GTalign: High-performance protein structure alignment, superposition, and search

Mindaugas Margelevičius[1*]

[1*]Institute of Biotechnology, Life Sciences Center, Vilnius University, Saulėtekio av. 7, Vilnius, 10257, Lithuania.

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

Abstract

In response to the challenge posed by vast protein structure repositories, GTalign offers an innovative solution to protein structure alignment and search—an algorithm that achieves optimal superposition at unprecedented speeds. Through the design and implementation of spatial structure indexing, GTalign parallelizes all stages of superposition search across residues and protein structure pairs, yielding rapid identification of optimal superpositions. Rigorous evaluation across diverse datasets reveals GTalign as the most accurate among structure aligners while presenting orders of magnitude in speedup at state-of-the-art accuracy. GTalign's high speed and accuracy make it useful for numerous applications, including functional inference, evolutionary analyses, protein design, and drug discovery, contributing to advancing understanding of protein structure and function.

Keywords: Protein structure similarity; Protein evolution; Protein function; High-performance computing

1 Introduction

In contemporary structural bioinformatics, the advent of advanced artificial neural network architectures [1–3] has ushered in an era where protein structures are predicted with unprecedented accuracy for a myriad of protein sequences [4, 5]. This surge in structural data has presented a challenge: the need for efficient and rapid protein structure comparison to distill meaningful insights from the burgeoning repositories of three-dimensional protein structures. Such tools enable the extraction of biologically relevant information, decipher evolutionary relationships [6–8], and contribute significantly to understanding functional mechanisms encoded within protein structures [9–11]. In this context, the development of computational tools for rapid large-scale protein structure alignments represents an important step forward.

The need for efficient tools gave rise to a variety of computational techniques, each with its own strengths and limitations, aiming to achieve accurate and rapid comparisons of protein structures. Two fundamental approaches have emerged: local pattern matching and rigid-body superposition optimization. Local pattern matching involves analyzing structures independently, making it suitable for handling flexible regions like linkers between protein domains. This approach encompasses various strategies such as optimizing the match between protein inter-residue distance matrices [12] or probability distributions [13], aligning secondary structure elements using double dynamic programming [14] or graph matching [15], finding and extending [16] or chaining [17] aligned fragment pairs with optimal inter-residue distance matching, and quantifying evolutionary similarity to infer initial alignments [13, 18].

On the other hand, methods based on rigid-body superposition treat structures as rigid bodies and focus on optimizing their local or global spatial agreement. These methods may involve optimizing superposition for protein fragments [19], iterative superposition at different distance cutoffs [20], or multi-stage fragment-based superposition optimization [21]. To achieve accurate pairwise protein alignments, these methods [20, 21], and those employing local pattern matching [14–18], apply iterative alignment refinement that includes rigid-body superposition.

While approaches like TM-align [21] and Dali [22] have been essential in advancing protein structure comparison, their computational complexity has made them less practical for large-scale applications

to growing protein structure databases. To address this challenge, some strategies involve preprocessing database structures to compute computationally intensive parts in advance [15]. Alternatively, accelerating protein structure searches involves conducting searches across preclustered databases [23, 24]. However, these approaches require compute-intensive preprocessing and constant updating.

Recently, Foldseek [25] has introduced a novel approach of leveraging a structural alphabet to reduce the problem to a sequence comparison and employed k-mer matching to filter out insignificant matches. Despite its notable speed, Foldseek contends with inherent limitations in alignment accuracy, underscoring the persistent challenge of achieving a balance between computational efficiency and accuracy.

We bridge this gap by introducing GTalign (Giga-scale Targeted alignment), a new computational tool designed for extensive protein structure similarity exploration and alignment. GTalign's primary objective is to find an optimal spatial overlay for a given pair of structures and subsequently derive an alignment from it. In that sense, GTalign treats each structure as a rigid body and evaluates protein similarity through the identification of the best superposition, thereby ensuring an accurate structural alignment. High computational performance is achieved by novel algorithms developed for low-complexity superposition search and to harness the power of modern computer processors and accelerators. Consequently, GTalign combines high accuracy and fast execution times, offering a solution for accelerated and accurate structural analyses.

2 Results and Discussion

2.1 GTalign approach

For a given protein pair, GTalign employs an iterative process that involves (i) selecting a subset of atom pairs, (ii) calculating the transformation matrix, (iii) deriving an alignment based on the resulting superposition, and finally, selecting the alignment that maximizes the TM-score [26]—a strategy akin to TM-align [21]. The first two steps can be naturally parallelized for speed. However, alignment derivation entails finding, for each residue, the spatially closest residue of the other structure after superposition. Preserving sequence order is crucial for assessing protein topology similarity, and thus, this optimization typically requires dynamic programming (DP), a standard choice for solving such problems. Handling positional dependencies while maintaining sequence order poses challenges to DP implementation with favorable time complexity. Therefore, this, coupled with the need for multiple alignments per structure pair, represents a bottleneck for large-scale computation despite optimized parallel DP implementation efforts [27].

GTalign tackles this challenge by introducing a spatial index for each structure, which allows for considering atoms independently and ensures $O(1)$ time complexity for the alignment problem (Fig. 1a). Although post-processing is necessary to preserve sequence order, it has sublinear rather than quadratic time complexity. This methodology enables GTalign to effectively parallelize all steps, efficiently navigating through an extensive superposition space. Coupled with parallel processing of numerous protein structure pairs, it significantly accelerates the entire protein similarity search process.

2.2 Comprehensive reference-free performance evaluation

We benchmarked GTalign against a set of well-established protein structure aligners: TM-align [21], Dali [22], FATCAT [28], DeepAlign [18], and Foldseek [25]. We also evaluated the performance of TM-align's fast variant (option `-fast`) and Foldseek variants utilizing both fast (`--tmalign-fast 1`; FoldseekTM) and regular (`--tmalign-fast 0`) versions of TM-align for aligning protein structures that passed sequence similarity filters. Various parameterizations of GTalign were also benchmarked. The `--speed` option controls the algorithm's execution speed, with higher values prioritizing speed over accuracy. Additionally, the `--pre-similarity` and `--pre-score` options were employed for initial similarity screening in the sequence and structure space, respectively.

Comprehensive evaluations were conducted across four diverse datasets representing different protein analysis scenarios: (i) SCOPe 2.08 [29] protein domains filtered to 40% sequence identity, (ii) PDB [30] full-length structures filtered to 20% sequence identity, (iii) the UniProtKB/Swiss-Prot [31] protein structures from the AlphaFold Database [4], and (iv) the HOMSTRAD database [32].

The evaluation employed an unbiased and reference-free approach that reveals alignment accuracy through the superposition of structures based on the produced alignments (see Section 4.11). We calculated the superposition score, TM-score [26], for alignments produced by each tool. TM-score ranges from 0 to 1, with 1 representing a perfect match. Therefore, the progression of TM-scores serves as an indicator of a tool's accuracy. The range TM-score $\in [0.5; 1]$ is particularly important because it reflects sensitivity, i.e., a tool's ability to detect proteins sharing the same fold [33].

TM-score can be normalized by the length of either of the two proteins being compared. We first discuss the results obtained using the TM-score normalized by the length of the shorter protein. This TM-score

evaluates structural similarity irrespective of the ratio of the two protein lengths, effectively capturing similarities of smaller proteins or domains matching regions of larger proteins.

The results (Fig. 1b–d, Supplementary Section S1.1, Supplementary Fig. S1a, Supplementary Table S1) show that GTalign consistently outperforms all the aligners in terms of accuracy. GTalign (option `--speed=0`) produces up to 7% more alignments with a TM-score ≥ 0.5 than TM-align, the second most accurate tool (732,024 vs. 683,996, SCOPe40 2.08 dataset).

This trend persists across the entire TM-score significance range from 0.5 to 1.0 (Supplementary Table S2). However, there are exceptions: GTalign reports 191 fewer alignments than TM-align for TM-scores ≥ 0.7 in the PDB20 dataset and 31 fewer alignments for TM-scores ≥ 0.8 in the Swiss-Prot dataset. Further investigation revealed that these differences stem from aligning queries with short proteins with fewer than 30 residues. For a pair of proteins, GTalign employs approximate partial sorting to select candidate alignments for detailed refinement. Alignments for very small proteins or peptides score similarly, and this approximation can occasionally lead to a suboptimal final alignment.

While the issue of short proteins is recognized and left for future resolution, analysis based on the TM-score normalized by the query length effectively diminishes the significance of alignments between queries and much smaller proteins. In this evaluation setting (Fig. 2, Supplementary Fig. S1b, Supplementary Table S3), GTalign demonstrates superior accuracy over the other aligners again. For example, in the SCOPe40 2.08 dataset, GTalign (option `--speed=0`) produces up to 7% more alignments with a TM-score ≥ 0.5 than TM-align (492,887 vs. 460,847). Disregarding insignificant differences of one alignment, GTalign exhibits superiority across the full TM-score significance range (Supplementary Table S4).

GTalign also demonstrates higher accuracy on the HOMSTRAD dataset (Fig. 3), which contains reference structure alignments grouped into evolutionarily and structurally related protein families. These families exhibit relatively high structural similarity, as evidenced by the TM-score distribution of the reference alignments (Fig. 3). Therefore, these results highlight GTalign's utility in improving structural alignments, even among highly similar proteins. This is further supported by the greater accuracy of GTalign-produced alignments compared to the reference alignments, suggesting potential applications in constructing reference datasets and classifying proteins. A similar trend was observed previously [34].

Additionally, we provide structural examples in Supplementary Figs. S2–S5, each corresponding to one dataset used in our benchmark study. These examples highlight nontrivial structural similarities identified by GTalign but overlooked or misaligned by all other benchmarked tools. Notably, all showcased examples demonstrate domains (even from different folds; see a subsection below) or larger significant structural segments, with insertions and deletions, sharing the same topology, as confirmed by the TM-scores and structural alignments. Therefore, Supplementary Figs. S2–S5 exemplify GTalign's primary objective of achieving optimal protein spatial superpositions and detecting subtle yet significant structural similarities.

While GTalign excels in uncovering structural similarities, its efficiency also sets it apart (Fig. 1b–d). GTalign is up to 104–1424x faster (618–8454 vs. 879,965 seconds, Swiss-Prot dataset) than TM-align parallelized on all 40 CPU threads. It achieves a 177x speedup (options `--speed=13` `--pre-score=0.4`) over the fast (`-fast`) TM-align version (618 vs. 109,319 seconds, Swiss-Prot dataset) and is the fastest among the tools except Foldseek.

Clearly, the sequence prefiltering strategy contributes to Foldseek's high speed. However, this comes at a high price in accuracy and sensitivity (only 13,371 alignments with a TM-score ≥ 0.5; SCOPe40 2.08 dataset, Fig. 1b; see also Fig. 2 and Supplementary Table S2 and S4). When GTalign is configured to use sequence prefiltering (options `--speed=13` `--pre-similarity=15`), a similar pattern emerges, with runtimes comparable to Foldseek (428 vs. 196 seconds, Swiss-Prot dataset) but decreased sensitivity. In contrast, no such effect is observed for prescreening in the structure space (option `--pre-score`; Fig. 1 and 2).

This phenomenon can be attributed to at least two factors. First, low sequence similarity does not necessarily correlate with low structural similarity, as demonstrated by the results. Second, the generation of accurate structural alignments using a single scoring scheme per protein pair may lack consistency. To address this, we explored the use of scores derived from spectral analysis of rotation-invariant two-dimensional representations of geometric features, such as angles and distances between residues, in the frequency (Fourier) domain. However, this approach demonstrated inconsistent results and requires further investigation. Despite these observations, leveraging prescreening in the sequence space can prove valuable for high-similarity searches.

GTalign offers additional computational advantages by providing the option to utilize multiple GPUs for computation. This feature was effectively leveraged for processing the SCOPe40 2.08, PDB20, and Swiss-Prot datasets, where GTalign exploited the computational power of all three Tesla V100 GPUs available on the system. Supplementary Table S1 provides GTalign runtimes on one, two, and three GPUs, demonstrating scalability across all benchmarked parametrizations. Furthermore, the results presented in Supplementary Section S1.2 and Supplementary Table S5 unveil a noteworthy performance trend: A more recent desktop-grade GPU consistently outperforms the computational capabilities of the three server-grade V100 GPUs,

effectively conveying GTalign's remarkable performance even when run on a single, relatively inexpensive GPU.

2.3 Alignment accuracy evaluation using RMSD and GDT_TS

TM-score, used to evaluate alignment accuracy in the previous subsection, is a global measure sensitive to alignment coverage due to normalization by protein length. Here, we turn to root-mean-squared deviation (RMSD) and global distance test (GDT) [20] for evaluation.

RMSD, a measure of spatial proximity, is normalized by the number of aligned residue pairs and is effective at capturing accurately aligned local protein regions (see Supplementary Fig. S6 in Supplementary Section S1.3 for an example). However, optimizing alignments based solely on RMSD can yield short aligned fragments that provide limited insight into structural similarity at the domain or protein level. Previous approaches [14, 15, 20] sought to find a balance between RMSD and alignment coverage to generate alignments sufficiently long to assess structural similarity without being overly divergent.

The GDT score (GDT_TS) [20] is another measure of spatial proximity, calculated at four different distance thresholds (1, 2, 4, and 8 Å), which does not over-penalize spatially unmatched residue pairs.

In this section, our RMSD and GDT_TS-based evaluation focuses on local alignments within alignment boundaries, providing insight into the extent to which alignments can be shortened to increase local precision by reducing alignment coverage. The results (Fig. 4) reveal that even among local alignment methods such as Foldseek (default parametrization), FATCAT, and DeepAlign, only DeepAlign produces alignments with lower RMSDs and higher GDT_TS scores than GTalign (Supplementary Table S6). However, DeepAlign achieves 15% and 18% lower RMSDs (2.48 and 2.03 vs. 2.91 and 2.47 for the SCOPe40 2.08 and PDB20 datasets) with 20% and 17% fewer aligned residues on average (56.0 and 67.4 vs. 47.6 and 57.1) compared to GTalign (`--speed=0`). The average TM-scores of GTalign alignments (Supplementary Table S1) exceeding those of DeepAlign alignments suggest that the difference in the number of aligned residues does not imply misaligned pairs. Indeed, Supplementary Table S1 and Supplementary Table S6 demonstrate that both GTalign and TM-align produce alignments that strike a good balance between coverage and precision. On average, GTalign (`--speed=0`) achieves lower RMSDs with higher alignment coverage compared to TM-align. In some cases, this is achieved by identifying different, more optimal spatial superpositions (see Supplementary Figs. S2–S4 for examples).

2.4 Benchmarking against the SCOPe dataset reference

GTalign's main objective is to achieve an optimal superposition for a pair of structures for inferring their structural similarity, not rarely indicating an evolutionary relationship. This section assesses GTalign and the other tools from an evolutionary standpoint by examining their ability to replicate SCOPe classification. Importantly, this benchmark does not measure alignment accuracy or the rate of accurate alignments (Figs. 1–3) but rather the consistency between a tool's ranking of structure pairs and SCOPe classification.

The SCOPe knowledgebase categorizes protein domains into families, superfamilies, folds, and classes. Families group domains based on sequence similarity, with those sharing a common ancestor organized into superfamilies. Folds comprise structurally similar superfamilies, while classes are arranged by secondary structure content and organization [29].

Fig. 5 shows the relationship between the precision and recall (PR) of matching domains of the same SCOPe 2.08 family, superfamily, and fold for each tool. As discussed in the next subsection, significant structural similarities extend even across folds and classes. Consequently, Fig. 5a displays the PR curves obtained with false positives (FPs) as pairs of domains from different folds, excluding well-known cross-fold relationships across Rossman-like (c.2–c.5, c.27, c.28, c.30, and c.31) and beta-propeller (b.66–b.70) folds [35]. Additionally, Fig. 5b shows the PR curves when disregarding cross-fold relationships, with FPs corresponding to pairs from different SCOPe 2.08 folds.

The results in Fig. 5 and the areas under the PR curves (AUPRCs) reported in Supplementary Table S7 (Supplementary Section S1.4) demonstrate that GTalign (variants `--speed=0` and `--speed=13`) generally outperforms the other tools, except for Dali in the evaluation that ignores cross-fold relationships at the fold level, where the difference in AUPRC is <1%. When evaluating sensitivity in identifying related domains before encountering the first FP (Supplementary Fig. S7), GTalign shows lower average sensitivity compared to both TM-align versions at the family and fold levels, and compared to Dali and DeepAlign at the superfamily level. However, these differences in the distributions of sensitivity values are statistically insignificant (Supplementary Table S8).

Fig. 5 and Supplementary Table S7 also highlight two observations. First, GTalign's performance diminishes when employing sequence prefiltering (options `--speed=13 --pre-similarity=15`), due to overly stringent criteria in filtering out dissimilar structures, despite relatively high alignment accuracy for pairs passing the filter (Fig. 1, 2, and 4). Future enhancements may involve refining sequence prefiltering options.

Second, disregarding cross-fold relationships considerably affects precision, recall, and AUPRCs, rendering evaluations unstable. Further discussion on this aspect follows in the subsequent subsection.

2.5 Limitations of SCOPe-based evaluation

In the SCOPe classification, the traditional emphasis on organizing proteins according to their sequence similarity to classified entries [36] reflects a primary focus on evolutionary relationships. However, structural similarities among proteins extend beyond evolutionary connections [37]. Classifying proteins into discrete folds presents challenges due to inherent ambiguity in defining folds [38, 39]. For example, what level of insertions or deletions can be considered critical for classifying a domain into a different fold? A more nuanced perspective suggests that protein fold space may exhibit both discrete and continuous characteristics [37, 39]. While high structural or evolutionary similarities may support discrete fold assignments, lower yet significant similarities imply a continuous nature of fold space.

Our analysis supports these assertions. Examination of the distribution of statistically significant TM-scores, obtained by aligning query and subject structures from the SCOPe40 2.08 dataset using TM-align, reveals that only highly similar protein domains (TM-score > 0.8) within the same folds outnumber those from different folds (Fig. 6, Supplementary Fig. S8). Domains from different folds span the entire TM-score significance range, with all of them deemed errors (false positives) despite evident structural and topological similarities (Fig. 6).

These insights underscore the limitations of the SCOPe-based evaluation. While the reference SCOPe-based evaluation provides a convenient approach for benchmarking structure alignment tools, its inability to capture the full complexity of protein structure space is evident.

3 Conclusion

GTalign provides an efficient solution for searching vast protein structure datasets at different levels of accuracy. Its high efficiency is exemplified by a speedup of 6 orders of magnitude over TM-align when aligning large protein complexes (Supplementary Fig. S9). GTalign's cross-platform implementation, user-friendly interface, and high configurability, including the option for clustering structures (Supplementary Section S2), underscore its accessibility and versatility. Providing orders of magnitude in speedup at state-of-the-art accuracy, GTalign positions itself as a valuable tool among existing structure aligners.

4 Methods

4.1 Structure representation

GTalign offers users the flexibility to configure and choose which protein structure atoms will serve as representatives. By default, protein structures are represented using alpha-carbon atoms. All experiments conducted with GTalign were performed using alpha-carbon atoms as representatives.

4.2 Algorithm outline

The GTalign software takes as inputs query and subject (referred to as "reference" in the software) structure databases of arbitrary size. GTalign processes this data in chunks, aligning batches of query structures with batches of subject structures, both sorted by length, iteratively until all possible batch pairs are completed. A similar batch-oriented approach to processing large databases has been described previously [27]. Below, we outline the (sequential) algorithmic steps representing the actions performed on a pair of batch query and subject structures.

Certain steps in the algorithm involve the alignment refinement procedure described in Algorithm 10 (RefineBestAlignments) specified in Supplementary Section S3. Algorithm 10 optimizes TM-scores and refines alignments by considering differently positioned alignment fragments of different lengths. It takes three parameters: the numbers of query and subject structures, where all possible pairs are processed in parallel, and a gap opening penalty for the COMER2 DP algorithm [27] to generate alignments that optimize TM-scores given the superpositions. The complete outline of the GTalign algorithm is provided below.

1. Index query and subject structures and store the spatial indices in a k-d tree data structure.
2. Assign secondary structure states to the structures at each residue in parallel. This assignment is determined by the coordinates of five residues centered around the residue under consideration, with distance cutoffs between residues optimized in [21].
3. Calculate transformation matrices based on continuous fragment pairs in parallel for all queries and subjects, their matched fragment pairs, and fragment positions.
4. Apply Algorithm 10 with parameters (n_{Q}, n_{S}, 0) to refine alignments obtained from the superpositions found in the previous step to maximize the TM-score (always normalized by the length of the shorter protein). n_{Q} and n_{S} are the numbers of query and subject structures in batches. Here and in the following steps, keep track of the maximum TM-score and the corresponding transformation matrix for all query-subject pairs.
5. If the option `--add-search-by-ss` is specified, apply Algorithm 10 with parameters (n_{Q}, n_{S}, $\{-0.6, 0\}$) to refine alignments obtained from the application of the COMER2 DP algorithm [27] using a scoring function based on secondary structure matching and sequence similarity score [40].
6. Apply Algorithm 1 (DeepSuperpositionSearch) to find the most favorable superpositions through a deep search using spatial indices. The search depth is controlled with the `--speed` option. This step is central to the GTalign method because it enables rapid exploration of the superposition space, resulting in accurate alignments. We provide a detailed specification of Algorithm 1 in Supplementary Section S3.
7. Apply Algorithm 10 with parameters (n_{Q}, n_{S}, $\{-0.6, 0\}$) to refine alignments obtained from the application of the COMER2 DP algorithm using a scoring function based on secondary structure matching and TM-score, given the optimal transformation matrices obtained so far.
8. Apply Algorithm 10 with parameters (n_{Q}, n_{S}, $\{-0.6, 0\}$) to refine alignments obtained from the application of the COMER2 DP algorithm using TM-score as a scoring function. Here, the number of repetitions in Algorithm 10 is configurable (option `--convergence`).
9. Produce final alignments using the COMER2 DP algorithm based on the optimal transformation matrices in parallel for all queries and subjects.
10. Calculate TM-scores, root-mean-squared differences (RMSDs), and other alignment-related statistics in parallel for all queries and subjects.

Steps 1 and 2 prepare data for processing. Steps 3 and 4 identify protein superpositions by matching continuous protein segments, similar to TM-align's initial gapless matching [21]. These steps are sufficient to capture optimal superpositions of proteins sharing high structural similarity over a significant fraction of the length of at least one protein of a pair. Step 5 occasionally improves superpositions found in the previous steps. Step 6 conducts an extensive superposition search by matching different protein spatial regions. Steps 7 and 8 represent the refinement of transformation matrices and related alignments obtained earlier, meaning that alignment regions typically do not change or change slightly. Steps 9 and 10 prepare results for output. All the steps are based on algorithms and data structures designed to maximize instruction and memory throughput.

4.3 Spatial index data structure

To accelerate the superposition search, protein structures are initially indexed (step 1 of the algorithm outline). Each structure's index is stored in a k-d tree data structure, which hierarchically organizes protein atom coordinates. This organization allows for the retrieval of the nearest neighbor in the tree for a query atom with specified coordinates in constant, $O(1)$ time.

4.4 Accelerated superposition search using spatial indexing

The accelerated superposition search process (step 6 of the algorithm outline) leverages spatial indexing to find optimal superpositions for query and subject proteins within a data chunk. Conducted in parallel for all query-subject protein pairs in the chunk, this process explores numerous initial superposition configurations per protein pair simultaneously.

Initially, the process calculates initial superpositions, or transformation matrices, based on continuous query and subject protein fragments spanning the entire extent of both proteins. The search depth, determining the number of superpositions to explore, and the fragment length depend on the the query and subject protein lengths, with the fragment length not exceeding 100 residues. Regions with low local secondary structure similarity between query-subject fragment pairs avoid the calculation and exploration of initial superpositions.

Upon completing initial superpositions, the shorter protein undergoes transformation to obtain spatial overlays. Then, alignments are generated between the shorter protein and the other using the longer protein's index in parallel over residues, achieved in constant time complexity. This routine repeats twice: Initially produced alignments refine spatial overlays, followed by repeated alignment production using the protein index while ensuring matching protein secondary structure this time.

The most favorable alignment for the query-subject pair is then selected based on the highest TM-score. However, alignments obtained using spatial indices are sequence order-independent. Therefore, approximate sequence order-dependent TM-scores are computed from these alignments, with one structure transformed, in sub-linear time considering a maximum of 512 aligned residues.

Next, a small subset of transformation matrices with the highest approximate scores is chosen for TM-score calculation using the COMER2 DP algorithm. Further refinement involves selecting an even smaller subset of transformation matrices corresponding to the highest TM-scores to optimize alignments, considering different-length and differently positioned alignment fragments. Finally, the best alignment for the query-subject pair is selected and refined similarly, employing full DP and TM-score optimization.

A detailed specification of this procedure is provided in Supplementary Section S3.

4.5 Calculation of rotation matrices

GTalign computes rotation matrices using the Kabsch algorithm [41, 42]. Solving for the eigenvalues and eigenvectors of a cross-covariance matrix, $K \in \mathbb{R}^{3\times 3}$ (R in the original notation), requires double-precision arithmetic. To render the problem solvable in single precision, thereby boosting instruction and memory throughput, K is normalized by the mean of the absolute values of its elements. It is easy to show that this operation corresponds to scaling the coordinates of protein atoms. Effectively, rotation matrices for large proteins can be considered as obtained using coordinates expressed in nanometers instead of Angstroms, preventing single-precision arithmetic overflow and underflow. The resulting rotation matrices exhibit an insignificant error (on the order of 10^{-5} on average) with no discernible impact on superposition and structural alignment while still ensuring high performance.

4.6 Dynamic programming implementation

A previously published algorithm [27] was employed to implement the dynamic programming (DP) algorithms. The time complexity to calculate DP matrices is $O(\max_q l_q + \max_s \tilde{l}_s)$, with a constant factor dependent on the number of threads running in parallel. (The computation involves $(\max_q l_q + \max_s \tilde{l}_s)/32$ iterations of independent and parallelized calculations, executed in $O(32)$ time by GPU threads.) Here, the maximums are taken over the lengths of all query (l_q) and subject ($\tilde{l}_s$) proteins in a data *chunk*. In instances where DP matrix values are only required to update backtracking information, the memory complexity for DP matrices is $O(n_\mathrm{Q} \sum_s \tilde{l}_s)$ [27], with n_Q representing the number of query proteins in the chunk.

DP matrices are built using a gap opening penalty, while the gap extension cost is set to 0 unless otherwise specified. For optimizing memory usage, match scores are modified before writing to memory—either by negating them for non-negative alignment scores or subtracting a large constant for potentially negative scores. The reverse operation is subsequently applied upon reading from memory. This approach minimizes memory requirements, facilitating greater data accommodation and parallelization.

4.7 Algorithm efficiency

The high efficiency of the developed algorithms, particularly the spatial indices, is best demonstrated by aligning large protein structures. For instance, GTalign only took seconds to align and provide a superposition for two virus nucleocapsid variants, 7a4i and 7a4j (37,860 residues each), featuring different chain orders on a single Tesla V100 GPU (Supplementary Fig. S9). In contrast, aligning these complexes using TM-align took more than three months. Although TM-align is not typically used for aligning complexes, GTalign's efficiency may open new possibilities for exploring large complexes when chain order preservation is important.

4.8 Performance improvement potential

GTalign's efficiency can be further enhanced by considering three key aspects. First, GTalign currently uses 32-bit floating-point precision (FP32) operations. Exploring the adoption of 16-bit (FP16) or even 8-bit (FP8) floating point precision before the final stages of alignment has the potential to increase the degree of parallelization by 2 to 4-fold.

Second, the COMER2 DP algorithm, a critical component for accuracy, is employed several times throughout the structural alignment search procedure. Substituting it with spatial matching, as outlined in Supplementary Section S3, at all intermediate stages and reserving it solely for the final alignment stage could result in a significant speedup.

Finally, the third aspect involves similarity selection on the coarse scale. By encoding structures with embeddings and utilizing indexed vector databases [43], GTalign could achieve nearly instantaneous selection of similar protein candidates and a constant-time database search and alignment, regardless of the database size.

4.9 Prescreening for similarities in sequence and structure space

GTalign allows for an initial screening in the sequence space (option `--pre-similarity`) to identify potential similarities before engaging in more detailed structural analysis. The implementation of this procedure is based on calculating local ungapped alignment scores between protein sequences using a sequence similarity score table [40] and does not involve dynamic programming. Protein pairs with alignment scores exceeding a specified threshold progress to the subsequent stages of structural analysis.

Additionally, an initial screening for similarities is available in the structure space using the `--pre-score` option. With this option, protein pairs with provisional TM-scores, obtained in step 4 of the algorithm outline, lower than a specified threshold are excluded from further processing.

4.10 GTalign software

GTalign incorporates several key features that contribute to its versatility and user-friendly nature. Developed using the OpenMP standard for CPUs and CUDA architecture for GPUs, GTalign is compatible with various computing architectures, including NVIDIA Pascal, Turing, Volta, Ampere, Ada Lovelace, and subsequent GPU architectures. (The GPU version exhibits a 10–20x increase in speed.) Its independence from external packages ensures seamless operation across different compilers (GCC, LLVM/Clang, MSVC) and their respective versions. GTalign is cross-platform software, with binary packages precompiled for Linux and Windows x64 operating systems. For other platforms, users have the flexibility to compile GTalign from its source code. GTalign usage is straightforward: No structure database preprocessing is required. Users can effortlessly employ GTalign by directly providing files, compressed files (`gzip`), directories, and/or archives (`tar`) of protein structures as command-line arguments. This user-centric design enhances accessibility and facilitates streamlined integration into diverse computational environments.

4.11 Alignment accuracy evaluation

The evaluation of structural alignment accuracy is based on assessing how accurately the structural alignments of protein pairs translate to spatial agreement in their respective structures. This self-contained evaluation is unbiased, as it does not depend on external classifications, which may be constructed using specific sequence and structure alignment tools.

The superposition of two aligned proteins is evaluated by the TM-score and RMSD, calculated by the established method TM-align [21] using the `-I` option (Fig. 7). Notably, in this setting, TM-align does not perform a global superposition search but instead optimizes superposition constrained by a given alignment, leaving it unchanged.

GDT_TS scores were calculated using the TM-score tool [26], with minimal modifications to the source code to normalize GDT_TS by the number of aligned residue pairs. The adapted TM-score code is publicly available.

In the benchmarks, alignments are evaluated based on (i) the TM-score normalized by the length of the shorter protein and (ii) the TM-score normalized by the query length. The first scenario considers all structural similarities, including instances where smaller proteins match regions of larger proteins. The second scenario downgrades the importance of alignments between the query and a much shorter subject protein, providing a more favorable position for some methods (e.g., Dali [22]) as their measures (e.g., Z-score) reduce the significance of such alignments.

4.12 SCOPe-based evaluation

This evaluation aimed to assess the ability of the tools to match SCOPe 2.08 [29] domains to families, superfamilies, and folds. True positives (TPs) at the family, superfamily, and fold level were defined as pairs of structures from the same family, the same superfamily but different families, and the same fold but different superfamilies, respectively. Self-matches were excluded. The sizes of these groups are referred to as effective sizes. False positives (FPs) were identified as pairs from different folds.

Precision and recall were calculated as #TP/(#TP + #FP) and #TP/P, respectively, where P represents the total number of positive pairs. The number of TPs, #TP, and P for precision-recall (PR) curves were downweighted by the effective size of family, superfamily, and fold for respective-level calculations. The number of FPs, #FP, was downweighted by the effective fold size. The weighting for counts was consistent with the approach used in [25].

Before conducting sensitivity and PR analyses, alignments generated by the tools were sorted by their significance measure. Foldseek (default parametrization) alignments were sorted by E-value, while FATCAT alignments were sorted by P-value, and Dali alignments by Z-score. DeepAlign alignments were sorted by DeepScore. TM-align and GTalign alignments were sorted by the harmonic mean of the TM-scores normalized by the query and subject lengths. The harmonic mean proved superior to the arithmetic mean for TM-align and GTalign alignments due to its ability to reduce significance for structure pairs with large length differences. However, the arithmetic mean was more suitable for Foldseek `--tmalign-fast 1` and `--tmalign-fast 0` alignments, as most of such pairs had already been filtered out.

Secondary TM-scores, referred to as 2TM-scores, were introduced to rank GTalign alignments in the SCOPe-based evaluation. The 2TM-score is calculated over the alignment excluding unmatched helices and provided slightly improved results for fold-level evaluations. Options to calculate 2TM-scores (`--2tm-score`) and rank alignments by the harmonic mean of the TM-scores or 2TM-scores are available starting with version 0.15.0.

4.13 The SCOPe40 2.08 dataset

All protein domains from the SCOPe 2.08 [29] database filtered to 40% sequence identity (SCOPe40 2.08), totaling 15,177, were searched with query protein domains selected randomly, one per superfamily, from the same SCOPe40 2.08 dataset. Representatives that Dali [22] failed to reformat for its initial structural representation were omitted, resulting in a total of 2045 queries.

To ensure consistent structure interpretation between TM-align and the other tools, the structure files underwent the following changes: (i) the first model of multi-model files was retained; (ii) the chain identifier was set to 'A' to make a single-chain structure; (iii) residues were renumbered sequentially; (iv) residues lacking at least one of the N, CA, C, and O atoms were removed. HETATM records were disregarded when using Foldseek as its interpretation of these records differed from that of TM-align.

4.14 The PDB20 dataset

PDB [30] structures filtered using `blastclust` [44] to 20% sequence identity with a length coverage threshold of 70% (PDB20), totaling 18,801, were queried with 186 CAMEO protein structure targets [45] released over three months from 07/24/2021 through 10/16/2021. The CAMEO targets and the PDB20 structures maintained no more than 20% sequence identity.

The structure files were preprocessed to ensure consistent structure interpretation across the tools: HETATM records and residues lacking at least one of the N, CA, C, and O atoms were removed. Also, the first model of multi-model files was retained.

4.15 The Swiss-Prot dataset

All UniProtKB/Swiss-Prot [31] protein structures (542,378) from the AlphaFold Database [4] were searched with 40 proteins representative of structurally diverse CRISPR-Cas systems [46]. The selection of the 40 query proteins followed a specific process: First, the 5831 PDB protein chains associated with CRISPR-Cas systems (downloaded on 10/19/2023) were clustered at a TM-score threshold of 0.4 with a length coverage threshold of 40% using GTalign with options `--speed=13 --add-search-by-ss --cls-coverage=0.4`

--cls-threshold=0.4 --ter=0 --split=2. Subsequently, the top 40 members from every third singleton cluster, sorted by length, were chosen as queries, with an average length of 382 residues. The query structures underwent preprocessing, involving the removal of HETATM records and residues lacking at least one of the N, CA, C, and O atoms.

4.16 The HOMSTRAD dataset

The HOMSTRAD dataset, comprising reference structural alignments of protein families and accompanying structure files, was obtained from Ref. [34], containing 398 multiple protein structure alignments from the HOMSTRAD database [32]. (The original data were inaccessible.) For benchmarking purposes, each family's first protein from the reference alignments was aligned with every other protein of the same family, resulting in a total of 1722 pairwise alignments.

4.17 Computer system configuration

Unless otherwise specified, all benchmark tests were conducted on a server equipped with two Intel Xeon Gold 5115 CPUs @ 2.4 GHz (20 hardware threads per CPU), 128GB DDR4 RAM, and three NVIDIA Tesla V100-PCIE-16GB GPU accelerators, running the CentOS 7 operating system.

4.18 Runtime evaluation

The runtimes of all tools were measured by the Linux `time` command.

4.19 GTalign settings

Unless otherwise specified, all analyses were performed using GTalign version 0.14.0, compiled with GPU support. For protein pairs in the HOMSTRAD dataset, alignments were generated with the command

```
gtalign --qrs=<query_file> --rfs=<subject_file> -o <output_dir> --hetatm --dev-min-
    length=3 --speed=<speed> --pre-score=0 -s 0 --add-search-by-ss --dev-mem=4096
```

The command used to process queries for the SCOPe40 2.08, PDB20, and Swiss-Prot datasets was

```
gtalign --qrs=<query_dir> --rfs=<db_dir> -o <output_dir> --hetatm --dev-queries-total-
    length-per-chunk=1500 --dev-min-length=3 --dev-max-length=<max_len> --speed=<speed> -
    s 0.44 --add-search-by-ss --nhits=<n_hits> --nalns=<n_hits> --dev-N=3
```

where `<query_dir>` represents the directory of query structures, `<db_dir>` denotes the directory or `tar` archive (Swiss-Prot dataset) containing subject structures, `<max_len>` is 5000 for the PDB20 dataset and 4000 for the others, and `<n_hits>` is 10000, 4000, and 50000 for the SCOPe40 2.08, PDB20, and Swiss-Prot datasets, respectively. The value of the `--speed` option, along with the usage of several additional options (`--pre-score` and `--pre-similarity`), is indicated in Figs. 1–5 and Supplementary Tables and Figures. When employing the initial screening in the sequence space (option `--pre-similarity`), options `-s 0.3` and `-c cachedir` were specified. The latter was used unconditionally for the Swiss-Prot dataset. In the SCOPe-based evaluation, the `--2tm-score` option was specified to calculate 2TM-scores (version 0.15.0).

GTalign calculates and outputs TM-scores normalized by the length of both proteins in a pair. Consequently, the corresponding TM-scores were utilized as GTalign's measures to sort alignments (left panels of Fig. 1b–d and Fig. 2a–c).

4.20 TM-align settings

Parallel processing of all queries for each dataset was achieved by iteratively running 40 instances of TM-align [21] version 20220412 simultaneously. For the HOMSTRAD and three other datasets, each process instance was executed with the following options, respectively: `<query> <subject_file> -het 1` and `<query> -dir2 <db_dir> <lst_file> -het 1`. Here, `<query>` represents a query file, `<db_dir>` is a directory of subject structure files, and `<lst_file>` is a list file of all subjects. For the Swiss-Prot dataset, `-outfmt 2` was included to reduce disk space usage. The fast version (TM-align -fast) utilized an additional option, `-fast`.

4.21 Dali settings

The standalone version DaliLite.v5 [22] was employed in the benchmark tests. Prior to initiating searches, structure files underwent reformatting to an initial representation using the command `import.pl --pdbfile <struct_file> --dat <dir> --pdbid <id>`. In this command, `<struct_file>` refers to a

structure file, <dir> is a directory for reformatted structures, and <id> is an assigned structure identifier. Reformatting failed for 104 and 525 subject structures from the SCOPe40 2.08 and PDB20 datasets, respectively. The time taken for reformatting was excluded from runtime evaluations.

For parallelizing searches, 40 process instances were executed using the command `dali.pl --np 40 --query <query_list> --db <sbjct_list> --dat1 <query_dir> --dat2 <db_dir> --outfmt alignments`. In this command, <query_list> and <sbjct_list> represent the list files of query and subject structures, respectively, while <query_dir> and <db_dir> indicate the directories of (reformatted) query and subject structures. Dali's Z-score served as the sorting measure, arranging the alignments in descending order based on it.

For the HOMSTRAD dataset, reformatting was not used, and alignments were directly generated with the command `dali.pl --pdbfile1 <query_file> --pdbfile2 <subject_file> --dat1 <query_dir> --dat2 <subject_dir> --outfmt alignments`.

4.22 DeepAlign settings

All query-subject structure pairs in each dataset underwent parallel processing through the iterative execution of 40 simultaneous instances of DeepAlign [18] version v1.4 Aug-20-2018 (https://github.com/realbigws/DeepAlign), using the command `DeepAlign <query_file> <subject_file>`.

DeepAlign outputs the TM-score normalized by the length of the shorter protein, which was used as its measure for sorting alignments in the corresponding evaluations (left panel of Fig. 1b and c). When evaluating alignments based on the TM-score normalized by the query length, DeepAlign's DeepScore was utilized as its measure to sort the alignments (left panel of Fig. 2a and b).

4.23 FATCAT settings

FATCAT 2.0 [28] searches were conducted iteratively for all queries in three datasets using the rigid structural alignment setting. The command `FATCATSearch <query_file> <sbjct_list> -i2 <db_dir> -r -o <output_file> -m` was utilized, with <sbjct_list> and <db_dir> representing the list file and the directory of subject structures, respectively. For each query, FATCAT automatically initiated parallel processes corresponding to the number of processors in the system, in this case, 40. FATCAT's P-value served as the sorting measure, i.e., the alignments were sorted in ascending order based on it. It is noteworthy that FATCAT disregards HETATM records, and thus, these records were also omitted during alignment accuracy evaluation with TM-align.

For the HOMSTRAD dataset, alignments were generated with the following command: `FATCAT -p1 <query_file> -p2 <subject_file> -r -o <output_file> -m`.

4.24 Foldseek settings

Foldseek [25] version `d1d1b868a571a9a0c62ae50b07139ebdd224f879` was used (downloaded on 6/25/2023). All queries in the SCOPe40 2.08, PDB20, and Swiss-Prot datasets were parallelized using the following command:

```
foldseek easy-search <query_dir> <db_dir> <output_file> <tmp_dir> --threads 40 --max-
    seqs 4000 --format-output query,target,evalue,qtmscore,ttmscore,alntmscore,rmsd,
    qstart,qend,qlen,tstart,tend,tlen,qaln,taln
```

In this command, <query_dir> denotes the directory of query structures, <db_dir> indicates the directory or `tar` archive (Swiss-Prot dataset) containing subject structures, and <tmp_dir> is a temporary directory. For the Swiss-Prot dataset, the option value `--max-seqs 20000` was specified. Foldseek was additionally parameterized with the options `--alignment-type 1` and `--tmalign-fast 0` to enable the use of the fast (Foldseek --tmalign-fast 1) and regular (Foldseek --tmalign-fast 0) versions of TM-align for alignment production.

As Foldseek generates TM-scores, the TM-score normalized by the length of the shorter protein was employed as the sorting measure for alignments in the left panel of Fig. 1b–d. When evaluating alignments based on the TM-score normalized by the query length (Fig. 2a–c), E-value and the average TM-score (Foldseek --tmalign-fast 0/1), a recommended metric [25], were utilized as measures to sort the alignments in ascending and descending order, respectively.

For the HOMSTRAD dataset, the command was modified to specify individual query and subject structure files instead of their directories. Also, the option `--threads` was set to 1, and the additional options `--prefilter-mode 2` and `-e 1e6` were included.

4.25 Figure preparation

Molecular graphics images were generated using UCSF Chimera [47]. Plots were created using the ggplot2 package [48] in R [49], versions 3.6.0 and 4.3.2.

Data availability

The SCOPe 2.08 PDB-style files with coordinates for the SCOPe40 2.08 dataset are available at https://scop.berkeley.edu/downloads/. The PDB files for the PDB20 dataset are available at https://www.rcsb.org/downloads. The archive of the Swiss-Prot protein structures is available at https://ftp.ebi.ac.uk/pub/databases/alphafold/latest/swissprot_pdb_v4.tar. The HOMSTRAD dataset was obtained from http://yanglab.nankai.edu.cn/mTM-align/benchmark. Benchmark data are deposited to Zenodo at https://doi.org/10.5281/zenodo.11148017.

Code availability

The source code and software packages of GTalign are available at https://github.com/minmarg/gtalign_alpha, and the corresponding releases are archived on Zenodo: https://zenodo.org/records/10433420 (version 0.14.0; doi: 10.5281/zenodo.10433420) and https://zenodo.org/records/10433419 (versions >0.14.0; doi: 10.5281/zenodo.10433419). The lists of structure identifiers, as well as scripts and programs for structure preprocessing, benchmarking, and producing graphs, are available at https://github.com/minmarg/gtalign-evaluation.

Acknowledgements

This work was inspired by the success of TM-align [21]. The research was supported by Research Council of Lithuania (LMTLT) grant S-MIP-23-104.

Author contributions

M.M. conceived and supervised the project, designed the algorithms, developed GTalign, conducted the analyses, and wrote the manuscript.

Competing interests

M.M. is a co-founder of Hiomics digital and declares non-financial competing interests.

References

[1] Vaswani, A. *et al.* Guyon, I. *et al.* (eds) *Attention is all you need.* (eds Guyon, I. *et al.*) *Advances in Neural Information Processing Systems*, Vol. 30 of *NIPS'17*, 6000–6010 (Curran Associates, Inc., Red Hook, NY, USA, 2017).

[2] Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci. U.S.A.* **118**, e2016239118 (2021).

[3] Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).

[4] Varadi, M. *et al.* Alphafold protein structure database in 2024: providing structure coverage for over 214 million protein sequences. *Nucleic Acids Res.* gkad1011 (2023).

[5] Lin, Z. *et al.* Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).

[6] Pavlopoulos, G. A. *et al.* Unraveling the functional dark matter through global metagenomics. *Nature* **622**, 594–602 (2023).

[7] Barrio-Hernandez, I. *et al.* Clustering predicted structures at the scale of the known protein universe. *Nature* **622**, 637–645 (2023).

[8] Durairaj, J. *et al.* Uncovering new families and folds in the natural protein universe. *Nature* **622**, 646–653 (2023).

[9] van Montfort, R. L. M. & Workman, P. Structure-based drug design: aiming for a perfect fit. *Essays Biochem.* **61**, 431–437 (2017).

[10] Goodsell, D. S. *et al.* RCSB Protein Data Bank: Enabling biomedical research and drug discovery. *Protein Sci.* **29**, 52–65 (2020).

[11] Borkakoti, N. & Thornton, J. M. AlphaFold2 protein structure prediction: Implications for drug discovery. *Curr. Opin. Struct. Biol.* **78**, 102526 (2023).

[12] Holm, L. & Sander, C. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.* **233**, 123–138 (1993).

[13] Kawabata, T. & Nishikawa, K. Protein structure comparison using the Markov transition model of evolution. *Proteins* **41**, 108–122 (2000).

[14] Yang, A.-S. & Honig, B. An integrated approach to the analysis and modeling of protein sequences and structures. I. protein structural alignment and a quantitative measure for protein structural distance. *J. Mol. Biol.* **301**, 665–678 (2000).

[15] Krissinel, E. & Henrick, K. Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions. *Acta Crystallogr. D Biol. Crystallogr.* **60**, 2256–2268 (2004).

[16] Shindyalov, I. N. & Bourne, P. E. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng.* **11**, 739–747 (1998).

[17] Ye, Y. & Godzik, A. Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinformatics* **19**, ii246–ii255 (2003).

[18] Wang, S., Ma, J., Peng, J. & Xu, J. Protein structure alignment beyond spatial proximity. *Sci. Rep.* **3**, 1448 (2013).

[19] Ortiz, A. R., Strauss, C. E. & Olmea, O. MAMMOTH (matching molecular models obtained from theory): An automated method for model comparison. *Protein Sci.* **11**, 2606–2621 (2002).

[20] Zemla, A. LGA: a method for finding 3d similarities in protein structures. *Nucleic Acids Res.* **31**, 3370–3374 (2003).

[21] Zhang, Y. & Skolnick, J. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Res.* **33**, 2302–2309 (2005).

[22] L., H. Dali server: structural unification of protein families. *Nucleic Acids Res.* **50**, W210–W215 (2022).

[23] Holm, L. Benchmarking fold detection by DaliLite v.5. *Bioinformatics* **35**, 5326–5327 (2019).

[24] Liu, Z., Zhang, C., Zhang, Q., Zhang, Y. & Yu, D.-J. TM-search: An efficient and effective tool for protein structure database search. *J. Chem. Inf. Model.* **64**, 1043–1049 (2024).

[25] van Kempen, M. *et al.* Fast and accurate protein structure search with foldseek. *Nat. Biotechnol.* (2023).

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

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

[28] Li, Z., Jaroszewski, L., Iyer, M., Sedova, M. & Godzik, A. FATCAT 2.0: towards a better understanding of the structural diversity of proteins. *Nucleic Acids Res.* **48**, W60–W64 (2020).

[29] Chandonia, J. *et al.* SCOPe: improvements to the structural classification of proteins — extended database to facilitate variant interpretation and machine learning. *Nucleic Acids Res.* **50**, D553–D559

(2021).

[30] Burley, S. *et al.* RCSB Protein Data Bank: powerful new tools for exploring 3d structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic Acids Res.* **49**, D437–D451 (2020).

[31] UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.* **49**, D480–D489 (2021).

[32] Stebbings, L. A. & Mizuguchi, K. HOMSTRAD: recent developments of the homologous protein structure alignment database. *Nucleic Acids Res.* **32**, D203–D207 (2004).

[33] Xu, J. & Zhang, Y. How significant is a protein structure similarity with tm-score=0.5? *Bioinformatics* **26**, 889–895 (2010).

[34] Dong, R., Peng, Z., Zhang, Y. & Yang, J. mTM-align: an algorithm for fast and accurate multiple protein structure alignment. *Bioinformatics* **34**, 1719–1725 (2018).

[35] Söding, J. & Remmert, M. Protein sequence comparison and fold recognition: progress and good-practice benchmarking. *Curr. Opin. Struct. Biol.* **21**, 404–411 (2011).

[36] Fox, N. K., Brenner, S. E. & Chandonia, J.-M. SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Res.* **42**, D304–D309 (2013).

[37] Skolnick, J., Arakaki, A. K., Lee, S. Y. & Brylinski, M. The continuity of protein structure space is an intrinsic property of proteins. *Proc. Natl. Acad. Sci. U.S.A.* **106**, 15690–15695 (2009).

[38] Kolodny, R., Petrey, D. & Honig, B. Protein structure comparison: implications for the nature of 'fold space', and structure and function prediction. *Curr. Opin. Struct. Biol.* **16**, 393–398 (2006).

[39] Sadreyev, R. I., Kim, B.-H. & Grishin, N. V. Discrete-continuous duality of protein structure space. *Curr. Opin. Struct. Biol.* **19**, 321–328 (2009).

[40] Gonnet, G. H., Cohen, M. A. & Benner, S. A. Exhaustive matching of the entire protein sequence database. *Science* **256**, 1443–1445 (1992).

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

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

[43] Hamamsy, T. *et al.* Protein remote homology detection and structural alignment using deep learning. *Nat. Biotechnol.* (2023).

[44] Altschul, S. F. *et al.* Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402 (1997).

[45] Robin, X. *et al.* Continuous automated model evaluation (CAMEO)—perspectives on the future of fully automated evaluation of structure prediction methods. *Proteins* **89**, 1977–1986 (2021).

[46] Wang, J. Y., Pausch, P. & Doudna, J. A. Structural biology of CRISPR-cas immunity and genome editing enzymes. *Nat. Rev. Microbiol.* **20**, 641–656 (2022).

[47] Pettersen, E. *et al.* UCSF Chimera—a visualization system for exploratory research and analysis. *J. Comput. Chem.* **25**, 1605–1612 (2004).

[48] Wickham, H. *Ggplot2: Elegant graphics for data analysis* 2 edn. Use R! (Springer International Publishing, Cham, Switzerland, 2016).

[49] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria (2021). URL https://www.R-project.org/.

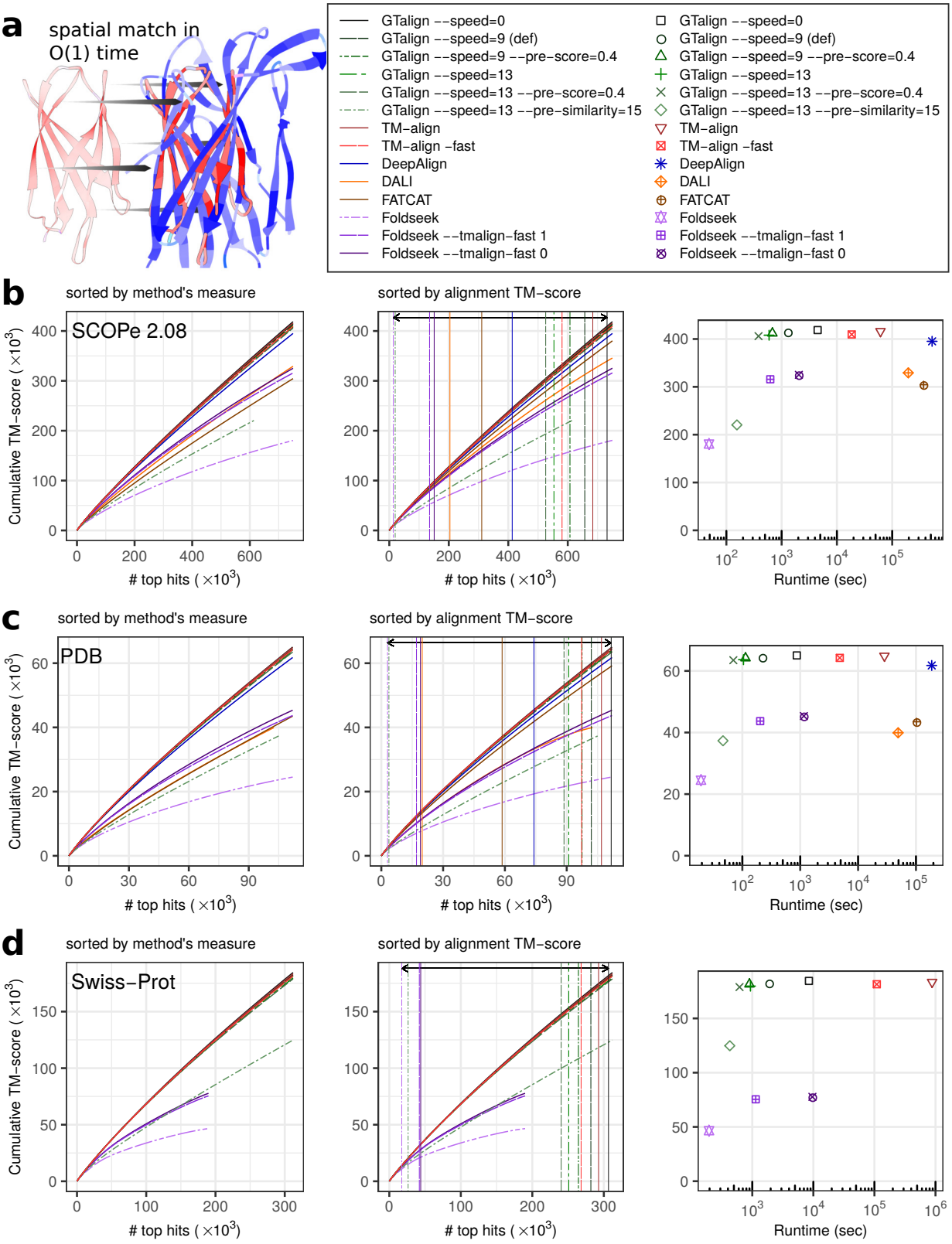

a
spatial match in
O(1) time
GTalign --speed=0
GTalign --speed=9 (def)
GTalign --speed=9 --pre-score=0.4
GTalign --speed=13
GTalign --speed=13 --pre-score=0.4
GTalign --speed=13 --pre-similarity=15
TM-align
TM-align -fast
DeepAlign
DALI
FATCAT
Foldseek
Foldseek --tmalign-fast 1
Foldseek --tmalign-fast 0
b
sorted by method's measure
sorted by alignment TM-score
SCOPe 2.08
Cumulative TM-score (×10^3)
top hits (×10^3)
Runtime (sec)
c
PDB
d
Swiss-Prot

Fig. 1 Results. a, Illustration of matching protein structures with $O(1)$ time complexity. GTalign explores numerous superpositions in parallel. Upon obtaining a superposition, the alignment between the query protein (red) and the subject protein (blue) is generated using the subject protein's spatial index. This index allows for the independent retrieval of the nearest residue in the subject protein for each residue in the query protein, enabling parallel processing. b, Benchmarking results on the SCOPe40 2.08 dataset with 2045 queries and 15,177 database entries. Parameterized runs of GTalign, TM-align, and Foldseek are included. The left panel plots the cumulative TM-score (normalized by the shorter protein length) against the number of top alignments ranked by a tool's measure (TM-score, Z-score, or P-value). In the middle panel, the alignments are sorted by their (TM-align-obtained) TM-score. Vertical lines indicate the number of alignments with a TM-score ≥ 0.5. The arrow denotes the largest difference in that number between GTalign (732,024) and Foldseek (13,371). The right panel shows the cumulative TM-score plotted against runtime in seconds. c, Benchmarking results on the PDB20 dataset with 186 queries and 18,801 database entries. d, Benchmarking results on the Swiss-Prot dataset with 40 queries and 542,378 database entries. DeepAlign, Dali, and FATCAT are excluded due to their long projected execution times. The Foldseek curves appear truncated due to the total number of hits it produced. The axes scales in Panels b–d are chosen to accommodate the maximum values of the cumulative TM-score and the number of top hits with a TM-score ≥ 0.5.

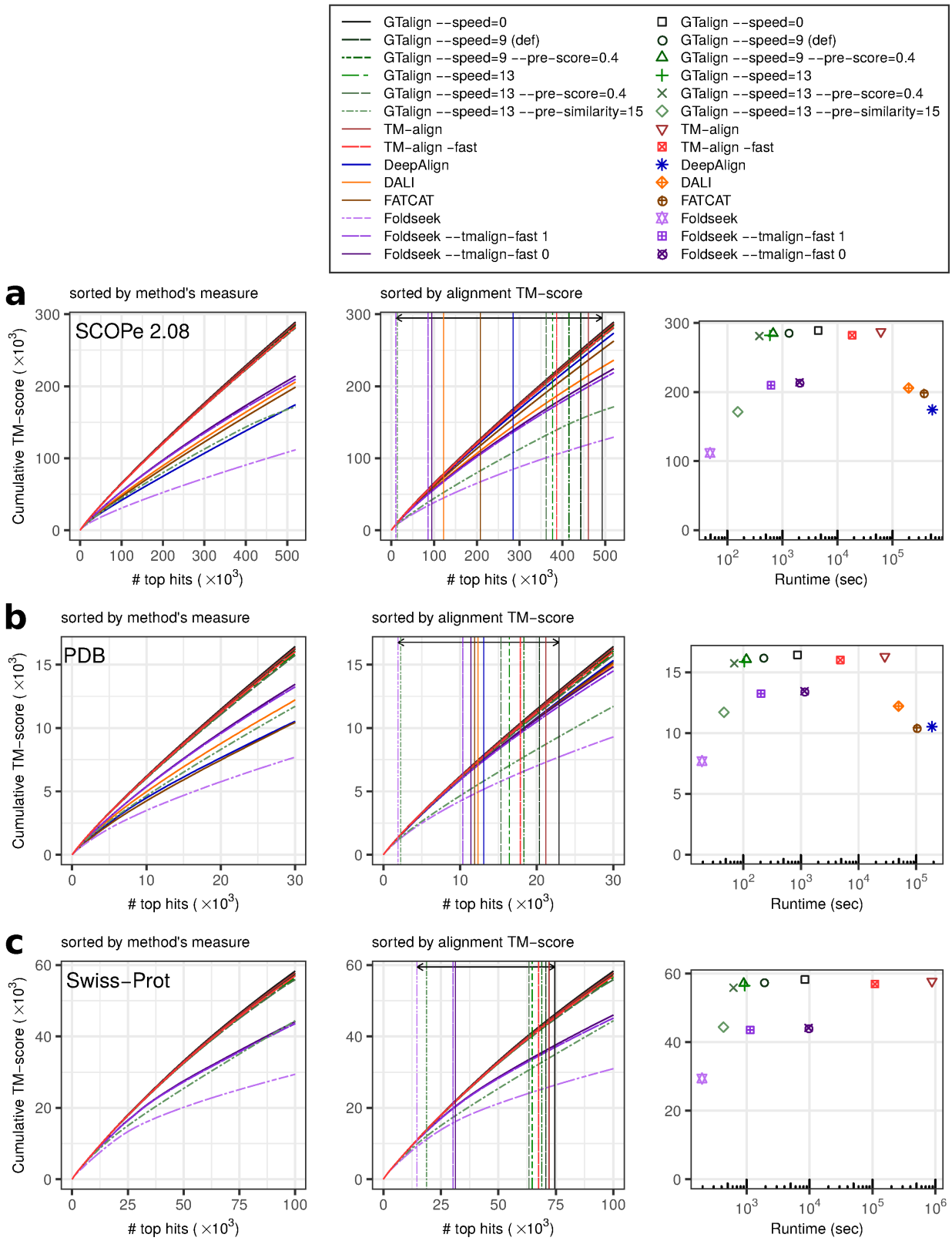


Fig. 2 Alignment evaluation results based on the TM-score normalized by the query length. a, Benchmarking results on the SCOPe40 2.08 dataset with 2045 queries and 15,177 database entries. The left panel plots the cumulative TM-score (normalized by the query length) against the number of top alignments ranked by a tool's measure (TM-score, Z-score, or P-value). In the middle panel, the alignments are sorted by the TM-align-obtained TM-score. Vertical lines denote the number of alignments with a TM-score ≥ 0.5. The arrow highlights the largest difference in that number between GTalign (492,887) and Foldseek (10,375). The right panel shows the cumulative TM-score plotted against runtime in seconds. b, Results on the PDB20 dataset with 186 queries and 18,801 database entries. c, Results on the Swiss-Prot dataset with 40 queries and 542,378 database entries. Excluding DeepAlign, Dali, and FATCAT due to their long projected execution times.

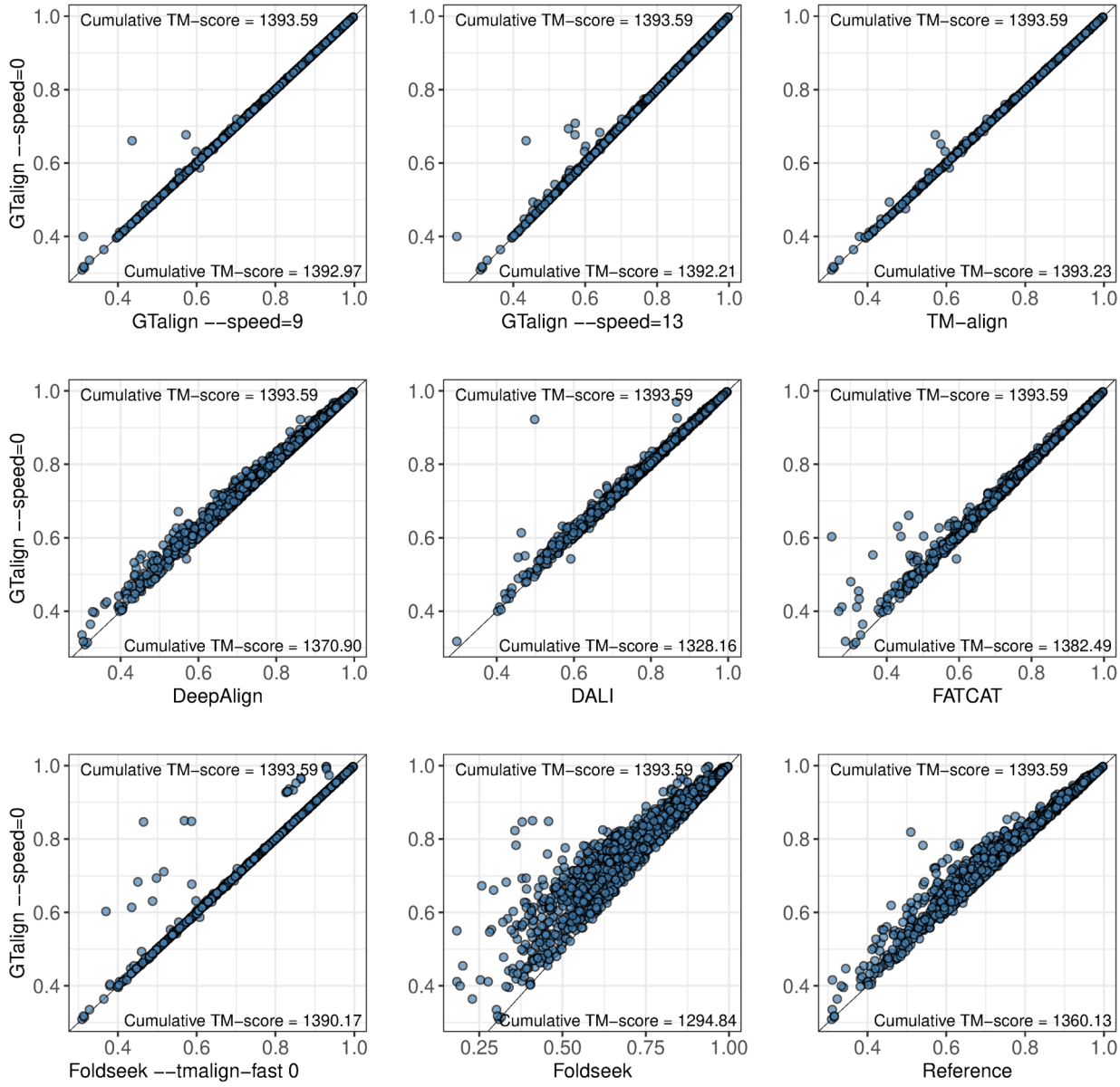


Fig. 3 Benchmarking results on the HOMSTRAD dataset. Each plot illustrates the distribution of TM-scores between two tools. The black line indicates identical distributions. 100 structure pairs for which Dali did not produce any alignment are excluded from the corresponding plot. "Reference" represents the original alignments in the database.

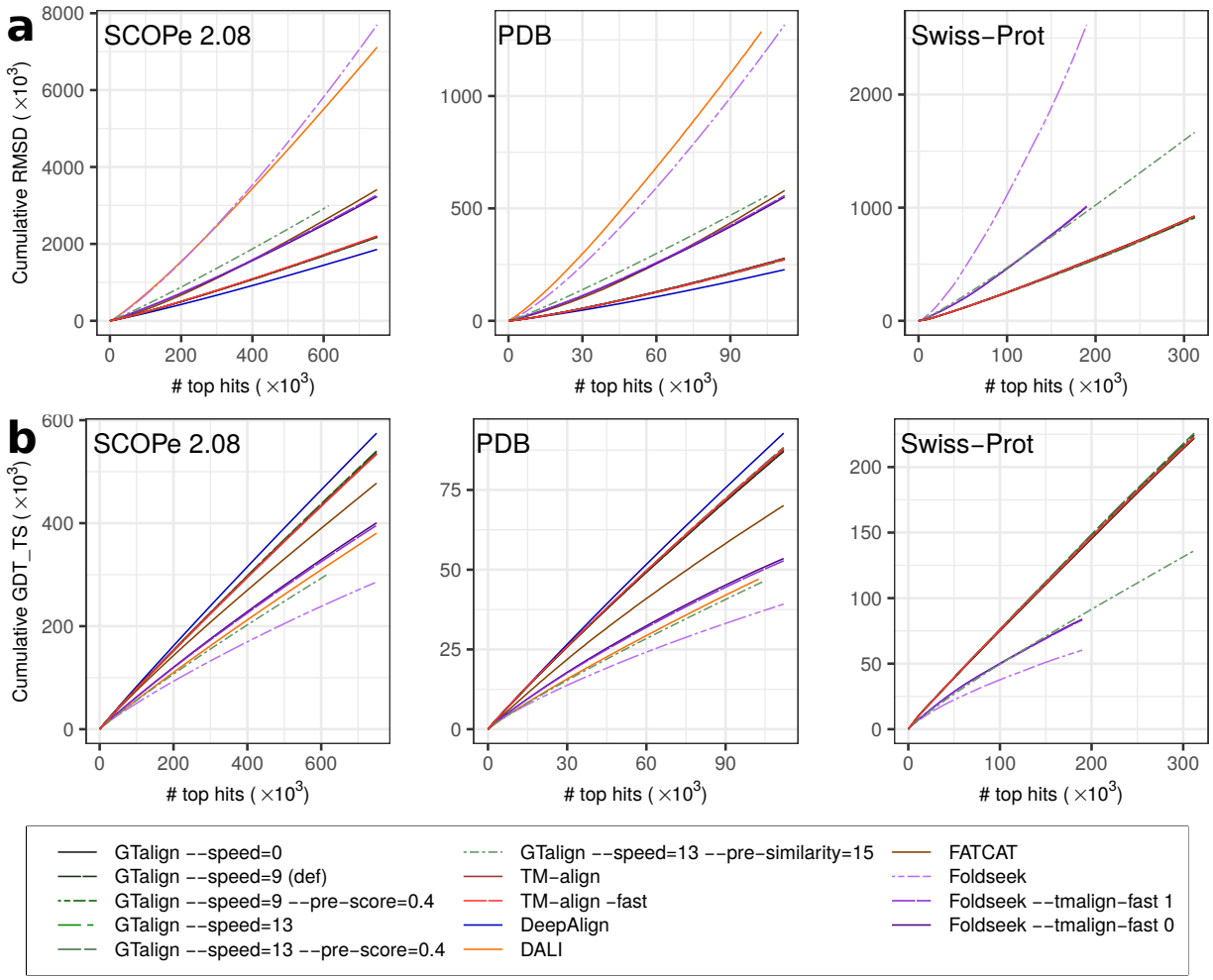


Fig. 4 Alignment evaluation results for the SCOPe40 2.08, PDB20, and Swiss-Prot datasets using RMSD (a) and GDT_TS scores normalized by the number of aligned residue pairs (b). The panels display the cumulative RMSD and GDT_TS scores plotted against the number of top alignments ranked by a tool's measure (TM-score, Z-score, or P-value). Foldseek alignments were ranked by Foldseek's TM-score, resulting in lower RMSDs and higher GDT_TS scores. The curves of the TM-align and GTalign variants, excluding GTalign `--speed=13 --pre-similarity=15`, closely match.

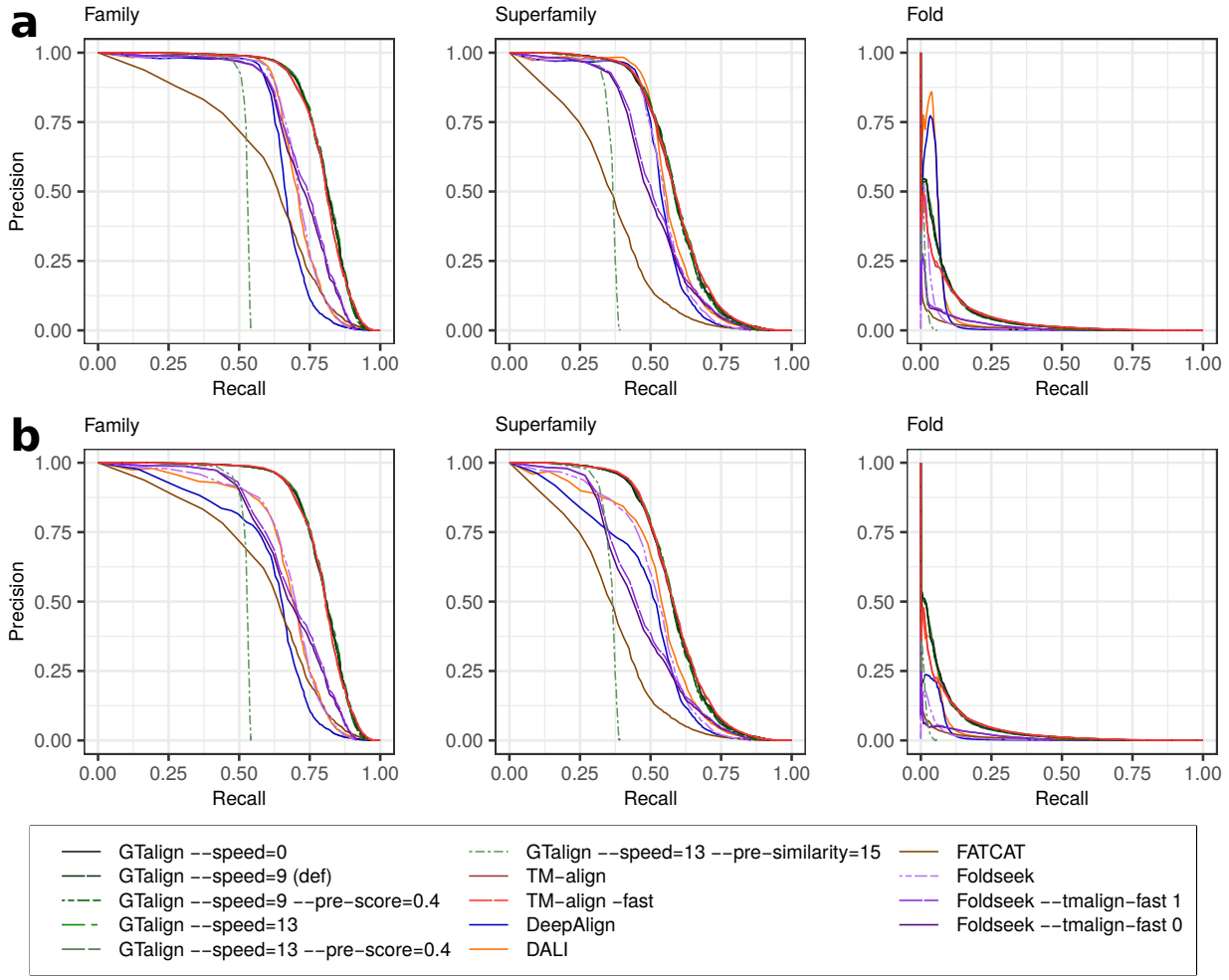


Fig. 5 Weighted precision-recall (PR) curves at the family, superfamily, and fold levels. The areas under these PR curves are reported in Supplementary Table S7. a, False positives for calculating precision and recall are pairs of structures from different SCOPe 2.08 folds, with the exception that those pairs belonging to Rossman-like (c.2–c.5, c.27, c.28, c.30, and c.31) or beta-propeller (b.66–b.70) folds are ignored [35]. b, False positives correspond to pairs from different SCOPe 2.08 folds without exceptions.

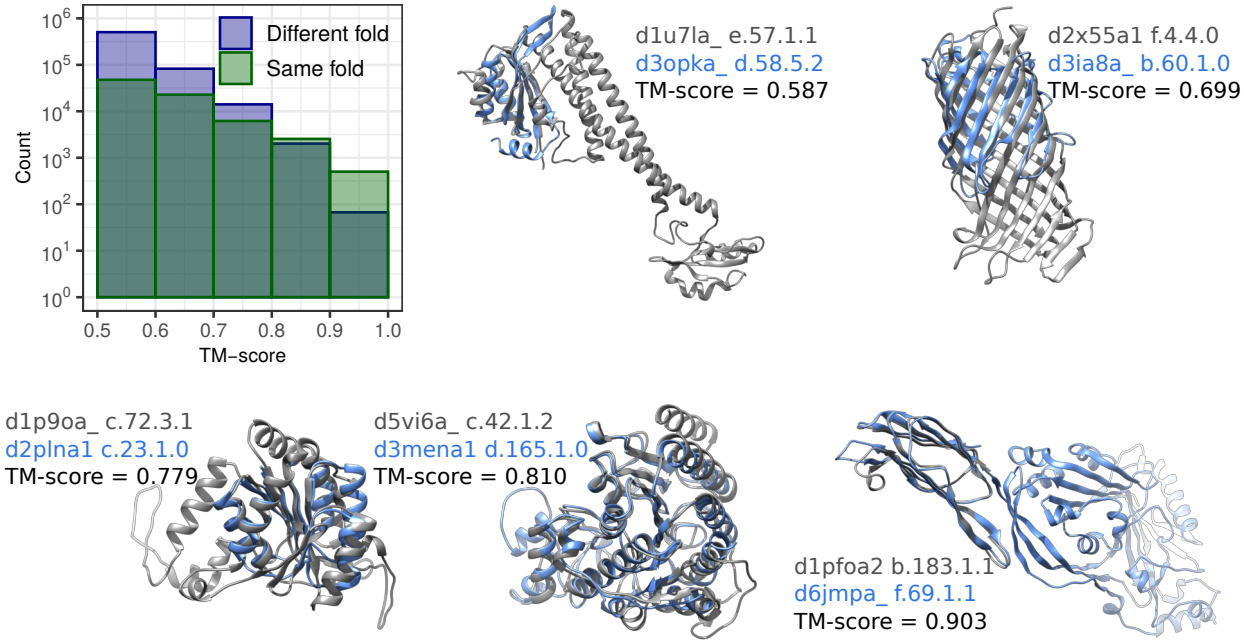


Fig. 6 Distributions of TM-scores (> 0.5) for domain pairs within the same and different SCOPe 2.08 folds, presented in bins of width 0.1. TM-scores, normalized by the length of the shorter protein, were calculated by aligning query and subject structures from the SCOPe40 2.08 dataset using TM-align. Representative examples of structure pairs from different folds and classes are provided for each TM-score bin. These examples, along with numerous other significantly structurally similar pairs from different folds and classes, are considered errors (false positives) in the reference SCOPe-based evaluation.

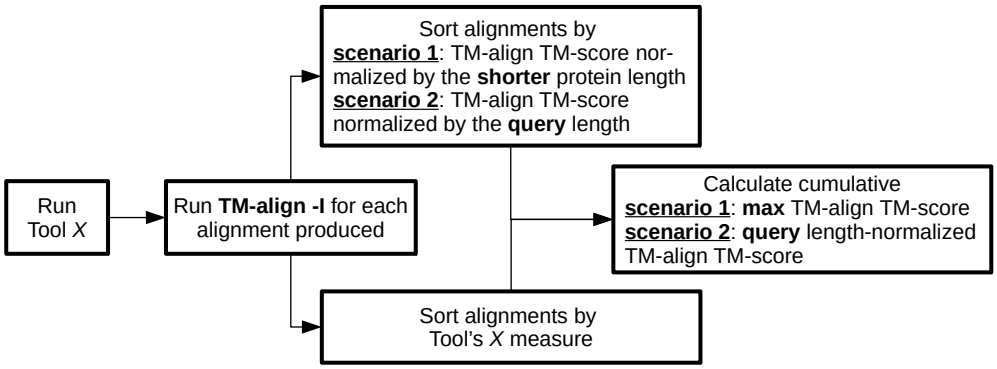


Fig. 7 Schematic for benchmarking structure alignment tools. The entire procedure can be described as follows: (i) Run a structure alignment tool; (ii) Use TM-align to calculate the TM-score for each produced alignment; (iii) Sort the alignments by the tool's measure (e.g., P-value, Z-score, etc.); (iv) Additionally, sort the alignments by the TM-score calculated by TM-align; (v) Finally, calculate the cumulative TM-score for the results, considering both the sorting by the tool's measure and the sorting by the TM-align-obtained TM-score. This provides a comprehensive measure of how accurately the tool produces alignments and their rate. It's worth noting that seemingly subtle differences in cumulative TM-score can be significant, especially considering the narrow gap between successive TM-scores; a mere 0.2 difference can distinguish between an accurate and inaccurate alignment.

Supplementary Information for: GTalign: High-performance protein structure alignment, superposition, and search

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 benchmarking results **2**
S1.1 Reference-free performance evaluation . . . 2
S1.1.1 Supplementary tables . . . 2
S1.1.2 Supplementary figures . . . 5
S1.2 GTalign runtimes on different machines . . . 10
S1.3 Alignment evaluation using RMSD . . . 11
S1.4 Benchmarking against SCOPe reference . . . 13
S1.5 Alignment of large complexes . . . 16

S2 Clustering protein structures with GTalign **17**

S3 Supplementary methods **18**
S3.1 Extensive superposition search . . . 18

Bibliography **23**

S1 Supplementary benchmarking results

S1.1 Reference-free performance evaluation

S1.1.1 Supplementary tables.

Table S1 Runtimes and cumulative TM-score, normalized by the length of the shorter protein, for each dataset. Additionally, runtimes are provided for GTalign evaluated on one, two, and three Tesla V100 GPUs. Numbers in parentheses indicate the maximum number of top hits evaluated.

	SCOPe40 2.08 (750K)		*PDB20* (112K)		*Swiss-Prot* (312K)	
	Runtime (*Seconds*)	Cumulative TM-score	Runtime (*Seconds*)	Cumulative TM-score	Runtime (*Seconds*)	Cumulative TM-score
3× *Tesla V100*						
GTalign --speed=0	4476.2	418682.3	872.6	65030.2	8453.8	184532.0
GTalign --speed=9	1301.4	414564.9	223.9	64414.0	1896.3	182556.0
GTalign --speed=9 --pre-score=0.4	679.7	411984.3	113.5	64156.6	899.2	181464.4
GTalign --speed=13	592.5	407922.2	104.6	63658.3	933.2	179738.8
GTalign --speed=13 --pre-score=0.4	381.7	405980.2	69.9	63482.3	618.0	178865.1
GTalign --speed=13 --pre-similarity=15	155.2	220411.3	46.3	37321.9	428.4	124803.3
2× *Tesla V100*						
GTalign --speed=0	6458.1	418682.3	1236.6	65030.2	11417.8	184532.0
GTalign --speed=9	1872.2	414564.9	316.2	64414.0	2660.0	182556.0
GTalign --speed=9 --pre-score=0.4	953.5	411984.3	157.7	64156.6	1253.0	181464.4
GTalign --speed=13	857.6	407922.2	147.0	63658.3	1325.7	179738.8
GTalign --speed=13 --pre-score=0.4	543.1	405980.2	96.5	63482.3	896.9	178865.1
GTalign --speed=13 --pre-similarity=15	214.2	220411.3	61.3	37321.9	611.5	124803.3
1× *Tesla V100*						
GTalign --speed=0	11902.0	418682.3	2261.2	65030.2	19071.6	184532.0
GTalign --speed=9	3472.7	414564.9	585.6	64414.0	4551.6	182556.0
GTalign --speed=9 --pre-score=0.4	1680.8	411984.3	280.4	64156.6	2075.7	181464.4
GTalign --speed=13	1601.2	407922.2	272.3	63658.3	2345.3	179738.8
GTalign --speed=13 --pre-score=0.4	997.8	405980.2	177.4	63482.3	1573.1	178865.1
GTalign --speed=13 --pre-similarity=15	386.2	220411.3	108.7	37321.9	1055.0	124803.3
TM-align	61380.4	416074.0	28504.8	64870.4	879965.5	183432.0
TM-align -fast	18501.2	409503.4	4826.5	64236.8	109319.0	181464.3
DeepAlign	527135.4	395118.6	187205.0	61771.6	–	–
DALI	195576.3	329225.8	49194.4	39914.8	–	–
FATCAT	369843.5	304491.7	102395.7	43501.8	–	–
Foldseek	48.5	180361.8	19.3	24513.0	195.6	46537.8
Foldseek --tmalign-fast 1	622.8	315751.3	203.3	43699.7	1138.8	75453.9
Foldseek --tmalign-fast 0	2048.1	325148.9	1156.0	45350.8	9649.5	77717.0

Table S2 Number of top hits at different levels of TM-score, normalized by the shorter protein length, for each dataset

	# top hits with TM-score≥				
	0.5	0.6	0.7	0.8	0.9
SCOPe40 2.08					
GTalign --speed=0	732024	138508	26891	5261	579
GTalign --speed=9	657116	127698	25114	5121	571
GTalign --speed=9 --pre-score=0.4	607322	123361	24871	5107	571
GTalign --speed=13	553470	113465	23378	4966	567
GTalign --speed=13 --pre-score=0.4	525719	110522	23186	4956	567
GTalign --speed=13 --pre-similarity=15	19757	7357	3756	2085	434
TM-align	683996	131059	25466	5140	573
TM-align -fast	580689	115546	22483	4655	558
DeepAlign	413179	92809	20251	4443	477
DALI	203818	43864	9300	2850	469
FATCAT	310773	66248	13951	3474	477
Foldseek	13371	6136	3292	1410	195
Foldseek --tmalign-fast 1	135399	38086	9454	2984	486
Foldseek --tmalign-fast 0	150894	41461	9862	3001	486
PDB20					
GTalign --speed=0	111603	32860	10233	2558	308
GTalign --speed=9	101965	31051	9869	2487	293
GTalign --speed=9 --pre-score=0.4	97440	30501	9794	2483	293
GTalign --speed=13	90983	29288	9537	2418	289
GTalign --speed=13 --pre-score=0.4	88693	28916	9464	2415	289
GTalign --speed=13 --pre-similarity=15	3692	1480	651	177	30
TM-align	106869	32428	10424	2694	335
TM-align -fast	97229	30894	10010	2543	313
DeepAlign	74107	24883	8047	1921	163
DALI	19856	5908	1647	228	30
FATCAT	58644	18813	5485	1056	81
Foldseek	3018	1227	352	78	14
Foldseek --tmalign-fast 1	17066	6086	1864	270	35
Foldseek --tmalign-fast 0	18987	6539	1901	274	35
Swiss-Prot					
GTalign --speed=0	306488	116253	34008	6864	1581
GTalign --speed=9	281858	108747	32867	6850	1580
GTalign --speed=9 --pre-score=0.4	264427	106267	32784	6850	1580
GTalign --speed=13	250953	99489	31463	6847	1580
GTalign --speed=13 --pre-score=0.4	240320	97986	31402	6847	1580
GTalign --speed=13 --pre-similarity=15	26288	13245	5593	2403	1356
TM-align	292670	111400	33415	6895	1582
TM-align -fast	268250	105747	31686	6535	1562
Foldseek	17400	7201	2912	1622	1081
Foldseek --tmalign-fast 1	42007	25957	11303	2783	1461
Foldseek --tmalign-fast 0	43913	27242	11730	2785	1461

Table S3 Cumulative TM-score, normalized by the query length, for each dataset. Numbers in parentheses indicate the maximum number of top hits evaluated.

	SCOPe40 2.08 (520K) Cumulative TM-score	*PDB20* (30K) Cumulative TM-score	*Swiss-Prot* (100K) Cumulative TM-score
GTalign --speed=0	289050.2	16426.2	58293.0
GTalign --speed=9	286252.7	16232.2	57587.2
GTalign --speed=9 --pre-score=0.4	284808.9	16057.3	57143.3
GTalign --speed=13	281944.0	15865.4	56483.2
GTalign --speed=13 --pre-score=0.4	280877.5	15739.2	55886.1
GTalign --speed=13 --pre-similarity=15	171378.9	11707.7	44383.7
TM-align	287250.9	16293.6	57764.3
TM-align -fast	282231.7	16010.5	56964.2
DeepAlign	174382.5	10526.1	–
DALI	205760.3	12216.4	–
FATCAT	198691.8	10438.4	–
Foldseek	111670.0	7698.6	29400.7
Foldseek --tmalign-fast 1	209747.8	13246.8	43590.1
Foldseek --tmalign-fast 0	213999.1	13446.6	44166.4

Table S4 Number of top hits at different levels of TM-score, normalized by the query length, for each dataset

	# top hits with TM-score≥				
	0.5	0.6	0.7	0.8	0.9
SCOPe40 2.08					
GTalign --speed=0	492887	90869	18336	3715	359
GTalign --speed=9	442998	83972	17156	3612	354
GTalign --speed=9 --pre-score=0.4	415248	81732	17003	3601	354
GTalign --speed=13	377275	75003	15986	3505	352
GTalign --speed=13 --pre-score=0.4	361914	73491	15864	3497	352
GTalign --speed=13 --pre-similarity=15	13879	5657	3147	1662	273
TM-align	460847	86062	17332	3624	354
TM-align -fast	386777	74277	14860	3228	340
DeepAlign	284743	63018	14048	3192	312
DALI	122228	26851	6581	2155	289
FATCAT	208110	43679	9961	2582	296
Foldseek	10375	5066	2787	1062	127
Foldseek --tmalign-fast 1	85776	24238	6943	2262	297
Foldseek --tmalign-fast 0	94430	25964	7204	2271	297
PDB20					
GTalign --speed=0	22913	5024	1061	166	23
GTalign --speed=9	20369	4665	1017	166	23
GTalign --speed=9 --pre-score=0.4	18339	4415	1006	166	23
GTalign --speed=13	16426	4137	973	166	23
GTalign --speed=13 --pre-score=0.4	15339	3983	963	166	23
GTalign --speed=13 --pre-similarity=15	2261	974	412	125	22
TM-align	21209	4739	1024	167	23
TM-align -fast	17861	4341	985	166	23
DeepAlign	13086	3261	701	114	15
DALI	12341	3562	840	149	20
FATCAT	11906	3498	842	143	20
Foldseek	1894	711	226	55	7
Foldseek --tmalign-fast 1	10356	3534	949	164	23
Foldseek --tmalign-fast 0	11423	3769	964	165	23
Swiss-Prot					
GTalign --speed=0	74533	38977	14016	2482	1277
GTalign --speed=9	70619	36366	13526	2483	1276
GTalign --speed=9 --pre-score=0.4	68870	35791	13452	2483	1276
GTalign --speed=13	64699	33200	12478	2482	1276
GTalign --speed=13 --pre-score=0.4	63355	32791	12423	2482	1276
GTalign --speed=13 --pre-similarity=15	18785	10385	4906	2180	1180
TM-align	72030	36596	13679	2483	1276
TM-align -fast	67516	34336	12882	2482	1276
Foldseek	14617	6737	2665	1480	891
Foldseek --tmalign-fast 1	30238	19396	9954	2482	1277
Foldseek --tmalign-fast 0	31357	20229	10343	2483	1276

S1.1.2 Supplementary figures.

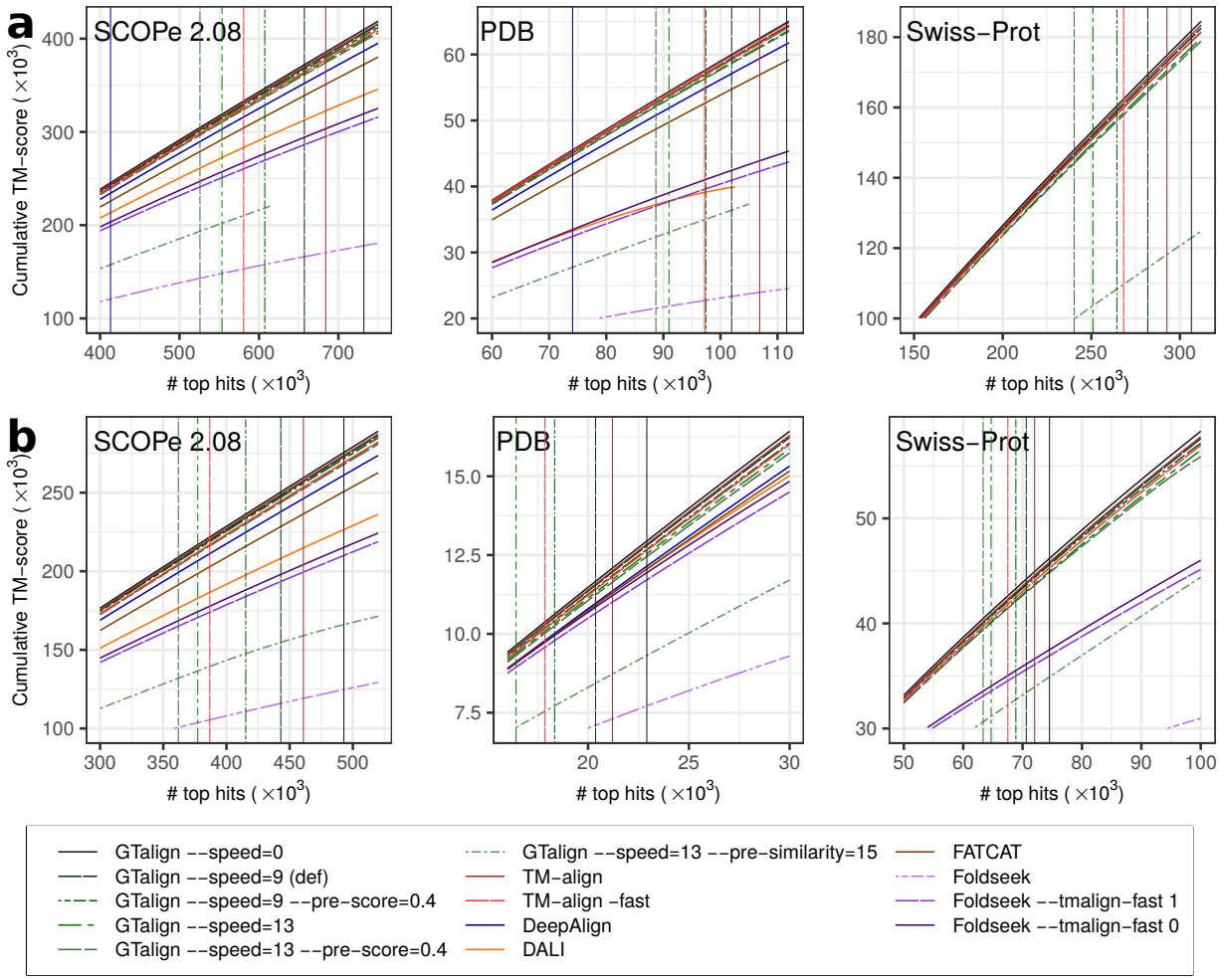


Fig. S1 Benchmarking results on the SCOPe40 2.08, PDB20, and Swiss-Prot datasets. The figure plots the cumulative TM-score against the number of top alignments sorted by TM-align-obtained TM-score and represents zoomed-in sections of Fig. 1 and 2 from the main text for detailed analysis. Vertical lines indicate the number of alignments with a TM-score ≥ 0.5. Panels (a) and (b) show TM-scores normalized by the length of the shorter protein and by the query length, respectively.

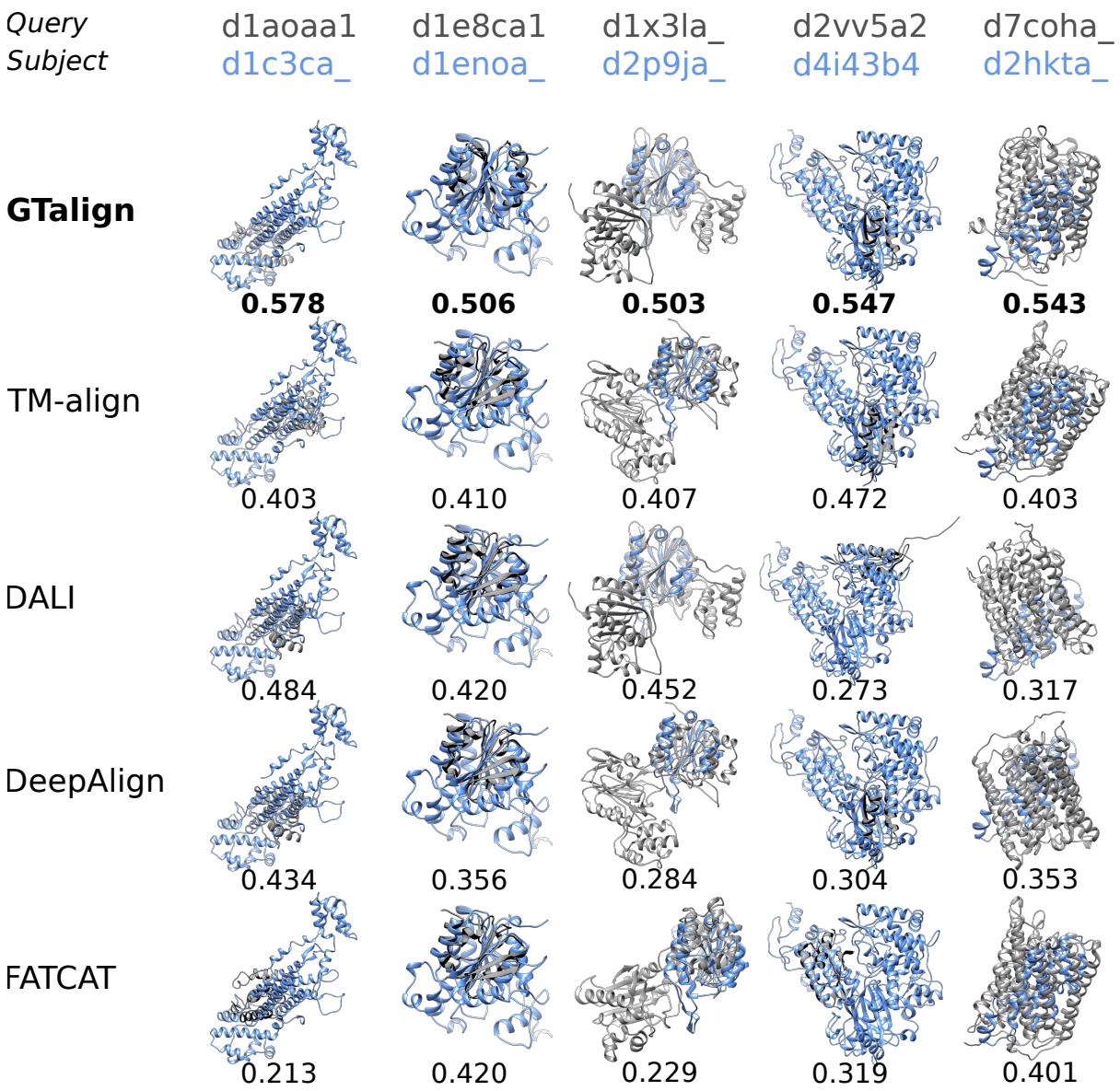


Fig. S2 Structural alignment examples from benchmarking on the SCOPe40 2.08 dataset. Query structures are in grey, and subject structures are colored. The subject structures share the same orientation across the tools for visual inspection. Numbers represent TM-scores normalized by the length of the shorter protein. It's noteworthy that some superpositions appear similar despite having different TM-scores. In such instances, TM-align identifies a favorable superposition even when provided with an inferior alignment. Foldseek (all parameterized versions) did not produce alignments for the given protein pairs.

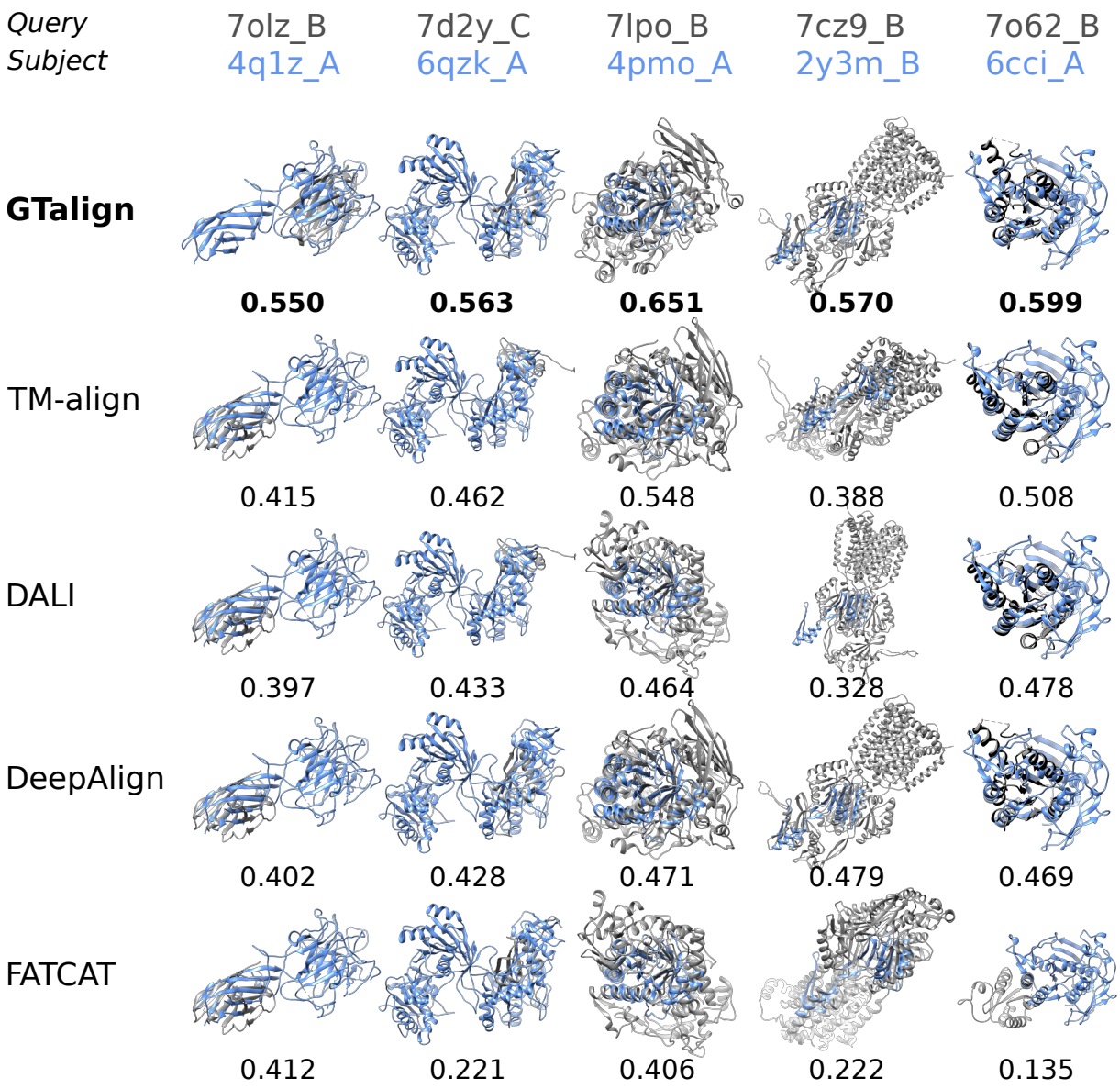


Fig. S3 Illustrative examples from benchmarking on the PDB20 dataset. Query structures are in grey, and subject structures are colored. The subject structures share the same orientation across the tools. Numbers represent TM-scores normalized by the length of the shorter protein. Foldseek (all parameterized versions) did not produce alignments for the given protein pairs.

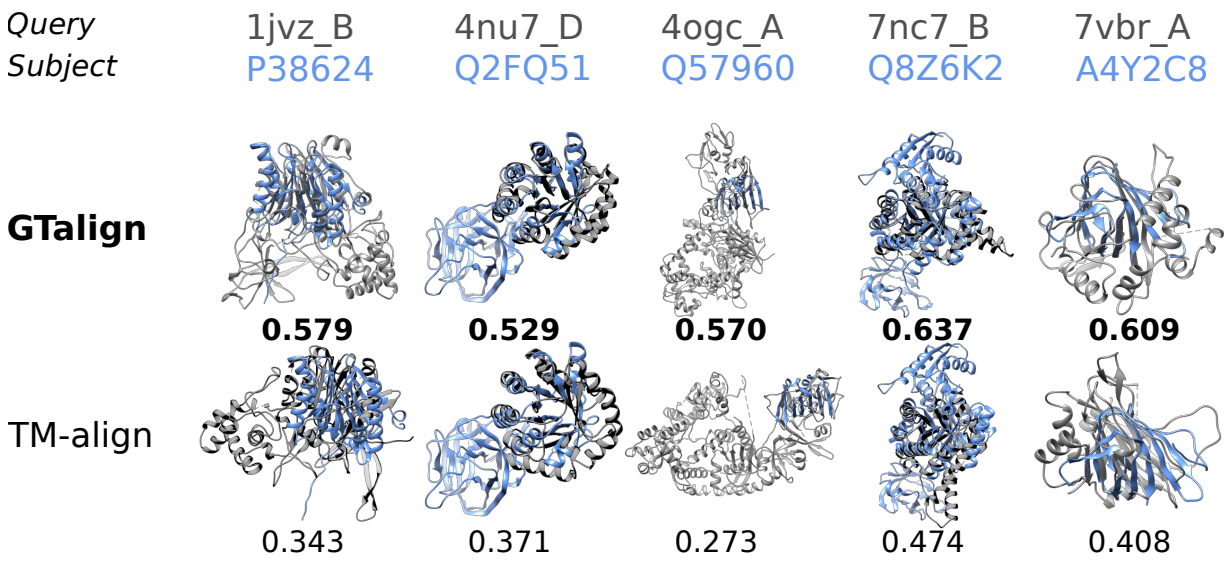


Fig. S4 Illustrative examples from benchmarking on the Swiss-Prot dataset. Query structures are in grey. Subject structures are colored. The subject structures share the same orientation across the tools. Numbers represent TM-scores normalized by the length of the shorter protein. Foldseek (all parameterized versions) did not produce alignments for the given protein pairs.

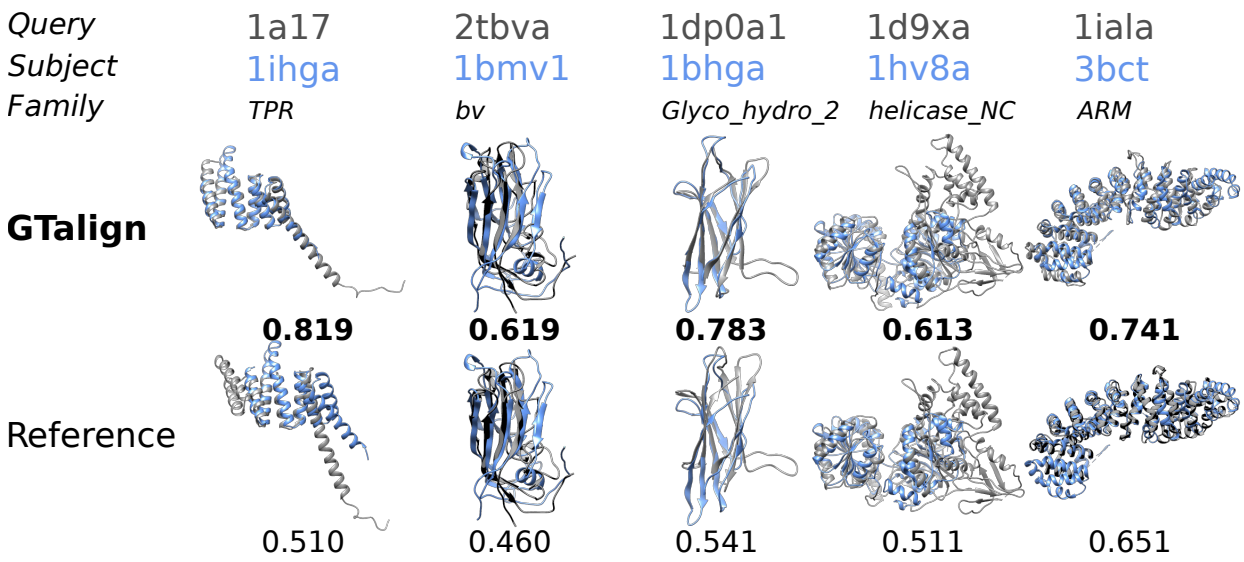


Fig. S5 Illustrative examples of GTalign and reference (original) alignments from benchmarking on the HOMSTRAD dataset. Query structures are depicted in grey, while subject structures are colored. The same subject structures maintain the same orientation for comparison. Numbers represent TM-scores normalized by the length of the shorter protein.

S1.2 GTalign runtimes on different machines

In assessing the performance of GTalign parameterized variants across diverse hardware configurations, we conducted runtime evaluations on three distinct machines: a server-grade system equipped with three Tesla V100 GPU accelerators, a desktop-grade machine featuring a GeForce RTX 4090 GPU, and a laptop with a GeForce RTX 4090 Laptop GPU. GTalign was configured to use 16GB of GPU RAM (option `--dev-mem`) for all tests.

Notably, the desktop-grade machine, housing a more recent and affordable GeForce RTX 4090 GPU, outpaced the server with three Tesla V100 GPU cards when running GTalign. The detailed runtimes for each GTalign parameterized variant on these diverse machines are presented in Table S5.

Table S5 Runtimes (in seconds) of parameterized GTalign variants on different machines for each dataset

	3×Tesla V100 (*Server*)	1×GeForce RTX 4090 (*Desktop*)	1×GeForce RTX 4090 Laptop (*Laptop*)
SCOPe40 2.08			
GTalign --speed=0	4476.2	3762.5	8522.6
GTalign --speed=9	1301.4	1082.9	2332.5
GTalign --speed=13 --pre-similarity=15	155.2	159.6	210.7
PDB20			
GTalign --speed=0	872.6	781.7	1702.3
GTalign --speed=9	223.9	193.5	383.4
GTalign --speed=13 --pre-similarity=15	46.3	40.8	55.8
Swiss-Prot			
GTalign --speed=0	8453.8	6544.2	15612.2
GTalign --speed=9	1896.3	1400.7	3080.8
GTalign --speed=13 --pre-similarity=15	428.4	371.0	607.5

S1.3 Alignment evaluation using RMSD

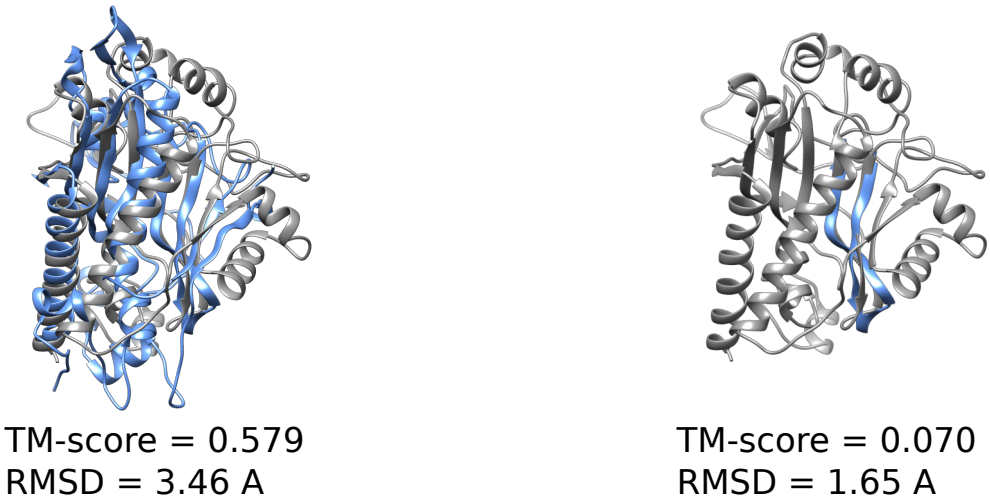


Fig. S6 Illustrative example of alignment accuracy evaluation for the SCOPe 2.08 domains d12asa_ (d.104.1.1; shown in grey) and d1seta2 (d.104.1.1; shown in color) using TM-score normalized by the length of the shorter protein and RMSD. TM-score is sensitive to alignment coverage due to normalization by protein length. RMSD, on the other hand, is normalized by the number of aligned residue pairs and measures their spatial proximity after superposition, similar to TM-score. Therefore, a short accurately aligned fragment (right panel; unaligned d1seta2 parts not shown) implies a lower RMSD compared to the full alignment (left panel), which includes more divergent structural segments. Alignments, TM-scores, and RMSDs were calculated using TM-align.

Table S6 Mean RMSDs (in angstroms), GDT_TS scores normalized by the number of aligned residues, and numbers of aligned residues (Length) for each dataset. Numbers in parentheses following the dataset names indicate the maximum number of top hits evaluated. s.d., standard deviation.

	RMSD (s.d.)	GDT_TS (s.d.)	Length (s.d.)
SCOPe40 2.08 (750K)			
GTalign --speed=0	2.91 (0.87)	0.7137 (0.1288)	67.4 (43.1)
GTalign --speed=9	2.92 (0.88)	0.7128 (0.1312)	67.0 (43.1)
GTalign --speed=9 --pre-score=0.4	2.90 (0.88)	0.7182 (0.1310)	65.0 (42.9)
GTalign --speed=13	2.92 (0.91)	0.7164 (0.1345)	65.0 (43.2)
GTalign --speed=13 --pre-score=0.4	2.90 (0.90)	0.7206 (0.1341)	63.6 (43.0)
GTalign --speed=13 --pre-similarity=15	4.87 (0.95)	0.4872 (0.1067)	94.0 (42.8)
TM-align	2.92 (0.88)	0.7144 (0.1315)	66.7 (43.3)
TM-align -fast	2.94 (0.90)	0.7121 (0.1333)	66.4 (43.5)
DeepAlign	2.48 (0.97)	0.7663 (0.1375)	56.0 (39.2)
DALI	9.49 (5.12)	0.5075 (0.1107)	109.6 (44.3)
FATCAT	4.55 (3.11)	0.6368 (0.2089)	69.7 (50.6)
Foldseek	10.26 (4.20)	0.3810 (0.1433)	85.4 (44.4)
Foldseek --tmalign-fast 1	4.37 (0.91)	0.5277 (0.0988)	96.7 (42.6)
Foldseek --tmalign-fast 0	4.31 (0.90)	0.5342 (0.0978)	98.0 (42.7)
PDB20 (112K)			
GTalign --speed=0	2.47 (1.05)	0.7781 (0.1462)	57.1 (49.0)
GTalign --speed=9	2.48 (1.06)	0.7779 (0.1479)	56.8 (48.9)
GTalign --speed=9 --pre-score=0.4	2.45 (1.05)	0.7832 (0.1453)	54.9 (47.8)
GTalign --speed=13	2.44 (1.07)	0.7852 (0.1479)	54.7 (48.6)
GTalign --speed=13 --pre-score=0.4	2.42 (1.06)	0.7887 (0.1459)	53.3 (47.5)
GTalign --speed=13 --pre-similarity=15	5.30 (1.09)	0.4436 (0.1149)	120.0 (57.5)
TM-align	2.47 (1.07)	0.7819 (0.1476)	56.3 (49.4)
TM-align -fast	2.44 (1.06)	0.7858 (0.1477)	54.9 (49.3)
DeepAlign	2.03 (1.04)	0.8285 (0.1416)	47.6 (49.0)
DALI	12.55 (7.84)	0.4587 (0.1384)	136.4 (66.3)
FATCAT	5.18 (4.32)	0.6260 (0.2565)	85.6 (78.4)
Foldseek	11.76 (5.30)	0.3501 (0.1499)	103.5 (56.2)
Foldseek --tmalign-fast 1	4.97 (1.22)	0.4714 (0.1208)	122.1 (54.3)
Foldseek --tmalign-fast 0	4.92 (1.22)	0.4777 (0.1214)	125.4 (55.4)
Swiss-Prot (312K)			
GTalign --speed=0	2.96 (1.16)	0.7113 (0.1673)	103.6 (88.8)
GTalign --speed=9	2.97 (1.18)	0.7125 (0.1706)	101.9 (88.4)
GTalign --speed=9 --pre-score=0.4	2.93 (1.18)	0.7190 (0.1706)	99.1 (88.5)
GTalign --speed=13	2.94 (1.21)	0.7186 (0.1740)	98.0 (87.5)
GTalign --speed=13 --pre-score=0.4	2.91 (1.21)	0.7238 (0.1734)	95.7 (87.3)
GTalign --speed=13 --pre-similarity=15	5.34 (1.02)	0.4365 (0.1092)	154.7 (64.4)
TM-align	2.98 (1.19)	0.7126 (0.1711)	102.7 (89.2)
TM-align -fast	2.96 (1.19)	0.7155 (0.1722)	100.1 (88.4)
DeepAlign	–	–	–
DALI	–	–	–
FATCAT	–	–	–
Foldseek	13.78 (6.49)	0.3176 (0.1563)	157.5 (82.8)
Foldseek --tmalign-fast 1	5.34 (1.29)	0.4390 (0.1197)	165.6 (68.7)
Foldseek --tmalign-fast 0	5.31 (1.28)	0.4421 (0.1193)	169.9 (68.7)

S1.4 Benchmarking against SCOPe reference

Table S7 Area under the weighted precision-recall curve (AUPRC) at the family, superfamily, and fold level. The results are presented in two parts. In one part, false positives for calculating precision and recall correspond to pairs of structures from different SCOPe 2.08 folds (No exceptions). In the second part, false positives are pairs from different SCOPe 2.08 folds; however, these pairs are ignored if the structures belong to Rossman-like (c.2–c.5, c.27, c.28, c.30, and c.31) or beta-propeller (b.66–b.70) folds [1] (Cross-fold relationships ignored).

	Cross-fold relationships ignored			*No exceptions*		
	Family	Superfamily	Fold	Family	Superfamily	Fold
GTalign --speed=0	0.807	0.591	0.0582	0.802	0.584	0.0536
GTalign --speed=9	0.808	0.592	0.0574	0.803	0.585	0.0530
GTalign --speed=9 --pre-score=0.4	0.806	0.587	0.0554	0.800	0.581	0.0509
GTalign --speed=13	0.811	0.596	0.0567	0.806	0.589	0.0521
GTalign --speed=13 --pre-score=0.4	0.808	0.590	0.0548	0.803	0.583	0.0503
GTalign --speed=13 --pre-similarity=15	0.527	0.364	0.0090	0.524	0.360	0.0069
TM-align	0.804	0.593	0.0514	0.800	0.588	0.0486
TM-align -fast	0.805	0.595	0.0505	0.801	0.590	0.0477
DeepAlign	0.669	0.540	0.0487	0.611	0.471	0.0195
DALI	0.708	0.567	0.0586	0.672	0.523	0.0339
FATCAT	0.603	0.366	0.0123	0.603	0.365	0.0123
Foldseek	0.710	0.548	0.0201	0.682	0.514	0.0109
Foldseek --tmalign-fast 1	0.733	0.523	0.0191	0.696	0.480	0.0134
Foldseek --tmalign-fast 0	0.726	0.515	0.0183	0.688	0.471	0.0129

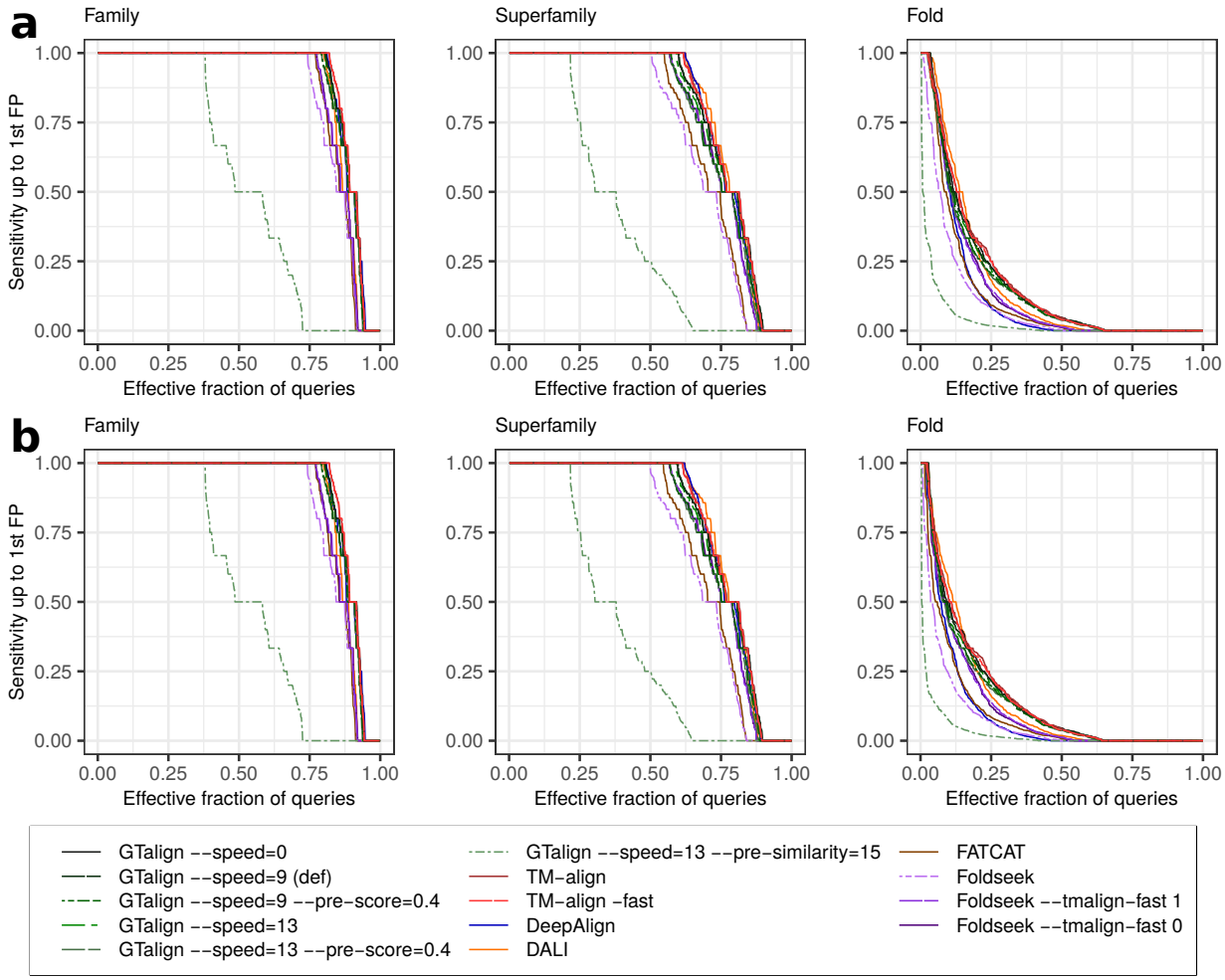


Fig. S7 Sensitivity up to the first false positive against the effective fraction of queries at the family, superfamily, and fold level. Sensitivity represents the fraction of true positives identified before encountering the first false positive. The effective fractions of queries refer to the proportion of queries originating from families with at least two members (953 queries), superfamilies with at least two families (858 queries), and folds with at least two superfamilies (941 queries), respectively. a, False positives are pairs from different SCOPe 2.08 folds, excluding those pairs belonging to Rossman-like (c.2–c.5, c.27, c.28, c.30, and c.31) or beta-propeller (b.66–b.70) folds. b, False positives are pairs from different SCOPe 2.08 folds without exceptions.

Table S8 Average sensitivity up to the first false positive at the family, superfamily, and fold levels. Sensitivity represents the fraction of true positives identified before the first false positive is encountered. In one analysis, false positives are defined as pairs from different SCOPe 2.08 folds, ignoring those pairs from Rossman-like (c.2–c.5, c.27, c.28, c.30, and c.31) or beta-propeller (b.66–b.70) folds (Cross-fold relationships ignored). In the second analysis, false positives are defined as pairs from different SCOPe 2.08 folds (No exceptions). *P*-values in parentheses are obtained from the nonparametric unpaired Mann-Whitney test on the sensitivity values (Fig. S7) between GTalign --speed=0 and all other parameterized tools at the family, superfamily, and fold levels.

	Family (*p*-value)	Superfamily (*p*-value)	Fold (*p*-value)
Cross-fold relationships ignored			
GTalign --speed=0	0.8933 (–)	0.7723 (–)	0.1750 (–)
GTalign --speed=9	0.8928 (0.920)	0.7723 (0.951)	0.1718 (0.853)
GTalign --speed=9 --pre-score=0.4	0.8859 (0.409)	0.7541 (0.257)	0.1622 (0.380)
GTalign --speed=13	0.8912 (0.823)	0.7664 (0.756)	0.1676 (0.569)
GTalign --speed=13 --pre-score=0.4	0.8844 (0.392)	0.7497 (0.182)	0.1595 (0.242)
GTalign --speed=13 --pre-similarity=15	0.5380 (1.1×10^{-87})	0.3752 (8.4×10^{-79})	0.0327 (1.1×10^{-52})
TM-align	0.8958 (0.558)	0.7782 (0.525)	0.1802 (0.931)
TM-align -fast	0.8966 (0.547)	0.7767 (0.577)	0.1759 (0.854)
DeepAlign	0.8908 (0.922)	0.7737 (0.498)	0.1258 (1.6×10^{-17})
DALI	0.8722 (0.556)	0.7792 (0.105)	0.1682 (0.031)
FATCAT	0.8556 (0.022)	0.7098 (0.003)	0.1219 (6.9×10^{-11})
Foldseek	0.8457 (0.003)	0.6916 (2.3×10^{-5})	0.0973 (5.0×10^{-15})
Foldseek --tmalign-fast 1	0.8616 (0.175)	0.7481 (0.263)	0.1431 (4.7×10^{-5})
Foldseek --tmalign-fast 0	0.8622 (0.193)	0.7496 (0.258)	0.1428 (2.9×10^{-5})
No exceptions			
GTalign --speed=0	0.8923 (–)	0.7705 (–)	0.1580 (–)
GTalign --speed=9	0.8918 (0.920)	0.7706 (0.952)	0.1547 (0.814)
GTalign --speed=9 --pre-score=0.4	0.8849 (0.410)	0.7526 (0.261)	0.1453 (0.347)
GTalign --speed=13	0.8902 (0.823)	0.7651 (0.810)	0.1509 (0.525)
GTalign --speed=13 --pre-score=0.4	0.8834 (0.393)	0.7487 (0.208)	0.1429 (0.212)
GTalign --speed=13 --pre-similarity=15	0.5380 (4.1×10^{-87})	0.3747 (4.8×10^{-78})	0.0261 (4.9×10^{-54})
TM-align	0.8954 (0.520)	0.7765 (0.522)	0.1630 (0.938)
TM-align -fast	0.8962 (0.510)	0.7752 (0.553)	0.1587 (0.844)
DeepAlign	0.8889 (0.973)	0.7717 (0.487)	0.1001 (5.2×10^{-20})
DALI	0.8717 (0.551)	0.7777 (0.099)	0.1486 (0.029)
FATCAT	0.8554 (0.026)	0.7082 (0.003)	0.0980 (2.2×10^{-12})
Foldseek	0.8451 (0.003)	0.6897 (1.8×10^{-5})	0.0749 (2.0×10^{-18})
Foldseek --tmalign-fast 1	0.8608 (0.163)	0.7470 (0.298)	0.1253 (3.2×10^{-5})
Foldseek --tmalign-fast 0	0.8605 (0.161)	0.7485 (0.293)	0.1242 (1.8×10^{-5})

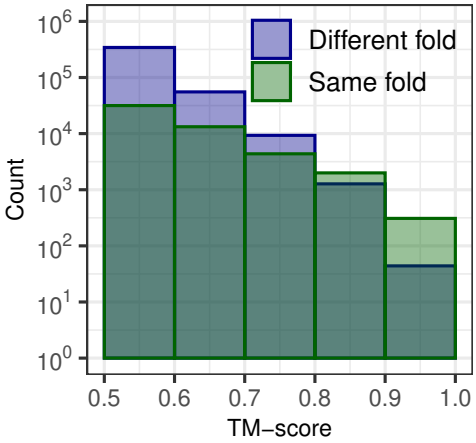
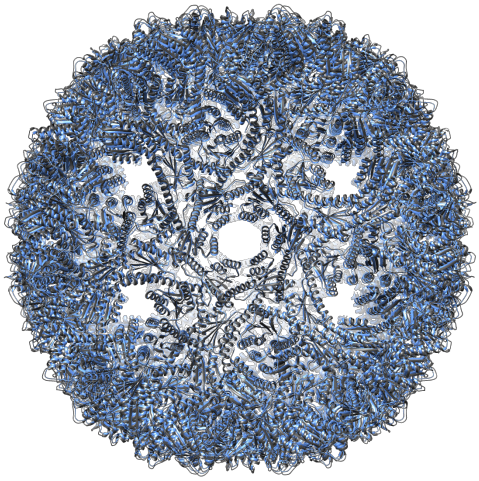


Fig. S8 Distributions of TM-scores (> 0.5) for domain pairs within the same and different SCOPe 2.08 folds, presented in bins of width 0.1. TM-scores, normalized by the query length, were calculated by aligning query and subject structures from the SCOPe40 2.08 dataset using TM-align.

S1.5 Alignment of large complexes



	TM-score	Runtime (sec)
GTalign --speed=13	0.2448	1.6
GTalign --speed=9	0.2793	8.7
GTalign --speed=6	0.3154	23.5
TM-align	0.3154	8,084,903.5

Fig. S9 Superposition of two virus nucleocapsid variants, 7a4i and 7a4j (37,860 residues each), by GTalign and TM-align. The GTalign runtimes exclude GPU initialization time (3.5 seconds). The GTalign command line included the following options for aligning the complexes: `--dev-queries-per-chunk=1 --dev-queries-total-length-per-chunk=40000 --dev-max-length=40000 --pre-score=0 --ter=0 -s 0`.

S2 Clustering protein structures with GTalign

To showcase GTalign's clustering capabilities, we performed a clustering analysis on the entire PDB [2] database (accessed 08/18/2023) of protein structures. Using GTalign with parameters `--dev-queries-total-length-per-chunk=1500 --dev-min-length=3 --dev-max-length=1500 --speed=13 --add-search-by-ss --cls-coverage=0.7 --cls-threshold=0.5 -c cachedir` and the complete linkage clustering algorithm [3] (single linkage clustering is available too), we clustered protein structures at a TM-score threshold of 0.5 with a length coverage threshold of 70%. The clustering focused solely on the first protein chains, totaling 199,839 chains, and was accomplished in five days, leveraging three Tesla V100 GPUs. The distribution of cluster sizes resulting from this analysis is visualized in Fig. S10a.

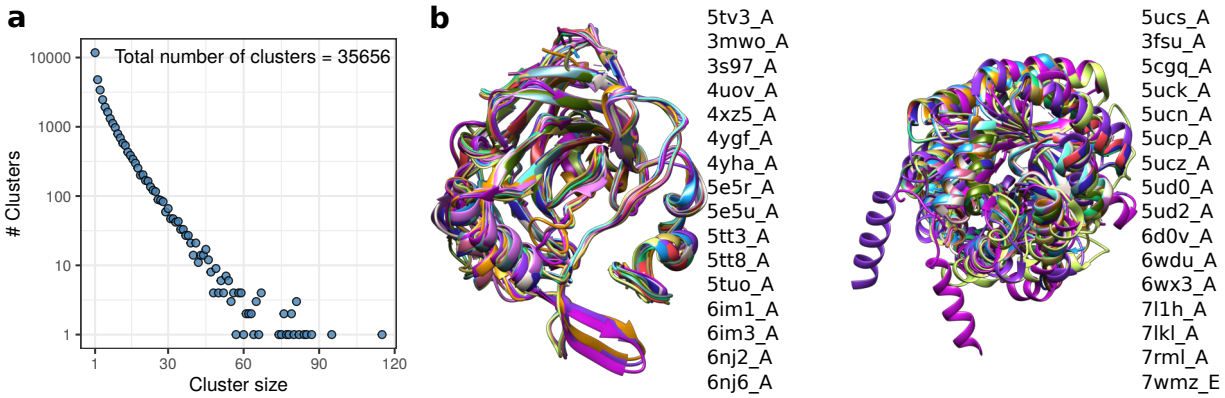


Fig. S10 Results of the GTalign clustering analysis on the entire PDB database. a, Distribution of cluster sizes. b, Superimpositions of proteins within selected clusters, representing the Carbonic anhydrase fold (left panel) and the TIM beta/alpha-barrel fold (right panel). Despite variations in sequence identity (21–35% with the first member of the first cluster and at most 7% for the second cluster, excluding trivial matches), GTalign reveals significant structural similarity within clusters. Superpositions were found by aligning the first member of each cluster with the rest using GTalign and the option `--referenced`, producing transformation matrices for subject structures.

The significance of clustering in the structural space becomes evident when compared to traditional sequence-based clustering approaches. As demonstrated in Fig. S10b, the superposition of cluster member proteins showcases known structural relationships also detected by GTalign. In one example, a cluster encompassing proteins with the TIM beta/alpha-barrel fold reveals diversity in sequence yet high structural similarity. More than half of the proteins within this cluster share, at most, 7% sequence identity with the cluster's first member. Despite this low sequence identity, their structural alignment yields significant TM-scores ranging from 0.553 to 0.708. In contrast, conventional sequence-based clustering would consider such protein pairs as unrelated due to the lack of sequence similarity. This example underscores the effectiveness and utility of fast clustering in the structural space, a feature offered by GTalign.

S3 Supplementary methods

S3.1 Extensive superposition search

This section presents algorithms for an extensive superposition search using spatial indices. The following notation is adopted: vectors are denoted by bold italic lowercase letters, and multi-dimensional matrices are represented by italic uppercase letters. The operator $[\cdot]_{\text{mut}}$ in Algorithms 2 and 8 signifies a mutually exclusive operation that ensures no two threads access the same data simultaneously.

The top-level algorithm is given in Algorithm 1. It has the following parameters: n_{Q}, n_{S}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, l_{f}, C, $\tilde{C}$, T, $\tilde{T}$, n_{brn}, and c_{thr}. n_{Q} and n_{S} represent the numbers of query and subject structures in their respective batches, while $\boldsymbol{l}$ and $\tilde{\boldsymbol{l}}$ denote the lengths of the query and subject structures, respectively. l_{f} represents the length of a protein sequence fragment used for initial protein superpositions within the scope of the algorithm. C and $\tilde{C}$ are the coordinates of the query and subject structures, with $C_{q,p_q} \in \mathbb{R}^{3\times 1}$ indicating the coordinates of residue p_q of query structure q and $\tilde{C}_{s,\tilde{p}_s} \in \mathbb{R}^{3\times 1}$ having the corresponding meaning for subject structure s. T and $\tilde{T}$ represent the secondary structure assignments for the query and subject structures. n_{brn} is a configurable number of top-performing superposition branches to explore in detail, and c_{thr} is a local structure similarity threshold for triggering superposition analysis initiated by a specific configuration of continuous fragment matching.

The algorithm initially computes the dynamic programming (DP) matrix of local scores using the following recursive equation:

$$L_{qsij} = \max\{L_{q,s,i-1,j-1} + 2 \cdot \mathbf{1}_{T_{qi}=\tilde{T}_{sj}} - 1, L_{q,s,i-1,j} - 0.8, L_{q,s,i,j-1} - 0.8, 0\}, \tag{S1}$$

with the values L_{qsij} capped at a maximum value of 255 to reduce memory footprint. Subsequently, initial superpositions (lines 6–7) are determined based on protein regions exhibiting substantial local similarity (line 5). Following the calculation of initial superpositions, several rounds of constant-time alignment, coupled with superposition inference, are executed (lines 8–13). These steps enable the rapid computation of numerous (n_{ext}) alignments based on different spatial configurations. However, the resulting alignments are sequence order-independent, some of which correspond to only good spatial agreement rather than similar folds, and necessitate post-processing.

Algorithm 1 Find most favorable superpositions by deep search

1: **procedure** DeepSuperpositionSearch(n_{Q}, n_{S}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, l_{f}, C, $\tilde{C}$, T, $\tilde{T}$, n_{brn}, c_{thr})
2: Calculate L using Eq. (S1) in parallel $\forall\, 0 \le q < n_{\text{Q}}$ and $0 \le s < n_{\text{S}}$
3: Calculate n_{ext} based on $\max_q l_q$ and $\max_s \tilde{l}_s$ and depth specification
4: Set n_{tfm} to 32, 64, or 96 based on depth specification
5: CalculateLocalSimilarity(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, l_{f}, L, c_{thr})
6: CalculateXCovariances(n_{Q}, n_{S}, n_{ext}, l_{f}, C, $\tilde{C}$)
7: CalculateTransformations(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, 1)
8: AlignInConstantTime(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, C, $\tilde{C}$, T, $\tilde{T}$, 1)
9: CalculateXCovariancesA(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$)
10: CalculateTransformations(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, 1)
11: AlignInConstantTime(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, C, $\tilde{C}$, T, $\tilde{T}$, 0)
12: CalculateXCovariancesA(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$)
13: CalculateTransformations(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, 0)
14: CalculateApproxScores(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$)
15: GetTopNTransformations(n_{Q}, n_{S}, n_{ext}, n_{tfm}, 0)
16: Calculate TM-scores [4] in parallel $\forall\, 0 \le q < n_{\text{Q}}$, $0 \le s < n_{\text{S}}$, and
TF matrices $0 \le i < n_{\text{tfm}}$ using the Comer2 DP algorithm [5]
17: GetTopNTransformations(n_{Q}, n_{S}, n_{tfm}, n_{brn}, 1)
18: OptimizeSelectedAlignments(n_{Q}, n_{S}, n_{brn})
19: RefineBestAlignments(n_{Q}, n_{S}, -0.6)
20: RefineBestAlignments(n_{Q}, n_{S}, 0)
21: **end procedure**

The post-processing phase (line 14) of order-independent alignments involves calculating approximate order-dependent TM-scores in sublinear time. A small number (n_{tfm}) of transformation matrices with the highest approximate scores are selected (line 15) to calculate TM-scores using the COMER2 DP algorithm

[5]. Requiring no backtracking information in line 16 reduces memory demands and increases the parallelization degree. A further refinement is made by selecting (line 17) an even smaller number ($n_{\text{brn}} < n_{\text{tfm}}$) of transformation matrices corresponding to the highest TM-scores to optimize structural alignments derived from them (line 18). Finally, the alignments—each representing one query-subject pair—with the highest TM-score are refined (lines 19–20).

Algorithm 2 Calculate Local Similarity

1: **procedure** CALCULATELOCALSIMILARITY(n_{Q}, n_{S}, n_{ext}, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, l_{f}, L, c_{thr})
2: **for all** $(q, s, f_{\text{ext}}) \in [0, n_{\text{Q}}) \times [0, n_{\text{S}}) \times [0, n_{\text{ext}})$ **do in parallel**
3: Calculate query and subject protein positions p_q and $\tilde{p}_s$ from index f_{ext}
4: **if** $p_q + l_{\text{f}} > l_q$ **or** $\tilde{p}_s + l_{\text{f}} > \tilde{l}_s$ **then**
5: Set *skip* flag for configuration $\{q, s, f_{\text{ext}}\}$
6: **return**
7: **end if**
8: $S \leftarrow 0_{32,32}$
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: $[\ S_{i \bmod 32, j \bmod 32} \leftarrow \max\{S_{i \bmod 32, j \bmod 32}, L_{q,s,p_q+i,\tilde{p}_s+j}\}\ ]_{\text{mut}}$
12: **end for**
13: **end for**
14: $m \leftarrow \max_{i,j} S_{ij}$ ▷ two-dimensional parallel reduction
15: **if** $m < l_{\text{f}} \times c_{\text{thr}}$ **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 cross-covariance matrices

1: **procedure** CALCULATEXCOVARIANCES(n_{Q}, n_{S}, n_{ext}, l_{f}, C, $\tilde{C}$)
2: **for all** $(q, s, f_{\text{ext}}) \in [0, n_{\text{Q}}) \times [0, n_{\text{S}}) \times [0, n_{\text{ext}})$ **do in parallel**
3: **continue if** *skip* flag is set for configuration $\{q, s, f_{\text{ext}}\}$
4: Calculate query and subject protein positions p_q and $\tilde{p}_s$ from index f_{ext}
5: $(K_{qsf_{\text{ext}}}, \boldsymbol{c}_{qf_{\text{ext}}}, \tilde{\boldsymbol{c}}_{sf_{\text{ext}}})$
$\leftarrow \sum_{i=0}^{l_{\text{f}}-1} (C_{q,p_q+i}\tilde{C}^{\text{T}}_{s,\tilde{p}_s+i}, C_{q,p_q+i}, \tilde{C}_{s,\tilde{p}_s+i})$ ▷ parallel sum reduction
6: Store $(K_{qsf_{\text{ext}}}, \boldsymbol{c}_{qf_{\text{ext}}}, \tilde{\boldsymbol{c}}_{sf_{\text{ext}}})$ in memory
7: **end for**
8: **end procedure**

Algorithm 4 Calculate transformation matrices

procedure CALCULATETRANSFORMATIONS(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, b_dyn)
  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**
    continue if *skip* flag is set for configuration $\{q, s, f_\mathrm{ext}\}$
    Load $(K_{qsf_\mathrm{ext}}, \boldsymbol{c}_{qf_\mathrm{ext}}, \tilde{\boldsymbol{c}}_{sf_\mathrm{ext}})$ from memory
    if $b_\mathrm{dyn} = 1$ **and** $l_q \geq \tilde{l}_s$ **then**
      $(K_{qsf_\mathrm{ext}}, \boldsymbol{c}_{qf_\mathrm{ext}}, \tilde{\boldsymbol{c}}_{sf_\mathrm{ext}}) \leftarrow (K^\mathrm{T}_{qsf_\mathrm{ext}}, \tilde{\boldsymbol{c}}_{sf_\mathrm{ext}}, \boldsymbol{c}_{qf_\mathrm{ext}})$
    end if
    Calculate R_{qsf_ext} by the Kabsch algorithm [6, 7]
      based on $(K_{qsf_\mathrm{ext}}, \boldsymbol{c}_{qf_\mathrm{ext}}, \tilde{\boldsymbol{c}}_{sf_\mathrm{ext}})$
    $\boldsymbol{t}_{qsf_\mathrm{ext}} \leftarrow \bar{\tilde{\boldsymbol{c}}}_{sf_\mathrm{ext}} - R_{qsf_\mathrm{ext}} \bar{\boldsymbol{c}}_{qf_\mathrm{ext}}$
    Store $(R_{qsf_\mathrm{ext}}, \boldsymbol{t}_{qsf_\mathrm{ext}})$ in memory
  end for
end procedure

Algorithm 5 Produce pilot alignments in constant time using spatial indices

procedure ALIGNINCONSTANTTIME(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, C, $\tilde{C}$, T, $\tilde{T}$, b_SSM)
  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**
    continue if *skip* flag is set for configuration $\{q, s, f_\mathrm{ext}\}$
    Calculate query and subject protein positions p_q and $\tilde{p}_s$ from index f_ext
    $l_\mathrm{f} \leftarrow \min\{512, l_q, \tilde{l}_s\}$
    if $l_q \geq \tilde{l}_s$ **then** ▷ always search in the larger structure
      $r \leftarrow \max\{0, \min\{\tilde{l}_s - l_\mathrm{f}, \tilde{p}_s - l_\mathrm{f}/2\}\}$
      $D \leftarrow \tilde{C}_{s,r:r+l_\mathrm{f}-1}$; $\tilde{D} \leftarrow C_q$; $\boldsymbol{\theta} \leftarrow \tilde{T}_{s,r:r+l_\mathrm{f}-1}$
    else
      $r \leftarrow \max\{0, \min\{l_q - l_\mathrm{f}, p_q - l_\mathrm{f}/2\}\}$
      $D \leftarrow C_{q,r:r+l_\mathrm{f}-1}$; $\tilde{D} \leftarrow \tilde{C}_s$; $\boldsymbol{\theta} \leftarrow T_{q,r:r+l_\mathrm{f}-1}$
    end if
    Load $(R_{qsf_\mathrm{ext}}, \boldsymbol{t}_{qsf_\mathrm{ext}})$ from memory
    for all $i \in [0, l_\mathrm{f})$ **do in parallel**
      $D'_i \leftarrow R_{qsf_\mathrm{ext}} D_i + \boldsymbol{t}_{qsf_\mathrm{ext}}$ ▷ transformation
      $j \leftarrow$ nearest neighbour in $\tilde{D}$ for D'_i using index
        with $(b_\mathrm{SSM} = 1)$ or without $(b_\mathrm{SSM} = 0)$ information θ_i
      ▷ $O(1)$ time complexity
      SWAP(D_i, $\tilde{D}_j$) **if** $l_q \geq \tilde{l}_s$
      Store $(D_i, \tilde{D}_j, j)$ in memory indexed by $(q, s, f_\mathrm{ext}, i)$
    end for
  end for
end procedure

Algorithm 6 Calculate cross-covariance matrices from alignments

procedure CALCULATEXCOVARIANCESA(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$)
  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**
    continue if *skip* flag is set for configuration $\{q, s, f_\mathrm{ext}\}$
    $l_\mathrm{f} \leftarrow \min\{512, l_q, \tilde{l}_s\}$
    Load $(D_i, \tilde{D}_i, \cdot)_{i=0}^{l_\mathrm{f}-1}$ from memory at $(q, s, f_\mathrm{ext}, i)_{i=0}^{l_\mathrm{f}-1}$
    $(K_{qsf_\mathrm{ext}}, \boldsymbol{c}_{qf_\mathrm{ext}}, \tilde{\boldsymbol{c}}_{sf_\mathrm{ext}}) \leftarrow \sum_{i=0}^{l_\mathrm{f}-1} (D_i \tilde{D}_i^\mathrm{T}, D_i, \tilde{D}_i)$ ▷ parallel sum reduction
    Store $(K_{qsf_\mathrm{ext}}, \boldsymbol{c}_{qf_\mathrm{ext}}, \tilde{\boldsymbol{c}}_{sf_\mathrm{ext}})$ in memory
  end for
end procedure

Algorithm 7 Calculate approximate order-dependent scores

procedure CalculateApproxScores(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$)
  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**
    continue if *skip* flag is set for configuration $\{q, s, f_\mathrm{ext}\}$
    $l_\mathrm{f} \leftarrow \min\{512, l_q, \tilde{l}_s\}$
    $\boldsymbol{x} \leftarrow -\mathbf{1}_{512}$; $\boldsymbol{m} \leftarrow \mathbf{0}_{512}$
    Load $(R_{qsf_\mathrm{ext}}, \boldsymbol{t}_{qsf_\mathrm{ext}})$ from memory
    for all $i \in [0, l_\mathrm{f})$ **do in parallel**
      Load $(D_i, \tilde{D}_i, j)$ from memory at $(q, s, f_\mathrm{ext}, i)$
      $D'_i \leftarrow R_{qsf_\mathrm{ext}} D_i + \boldsymbol{t}_{qsf_\mathrm{ext}}$ ▷ transformation
      $z_i \leftarrow j$; $a_i \leftarrow d_0^2/(d_0^2 + \|D'_i - \tilde{D}_i\|^2)$ ▷ d_0 defined as in [4]
    end for
    for $i = 0, \ldots, l_\mathrm{f} - 1$ **do**
      $\omega \leftarrow \max_{j: x_j < z_i} m_j$ ▷ parallel max reduction
      $c \leftarrow z_i \bmod 512$ ▷ trivial hash function
      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**
        $x_c \leftarrow z_i$; $m_c \leftarrow \omega + a_i$
      end if
    end for
    $w_{qsf_\mathrm{ext}} \leftarrow \max_j m_j$ ▷ parallel reduction
    Store w_{qsf_ext} in memory
  end for
end procedure

Algorithm 8 Select n_tfm top-performing transformation matrices

procedure GetTopNTransformations(n_Q, n_S, n_ext, n_tfm, b_srt)
  for all $(q, s) \in [0, n_\mathrm{Q}) \times [0, n_\mathrm{S})$ **do in parallel**
    $\boldsymbol{x} \leftarrow -\mathbf{1}_{n_\mathrm{tfm}}$; $\boldsymbol{m} \leftarrow \mathbf{0}_{n_\mathrm{tfm}}$
    if $b_\mathrm{srt} = 1$ **then**
      Load $(w_{qsf_\mathrm{ext}})_{f_\mathrm{ext}=0}^{n_\mathrm{ext}-1}$ from memory ▷ TM-scores if obtained by DP
      $\boldsymbol{x}' \leftarrow \mathrm{Sort}((w_{qsf_\mathrm{ext}})_{f_\mathrm{ext}})$ ▷ Batcher's sort [8] in $O(\log_2^2 n_\mathrm{ext})$ time
      $\boldsymbol{x} \leftarrow (x'_i)_{i=0}^{n_\mathrm{tfm}-1}$ ▷ $x_i = -1$ if $w_{qsx'_i} = 0$ (*skip* flag set)
    else
      for all $(f_\mathrm{ext}) \in [0, n_\mathrm{ext})$ **do in parallel** ▷ approx. to partial sorting
        continue if *skip* flag is set for configuration $\{q, s, f_\mathrm{ext}\}$
        Load w_{qsf_ext} from memory
        $f_\mathrm{m} \leftarrow f_\mathrm{ext} \bmod n_\mathrm{tfm}$
        $[\ (m_{f_\mathrm{m}}, x_{f_\mathrm{m}}) \leftarrow (w_{qsf_\mathrm{ext}}, f_\mathrm{ext})$ **if** $m_{f_\mathrm{m}} < w_{qsf_\mathrm{ext}}\ ]_\mathrm{mut}$
      end for
    end if
    for all $i \in [0, n_\mathrm{tfm})$ **do in parallel**
      Set $(x_i < 0)$ or unset $(x_i \geq 0)$ *skip* flag for configuration $\{q, s, i\}$
      if $x_i \geq 0$ **then**
        Load $(R_{qsx_i}, \boldsymbol{t}_{qsx_i})$ from memory
        $(R'_{qsi}, \boldsymbol{t}'_{qsi}) \leftarrow (R_{qsx_i}, \boldsymbol{t}_{qsx_i})$
      end if
    end for
    for all $i \in [0, n_\mathrm{tfm})$ **do in parallel**
      if $x_i \geq 0$ **then**
        $(R_{qsi}, \boldsymbol{t}_{qsi}) \leftarrow (R'_{qsi}, \boldsymbol{t}'_{qsi})$
        Store $(R_{qsi}, \boldsymbol{t}_{qsi})$ in memory
      end if
    end for
  end for
end procedure

Algorithm 9 Optimize n_{brn} selected alignments

procedure OptimizeSelectedAlignments(n_{Q}, n_{S}, n_{brn})
  for $i = 0, \ldots, n_{\mathrm{brn}} - 1$ **do**
    for all $(q, s) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}})$ **do in parallel**
      continue if *skip* flag is set for configuration $\{q, s, i\}$
      Load TF matrix $(R_{qsi}, \boldsymbol{t}_{qsi})$ from memory
      Perform DP using $(R_{qsi}, \boldsymbol{t}_{qsi})$ and the Comer2 DP algorithm [5]
      Produce alignment using the Comer2 backtracking algorithm
      Store $l_{\mathrm{A}}(R_{qsi}, \boldsymbol{t}_{qsi})$ aligned residues' coordinates $(C_{qk}, \tilde{C}_{sl})_{k,l}$ in memory
      Find optimal $(R_{qsi}, \boldsymbol{t}_{qsi})$ by calculating TM-scores
        based on the superpositions obtained in parallel from
        $n_{\mathrm{A}}[l_{\mathrm{A}}(R_{qsi}, \boldsymbol{t}_{qsi})]$ different-length and differently positioned
        alignment fragments (similarly to "Search Engine" in [4])
    end for
  end for
  for all $(q, s) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}})$ **do in parallel**
    Store $(R_{qs}, \boldsymbol{t}_{qs})$
      $\equiv \operatorname{argmax}_{\{(R_{qsi}, \boldsymbol{t}_{qsi})\}_{i=0}^{n_{\mathrm{brn}}-1}}$ TM-score$(R_{qsi}, \boldsymbol{t}_{qsi})$ in memory
  end for
end procedure

Algorithm 10 Refine best alignments

procedure RefineBestAlignments(n_{Q}, n_{S}, c_{goc})
  repeat 2 **times**
    for all $(q, s) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}})$ **do in parallel**
      Load TF matrix $(R_{qs}, \boldsymbol{t}_{qs})$ from memory
      Perform DP using $(R_{qs}, \boldsymbol{t}_{qs})$ and the Comer2 DP algorithm [5]
        with gap open cost c_{goc}
      Produce alignment using the Comer2 backtracking algorithm
      Store $l_{\mathrm{A}}(R_{qs}, \boldsymbol{t}_{qs})$ aligned residues' coordinates $(C_{qk}, \tilde{C}_{sl})_{k,l}$ in memory
      Find $(R'_{qs}, \boldsymbol{t}'_{qs})$ that maximizes TM-score
        based on the superpositions obtained in parallel from
        $n_{\mathrm{A}}[l_{\mathrm{A}}(R_{qs}, \boldsymbol{t}_{qs})]$ different-length and differently positioned
        alignment fragments (similarly to "Search Engine" in [4])
      Store $(R_{qs}, \boldsymbol{t}_{qs}) = (R'_{qs}, \boldsymbol{t}'_{qs})$ in memory
    end for
  end
end procedure

Bibliography

[1] Söding, J. & Remmert, M. Protein sequence comparison and fold recognition: progress and good-practice benchmarking. *Curr. Opin. Struct. Biol.* **21**, 404–411 (2011).

[2] Burley, S. *et al.* RCSB Protein Data Bank: powerful new tools for exploring 3d structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. *Nucleic Acids Res.* **49**, D437–D451 (2020).

[3] Defays, D. An efficient algorithm for a complete link method. *Comput. J.* **20**, 364–366 (1977).

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

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

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

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

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