation, which can be used in a subsequent round of merging. As partitions are merged, the updated value represents the higher threshold at which a match is no longer unique.

**Algorithm 1.** Anchor-based MUM merging

**Input:** Two sorted arrays of maximal unique matches $\mathcal{M}_1$, $\mathcal{M}_2$
**Input:** `unique_threshold` arrays $T_1$, $T_2$
**Output:** Merged MUMs $\mathcal{M}_{\text{merged}}$ and merged `unique_threshold` $T_{\text{new}}$

▷ MUMs are sorted by anchor position
$n \leftarrow |T_1|$ ▷ Length of anchor reference sequence
$B_1 \leftarrow \mathrm{CreateBitVector}(\mathcal{M}_1)$ ▷ Bit vectors marking MUM start positions
$B_2 \leftarrow \mathrm{CreateBitVector}(\mathcal{M}_2)$
$\mathcal{M}_{\text{merged}} \leftarrow \emptyset$
$T_{\text{new}} \leftarrow \mathrm{Array}(n)$

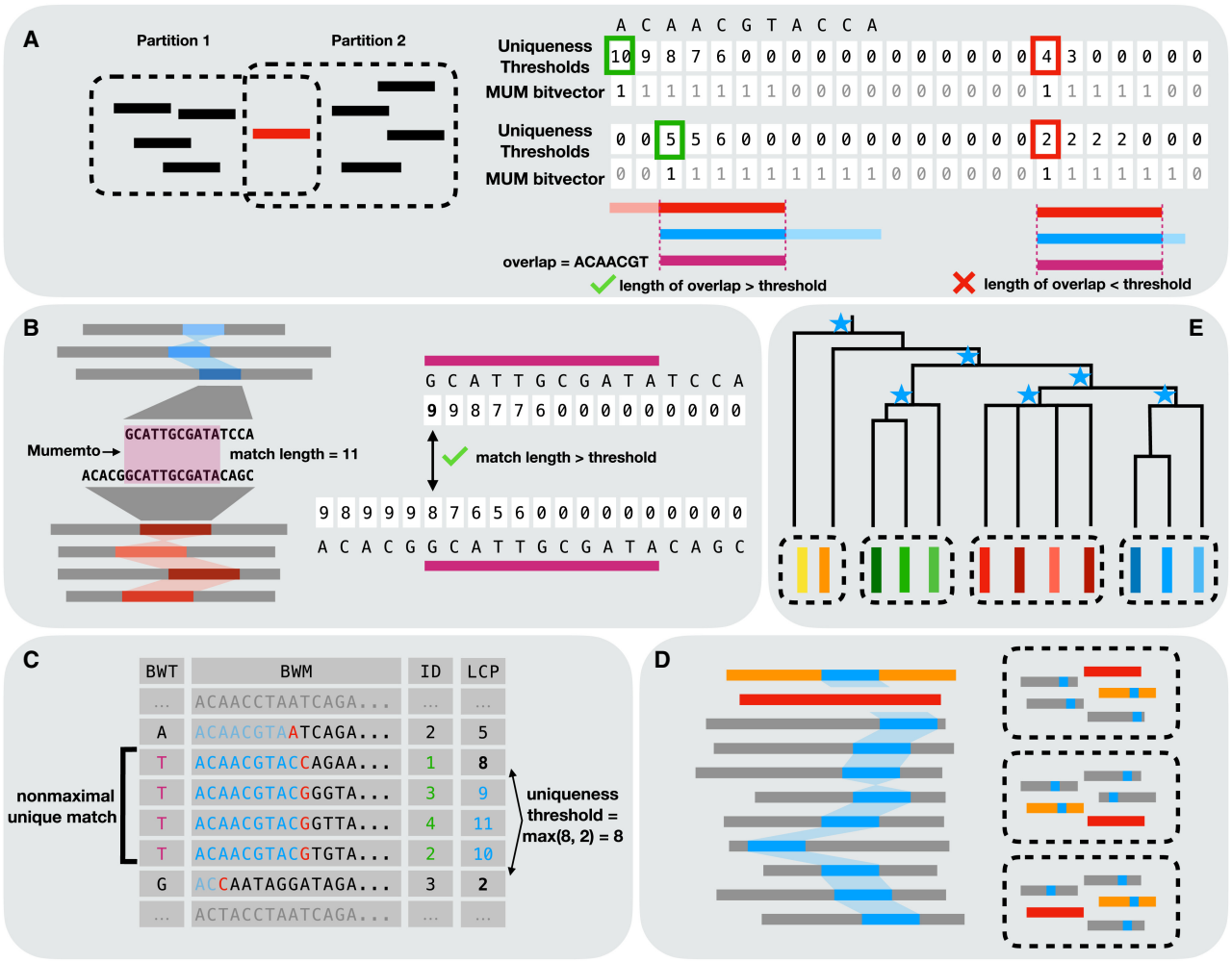


**Figure 1.** Overview of the MumemtoM algorithm. (*A*) Anchor-based merging requires a common sequence (red) present in each partition. Multi-MUMs are merged by identifying overlaps between partition-specific matches in the anchor coordinate space, and a uniqueness threshold determines if a MUM is still unique in each partition after truncation. (*B*) String-based merging enables computation of multi-MUMs between partitions without a common sequence. MUM overlaps are computed by running Mumemto on the MUM sequences, and the uniqueness threshold array ensures overlaps remain unique across the merged data set. (*C*) An example Burrows–Wheeler transform (BWT), matrix (BWM), and longest common prefix (LCP) array, with sequence IDs for each suffix shown (ID). A nonmaximal unique match (UM) is shown, and the uniqueness threshold for this match is found using the flanking LCP values. (*D*) A partial multi-MUM (in blue) is found in all-but-one sequence (excluded in red). Using two anchor sequences (red and orange), all-but-one partial MUMs can be computed using an augmented anchor-based merging method. (*E*) An example tree is shown, highlighting the string-based merging use case where partial multi-MUMs specific to internal nodes (starred) can be computed by merging subclade-based partitions up a tree.

6: **for** $i \leftarrow 0$ **to** $n-1$ **do**
7: **if** $T_1[i]>0$ **and** $T_2[i]>0$ **then** ▷ Positions covered by MUMs in both partitions
8: $T_{\text{new}}[i] \leftarrow \max(T_1[i], T_2[i])$
9: **if** $B_1[i]=1$ **or** $B_2[i]=1$ **then** ▷ Maximal in ≥1 partition
10: $m_1 \leftarrow \mathcal{M}_1[B_1.\text{rank}(i)]$
11: $m_2 \leftarrow \mathcal{M}_2[B_2.\text{rank}(i)]$
12: $m_{\text{new}} \leftarrow \text{ProcessMumPair}(m_1, m_2, i)$
13: **if** $m_{\text{new}}.\text{length} > T_{\text{new}}[i]$ **then**
14: $\mathcal{M}_{\text{merged}} \leftarrow \mathcal{M}_{\text{merged}} \cup \{m_{\text{new}}\}$
15: **end if**
16: **end if**
17: **end if**
18: **end for**
19: **return** $(\mathcal{M}_{\text{merged}}, T_{\text{new}})$

## String-based merging

The anchor-based merging strategy requires a common reference sequence across partitions. We now propose a method for multi-MUM merging that works for disjoint partitions with no sequences in common. Because this strategy uses the sequences of the multi-MUMs themselves as the basis for its merging decisions, we call it the string-based merging strategy.

We again store the `unique_threshold` array; as before, this array indicates how short a MUM or UM can be truncated before it loses uniqueness in a partition. However, we now store these values for each offset of each MUM yielding an array of size $O$(total length of MUMs), which is usually shorter than the length of the anchor reference. Additionally, we store this information for the reverse complement of each MUM or UM. This is used to construct the merged `unique_threshold` array for subsequent merging cycles.

To identify overlaps, we extract multi-MUM sequences from each of the $k$ partitions being merged and re-invoke Mumemto on those. That is, the collection of multi-MUMs from each partition becomes the new "genome" and we now find its multi-MUMs. When we detect an overlap between matches present in all $k$ partitions, we check the `unique_threshold` arrays for each to determine whether the multi-MUM should be kept as-is, truncated, or removed, similarly to the anchor-based method. Lastly, we build a new `unique_threshold` for the merged data set by computing an element-wise comparison over threshold values from the overlap regions across partitions. We use the reverse-complement thresholds when the overlap is on the negative strand of a MUM. Algorithm 2 details this procedure.

**Algorithm 2.** String-based MUM merging

**Input:** $k$ sorted arrays of maximal unique matches $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$
**Input:** $k$ `unique_threshold` arrays $\{T_1, \ldots, T_k\}$
**Output:** Combined MUMs $\mathcal{M}_{\text{merged}}$ and updated thresholds $T_{\text{new}}$

**for** $i \leftarrow 0$ **to** $k$ **do**
  $S_i \leftarrow \text{ExtractMUMSeq}(\mathcal{M}_i)$
**end for**
$\mathcal{M}_{\text{merged}} \leftarrow \emptyset$
$T_{\text{new}} \leftarrow \text{Array}(\emptyset)$
$\mathcal{O} \leftarrow$ **Mumemto**$(S_1, \ldots, S_k)$ ▷ Compute overlaps
**for** $o \in \mathcal{O}$ **do**
  $P \leftarrow o.\text{offsets}$ ▷ Overlap positions in each partition
  valid $\leftarrow$ true
  **for** $i \leftarrow 0$ **to** $k$ **do**
    **if** $T_i[P_i] \leq 0$ **or** $o.\text{length} \leq T_i[P_i]$ **then**
      valid $\leftarrow$ false
      **break**
    **end if**
  **end for**
  **if** valid **then**
    $m_{\text{new}} \leftarrow \text{ProcessMumPair}(o)$
    $\mathcal{M}_{\text{merged}} \leftarrow \mathcal{M}_{\text{merged}} \cup \{m_{\text{new}}\}$
    $t_{\text{cur}} \leftarrow \max\{T_i[p \cdots p + m_{\text{new}}.\text{length}]{:}i, p \in P\}$
      ▷ Element-wise maximum
    $\text{Extend}(T_{\text{new}}, t_{\text{cur}})$ ▷ Update thresholds
  **end if**
**end for**
**return** $(\mathcal{M}_{\text{merged}}, T_{\text{new}})$

### Extension to partial multi-MUMs

The method described so far works exclusively for multi-MUMs. In previous work (Shivakumar and Langmead 2025), we showed the utility of reporting other match types, for example, matches that appear in a large subset of the genomes but not all. We now describe an extension to the anchor-based merging method that enables computation of partial multi-MUMs appearing in $N-1$ sequences (all-but-one), as well as ideas for generalizing to the case of partial multi-MUMs appearing in at least $N-p$ sequences for small $p>1$.

To find partial multi-MUMs present in $N-1$ sequences, we must handle the case where a multi-MUM appears in all sequences except the anchor reference sequence. Anchor-based merging fails in this case because the MUM cannot be assigned an offset with respect to the anchor. To address this, we instead select *two* anchor sequences, one that serves as the primary anchor sequence for most partial MUMs, and a second that serves as a "backup" anchor sequence for the partial MUMs that do not appear in the primary anchor sequence. Additionally, we store a `unique_threshold` *for each anchor reference*. We compute partial multi-MUMs appearing in $N-1$ sequences for each partition (i.e., Mumemto flag `-k -1`). If a match appears in the first anchor reference, we store the `unique_threshold` at the first anchor offset. If a match does not appear in the first anchor sequence (which implies it must appear in all other sequences), we store the threshold information at the second anchor offset. We proceed with merging as normal, comparing the first and second anchor thresholds across partitions. However, we only process overlaps that are partial in exactly one partition, ensuring the final merged MUM appears in at least $N-1$ sequences.

To generalize to partial multi-MUMs appearing in at least $N-p$ sequences for $p>1$, we must now select a set of $p+1$ common anchor sequences and include all of them in all partitions. Assume that the anchor sequences have a priority order, with one being primary, another secondary, etc. Then each of these anchor references serves as a "backup" for the case of a partial multi-MUM that fails to appear in any of the higher-priority anchor references. Because the addition of each new anchor sequence adds overhead to the initial partition-wise MUM-finding step, this becomes impractical unless $p$ is small. We plan to implement this extension to partial MUMs in a future release of Mumemto.

## Results

We implemented MumemtoM as a part of the Mumemto software for computing multi-MUMs. Users can enable MumemtoM by running Mumemto with the `-M` option, which causes Mumemto to report the auxiliary `unique_threshold` array in addition to multi-MUMs. For anchor-based merging, this array uses $2n$ bytes, where $n$ is the length of the anchor reference sequence. For string-based merging, it uses $4m$ bytes, where $m$ is the total length of multi-MUMs.

To benchmark and apply the partition-merging method, we used assemblies from the Human Pangenome Reference Consortium (HPRC) (Liao et al. 2023), specifically 474 haplotype assemblies (available from Li 2025). To obtain chromosome-level assemblies for each haplotype, we performed reference-guided scaffolding against CHM13 using RagTag (default parameters) (Alonge et al. 2022). We only considered contigs which were included in the scaffolded assembly when referring to HPRC assemblies below.

### Time and memory tradeoffs

We implemented these methods in a workflow that (a) breaks the pangenome into partitions, (b) computes multi-UMs in each partition, and then (c) merges the per-partition matches into a global set of multi-MUMs. The user can trade between running time and peak memory usage by setting partition size and number of threads. Running parallel, per-partition Mumemto processes and merging the results reduces the running time but increases the peak memory footprint. Running a single Mumemto thread over each partition serially yields longer running time but a smaller memory footprint. Parallel processing with fewer threads than partitions can enable intermediate time–memory tradeoffs, however we have not yet implemented the load-balancing and scheduling required to handle these cases.

We measured the time and memory efficiency of running MumemtoM over a partitioned data set in parallel and serially for two data sets using anchor-based merging (see Table 1). The data sets were human (Liao et al. 2023) and *Arabidopsis thaliana* (Lian et al. 2024). Splitting 474 Chr 19 haplotypes into 40 partitions and processing those fully in parallel yielded an

**Table 1. Runtime and memory usage comparison between serial and parallel multi-MUM computation across different partition schemes**

| | | | Serial | | Parallel | |
|---|---|---|---|---|---|---|
| Data set | Partitions | Seqs per | Time (h) | Memory (GB) | Time (h) | Memory (GB) |
| Chr 19 HPRC ($N$ = 474) | 1 | 474 | 4.77 | 47.30 | – | – |
| | 5 | 96 | 5.00 | 14.99 | 1.11 | 71.71 |
| | 10 | 48 | 5.91 | 10.27 | 0.63 | 95.16 |
| | 20 | 24 | 7.21 | 7.69 | 0.40 | 134.53 |
| | 40 | 12 | 8.32 | **5.74** | **0.26** | 204.43 |
| *A. thaliana* ($N$ = 69) | 1 | 69 | 2.58 | 75.53 | – | – |
| | 5 | 15 | 3.75 | 29.15 | 0.80 | 137.52 |
| | 10 | 7 | 4.92 | 20.18 | 0.57 | 184.68 |
| | 20 | 4 | 6.48 | **14.58** | **0.40** | 260.23 |

We report execution time (in hours) and peak memory usage (in GB) for two data sets, Chr 19 haplotypes from the Human Pangenome Reference Consortium (HPRC) (Liao et al. 2023) and the *A. thaliana* pangenome (Lian et al. 2024). The lowest peak memory footprint and the fastest running times are bolded.

18.3× speedup (0.26 vs. 4.77 h for 1 thread) but used 4.3× more memory (190.39 GB vs. 44.05 GB for 1 thread). When partitioned and processed serially, the memory usage was reduced by 8.2× (5.35 GB vs. 44.05 GB for 1 thread), with a 1.7× increase in runtime (from 4.77 to 8.32 h). For *A. thaliana*, processing the 69 accession assemblies in 20 partitions in parallel yielded a 6.5× speedup (0.40 vs. 2.58 h for 1 thread) with a 3.4 × memory increase (242.36 GB vs. 70.34 GB for 1 thread). Run serially, a 5× memory reduction (from 70.34 GB to 13.58 GB) is achieved with a 2.5× increase in runtime (from 2.58 to 6.48 h).

The *A. thaliana* tradeoff is similar though smaller in magnitude compared to human assemblies. Recall that the anchor-based merging method introduces overhead by including the common anchor reference in each partition. When the sequences being indexed are highly similar, for example, human genome assemblies, then PFP compression can effectively mask this additional overhead. But for the more divergent *A. thaliana* data set, the additional overhead is more apparent, leading to the narrower trade-off space.

We also compared anchor-based and string-based merging in both time and memory (Supplemental Table S1). Though string-based merging was slightly faster due to a smaller partition size without the anchor sequence, the merging step is slower and more memory-intensive, introducing overhead in comparison to the anchor-based method due to finding overlaps between the multi-MUM sequences.

## Anchor-based merging across 474 human haplotypes

The Human Pangenome Reference Consortium (Liao et al. 2023) released 474 human haplotype assemblies (available at Li 2025), totaling ~1.4 Tbp (or 2.8 Tbp with reverse complements). Running Mumemto over the entire collection, even with PFP compression, would be infeasible, with a total PFP size of 145.6 GB, requiring over 2.7 TB of memory. However, we note that the primary bottleneck in this case is the size of the parse rather than the dictionary, which recursive extensions to PFP aim to address (Ferro et al. 2024).

With CHM13 as the common coordinate system for merging, we divided the HPRC data set into nine partitions (53–54 assemblies in each) and computed multi-MUMs within each subset. We note that although the choice of anchor reference does not affect the final multi-MUM output, CHM13 provides a convenient choice as the most complete T2T assembly for humans. We defined multi-MUM coverage as the proportion of bases in a sequence included in a multi-MUM and compute the average coverage across each sequence in the pangenome. Merging across subsets, we found 20,914,049 multi-MUMs across all 474 assemblies, totaling ~2 Gbp of nonoverlapping sequence in CHM13 and an average coverage of 73.2% across haplotypes. The updated HPRC v2 data set multi-MUM coverage is smaller than that of the previously released v1 (89 haplotypes) (84.8% coverage, totaling 2.43 Gbp of sequence). This is expected, as the data set grows, deletions and incomplete assemblies in newly added haplotypes will lower overall multi-MUM coverage. However, including partial MUMs in all-but-one sequences in HPRC v1 increases MUM coverage to 86.2%, motivating the merging algorithm extension for partial MUMs.

We combined collinear MUMs separated by <100 bp into "blocks," as described in previous work (Shivakumar and Langmead 2025). We found 254,440 collinear MUM blocks, covering 76.4% of CHM13. A collinear block contained 82 multi-MUMs on average, with over two-thirds of the gaps between collinear multi-MUMs being a single base wide (consistent with previous work; Shivakumar and Langmead 2025). Run serially, the MumemtoM pipeline would take 339.33 h (a little over 2 weeks), using 448.20 GB of memory. Using MumemtoM to run this same process across nine compute nodes in parallel, the pipeline took 40.36 h using roughly 434 GB of memory per node. Mumemto is the only method that can currently compute multi-sequence exact matches across this full set of 474 haplotypes. Furthermore, MumemtoM allows for incorporation of newly released human assemblies, without needing to recompute multi-MUMs over the assemblies indexed previously.

## Merging multi-MUMs across a phylogenetic tree

In cases where a pangenome spans individuals of multiple species or spans multiple diverse accessions, the sequences' natural groupings may be determined by a tree. For instance, the recent *A. thaliana* pangenome (Lian et al. 2024) spans genomes from geographically diverse accessions, which can be related in a phylogenetic tree. In this case, we can use string-based merging to allow MumemtoM to follow the structure of tree bottom-up, so that the intermediate results correspond to the tree's internal nodes (Fig. 1E). This is possible because string-based merging allows sibling subtrees to be merged without a common anchor reference. An additional benefit is that the intermediate computations will always be over clades of the tree, which consist of more similar genome sets than if we had partitioned arbitrarily. This maximizes PFP compression in each subproblem.

We applied this strategy to a phylogeny and collection of *A. thaliana* assemblies from diverse geographical regions (Lian et al. 2024) (Fig. 2A). Figure 2B shows the multi-MUM synteny across the full data set. Multi-MUM coverage across the full data set drops in the highly variable centromeric region; however, much of the synteny is revealed by local partition-specific MUMs. Highlighting these partition-specific MUMs reveals novel structural variation local to geographically isolated subgroups. For instance, a 778 kbp inversion is present in the Chr 4 centromere of two Madeiran accessions (Fig. 2D), and a complex inversion and

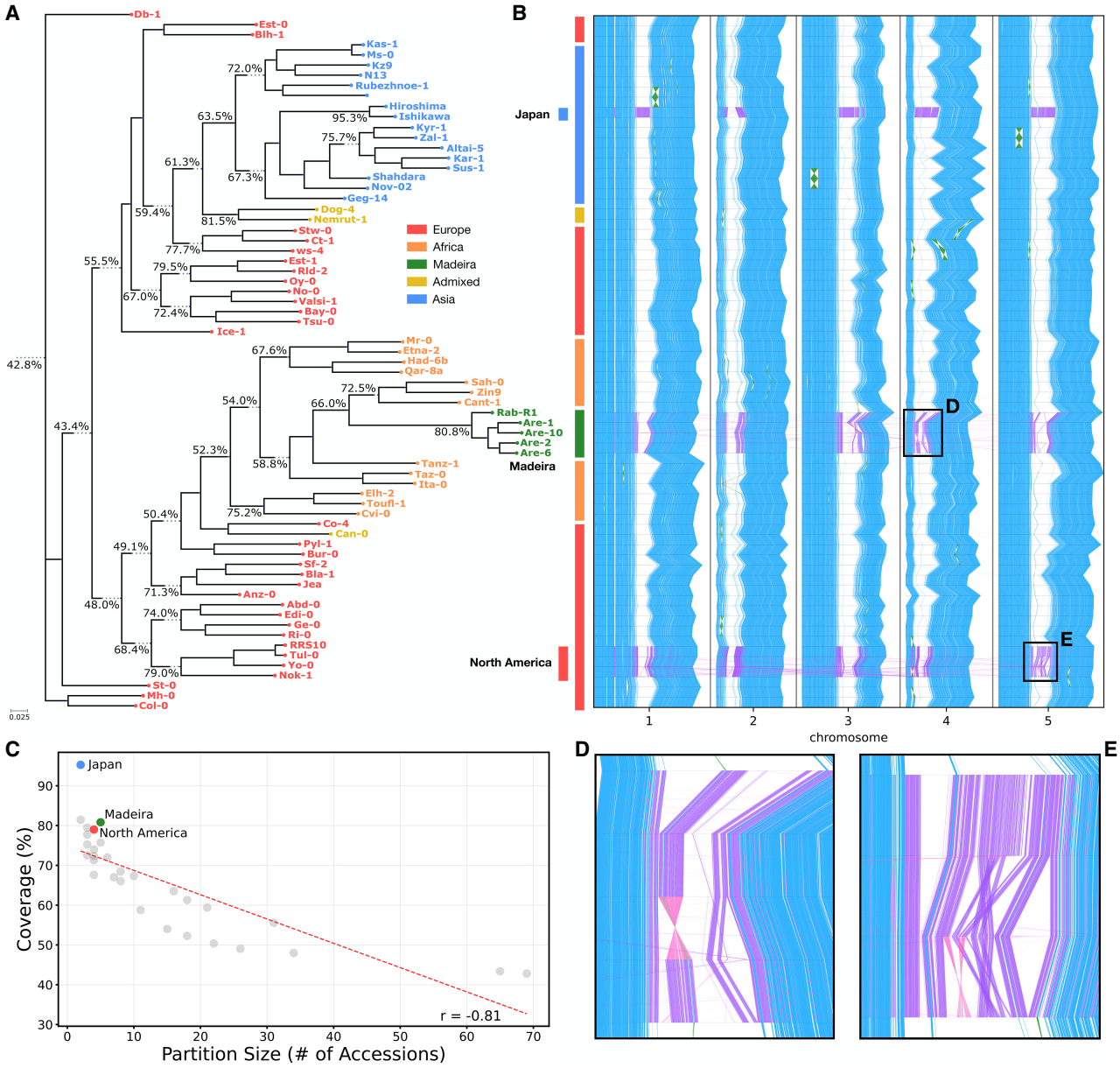


**Figure 2.** Applying MumemtoM to the *A. thaliana* pangenome reveals geographic region–specific variation. (*A*) Phylogeny of geographically diverse *A. thaliana* accessions (Lian et al. 2024), with broad geographical regions colored. Internal nodes are labeled with the coverage of partial multi-MUMs across the leaves of each node. Internal node partial MUMs are computed by merging subtree-based partitions progressively up the phylogeny. (*B*) Global multi-MUM synteny across the full data set shown in blue (with inversions in green). Global MUMs are computed by merging all partitions together (representing the root node). Additionally, three geographically distinct subgroups are highlighted and partition-specific multi-MUMs (in purple, with inversions in pink) reveal local structural variation in centromeric regions. Two examples are shown in zoomed-in panels: (*D*) a novel inversion located in accessions from the Madeira islands, and (*E*) a complex inversion/rearrangement in the North American subgroup Chr 5 centromere. (*C*) Coverage of internal nodes in the phylogeny compared to the number of leaves under node. Three highlighted subgroups show particularly high coverage, likely due to geographical isolation (Lian et al. 2024).

rearrangement region is revealed by partial MUMs among accessions from North America in the Chr 5 centromere (Fig. 2E).

Across the full data set, multi-MUM coverage is 42.8%, which corresponds well with previously reported measures of collinearity (Lian et al. 2024). As expected, MUM coverage drops as partitions are merged up the tree (Fig. 2A). Lian et al. also reported increased collinearity in certain small geographical groups. Similarly, we found an increased multi-MUM coverage of 95% in the Japanese accessions, 79% in the North American subgroup, and 81% in the Madeiran subgroup (Fig. 2C). By computing multi-MUMs in a hierarchical manner, we can find subgroup-specific conserved regions and data set–wide multi-MUMs with a single, partitioned Mumemto run.

## Merging enables interspecific multi-MUM computation

PFP is designed for repetitive inputs like pangenomes consisting of many same-species genomes. Interspecific data sets have greater

sequence diversity and are less compressible. Consistent with this, we observed that the prefix-free parse of 28 primate genome assemblies (from a curation by Hecker and Hiller 2020 and the T2T primate project; Yoo et al. 2025) was larger than the original sequence files. As a result, computing multi-MUMs across the 28 primate assemblies (~150 Gbp) with nonpartitioned Mumemto required over 2 TB of memory. Using MumemtoM, however, we can reduce the memory required for computing multi-MUMs across diverse, interspecific data sets like this one. By partitioning the genomes into four groups and processing the partitions serially, we were able to find multi-MUMs across the primates in 842 GB of memory and 37.4 h of running time.

To further explore this new multi-MUM data set, we studied the most conserved sequences across the primate species. Similar to ultraconserved elements (UCEs) (Bejerano et al. 2004; Hecker and Hiller 2020; Cummins et al. 2024), multi-MUMs across diverse genomes can be used to identify regions of evolutionary conservation. We identified collinear blocks from multi-MUMs present across 29 primate assemblies (including GRCh38 and CHM13). We considered all collinear blocks (with maximum space between MUMs of 100 bp) and noncollinear MUMs longer than 50 bp to be conserved elements across primates. We found 10,382 such elements, totaling ~1.56 Mbp of sequence. The average element length was 150 bp, with 198 elements longer than 500 bp. Most of these conserved sequences were located in non-protein-coding regions (83.81%), and 42.13% in intergenic regions (using MANE 1.4 to define genic and protein-coding regions; Morales et al. 2022). We also found 1410 (13.6%) elements spanned intron–exon boundaries. These results are consistent with previously reported genomic distributions of ultraconserved elements in humans (Cummins et al. 2024), highlighting the potential for interspecific multi-MUMs for identifying UCEs.

## Discussion

MumemtoM enables a partition-based, memory-efficient, and parallelizable Mumemto workflow, which now scales to hundreds of human genomes. These algorithms also simplify incorporating newly assembled genomes by dynamically updating multi-MUMs through merging, rather than recomputing MUMs across the full, updated data set. This combination of scalability and updatability makes Mumemto well-suited to a future where long-read sequencing projects produce steady streams of new assemblies.

The two merging schemes discussed each have their own limitations. Anchor-based merging is faster because finding multi-MUM overlaps is easily done with numerical comparisons in a common coordinate system. On the other hand, this requires that a common anchor reference be present across the partitions, adding both time and space overhead. This is still suitable when the goal is to simply compute multi-MUMs over a large data set such as HPRC v2. String-based merging is slower and more memory-intensive than anchor-based merging, because it requires an additional invocation of Mumemto to identify overlaps. However, it enables tree-based merging to find partial multi-MUMs at internal nodes, revealing conservation and variation between subgroups in the data set.

String-based merging relies on running Mumemto over the MUM sequences themselves to identify overlaps. We note that any exact match method could be used to find these overlaps (e.g., Mummer4; Marçais et al. 2018). We used Mumemto itself, in order to limit external dependencies. This enables merging multiple partitions at once, although we find that memory usage of the merging step can become the limiting factor for a large number of partitions. In the future, it will be important to investigate if another exact-matching method can accelerate string-based merging and close the speed gap with anchor-based merging. In addition, the merging step could be parallelized by merging in a progressive manner because the order of merging is not important, speeding up multipartition merging for both schemes.

Both merging methods are limited to computing strict multi-MUMs; however, we described an extension (not yet implemented) that enables partial multi-MUM finding in all-but-a-few sequences. A question for future work is whether a more sophisticated method could enable computing general partial multi-MUMs across any subsets.

We show that MumemtoM enables scalability to growing pangenome collections and expands the applicability of Mumemto to larger, more diverse data sets. This increases the scope for exploration of genomic conservation and variation and highlights the potential for Mumemto as a core method for future pangenomics and comparative genomics research.

## Code availability

The methods presented here are implemented in the Mumemto software (v1.3.0), available open-source at GitHub (https://github.com/vikshiv/mumemto). Mumemto v1.3.0 was used for all analyses and the source code is available as Supplemental Code.

## Competing interest statement

The authors declare no competing interests.

## Acknowledgments

We thank Christina Boucher and Travis Gagie for algorithmic discussions, Kuan-Hao Chao for valuable discussions on human genome annotations, and Korbinian Schneeberger and Qichao Lian for providing phylogenies for the *A. thaliana* pangenome. This work was carried out at the Advanced Research Computing at Hopkins (ARCH) core facility (rockfish.jhu.edu), supported by the National Science Foundation (NSF) grant OAC 1920103. V.S.S. and B.L. are supported by National Institutes of Health grants R35GM139602 (from the National Institute of General Medical Sciences) and R56HG01386523 (from the National Human Genome Research Institute).

*Author contributions*: V.S.S. and B.L. conceived the project. V.S.S. wrote the software and conducted the experiments. V.S.S. and B.L. wrote the manuscript.

## References

Abouelhoda MI, Kurtz S, Ohlebusch E. 2004. Replacing suffix trees with enhanced suffix arrays. *J Discrete Algorithms* **2:** 53–86. doi:10.1016/S1570-8667(03)00065-0

Alonge M, Lebeigle L, Kirsche M, Jenike K, Ou S, Aganezov S, Wang X, Lippman ZB, Schatz MC, Soyk S. 2022. Automated assembly scaffolding using RagTag elevates a new tomato system for high-throughput genome editing. *Genome Biol* **23:** 258. doi:10.1186/s13059-022-02823-7

Bejerano G, Pheasant M, Makunin I, Stephen S, Kent WJ, Mattick JS, Haussler D. 2004. Ultraconserved elements in the human genome. *Science* **304:** 1321–1325. doi:10.1126/science.1098119

Boucher C, Gagie T, Kuhnle A, Langmead B, Manzini G, Mun T. 2019. Prefix-free parsing for building big BWTs. *Algorithms Mol Biol* **14:** 13. doi:10.1186/s13015-019-0148-5

Brown NK, Shivakumar VS, Langmead B. 2025. Improved pangenomic classification accuracy with chain statistics. In *Research in computational*

*molecular biology* (ed. Sankararaman S), pp. 190–208. Springer Nature Switzerland, Cham, Switzerland.
Cummins M, Watson C, Edwards RJ, Mattick JS. 2024. The evolution of ultraconserved elements in vertebrates. *Mol Biol Evol* **41:** msae146. doi:10.1093/molbev/msae146
Darling AE, Mau B, Perna NT. 2010. progressiveMauve: Multiple genome alignment with gene gain, loss and rearrangement. *PLoS One* **5:** e11147. doi:10.1371/journal.pone.0011147
Deogun JS, Yang J, Ma F. 2004. Emagen: an efficient approach to multiple whole genome alignment. In *Proceedings of the Second Conference on Asia-Pacific Bioinformatics*, Dunedin, New Zealand, Vol. 29, pp. 113–122.
Ferro E, Oliva M, Gagie T, Boucher C. 2024. Building a pangenome alignment index via recursive prefix-free parsing. *iScience* **27:** 110933. doi:10.1016/j.isci.2024.110933
Garg V, Bohra A, Mascher M, Spannagl M, Xu X, Bevan MW, Bennetzen JL, Varshney RK. 2024. Unlocking plant genetics with telomere-to-telomere genome assemblies. *Nat Genet* **56:** 1788–1799. doi:10.1038/s41588-024-01830-7
Hecker N, Hiller M. 2020. A genome alignment of 120 mammals highlights ultraconserved element variability and placenta-associated enhancers. *GigaScience* **9:** giz159. doi:10.1093/gigascience/giz159
Hufford MB, Seetharam AS, Woodhouse MR, Chougule KM, Ou S, Liu J, Ricci WA, Guo T, Olson A, Qiu Y, et al. 2021. De novo assembly, annotation, and comparative analysis of 26 diverse maize genomes. *Science* **373:** 655–662. doi:10.1126/science.abg5289
Kille B, Nute MG, Huang V, Kim E, Phillippy AM, Treangen TJ. 2024. Parsnp 2.0: scalable core-genome alignment for massive microbial datasets. *Bioinformatics* **40:** btae311. doi:10.1093/bioinformatics/btae311
Li H. 2025. A collection of high-quality human assemblies [Data set]. In BWT construction and search at the terabase scale (3.0, Vol. 40, Number 12, p. btae717). Zenodo. https://doi.org/10.5281/zenodo.14854401
Lian Q, Huettel B, Walkemeier B, Mayjonade B, Lopez-Roques C, Gil L, Roux F, Schneeberger K, Mercier R. 2024. A pan-genome of 69 *Arabidopsis thaliana* accessions reveals a conserved genome structure throughout the global species range. *Nat Genet* **56:** 982–991. doi:10.1038/s41588-024-01715-9
Liao W-W, Asri M, Ebler J, Doerr D, Haukness M, Hickey G, Lu S, Lucas JK, Monlong J, Abel HJ, et al. 2023. A draft human pangenome reference. *Nature* **617:** 312–324. doi:10.1038/s41586-023-05896-x
Lucà S, Masillo F, Lipták Z. 2025. Measuring genomic data with prefix-free parsing. *Comput Biol Chem* **122:** 108870. doi:10.1016/j.compbiolchem.2025.108870
Marçais G, Delcher AL, Phillippy AM, Coston R, Salzberg SL, Zimin A. 2018. Mummer4: A fast and versatile genome alignment system. *PLoS Comput Biol* **14:** e1005944. doi:10.1371/journal.pcbi.1005944
Morales J, Pujar S, Loveland JE, Astashyn A, Bennett R, Berry A, Cox E, Davidson C, Ermolaeva O, Farrell CM, et al. 2022. A joint NCBI and EMBL-EBI transcript set for clinical genomics and research. *Nature* **604:** 310–315. doi:10.1038/s41586-022-04558-8
Olbrich J, Büchler T, Ohlebusch E. 2025. Generating multiple alignments on a pangenomic scale. *Bioinformatics* **41:** btaf104. doi:10.1093/bioinformatics/btaf104
Shivakumar VS, Langmead B. 2025. Mumemto: Efficient maximal matching across pangenomes. *Genome Biol* **26:** 169. doi:10.1186/s13059-025-03644-0
Taylor DJ, Eizenga JM, Li Q, Das A, Jenike KM, Kenny EE, Miga KH, Monlong J, McCoy RC, Paten B, et al. 2024. Beyond the human genome project: the age of complete human genome sequences and pangenome references. *Annu Rev Genomics Hum Genet* **25:** 77–104. doi:10.1146/annurev-genom-021623-081639
Treangen TJ. 2008. "Novel computational methods for large scale genome comparison." PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Departamento de Lenguajes y Sistemas Informáticos, Programa de Doctorado en Software.
Treangen TJ, Ondov BD, Koren S, Phillippy AM. 2014. The Harvest suite for rapid core-genome alignment and visualization of thousands of intraspecific microbial genomes. *Genome Biol* **15:** 524. doi:10.1186/s13059-014-0524-x
Yoo D, Rhie A, Hebbar P, Antonacci F, Logsdon GA, Solar SJ, Antipov D, Pickett BD, Safonova Y, Montinaro F, et al. 2025. Complete sequencing of ape genomes. *Nature* **641:** 401–418. doi:10.1038/s41586-025-08816-3

*Received May 16, 2025; accepted in revised form November 5, 2025.*