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. 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.

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.

While widely used tools like TM-align [12] and Dali [13] offer precision, their relatively slow speed limits feasibility for searching millions of protein structures. In contrast, the recent tool Foldseek [14] introduces a novel approach by leveraging a structural alphabet to reduce the problem to a sequence comparison task and employs 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 (short for 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

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 [15]—a strategy akin to TM-align [12]. While the first two steps can be naturally parallelized for speed, dynamic programming (DP), typically

employed to produce a structural alignment, poses challenges due to inherent positional dependencies. The requirement for multiple alignments per structure pair represents a bottleneck to large-scale computation, even with an optimized parallel DP implementation [16].

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, 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.

We benchmarked GTalign against a set of well-established protein structure aligners: TM-align [12], Dali [13], FATCAT [17], DeepAlign [18], and Foldseek [14]. Comprehensive evaluations were conducted across four diverse datasets representing different protein analysis scenarios: (i) SCOPe 2.08 [19] protein domains filtered to 40% sequence identity, (ii) PDB [20] full-length structures filtered to 20% sequence identity, (iii) the UniProtKB/Swiss-Prot [21] protein structures from the AlphaFold Database [4], and (iv) the HOMSTRAD database [22].

The evaluation employed an unbiased and reference-free approach that reveals alignment accuracy through the superposition of structures based on the produced alignments (Fig. 2). We calculated the superposition score, TM-score [15], 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 [23].

The results (Fig. 1, Figs. 3–4) show that GTalign consistently outperforms all the aligners in terms of accuracy (see examples in Figs. 5–8). GTalign (option `--speed=0`) produces up to 7% more alignments with a TM-score ≥ 0.5 than TM-align (Fig. 1), the second most accurate tool (732,024 vs. 683,996, SCOPe 2.08 dataset). 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; SCOPe 2.08 dataset). 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. (See the Methods section for a discussion of this phenomenon.)

3 Conclusion

In 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 (Fig. 9). GTalign's cross-platform implementation, user-friendly interface, and high configurability, including the option for clustering structures (Supplementary Section S3), underscore its accessibility and versatility. Providing orders of magnitude in speedup at state-of-the-art accuracy, GTalign stands out among existing structure aligners.

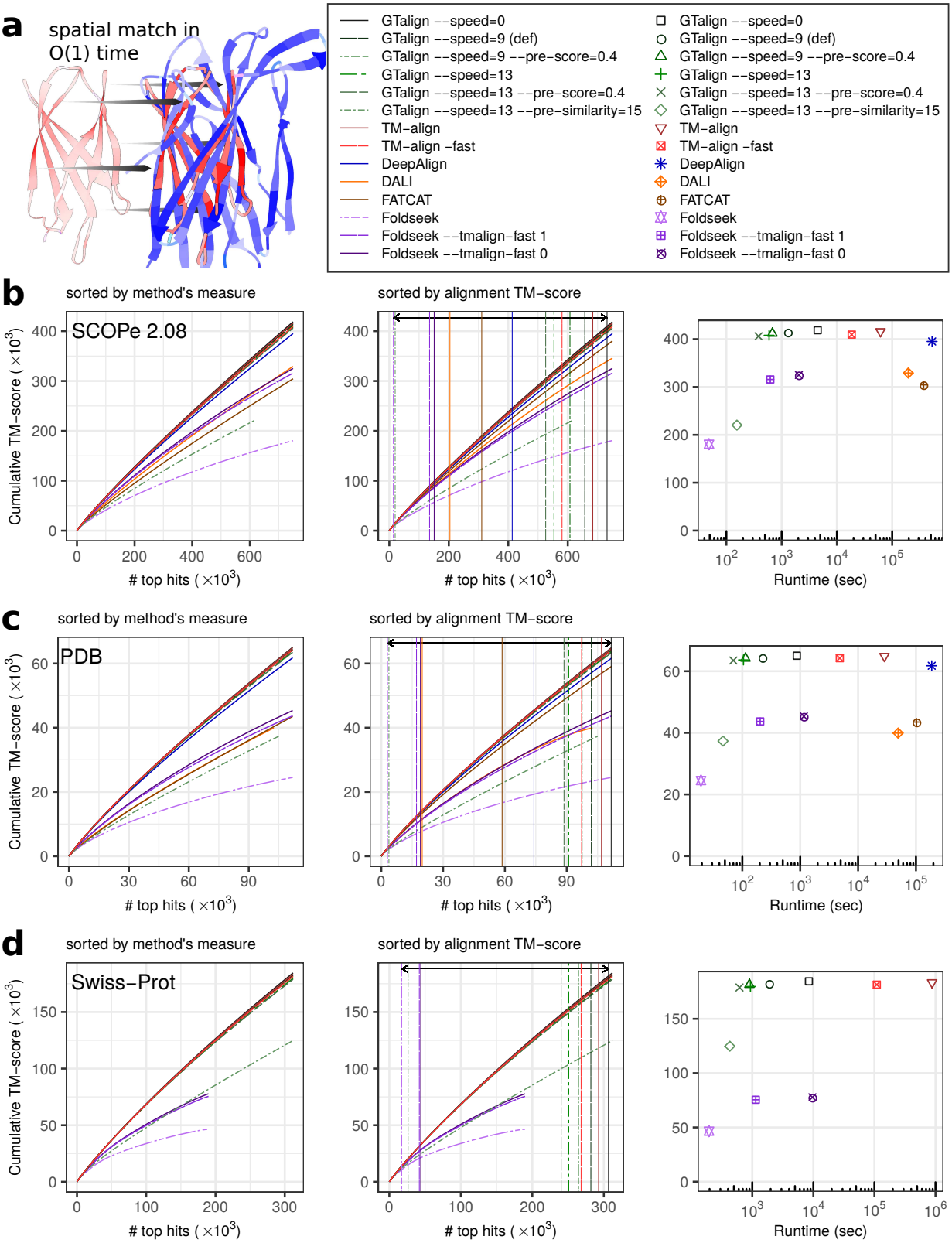


Fig. 1 Results. a, Illustration of matching protein structures with $O(1)$ time complexity. b, Benchmarking results on the SCOPe40 2.08 dataset with 2045 queries and 15,177 database entries. Parameterized runs of GTalign, TM-align, and Foldseek are included. The left panel plots the cumulative TM-score (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.

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. The following algorithmic steps represent actions performed on a pair of batch query and subject structures.

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 based on the parameters optimized in [12].
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 (RefineBestAlignments, Supplementary Section S1) 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 [16] using a scoring function based on secondary structure matching and sequence similarity score [24].
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 S1.
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 [12]. 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 Calculation of rotation matrices

GTalign computes rotation matrices using the Kabsch algorithm [25, 26]. Solving for the eigenvalues and eigenvectors of a cross-covariance matrix, $K \in \mathbb{R}^{3\times3}$ (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.

4.4 Dynamic programming implementation

A previously published algorithm [16] 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 $O(n_\mathrm{Q} \sum_s \tilde{l}_s)$ [16], 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.

4.5 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 (Fig. 9). 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.

4.6 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 S1, 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 [27], GTalign could achieve nearly instantaneous selection of similar protein candidates and a constant-time database search and alignment, regardless of the database size.

4.7 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 [24] 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.

The application of similarity prescreening in the sequence space results in a steep decrease in sensitivity. In contrast, no such effect is observed for prescreening in the structure space (Fig. 1 and Fig. 3). 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, even when based on spectral analysis of geometrical features in the frequency domain (as explored in our in-house research), may lack consistency and requires further investigation. Despite these observations, leveraging prescreening in the sequence space can prove valuable for high-similarity searches.

4.8 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.

4.9 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, calculated by the established method TM-align [12] using the `-I` option (Fig. 2). 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.

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 [13]) as their measures (e.g., Z-score) reduce the significance of such alignments.

4.10 The SCOPe40 2.08 dataset

All protein domains from the SCOPe 2.08 [19] 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 [13] 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.

4.11 The PDB20 dataset

PDB [20] structures filtered using `blastclust` [28] to 20% sequence identity with a length coverage threshold of 70% (PDB20), totaling 18,801, were queried with 186 CAMEO protein structure targets [29] 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.

4.12 The Swiss-Prot dataset

All UniProtKB/Swiss-Prot [21] protein structures (542,378) from the AlphaFold Database [4] were searched with 40 proteins representative of structurally diverse CRISPR-Cas systems [30]. 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.

4.13 The HOMSTRAD dataset

The HOMSTRAD dataset, comprising reference structural alignments of protein families and accompanying structure files, was obtained from Ref. [31], containing 398 multiple protein structure alignments from the HOMSTRAD database [22]. (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. The results (Figs. 4 and 8) demonstrate that GTalign-produced alignments exhibit greater accuracy in terms of TM-score compared to the reference alignments. A similar trend was observed previously [31].

4.14 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.

4.15 Runtime evaluation

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

4.16 GTalign settings

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 Fig. 1 and Figs. 3 and 4. 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.

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–c and Fig. 3a–b).

GTalign leveraged the computational power of three GPUs (option `--dev-N=3`). However, as detailed in Supplementary Section S2, the results unveil a noteworthy performance trend. A more recent desktop-grade GPU consistently outperforms the computational capabilities of the three server-grade GPUs.

4.17 TM-align settings

Parallel processing of all queries for each dataset was achieved by iteratively running 40 instances of TM-align [12] 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`.

4.18 Dali settings

The standalone version DaliLite.v5 [13] 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`.

4.19 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. 3a and b).

4.20 FATCAT settings

FATCAT 2.0 [17] 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`.

4.21 Foldseek settings

Foldseek [14] 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 and c. When evaluating alignments based on the TM-score normalized by the query length (Fig. 3a and b), E-value and the average TM-score (Foldseek --tmalign-fast 0/1), a recommended metric [14], 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.

4.22 Figure preparation

Molecular graphics images were generated using UCSF Chimera [32]. Plots were created using the ggplot2 package [33] in R [34], 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.

Code availability

The source code and software packages of GTalign are available at https://github.com/minmarg/gtalign_alpha and https://zenodo.org/records/10433420 (version 0.14.0; doi: 10.5281/zenodo.10433420). The lists of structure identifiers, as well as scripts 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 [12]. 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

The author declares no 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] Zhang, Y. & Skolnick, J. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Res.* **33**, 2302–2309 (2005).

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

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

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

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

[17] 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).

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

[19] 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).

[20] 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).

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

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

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

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

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

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

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

[28] 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).

[29] 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).

[30] 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).

[31] Dong, R., Peng, Z., Zhang, Y. & Yang, J. mTM-align: an algorithm for fast and accurate multiple protein structure alignment. *Bioinformatics* **34**, 1719–1725 (2017).

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

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

[34] 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/.

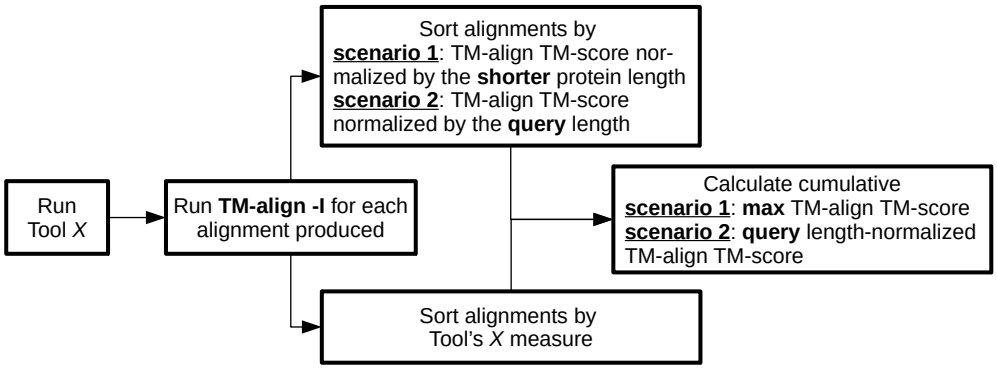


Fig. 2 Schematic for benchmarking structure alignment tools. The entire procedure can be described as follows: (i) Run a structure alignment tool; (ii) Use TM-align to calculate the TM-score for each produced alignment; (iii) Sort the alignments by the tool's measure (e.g., P-value, Z-score, etc.); (iv) 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.

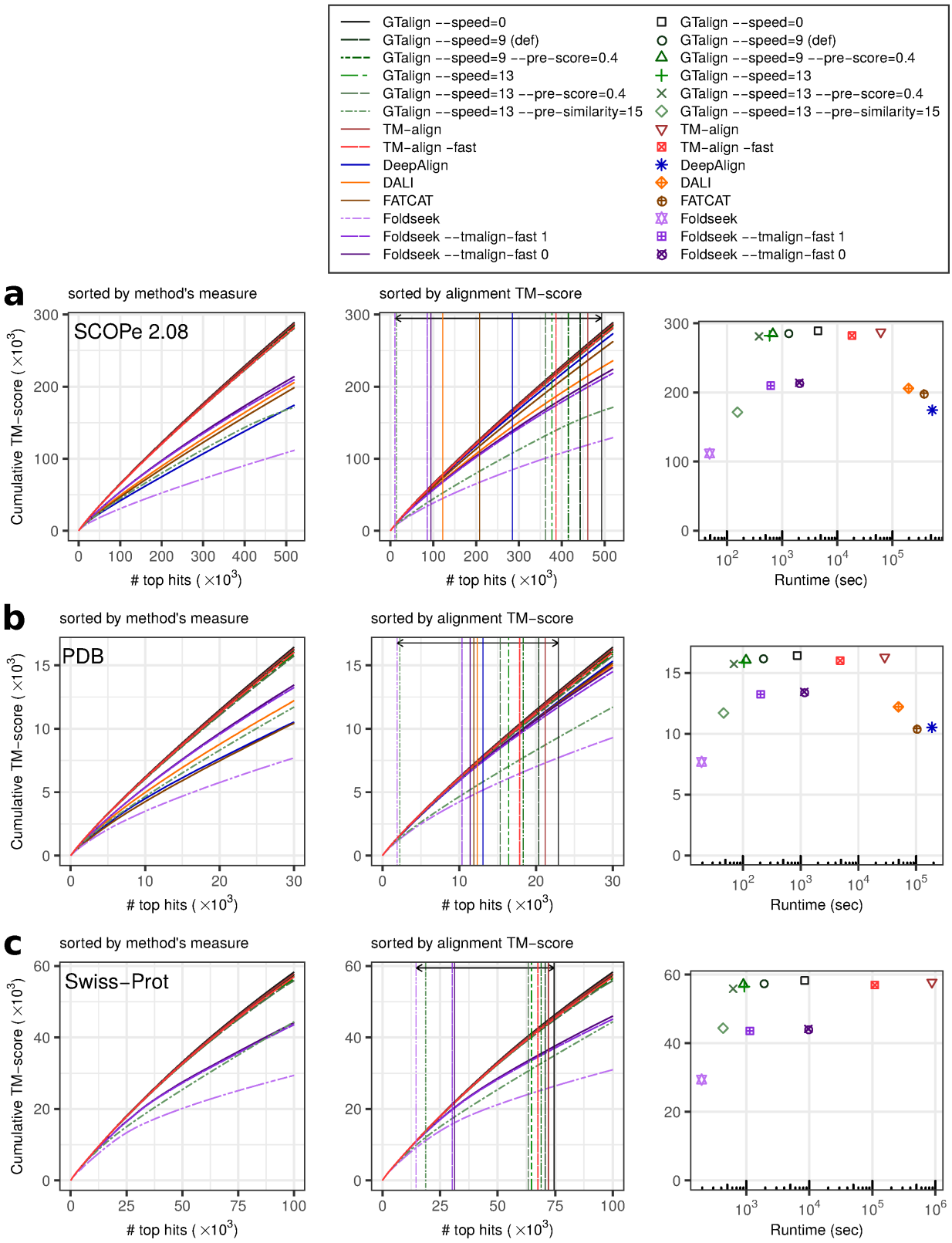


Fig. 3 Alignment evaluation results based on the TM-score normalized by the query length. a, Benchmarking results on the SCOPe40 2.08 dataset with 2045 queries and 15,177 database entries. The left panel plots the cumulative TM-score (normalized by the query length) against the number of top alignments 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 (10,376). 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.

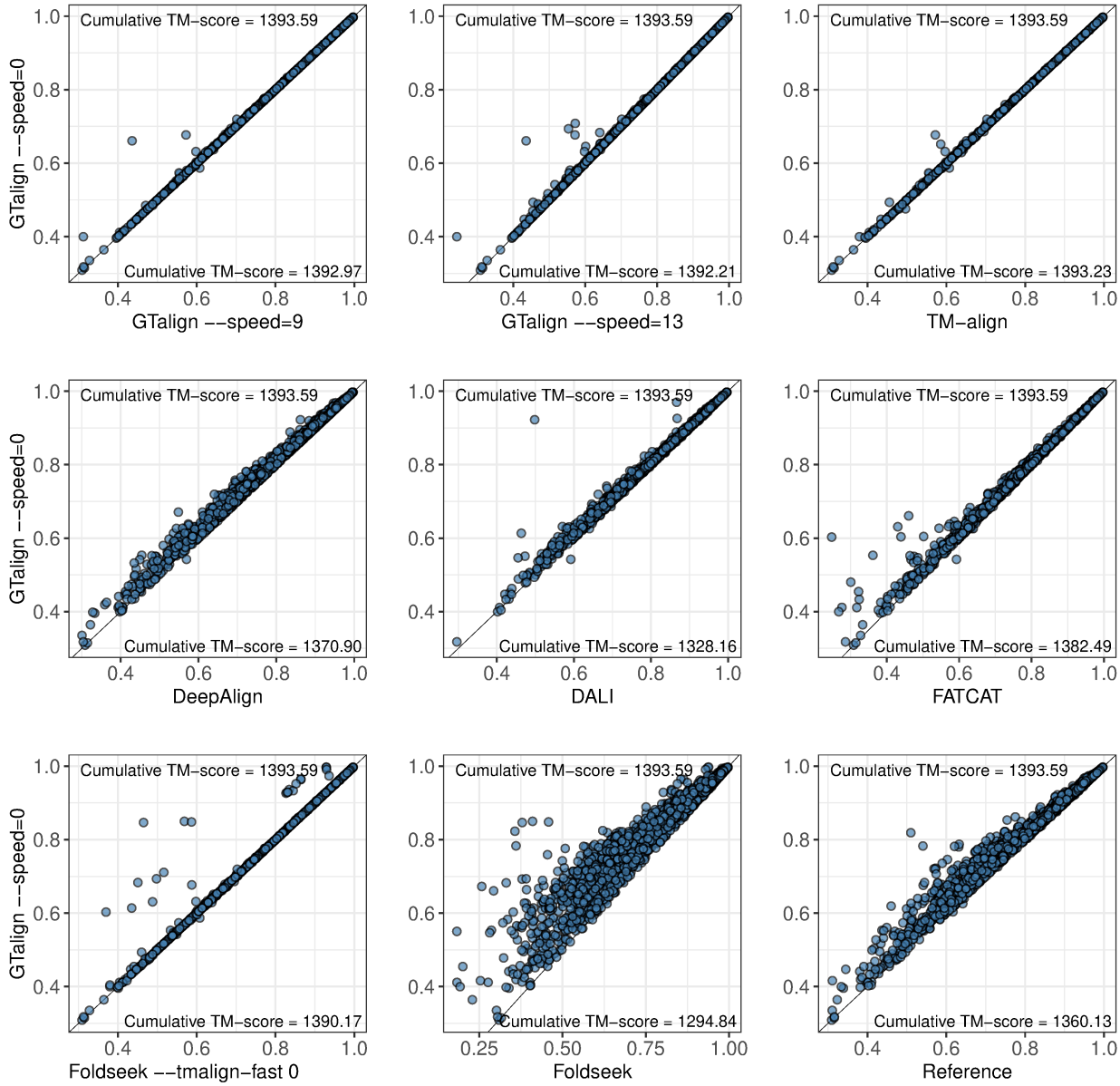


Fig. 4 Benchmarking results on the HOMSTRAD dataset. Each plot illustrates the distribution of TM-scores between two tools. The black line indicates identical distributions. 100 structure pairs for which Dali did not produce any alignment are excluded from the corresponding plot. "Reference" represents the original alignments in the database.

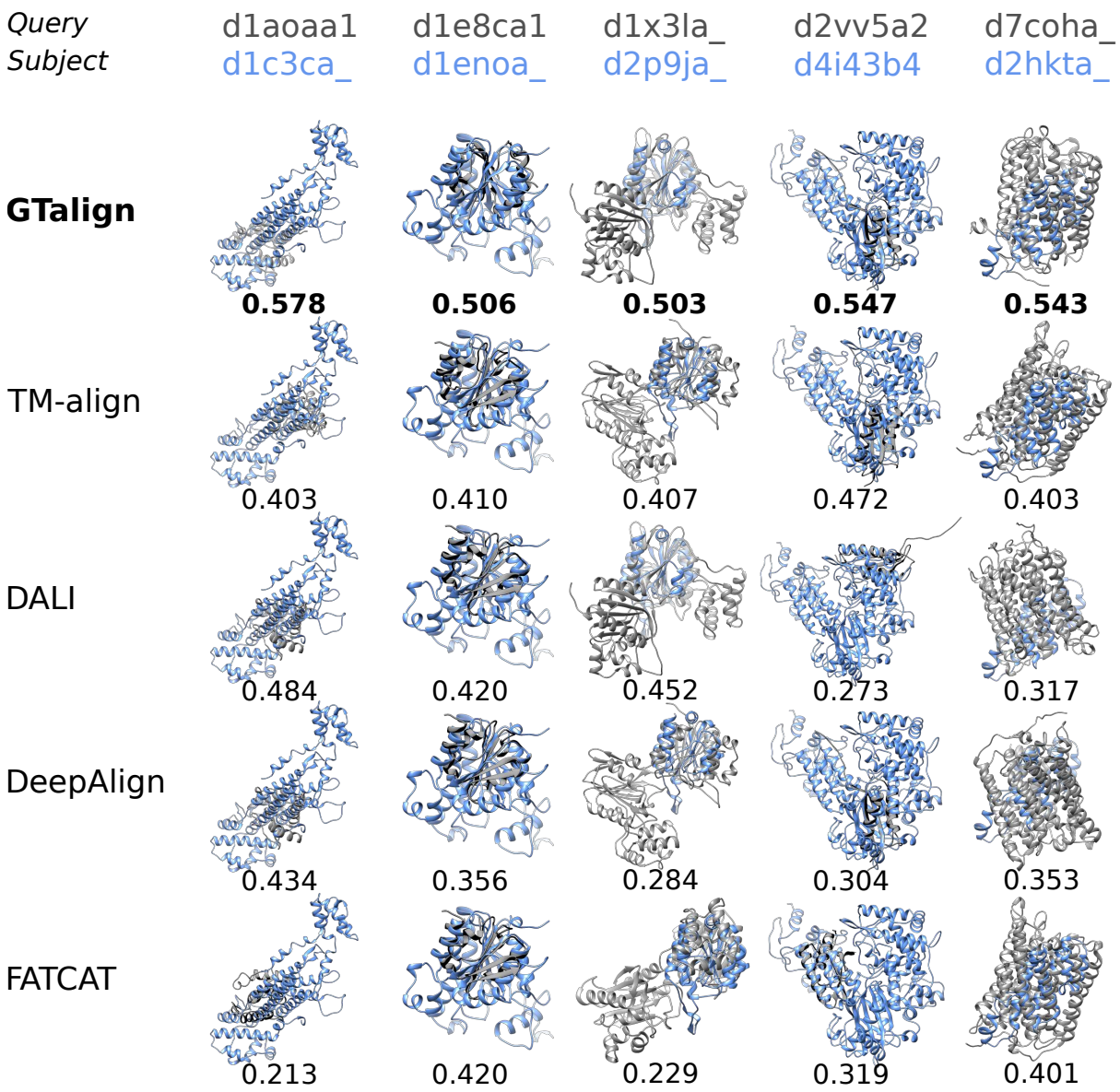


Fig. 5 Structural alignment examples from benchmarking on the SCOPe40 2.08 dataset. Query structures are in grey, and subject structures are colored. The subject structures share the same orientation across the tools for visual inspection. Numbers represent TM-scores normalized by the length of the shorter 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.

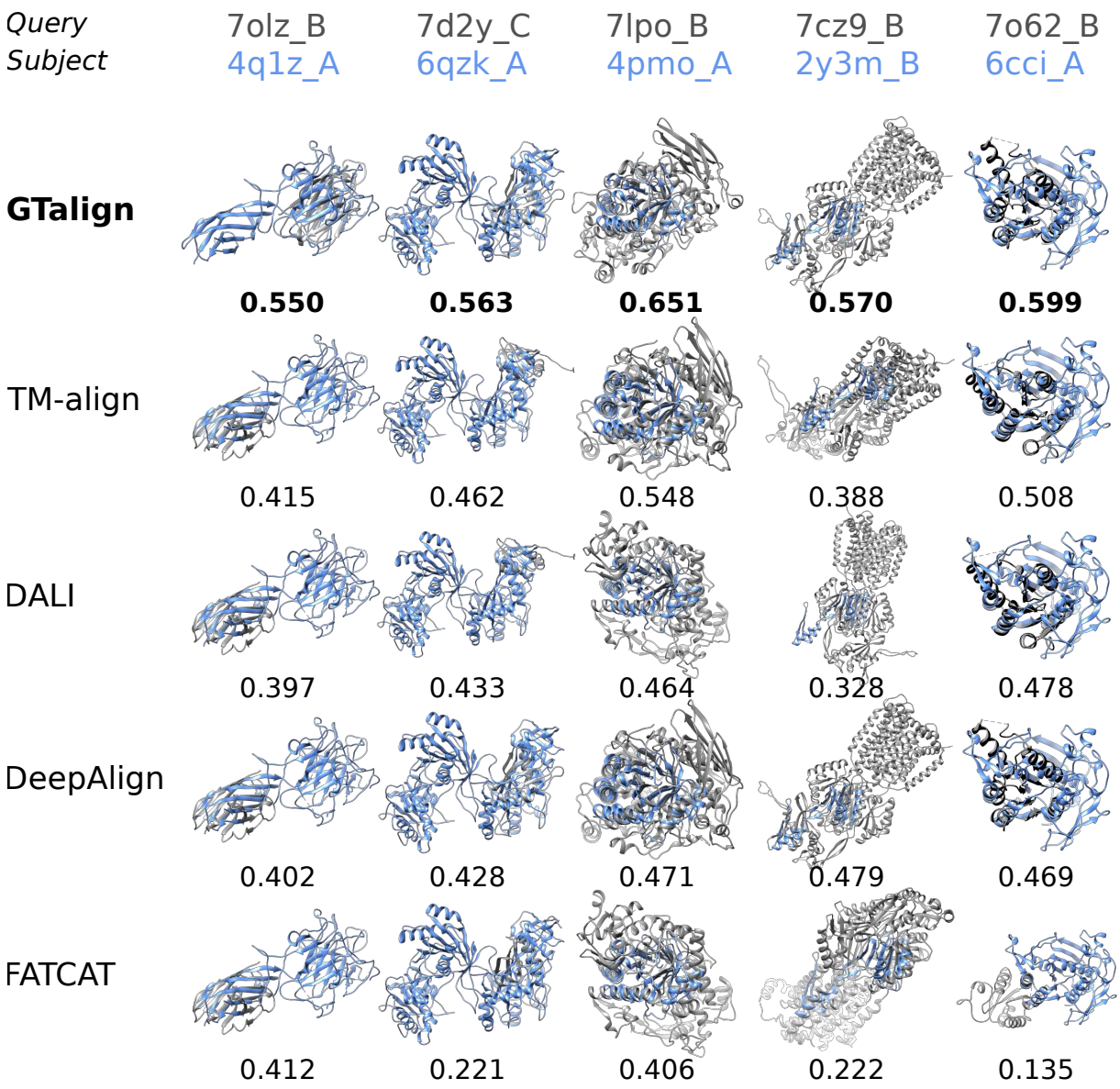


Fig. 6 Illustrative examples from benchmarking on the PDB20 dataset. Query structures are in grey, and subject structures are colored. The subject structures share the same orientation across the tools. Numbers represent TM-scores normalized by the length of the shorter protein. Foldseek (all parameterized versions) did not produce alignments for the given protein pairs.

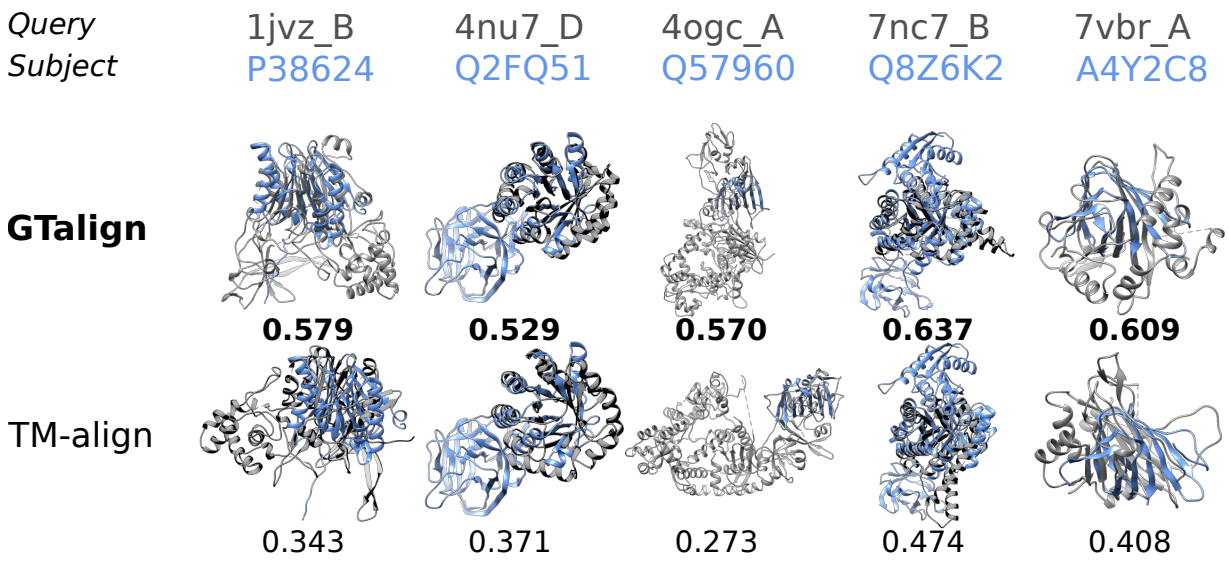


Fig. 7 Illustrative examples from benchmarking on the Swiss-Prot dataset. Query structures are in grey. Subject structures are colored. The subject structures share the same orientation across the tools. Numbers represent TM-scores normalized by the length of the shorter protein. Foldseek (all parameterized versions) did not produce alignments for the given protein pairs.

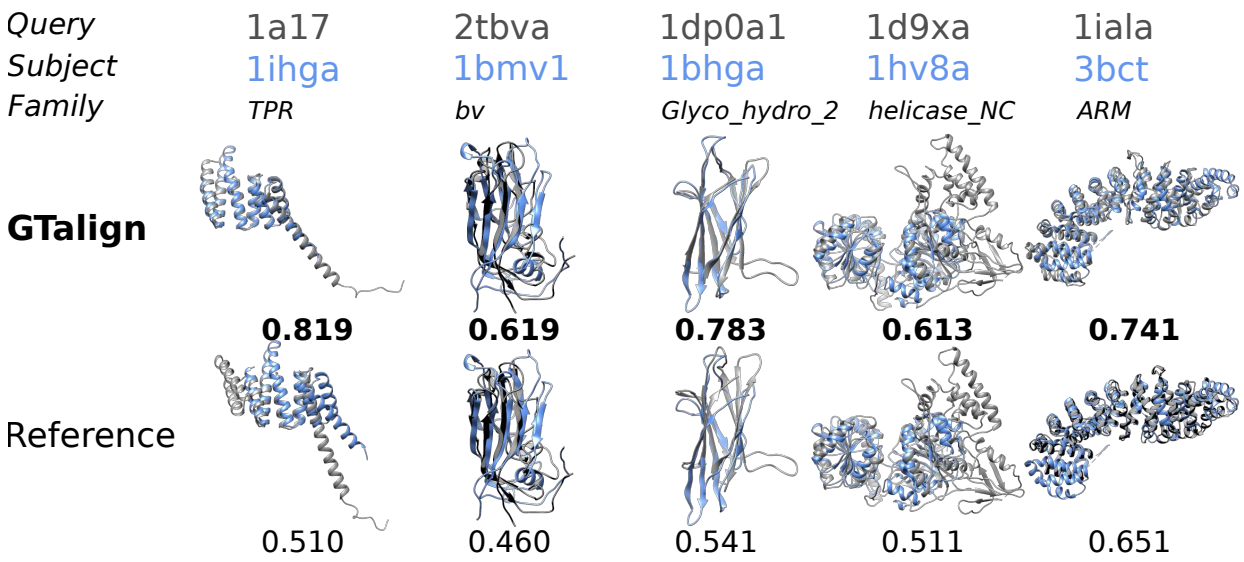
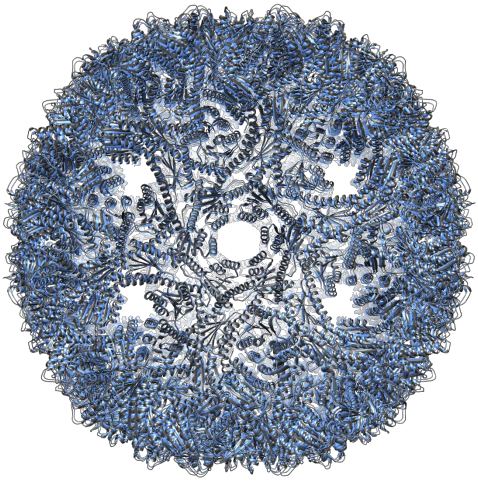


Fig. 8 Illustrative examples of GTalign and reference (original) alignments from benchmarking on the HOMSTRAD dataset. Query structures are depicted in grey, while subject structures are colored. The same subject structures maintain the same orientation for comparison. Numbers represent TM-scores normalized by the length of the shorter protein.



	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. 9 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`.

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;

S1 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.

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 [2]. 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 1 Find most favorable superpositions by deep search

procedure DeepSuperpositionSearch(n_Q, n_S, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, l_f, C, $\tilde{C}$, T, $\tilde{T}$, n_brn, c_thr)
  Calculate L using Eq. (S1) in parallel $\forall\, 0 \le q < n_\mathrm{Q}$ and $0 \le s < n_\mathrm{S}$
  Calculate n_ext based on $\max_q l_q$ and $\max_s \tilde{l}_s$ and depth specification
  Set n_tfm to 32, 64, or 96 based on depth specification
  CalculateLocalSimilarity(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, l_f, L, c_thr)
  CalculateXCovariances(n_Q, n_S, n_ext, l_f, C, $\tilde{C}$)
  CalculateTransformations(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, 1)
  AlignInConstantTime(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, C, $\tilde{C}$, T, $\tilde{T}$, 1)
  CalculateXCovariancesA(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$)
  CalculateTransformations(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, 1)
  AlignInConstantTime(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, C, $\tilde{C}$, T, $\tilde{T}$, 0)
  CalculateXCovariancesA(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$)
  CalculateTransformations(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, 0)
  CalculateApproxScores(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$)
  GetTopNTransformations(n_Q, n_S, n_ext, n_tfm, 0)
  Calculate TM-scores [1] in parallel $\forall\, 0 \le q < n_\mathrm{Q}$, $0 \le s < n_\mathrm{S}$, and
    TF matrices $0 \le i < n_\mathrm{tfm}$ using the Comer2 DP algorithm [2]
  GetTopNTransformations(n_Q, n_S, n_tfm, n_brn, 1)
  OptimizeSelectedAlignments(n_Q, n_S, n_brn)
  RefineBestAlignments(n_Q, n_S, -0.6)
  RefineBestAlignments(n_Q, n_S, 0)
end procedure

Algorithm 2 Calculate Local Similarity

procedure CalculateLocalSimilarity(n_Q, n_S, n_ext, $\boldsymbol{l}$, $\tilde{\boldsymbol{l}}$, l_f, L, c_thr)
  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**
    Calculate query and subject protein positions p_q and $\tilde{p}_s$ from index f_ext
    if $p_q + l_\mathrm{f} > l_q$ **or** $\tilde{p}_s + l_\mathrm{f} > \tilde{l}_s$ **then**
      Set *skip* flag for configuration $\{q, s, f_\mathrm{ext}\}$
      return
    end if
    $S \leftarrow 0_{32,32}$
    for all $i \in [0, \min\{96, l_q - p_q\})$ **do in parallel**
      for all $j \in [0, \min\{128, \tilde{l}_s - \tilde{p}_s\})$ **do in parallel**
        $[\ S_{i \bmod 32, j \bmod 32} \leftarrow \max\{S_{i \bmod 32, j \bmod 32}, L_{q,s,p_q+i,\tilde{p}_s+j}\}\ ]_\mathrm{mut}$
      end for
    end for
    $m \leftarrow \max_{i,j} S_{ij}$ ▷ two-dimensional parallel reduction
    if $m < l_\mathrm{f} \times c_\mathrm{thr}$ **then** ▷ fragment length fraction as a similarity threshold
      Set *skip* flag for configuration $\{q, s, f_\mathrm{ext}\}$
    end if
  end for
end procedure

Algorithm 3 Calculate cross-covariance matrices

procedure CalculateXCovariances(n_Q, n_S, n_ext, l_f, C, $\tilde{C}$)
  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
    $(K_{qsf_\mathrm{ext}}, \boldsymbol{c}_{qf_\mathrm{ext}}, \tilde{\boldsymbol{c}}_{sf_\mathrm{ext}})$
      $\leftarrow \sum_{i=0}^{l_\mathrm{f}-1} (C_{q,p_q+i}\tilde{C}^\mathrm{T}_{s,\tilde{p}_s+i}, C_{q,p_q+i}, \tilde{C}_{s,\tilde{p}_s+i})$ ▷ 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 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 [3, 4]
      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 [1]
    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 [5] 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 [2]
      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 [1])
    end for
  end for
  for all $(q, s) \in [0, n_{\mathrm{Q}}) \times [0, n_{\mathrm{S}})$ **do in parallel**
    Store $(R_{qs}, \boldsymbol{t}_{qs})$
      $\equiv \mathrm{argmax}_{\{(R_{qsi}, \boldsymbol{t}_{qsi})\}_{i=0}^{n_{\mathrm{brn}}-1}}$ 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 [2]
        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 [1])
      Store $(R_{qs}, \boldsymbol{t}_{qs}) = (R'_{qs}, \boldsymbol{t}'_{qs})$ in memory
    end for
  end
end procedure

S2 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 S1.

Table S1 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

S3 Clustering protein structures with GTalign

To showcase GTalign's clustering capabilities, we performed a clustering analysis on the entire PDB [6] 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 [7] (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. S1a.

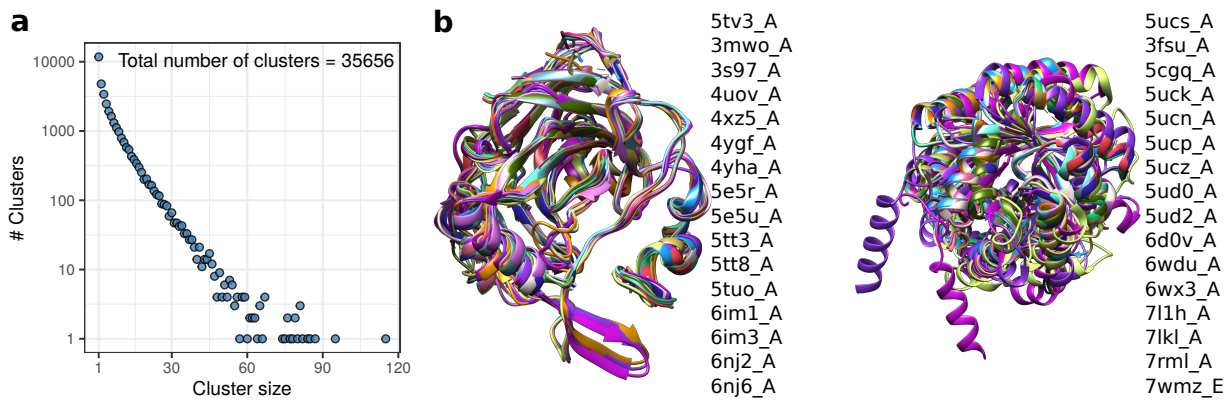


Fig. S1 Results of the GTalign clustering analysis on the entire PDB database. a, Distribution of cluster sizes. b, Superimpositions of proteins within selected clusters, representing the Carbonic anhydrase fold (left panel) and the TIM beta/alpha-barrel fold (right panel). Despite variations in sequence 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. S1b, 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.

References

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

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

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

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

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

[6] 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).

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