Supplementary Information for: GTcomplex: Spatial indexing-powered search and alignment of macromolecular complexes

Mindaugas Margelevičius[1*]

[1*]Institute of Biotechnology, Life Sciences Center, Vilnius University, Vilnius, Lithuania.

Corresponding author(s). E-mail(s): mindaugas.margelevicius@bti.vu.lt;

Contents

S1 Supplementary results

S1.1 Supplementary figures

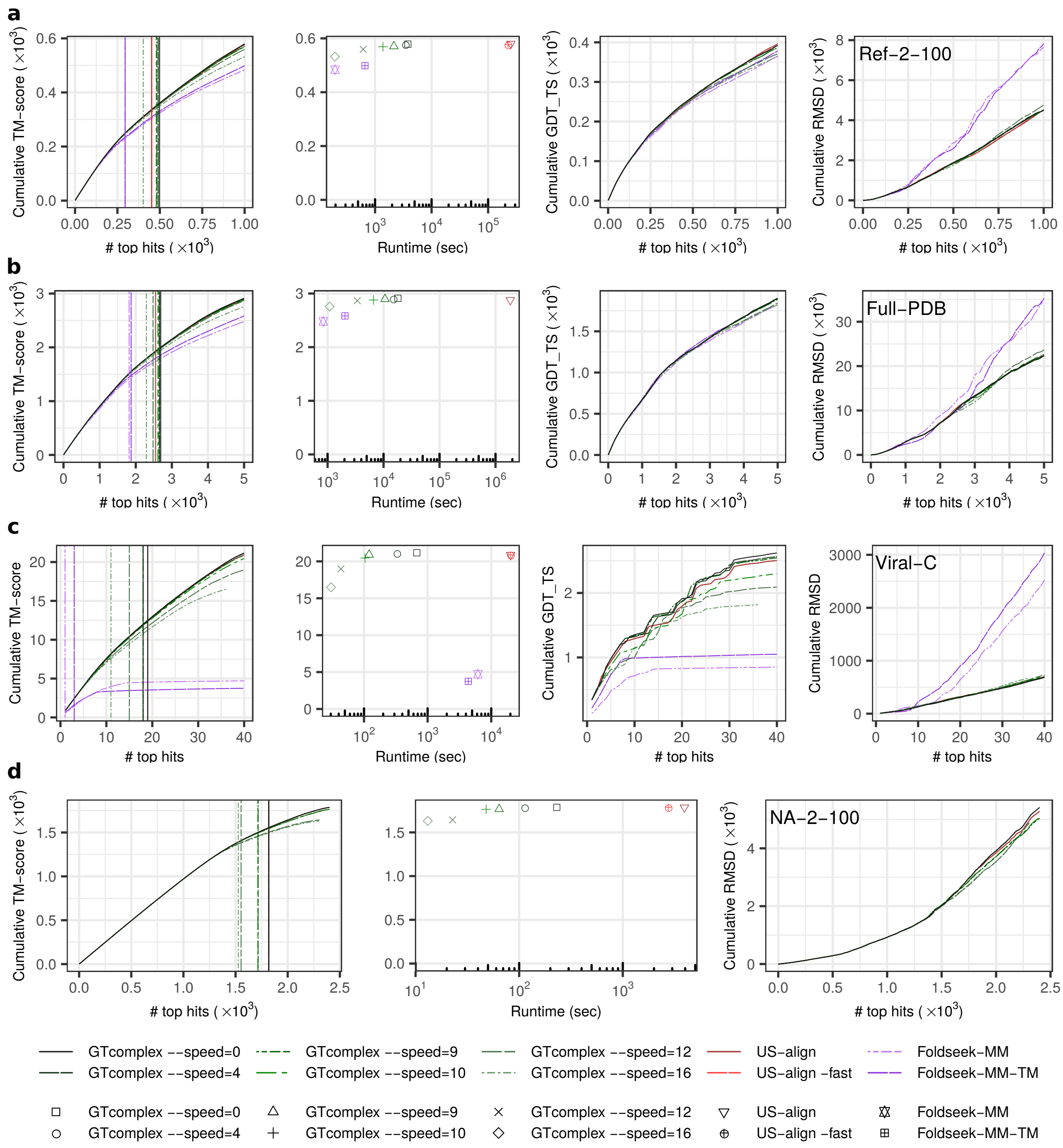

Fig. S1 Benchmarking results on the Ref-2-100 (a), Full-PDB (b), Viral-C (c), and NA-2-100 (d) datasets, evaluated using TM-score and GDT_TS metrics normalized by query length. Left panels show cumulative TM-score as a function of the number of top-ranked alignments (ranked by TM-score). Vertical lines indicate the number of alignments with TM-score ≥ 0.5. Middle-left panels in (a–c) and the central panel in (d) plot cumulative TM-score versus runtime (seconds). Middle-right panels in (a–c) plot cumulative GDT_TS, and right panels plot cumulative RMSD, each as a function of the number of top-ranked alignments. Alignments are ranked by TM-score normalized by the length of the query complex. In the left, middle-left (a–c), and central (d) panels, the TM-scores correspond to TM-align recalculations of the alignments, and the ranking of alignments is based on these recalculated TM-scores.

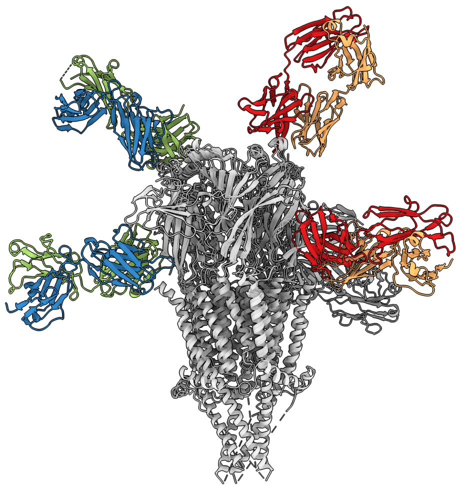

Fig. S2 Example of spatial mismatches produced by Foldseek-MM and Foldseek-MM-TM. The figure shows the superimposition of complexes 8ssz and 9gu1 as aligned by Foldseek-MM-TM. Spatially distant chains that are incorrectly paired and aligned, resulting in a large RMSD of 36.60 Å (36.96 Å for Foldseek-MM), are highlighted in matching colors. All other chains are shown in gray.

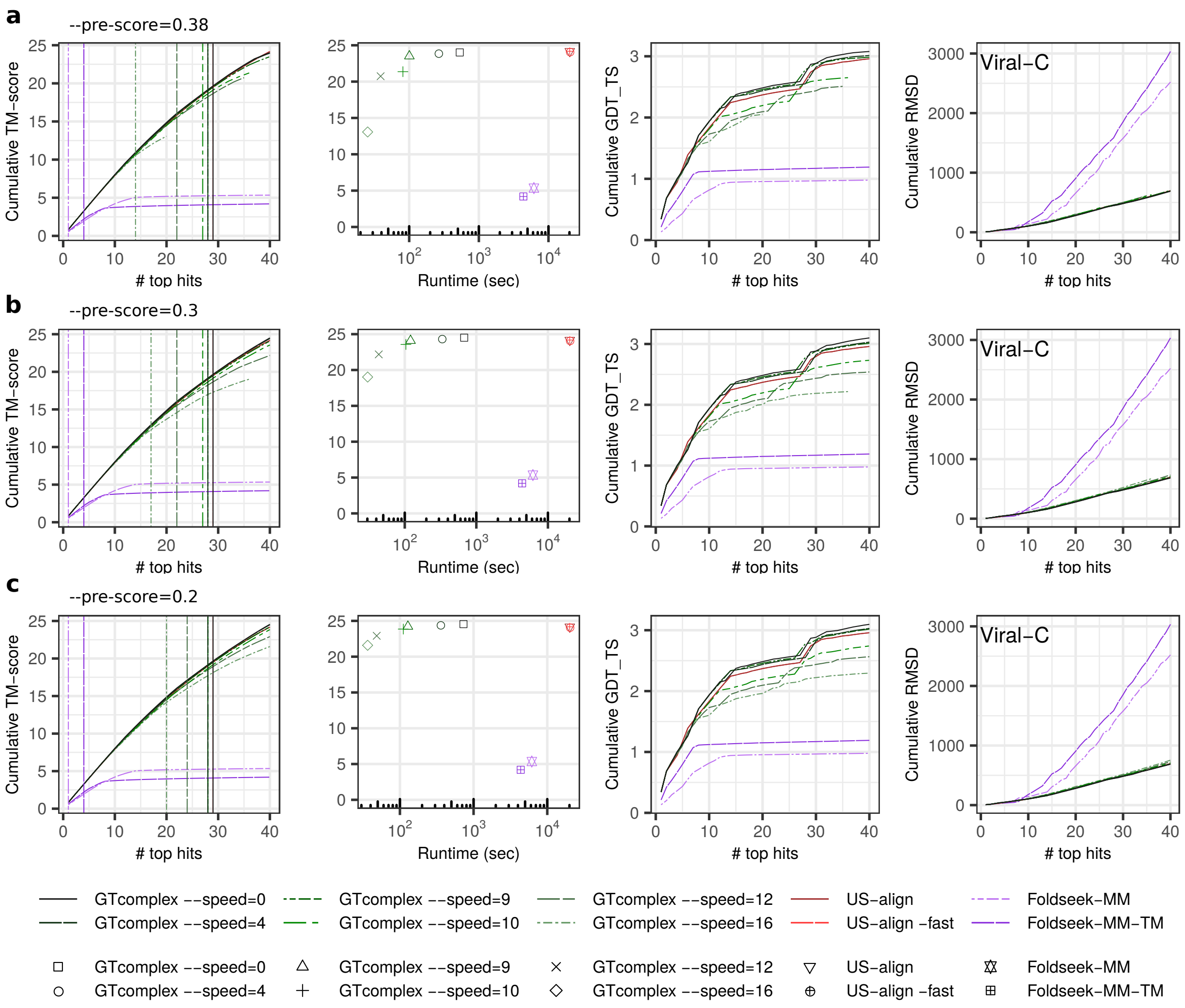

Fig. S3 Benchmarking results on the Viral-C dataset for three different prefiltering settings: `--pre-score=0.38` (a), `--pre-score=0.3` (b), and `--pre-score=0.2` (c). Alignments are ranked by TM-score normalized by the length of the shorter complex. Panel definitions follow those in Fig. 2 of the main text.

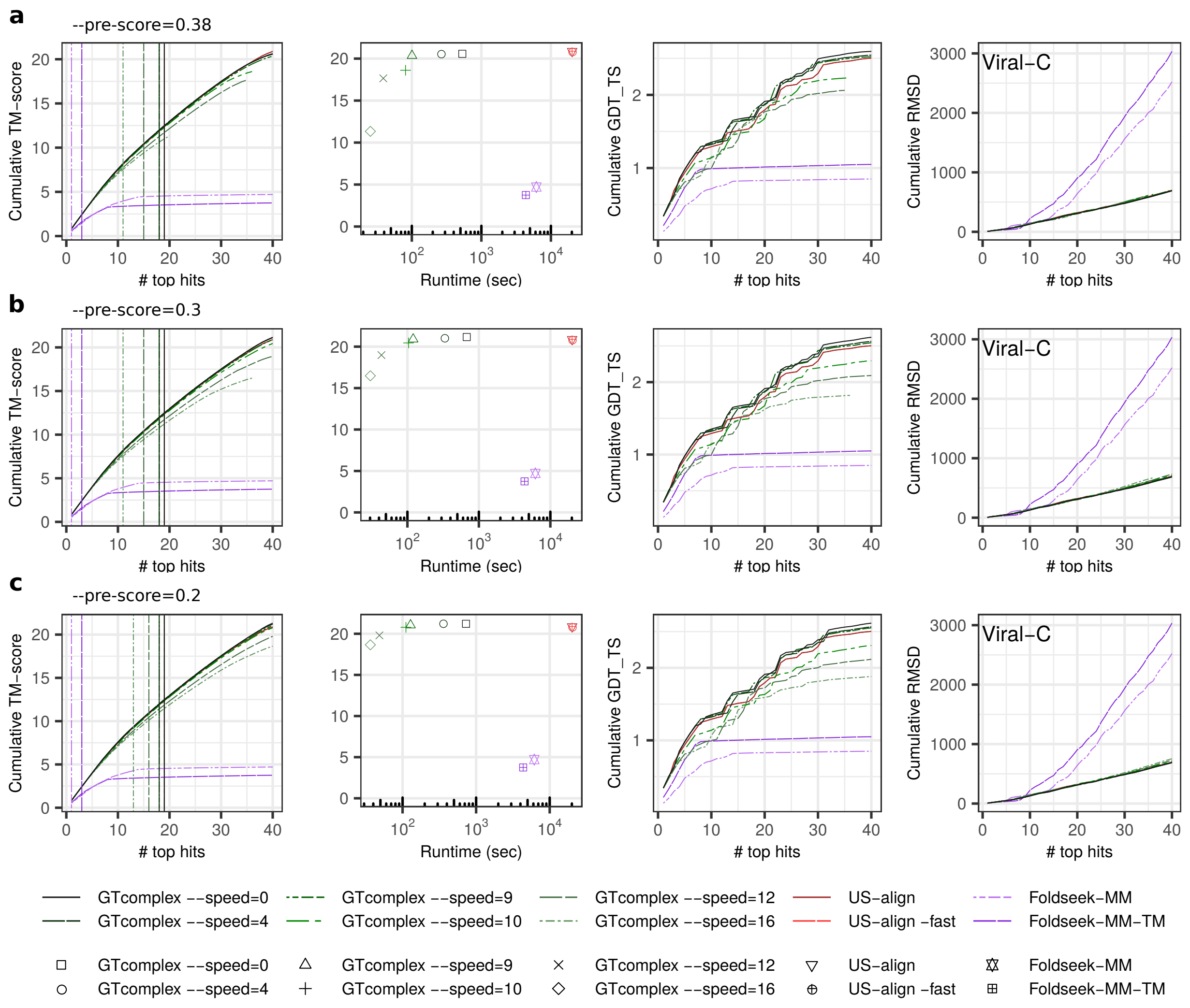

Fig. S4 Benchmarking results on the Viral-C dataset for three different prefiltering settings: `--pre-score=0.38` (a), `--pre-score=0.3` (b), and `--pre-score=0.2` (c). Alignments are ranked by TM-score normalized by the length of the query complex. Panel definitions follow those in Fig. S1.

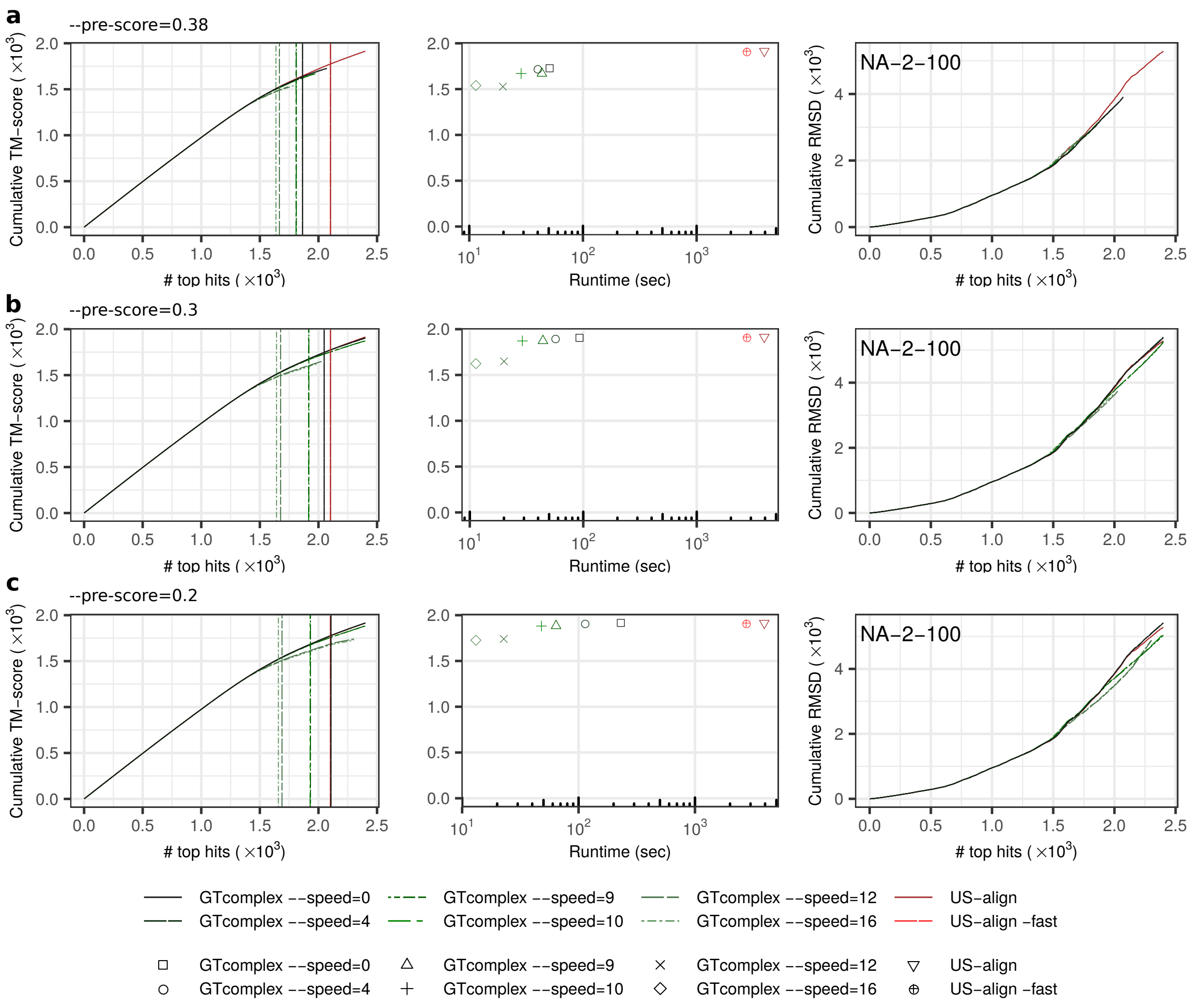

Fig. S5 Benchmarking results on the NA-2-100 dataset for three different prefiltering settings: `--pre-score=0.38` (a), `--pre-score=0.3` (b), and `--pre-score=0.2` (c). Alignments are ranked by TM-score normalized by the length of the shorter complex. Panel definitions follow those in Fig. 2d of the main text.

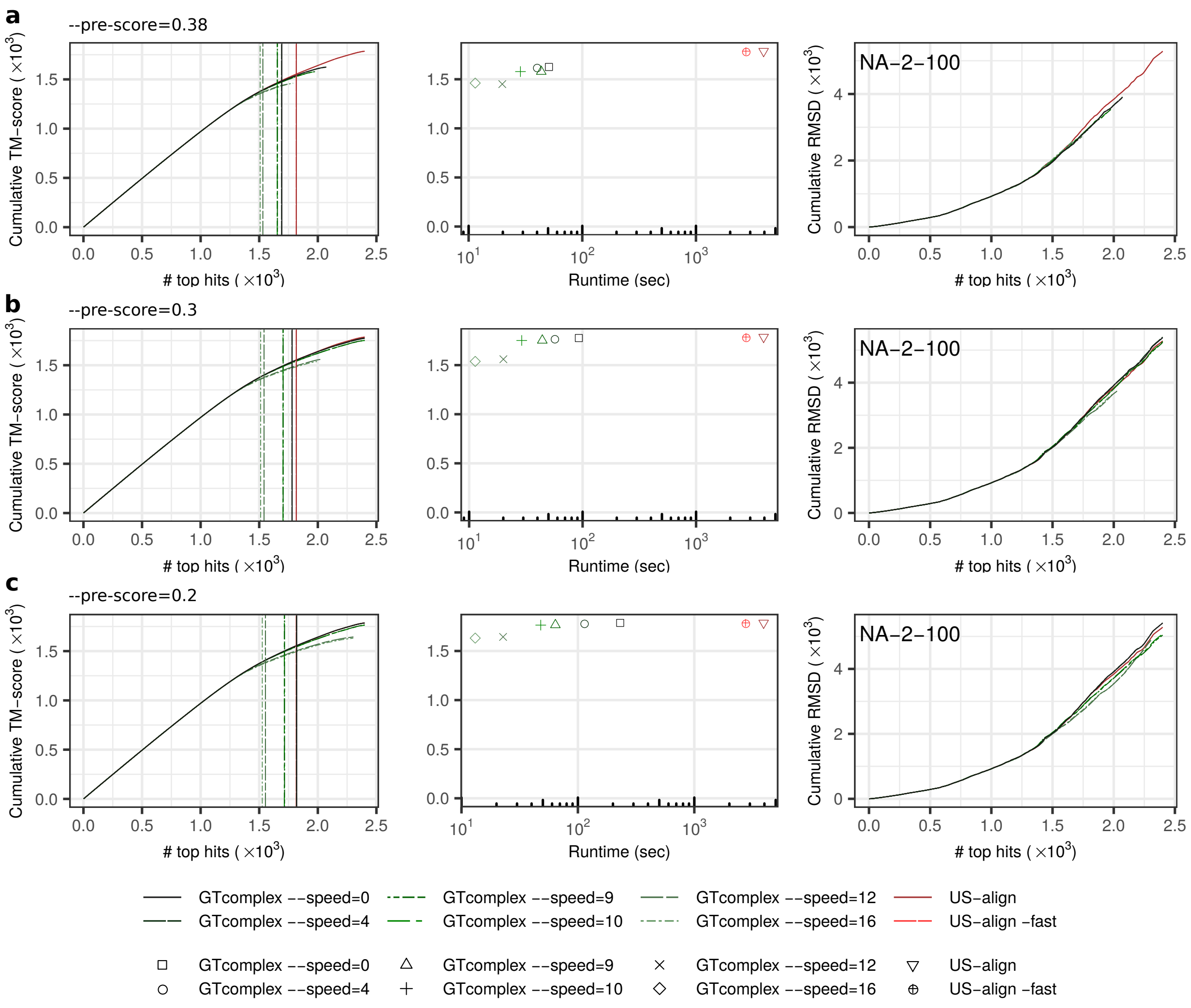

Fig. S6 Benchmarking results on the NA-2-100 dataset for three different prefiltering settings: `--pre-score=0.38` (a), `--pre-score=0.3` (b), and `--pre-score=0.2` (c). Alignments are ranked by TM-score normalized by the length of the query complex. Panel definitions follow those in Fig. S1d.

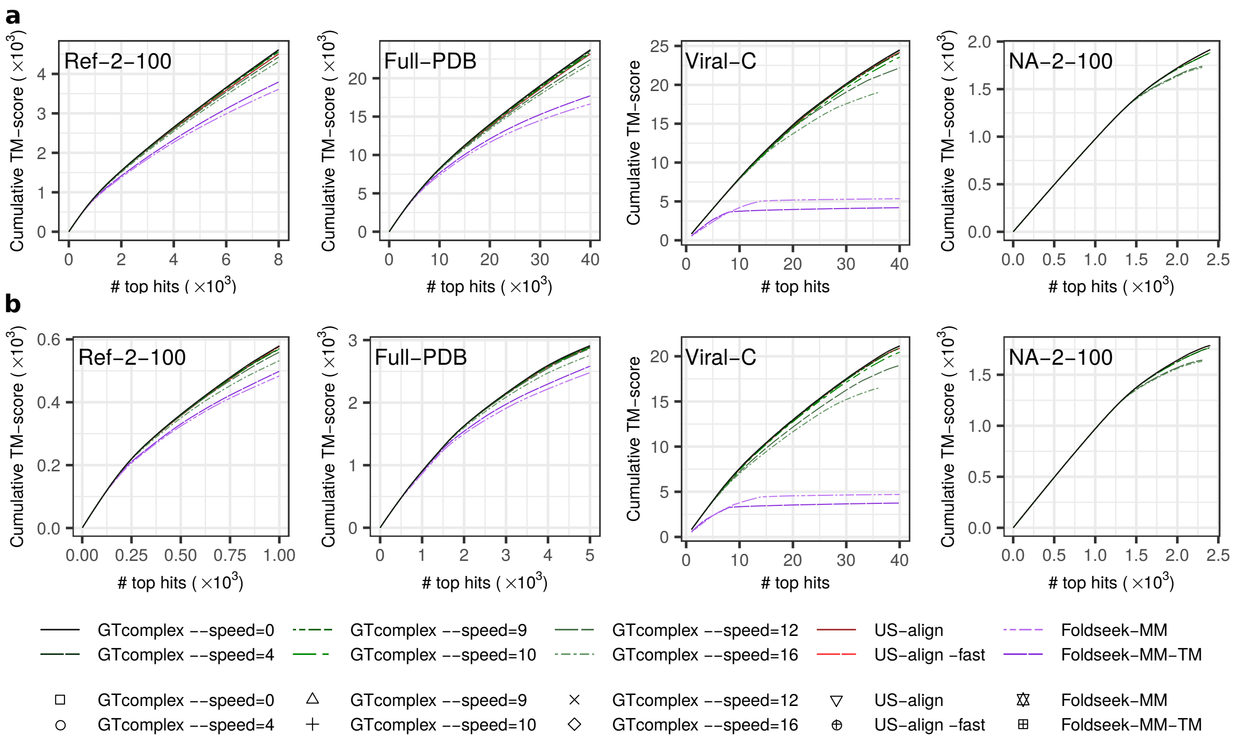

Fig. S7 Rate of accurate alignments, evaluated by cumulative TM-score as a function of the number of top-ranked alignments for the Ref-2-100, Full-PDB, Viral-C, and NA-2-100 datasets. Unlike the left panels of Fig. 2 in the main text and Fig. S1, here TM-scores computed by each tool are used to rank alignments. a, Alignments ranked by TM-score normalized by the length of the shorter complex. b, Alignments ranked by TM-score normalized by the length of the query complex.

S2 Supplementary methods

S2.1 Efficient identification of optimal superpositions

This section describes the core algorithms that enable GTcomplex to identify optimal structural superpositions of macromolecular complexes. To efficiently explore vast superposition space and converge on the most consistent superpositions, GTcomplex employs spatial indexing to generate candidate transformations, followed by iterative alignment, chain assignment, and refinement steps. Highly customizable, these algorithms enable sensitive and scalable structure alignment across large datasets.

The top-level algorithm is presented in Algorithm 1. It accepts as input two batches of macromolecular complex structures, associated structural descriptors, and user-defined parameters, and it produces as output optimal superpositions, expressed as rigid-body transformation matrices $(R, \mathbf{t})$, for all pairs of structures drawn from distinct batches. The two batches, representing the sets of query and subject structures, are processed in parallel.

In the algorithmic specification, bold lowercase letters denote vectors, and italic uppercase letters denote multi-dimensional matrices or collections. The operator $[\cdot]_{\text{mut}}$ indicates a mutually exclusive operation preventing concurrent access to the same data by different threads.

Algorithm 1 Find optimal structural superpositions through deep search

```
1: procedure DeepSuperpositionSearch($n_Q$, $n_S$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $l_f$, $L$, $\tilde{L}$, $C$, $\tilde{C}$, $T$, $\tilde{T}$, $n_{its}$, $n_{brn}$, $n_{rfn}$, $c_{thr}$)
2:     Calculate $\Lambda$ using Eq. (S1) in parallel for all $0 \le q < n_Q$ and $0 \le s < n_S$
3:     Determine $n_{ext}$ from $\max_q l_q$, $\max_s \tilde{l}_s$ and depth specification
4:     Set $n_{tfm}$ to 32, 64, or 96 depending on depth specification
5:     CalculateLocalSimilarity($n_Q$, $n_S$, $n_{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $l_f$, $\Lambda$, $c_{thr}$)          ▷ Algorithm 2
6:     CalculateLocalXCovariances($n_Q$, $n_S$, $n_{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $l_f$, $\Lambda$, $C$, $\tilde{C}$)          ▷ Algorithm 3
7:     CalculateTransformations($n_Q$, $n_S$, $n_{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $b_{dyn}$=1)          ▷ Algorithm 4
8:     for $i = 0, \ldots, n_{its} - 1$ do
9:         AlignInConstantTime($n_Q$, $n_S$, $n_{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $C$, $\tilde{C}$, $T$, $\tilde{T}$, $c_{thr}$, $[i+1 < n_{its}]$)          ▷ Algorithm 5
10:        CalculateXCovariancesA($n_Q$, $n_S$, $n_{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$)          ▷ Algorithm 6
11:        CalculateTransformations($n_Q$, $n_S$, $n_{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $[i+1 < n_{its}]$)          ▷ Algorithm 4
12:    end for
13:    CalculateApproxScores($n_Q$, $n_S$, $n_{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $L$, $\tilde{L}$)          ▷ Algorithm 7
14:    GetTopNTransformations($n_Q$, $n_S$, $n_{ext}$, $n_{tfm}$, $b_{srt}$=0)          ▷ Algorithm 8
15:    ComputeChainTMscores($n_Q$, $n_S$, $n_{tfm}$, $L$, $\tilde{L}$, $c_{goc}$=0)          ▷ Algorithm 9
16:    MakeChain2ChainAssignments($n_Q$, $n_S$, $n_{tfm}$, $L$, $\tilde{L}$)          ▷ Algorithm 10
17:    GetTopNTransformations($n_Q$, $n_S$, $n_{tfm}$, $n_{brn}$, $b_{srt}$=1)          ▷ Algorithm 8
18:    OptimizeSelectedAlignments($n_Q$, $n_S$, $n_{brn}$, $C$, $\tilde{C}$, $b_{runDP}$=1)          ▷ Algorithm 11
19:    RefineBestAlignments($n_Q$, $n_S$, $n_{rfn}$, $L$, $\tilde{L}$, $C$, $\tilde{C}$, $c_{goc}$=−0.6)          ▷ Algorithm 12
20:    RefineBestAlignments($n_Q$, $n_S$, $n_{rfn}$, $L$, $\tilde{L}$, $C$, $\tilde{C}$, $c_{goc}$=0)          ▷ Algorithm 12
21: end procedure
```

The parameters used in Algorithm 1 are defined as follows. n_Q and n_S denote the numbers of query and subject complex structures in their respective batches. $\mathbf{l}$ and $\tilde{\mathbf{l}}$ represent the lengths of the query and subject complexes, respectively. l_f specifies the context size used for local similarity evaluation and the generation of initial local alignment-based superpositions.

L and $\tilde{L}$ denote the chain-level lengths of the query and subject complexes, respectively, where L_{q,h_q} is the length of chain h_q of query complex q, and $\tilde{L}_{s,\tilde{h}_s}$ is defined analogously for subject complex s. C and $\tilde{C}$ represent the Cartesian coordinates of the query and subject structures. Specifically, $C_{q,p_q} \in \mathbb{R}^{3\times1}$ denotes the coordinates of residue or nucleotide p_q in query structure q, and $\tilde{C}_{s,\tilde{p}_s} \in \mathbb{R}^{3\times1}$ has the corresponding meaning for subject structure s. The position indices p_q and $\tilde{p}_s$ increase continuously along the entire complexes, spanning all constituent chains.

T and $\tilde{T}$ represent secondary structure assignments for protein query and subject complexes or nucleotide sequences for nucleic acid complexes. The complex type, protein or nucleic acid, is determined based on the predominant atom type, i.e., whether amino acids or nucleotides constitute the majority of the structure.

n_{its}, n_{brn}, and n_{rfn} are configurable parameters representing the number of deep, spatial index-driven superposition search iterations, top-performing superposition branches explored in detail, and refinement rounds, respectively. The parameter c_{thr} defines the local structural similarity threshold that triggers superposition analysis.

The algorithm operates on complete macromolecular complexes rather than treating them as separate chains. Chain-to-chain assignments are subsequently derived from the optimal complex-level superpositions identified during the search.

The algorithm begins by computing a dynamic programming (DP) matrix of local scores (line 2) using the following recursive relation:

$$\Lambda_{qsij} = \max\{\Lambda_{q,s,i-1,j-1} + 2 \cdot \mathbf{1}_{T_{qi}=\tilde{T}_{sj}} - 1,\ \Lambda_{q,s,i-1,j} - c_{\text{gap}},\ \Lambda_{q,s,i,j-1} - c_{\text{gap}},\ 0\}. \tag{S1}$$

The one-byte variables Λ_{qsij} are capped at a maximum value of 252, with the two least significant bits reserved for backtracking information. Initial superpositions (lines 6–7) are then identified based on structural regions exhibiting high local similarity (line 5). The total number of regions, n_{ext}, is determined by the lengths of the structures, and the regions are distributed evenly along them.

After the initial superpositions are computed, several rounds of constant-time, spatial index-driven alignment are performed to generate the corresponding superpositions (lines 9–11). These steps form the core of the algorithm, treating each position independently and enabling rapid exploration of the superposition space across different configurations.

The atom-independent structural analysis produces sequence-order-independent alignments that capture spatial correspondence but not topological similarity. To account for structural topology, a post-processing step is applied.

The post-processing step (line 13) converts sequence-order-independent alignments into approximate order-dependent TM-scores computed in sublinear time. A small number of transformation matrices (n_{tfm}) with the highest approximate scores are then selected (line 14) to calculate chain-level TM-scores (line 15) using the parallel COMER2 DP algorithm [1]. Note that only complex-level transformation matrices are used throughout all algorithmic stages.

After determining chain-to-chain correspondences by maximizing the sum of chain-level TM-scores (line 16), a smaller number of transformation matrices ($n_{\text{brn}} < n_{\text{tfm}}$) corresponding to the highest TM-scores are selected (line 17) for further optimization of the associated structural alignments (line 18). Finally, the best-performing alignments—each representing a unique query-subject pair—are refined through iterative complex alignment, chain reassignment, and optimization of the resulting complex alignments (lines 19–20).

Algorithm 1 produces transformation matrices for each query-subject complex pair, which are subsequently used in the final stages for fine-grained alignment refinement. The procedure for such refinement is summarized in Algorithm 12.

Algorithm 2 Calculate local similarity

```
1: procedure CalculateLocalSimilarity($n_Q$, $n_S$, $n_{\text{ext}}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $l_f$, $\Lambda$, $c_{\text{thr}}$)
2:     for all $(q, s, f_{\text{ext}}) \in [0, n_Q) \times [0, n_S) \times [0, n_{\text{ext}})$ do in parallel
3:         Calculate query and subject structure positions $p_q$ and $\tilde{p}_s$ from index $f_{\text{ext}}$
4:         if $p_q + l_f > l_q$ or $\tilde{p}_s + l_f > \tilde{l}_s$ then
5:             Set skip flag for configuration $\{q, s, f_{\text{ext}}\}$
6:             return
7:         end if
8:         $S \leftarrow \mathbf{0}_{32,32}$
9:         for all $i \in [0, \min\{96, l_q - p_q\})$ do in parallel
10:            for all $j \in [0, \min\{128, \tilde{l}_s - \tilde{p}_s\})$ do in parallel
11:                $[\ S_{i \bmod 32, j \bmod 32} \leftarrow \max\{S_{i \bmod 32, j \bmod 32},\ \Lambda_{q,s,p_q+i,\tilde{p}_s+j} \wedge \texttt{0xfc}\}\ ]_{\text{mut}}$
12:            end for
13:        end for
14:        $m \leftarrow \max_{i,j} S_{ij}$                          ▷ two-dimensional parallel reduction
15:        if $m < l_f \times c_{\text{thr}}$ then          ▷ fragment length fraction as a similarity threshold
16:            Set skip flag for configuration $\{q, s, f_{\text{ext}}\}$
17:        end if
18:    end for
19: end procedure
```

Algorithm 3 Calculate local alignment-based cross-covariance matrices

```
1:  procedure CalculateLocalXCovariances( $n_{\mathrm{Q}}$, $n_{\mathrm{S}}$, $n_{\mathrm{ext}}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $l_{\mathrm{f}}$, $\mathit{\Lambda}$, $C$, $\tilde{C}$)
2:      for all $(q, s, f_{\mathrm{ext}}) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}}) \times [0, n_{\mathrm{ext}})$ do in parallel
3:          continue if skip flag is set for configuration $\{q, s, f_{\mathrm{ext}}\}$
4:          Calculate query and subject structure positions $p_q$ and $\tilde{p}_s$ from index $f_{\mathrm{ext}}$
5:          $(p_q^*, \tilde{p}_s^*) \leftarrow \underset{1 \le x \le 4,\ p_q + l_{\mathrm{f}}/x < l_q,\ \tilde{p}_s + l_{\mathrm{f}}/x < l_s}{\operatorname{argmax}} (\mathit{\Lambda}_{q,\,s,\,p_q + l_{\mathrm{f}}/x,\,\tilde{p}_s + l_{\mathrm{f}}/x} \wedge \mathtt{0xfc})$      ▷ parallel reduction
6:          Construct the set of matched positions $P = \{(i, \tilde{\imath}) \in \mathbb{N}^2\}$
                by backtracking from $(p_q^*, \tilde{p}_s^*)$ to $(p_q, \tilde{p}_s)$ in $\mathit{\Lambda}_{q,s,\cdot,\cdot}$.
7:          $(K_{qsf_{\mathrm{ext}}}, \mathbf{c}_{qf_{\mathrm{ext}}}, \tilde{\mathbf{c}}_{sf_{\mathrm{ext}}})$
                $\leftarrow \sum_{(i,\tilde{\imath}) \in P} \left(C_{q,p_q+i} \tilde{C}^{\mathrm{T}}_{s,\tilde{p}_s+\tilde{\imath}},\ C_{q,p_q+i},\ \tilde{C}_{s,\tilde{p}_s+\tilde{\imath}}\right)$      ▷ parallel sum reduction
8:          Store $(K_{qsf_{\mathrm{ext}}}, \mathbf{c}_{qf_{\mathrm{ext}}}, \tilde{\mathbf{c}}_{sf_{\mathrm{ext}}})$ in memory
9:      end for
10: end procedure
```

Algorithm 4 Calculate transformation matrices

```
1:  procedure CalculateTransformations($n_{\mathrm{Q}}$, $n_{\mathrm{S}}$, $n_{\mathrm{ext}}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $b_{\mathrm{dyn}}$)
2:      for all $(q, s, f_{\mathrm{ext}}) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}}) \times [0, n_{\mathrm{ext}})$ do in parallel
3:          continue if skip flag is set for configuration $\{q, s, f_{\mathrm{ext}}\}$
4:          Load $(K_{qsf_{\mathrm{ext}}}, \mathbf{c}_{qf_{\mathrm{ext}}}, \tilde{\mathbf{c}}_{sf_{\mathrm{ext}}})$ from memory
5:          if $b_{\mathrm{dyn}} = 1$ and $l_q \ge \tilde{l}_s$ then
6:              $(K_{qsf_{\mathrm{ext}}}, \mathbf{c}_{qf_{\mathrm{ext}}}, \tilde{\mathbf{c}}_{sf_{\mathrm{ext}}}) \leftarrow (K^{\mathrm{T}}_{qsf_{\mathrm{ext}}}, \tilde{\mathbf{c}}_{sf_{\mathrm{ext}}}, \mathbf{c}_{qf_{\mathrm{ext}}})$
7:          end if
8:          Calculate $R_{qsf_{\mathrm{ext}}}$ by the Kabsch algorithm [2, 3]
                based on $(K_{qsf_{\mathrm{ext}}}, \mathbf{c}_{qf_{\mathrm{ext}}}, \tilde{\mathbf{c}}_{sf_{\mathrm{ext}}})$
9:          $\mathbf{t}_{qsf_{\mathrm{ext}}} \leftarrow \bar{\tilde{\mathbf{c}}}_{sf_{\mathrm{ext}}} - R_{qsf_{\mathrm{ext}}} \bar{\mathbf{c}}_{qf_{\mathrm{ext}}}$
10:         Store $(R_{qsf_{\mathrm{ext}}}, \mathbf{t}_{qsf_{\mathrm{ext}}})$ in memory
11:     end for
12: end procedure
```

Algorithm 5 Produce pilot alignments in constant time using spatial indices

```
1:  procedure AlignInConstantTime($n_{\mathrm{Q}}$, $n_{\mathrm{S}}$, $n_{\mathrm{ext}}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $C$, $\tilde{C}$, $T$, $\tilde{T}$, $c_{\mathrm{thr}}$, $b_{\mathrm{SSM}}$)
2:      for all $(q, s, f_{\mathrm{ext}}) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}}) \times [0, n_{\mathrm{ext}})$ do in parallel
3:          continue if skip flag is set for configuration $\{q, s, f_{\mathrm{ext}}\}$
4:          Calculate query and subject structure positions $p_q$ and $\tilde{p}_s$ from index $f_{\mathrm{ext}}$
5:          $b_{\mathrm{SSM}} \leftarrow 0$  if structure $q$ or $s$ does not represent a protein
6:          $l_{\mathrm{f}} \leftarrow \min\{256, l_q, \tilde{l}_s\}$
7:          if $l_q \ge \tilde{l}_s$ then                                  ▷ always search in the larger structure
8:              $r \leftarrow \max\{0, \min\{\tilde{l}_s - l_{\mathrm{f}}, \tilde{p}_s - l_{\mathrm{f}}/2\}\}$
9:              $D \leftarrow \tilde{C}_{s,r:r+l_{\mathrm{f}}-1}$; $\tilde{D}$, $\tilde{D}' \leftarrow C_q$; $\theta \leftarrow \tilde{T}_{s,r:r+l_{\mathrm{f}}-1}$
10:         else
11:             $r \leftarrow \max\{0, \min\{l_q - l_{\mathrm{f}}, p_q - l_{\mathrm{f}}/2\}\}$
12:             $D \leftarrow C_{q,r:r+l_{\mathrm{f}}-1}$; $\tilde{D}$, $\tilde{D}' \leftarrow \tilde{C}_s$; $\theta \leftarrow T_{q,r:r+l_{\mathrm{f}}-1}$
13:         end if
14:         Load $(R_{qsf_{\mathrm{ext}}}, \mathbf{t}_{qsf_{\mathrm{ext}}})$ from memory
15:         for all $i \in [0, l_{\mathrm{f}})$ do in parallel
16:             $D'_i \leftarrow R_{qsf_{\mathrm{ext}}} D_i + \mathbf{t}_{qsf_{\mathrm{ext}}}$                              ▷ transformation
17:             $j \leftarrow$ nearest neighbour in $\tilde{D}$ for $D'_i$ using index
                    with $(b_{\mathrm{SSM}} = 1)$ or without $(b_{\mathrm{SSM}} = 0)$ information $\theta_i$      ▷ $O(1)$ time complexity
18:             Swap($D_i$, $\tilde{D}_j$) if $l_q \ge \tilde{l}_s$
19:             Store $(D_i, \tilde{D}_j, j)$ in memory indexed by $(q, s, f_{\mathrm{ext}}, i)$
20:         end for
21:         if $b_{\mathrm{SSM}} = 1$ and $\sum_{i \in [0, l_{\mathrm{f}})} [\, \|D'_i - \tilde{D}'_{j(i)}\| > 8\,] > l_{\mathrm{f}} \times (1 - c_{\mathrm{thr}})$ then      ▷ parallel reduction
22:             Set skip flag for configuration $\{q, s, f_{\mathrm{ext}}\}$
23:         end if
24:     end for
25: end procedure
```

Algorithm 6 Calculate cross-covariance matrices from alignments

```
1:  procedure CalculateXCovariancesA($n_\mathrm{Q}$, $n_\mathrm{S}$, $n_\mathrm{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$)
2:      for all $(q, s, f_\mathrm{ext}) \in [0, n_\mathrm{Q}) \times [0, n_\mathrm{S}) \times [0, n_\mathrm{ext})$ do in parallel
3:          continue if *skip* flag is set for configuration $\{q, s, f_\mathrm{ext}\}$
4:          $l_\mathrm{f} \leftarrow \min\{256, l_q, \tilde{l}_s\}$
5:          Load $(D_i, \tilde{D}_i, \cdot)_{i=0}^{l_\mathrm{f}-1}$ from memory at $(q, s, f_\mathrm{ext}, i)_{i=0}^{l_\mathrm{f}-1}$
6:          $(K_{qsf_\mathrm{ext}}, \mathbf{c}_{qf_\mathrm{ext}}, \tilde{\mathbf{c}}_{sf_\mathrm{ext}}) \leftarrow \sum_{i=0}^{l_\mathrm{f}-1} (D_i \tilde{D}_i^\mathrm{T}, D_i, \tilde{D}_i)$      ▷ parallel sum reduction
7:          Store $(K_{qsf_\mathrm{ext}}, \mathbf{c}_{qf_\mathrm{ext}}, \tilde{\mathbf{c}}_{sf_\mathrm{ext}})$ in memory
8:      end for
9:  end procedure
```

Algorithm 7 Calculate approximate sequence-order-dependent scores

```
1:  procedure CalculateApproxScores($n_\mathrm{Q}$, $n_\mathrm{S}$, $n_\mathrm{ext}$, $\mathbf{l}$, $\tilde{\mathbf{l}}$, $L$, $\tilde{L}$)
2:      for all $(q, s, f_\mathrm{ext}) \in [0, n_\mathrm{Q}) \times [0, n_\mathrm{S}) \times [0, n_\mathrm{ext})$ do in parallel
3:          continue if *skip* flag is set for configuration $\{q, s, f_\mathrm{ext}\}$
4:          Calculate query and subject structure positions $p_q$ and $\tilde{p}_s$ from index $f_\mathrm{ext}$
5:          $l_\mathrm{f} \leftarrow \min\{256, l_q, \tilde{l}_s\}$
6:          $\mathbf{x} \leftarrow -\mathbf{1}_{512}$; $\mathbf{m} \leftarrow \mathbf{0}_{512}$; $\nu \leftarrow \mathbf{0}_{|L_{q,\cdot}|}$
7:          Load $(R_{qsf_\mathrm{ext}}, \mathbf{t}_{qsf_\mathrm{ext}})$ from memory
8:          for all $i \in [0, l_\mathrm{f})$ do in parallel
9:              Load $(D_i, \tilde{D}_i, j)$ from memory at $(q, s, f_\mathrm{ext}, i)$
10:             $D'_i \leftarrow R_{qsf_\mathrm{ext}} D_i + \mathbf{t}_{qsf_\mathrm{ext}}$      ▷ transformation
11:             $z_i \leftarrow j$; $a_i \leftarrow d_0^2/(d_0^2 + \|D'_i - \tilde{D}_i\|^2)$      ▷ $d_0$ defined as in [4]
                ▷ For clarity, indices $j$ and $i$ are assumed to correspond to the query and subject
                ▷ structures, respectively, according to the condition $l_q \geq \tilde{l}_s$ in Algorithm 5
12:             $g_i \leftarrow h_q(j)$      ▷ get $q$'s chain index from position $j$
13:         end for
14:         for $i = 0, \ldots, l_\mathrm{f} - 1$ do
15:             if $i > 0$ and $\tilde{h}_s(\tilde{p}_s + i) \neq \tilde{h}_s(\tilde{p}_s + i - 1)$ then      ▷ check for a different chain index value
16:                 continue if $\nu_{g_i} > 0.2 \times \tilde{L}_{s,\tilde{h}_s(\tilde{p}_s+i)}$      ▷ >20% of subject chain length
17:             end if
18:             $\nu_{g_i} \leftarrow \nu_{g_i} + 1$      ▷ #aligned positions for the query chain
19:             $\omega \leftarrow \max_{j:x_j<z_i} m_j$      ▷ parallel max reduction
20:             $c \leftarrow z_i \bmod 512$      ▷ trivial hash function
21:             if $x_c < 0$ or $(x_c = z_i$ and $m_c < \omega + a_i)$ or
                    $((i > l_\mathrm{f}/2)?\ x_c < z_i\colon x_c > z_i)$      ▷ heuristics upon hash collision
                 then
22:                 $x_c \leftarrow z_i$; $m_c \leftarrow \omega + a_i$
23:             end if
24:         end for
25:         $w_{qsf_\mathrm{ext}} \leftarrow \max_j m_j$      ▷ parallel reduction
26:         Store $w_{qsf_\mathrm{ext}}$ in memory
27:     end for
28: end procedure
```

Algorithm 8 Select the top n_{tfm} transformation matrices

```
procedure GetTopNTransformations($n_Q$, $n_S$, $n_{\text{ext}}$, $n_{\text{tfm}}$, $b_{\text{srt}}$)
    for all $(q, s) \in [0, n_Q) \times [0, n_S)$ do in parallel
        $\mathbf{x} \leftarrow -\mathbf{1}_{n_{\text{tfm}}}$; $\mathbf{m} \leftarrow \mathbf{0}_{n_{\text{tfm}}}$
        if $b_{\text{srt}} = 1$ then
            Load $(w_{qsf_{\text{ext}}})_{f_{\text{ext}}=0}^{n_{\text{ext}}-1}$ from memory        ▷ TM-scores obtained by DP
            $\mathbf{x}' \leftarrow$ Sort$((w_{qsf_{\text{ext}}})_{f_{\text{ext}}})$        ▷ Batcher's sort [5] in $O(\log_2^2 n_{\text{ext}})$ time
            $\mathbf{x} \leftarrow (x'_f)_{f=0}^{n_{\text{tfm}}-1}$        ▷ $x_f = -1$ if $w_{qsx'_f} = 0$ (skip flag set)
        else
            for all $(f_{\text{ext}}) \in [0, n_{\text{ext}})$ do in parallel        ▷ approximation to partial sorting
                continue if skip flag is set for configuration $\{q, s, f_{\text{ext}}\}$
                Load $w_{qsf_{\text{ext}}}$ from memory
                $f_{\text{m}} \leftarrow f_{\text{ext}} \bmod n_{\text{tfm}}$
                $[\,(m_{f_{\text{m}}}, x_{f_{\text{m}}}) \leftarrow (w_{qsf_{\text{ext}}}, f_{\text{ext}})$ if $m_{f_{\text{m}}} < w_{qsf_{\text{ext}}}\,]_{\text{mut}}$
            end for
        end if
        for all $f \in [0, n_{\text{tfm}})$ do in parallel
            Set $(x_f < 0)$ or unset $(x_f \geq 0)$ skip flag for configuration $\{q, s, f\}$
            if $x_f \geq 0$ then
                Load $(R_{qsx_f}, \mathbf{t}_{qsx_f})$ from memory
                $(R'_{qsf}, \mathbf{t}'_{qsf}) \leftarrow (R_{qsx_f}, \mathbf{t}_{qsx_f})$
                Load $A_{qsx_f}$ from memory into $A'_{qsf}$  if $b_{\text{srt}} = 1$        ▷ chain assignments
            end if
        end for
        for all $f \in [0, n_{\text{tfm}})$ do in parallel
            if $x_f \geq 0$ then
                $(R_{qsf}, \mathbf{t}_{qsf}) \leftarrow (R'_{qsf}, \mathbf{t}'_{qsf})$
                Store $(R_{qsf}, \mathbf{t}_{qsf})$ in memory
                Store $A_{qsf} = A'_{qsf}$ in memory  if $b_{\text{srt}} = 1$
            end if
        end for
    end for
end procedure
```

Algorithm 9 Compute chain-level TM-scores for the top n_{tfm} transformation matrices

```
procedure ComputeChainTMscores($n_Q$, $n_S$, $n_{\text{tfm}}$, $L$, $\tilde{L}$, $c_{\text{goc}}$)
    for all $(q, s, f) \in [0, n_Q) \times [0, n_S) \times [0, n_{\text{tfm}})$ do in parallel
        continue if skip flag is set for configuration $\{q, s, f\}$
        Load $(R_{qsf}, \mathbf{t}_{qsf})$ from memory
        for all $(h_q, \tilde{h}_s) \in [0, |L_{q,\cdot}|) \times [0, |\tilde{L}_{s,\cdot}|)$ do in parallel
            Perform DP with gap open cost $c_{\text{goc}}$ on chain pair $(h_q, \tilde{h}_s)$
                using $(R_{qsf}, \mathbf{t}_{qsf})$ and the Comer2 DP algorithm [1]
            Store the resulting TM-score in memory indexed by $(q, s, f, h_q, \tilde{h}_s)$
        end for
    end for
end procedure
```

Algorithm 10 Determine chain-to-chain assignments for the top n_{tfm} configurations

```
procedure MakeChain2ChainAssignments($n_Q$, $n_S$, $n_{\text{tfm}}$, $L$, $\tilde{L}$)
    for all $(q, s, f) \in [0, n_Q) \times [0, n_S) \times [0, n_{\text{tfm}})$ do in parallel
        continue if skip flag is set for configuration $\{q, s, f\}$
        Apply a parallelized Hungarian algorithm to obtain the chain assignment $A_{qsf} = \{(h_q^*, \tilde{h}_s^*)\}$
            that maximizes $\sum_{h_q=0}^{\min(|L_{q,\cdot}|,|\tilde{L}_{s,\cdot}|)}$ TM-score$_{q,s,f,h_q,\tilde{h}_s(h_q)}$
                                                    ▷ $\tilde{h}_s(h_q)$ denotes a one-to-one mapping to unique $\tilde{h}_s$
        Store $A_{qsf}$ in memory indexed by $(q, s, f)$
        Store TM-score$_{qsf} = \sum_{(h_q,\tilde{h}_s)\in A_{qsf}}$ TM-score$_{q,s,f,h_q,\tilde{h}_s}$ in memory indexed by $(q, s, f)$
    end for
end procedure
```

Algorithm 11 Optimize n_{brn} selected alignments

```
1: procedure OptimizeSelectedAlignments($n_{\mathrm{Q}}$, $n_{\mathrm{S}}$, $n_{\mathrm{brn}}$, $C$, $\tilde{C}$, $b_{\mathrm{runDP}}$)
2:     for $f = 0, \ldots, n_{\mathrm{brn}} - 1$ do
3:         for all $(q, s) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}})$ do in parallel
4:             continue if *skip* flag is set for configuration $\{q, s, f\}$
5:             Load transformation matrix $(R_{qsf}, \mathbf{t}_{qsf})$ from memory
6:             Load chain assignments $A_{qsf}$ from memory
7:             if $b_{\mathrm{runDP}} = 1$ then
8:                 Perform DP on chain pairs $(h_q, \tilde{h}_s) \in A_{qsf}$
                       using $(R_{qsf}, \mathbf{t}_{qsf})$ and the Comer2 DP algorithm [1]
9:             end if
10:            Construct alignment integrating all chain pairs $(h_q, \tilde{h}_s) \in A_{qsf}$
                   using the Comer2 backtracking algorithm
11:            Store coordinates $(C_{qk}, \tilde{C}_{sl})_{k,l}$ of $l_{\mathrm{A}}(R_{qsf}, \mathbf{t}_{qsf})$ aligned residues in memory
12:            Find optimal $(R_{qsf}, \mathbf{t}_{qsf})$ by calculating TM-scores
                   based on the superpositions obtained in parallel from
                   $n_{\mathrm{A}}[l_{\mathrm{A}}(R_{qsf}, \mathbf{t}_{qsf})]$ alignment fragments of varying lengths and positions
                   (similarly to "Search Engine" in [4])
13:        end for
14:    end for
15:    for all $(q, s) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}})$ do in parallel
16:        Store $(R_{qs}, \mathbf{t}_{qs})$
               $= \mathrm{argmax}_{\{(R_{qsf}, \mathbf{t}_{qsf})\}_{f=0}^{n_{\mathrm{brn}}-1}}$ TM-score$(R_{qsf}, \mathbf{t}_{qsf})$ in memory
17:    end for
18: end procedure
```

Algorithm 12 Refine the best alignments

```
1: procedure RefineBestAlignments($n_{\mathrm{Q}}$, $n_{\mathrm{S}}$, $n_{\mathrm{rfn}}$, $L$, $\tilde{L}$, $C$, $\tilde{C}$, $c_{\mathrm{goc}}$)
2:     repeat $n_{\mathrm{rfn}}$ times
3:         ComputeChainTMscores($n_{\mathrm{Q}}$, $n_{\mathrm{S}}$, $n_{\mathrm{tfm}}$=1, $L$, $\tilde{L}$, $c_{\mathrm{goc}}$)
4:         MakeChain2ChainAssignments($n_{\mathrm{Q}}$, $n_{\mathrm{S}}$, $n_{\mathrm{tfm}}$=1, $L$, $\tilde{L}$)
5:         OptimizeSelectedAlignments($n_{\mathrm{Q}}$, $n_{\mathrm{S}}$, $n_{\mathrm{brn}}$=1, $C$, $\tilde{C}$, $b_{\mathrm{runDP}}$=0)
6:     end
7: end procedure
```

Bibliography

[1] Margelevičius, M. COMER2: GPU-accelerated sensitive and specific homology searches. *Bioinformatics* **36**, 3570–3572 (2020).

[2] Kabsch, W. A solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. A* **32**, 922–923 (1976).

[3] Kabsch, W. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. A* **34**, 827–828 (1978).

[4] Zhang, Y. & Skolnick, J. Scoring function for automated assessment of protein structure template quality. *Proteins* **57**, 702–710 (2004).

[5] Batcher, K. E. Sorting networks and their applications. *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference* 307–314 (1968).