Theoretically and Practically Faster Algorithms for Protein Structure Alignment

Masahito Tsukahara
Human Genome Center, The Institute of Medical Science, The University of Tokyo, Japan
Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo, Japan

Tetsuo Shibuya
Human Genome Center, The Institute of Medical Science, The University of Tokyo, Japan

Abstract

Identifying shared substructures in 3D protein models is essential for structural bioinformatics. This task can be modeled as a sequential Largest Common Point-set (LCP) problem under the bottleneck distance. We propose a new $O(n^{13} \log n)$-time exact algorithm for this problem, which improves upon the previous best-known complexity of $O(n^{14})$, where n is the maximum size of the two input structures. Since these theoretical bounds are practically too large, an $O(n^7 \log n)$-time approximation algorithm with solution-size guarantees has been proposed; however, it remains too time-consuming for practical applications. Thus, we also propose a new filtering technique to enhance the approximation algorithm without increasing the theoretical time complexity or losing the solution-size guarantees. Experiments with PDB data show that our technique achieves over a 24-fold speedup at $n = 130$. While the previous algorithm required 3.80 hours on average for $n = 130$ in our experiments, making it difficult to test larger structures, our algorithm can process $n = 200$ in only 2.33 hours on average.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Applied computing → Molecular structural biology

Keywords and phrases Structural Bioinformatics, Pairwise Alignment, Exact Algorithm, Approximation Algorithm, Computational Geometry

Digital Object Identifier 10.4230/LIPIcs.WABI.2026.7

Supplementary Material
Software (Source Code): `https://github.com/masat03110/ProteinStructureAlignmentPlus` [18]
archived at `swh:1:dir:8bdeadc1f4304d35619d3600f97069c17bc38192`

Funding This work was partially supported by KAKENHI Grant Numbers 23K28035.

1 Introduction

Identifying shared 3D substructures among protein molecules is a fundamental task in structural bioinformatics. This task is commonly known as protein structure alignment, and it provides important clues about protein function, molecular interactions, and structure prediction [7, 15]. Several measures have been used to evaluate protein structure alignments, including Root Mean Square Deviation (RMSD) [3], the Global Distance Test (GDT) score, Local/Global Alignment (LGA) [24], and the TM-score [25]. Among them, GDT has been widely used in the Critical Assessment of protein Structure Prediction (CASP) experiments [16], because it is less sensitive than RMSD to local deviations and outliers [23]. Its relevance continues in recent CASP evaluations, including CASP16 [22].

Motivated by the GDT score, we study protein structure alignment through the Largest Common Point-set (LCP) problem under the bottleneck distance. Given two point sets P and Q in $\mathbb{R}^3$ and a threshold $\epsilon > 0$, the LCP problem asks for a maximum-cardinality subset $S \subseteq P$ such that, after a rigid transformation T, every point in $T(S)$ is within distance ϵ of

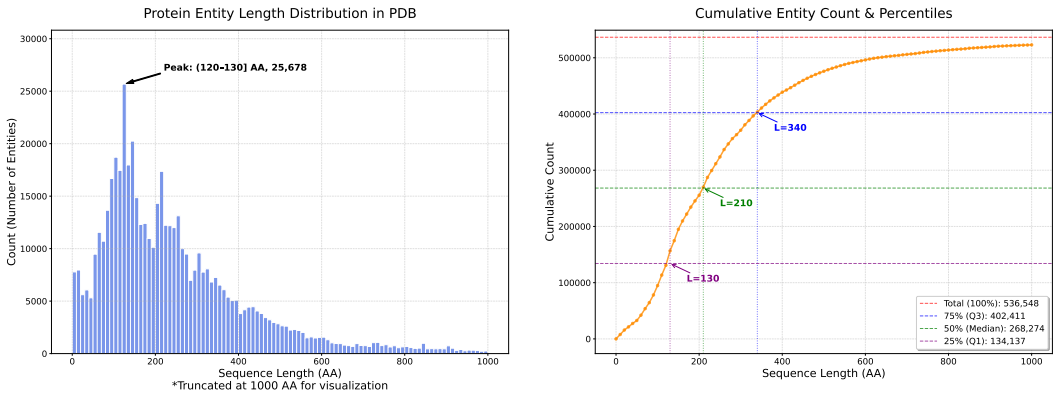

Figure 1 Distribution of protein sequence lengths in the Protein Data Bank (PDB) [6] as of May 9, 2026. **(Left)** Histogram of sequence lengths, truncated at 1,000 amino acids for visualization; the highest bin is (120, 130] residues. **(Right)** Cumulative count of protein entities ($N = 536{,}548$). The 25%, 50%, and 75% percentiles correspond to $L = 130$, $L = 210$, and $L = 340$, respectively.

a distinct point in Q [10]. For proteins, however, residues are not merely unordered points: their sequence order along the backbone is biologically essential. We therefore focus on the sequential LCP (sLCP) problem, which preserves residue order and naturally formulates the protein structure alignment problem considered in this paper.

A central difficulty is to obtain algorithms that are both theoretically guaranteed and practically scalable. The exact algorithm of Ambühl et al. [2] was previously reported to run in $O(n^{32})$ time, where n is the maximum size of the two input structures; however, the stated running-time bound does not necessarily follow from the reported analysis.[1] Although adapting Choi and Goyal's framework [9] gives an $O(n^{14})$-time exact algorithm, it remains impractical. Heuristic methods have also been developed [11, 20], but lack rigorous solution-quality guarantees. Approximation algorithms with theoretical guarantees have been developed to bridge this gap. Akutsu [1] gave an $O(n^8)$-time algorithm, followed by algorithms with stronger guarantees and better efficiency [8, 17]. Li and Ng [14] proposed an important framework with a reported running time of $O(\Delta_\epsilon^2 n^7/\delta^5)$, where Δ_ϵ and δ are defined later, but Tsukahara and Shibuya [19] showed that this bound does not hold in the worst case and proposed faster, more accurate corrected algorithms running in $O(\Delta_\epsilon n^7 \log n/\delta^5)$ time. Nevertheless, even their fastest algorithm remains costly in our experiments, requiring 1.54 hours at $n = 120$ and 3.80 hours at $n = 130$ on average. This matters for PDB-scale applications [6]: as shown in Figure 1, PDB entity lengths peak in (120, 130] residues, so under our experimental time limit, the previous fastest algorithms are practically limited to about the shortest 25% of PDB entities and become costly near the most common size range.

In this paper, we address both the theoretical and practical sides of this problem. First, we adapt and improve the framework of Choi and Goyal [9], obtaining an exact algorithm running in $O(\Delta_\epsilon n^{13} \log n)$ time. We then introduce a filtering technique for the approximation framework of Li and Ng [14] and Tsukahara and Shibuya [19], preserving its theoretical guarantees while improving practical performance. Experiments on PDB data show comparable solution quality for $n = 10, \ldots, 50$ and much better scalability: our method finishes in 5.83 and 9.50 minutes at $n = 120$ and $n = 130$, respectively, achieves over a 24-fold speedup at $n = 130$, and processes $n = 200$ and $n = 210$ in 2.33 and 3.33 hours, respectively, covering nearly 50% of PDB entities.

[1] This issue was pointed out via personal communication in May 2026, with acknowledgment to Bernd Gärtner for his insights.

2 Preliminaries

In this section, we introduce the notation used throughout the paper and formally define the Protein Structure Alignment problem. We model a protein as an ordered sequence of 3D points, where each point represents a residue, namely its $C\alpha$ atom. Thus, a protein $\mathcal{P}$ of length n is written as $\mathcal{P} = (p_1, \ldots, p_n)$, where $p_i \in \mathbb{R}^3$ is the position of the i-th residue. We use the following standard bounded-density property of protein structures.

▶ **Property 1** (Bounded Density [21]). *Due to steric constraints, residues in a protein cannot be arbitrarily close to each other. If the minimum distance between any two residues is at least D_1, then the number of residues contained in any ball of radius ϵ is bounded by $\Delta_\epsilon = O((1 + \frac{2\epsilon}{D_1})^3)$. In particular, for fixed ϵ, Δ_ϵ is independent of the protein length.*

A *rigid transformation* is a mapping $T : \mathbb{R}^3 \to \mathbb{R}^3$ consisting of a rotation and a translation. It preserves Euclidean distances, that is, $D(p, q) = D(T(p), T(q))$ for all $p, q \in \mathbb{R}^3$, where D denotes the Euclidean distance. For a sequence $\mathcal{P} = (p_1, \ldots, p_n)$, we write $T(\mathcal{P}) = (T(p_1), \ldots, T(p_n))$.

Let $\mathcal{S} = (s_1, \ldots, s_k)$ and $\mathcal{S}' = (s'_1, \ldots, s'_k)$ be two point sequences of the same length. A bijection $f : \mathcal{S} \to \mathcal{S}'$ is an order-preserving one-to-one correspondence defined by $f(s_i) = s'_i$ for each i. We say that $\mathcal{S}$ is ϵ-*congruent* to $\mathcal{S}'$ if there exists a rigid transformation T such that $D(T(\mathcal{S}), \mathcal{S}') \coloneqq \max_{1 \le i \le k} D(T(s_i), s'_i) \le \epsilon$.

For two proteins $\mathcal{P}$ and $\mathcal{Q}$, we seek the longest subsequence of $\mathcal{P}$ that is ϵ-congruent to a subsequence of $\mathcal{Q}$. Let $\mathfrak{S}$ be the set of all subsequences of $\mathcal{P}$ that are ϵ-congruent to some subsequence of $\mathcal{Q}$. We denote an optimal subsequence by $sLCP(\mathcal{P}, \mathcal{Q}, \epsilon) \in \arg\max_{\mathcal{S} \in \mathfrak{S}} |\mathcal{S}|$. Based on these definitions, the main problem considered in this paper is formulated as follows.

Protein Structure Alignment Problem

Input: Two proteins $\mathcal{P}$ and $\mathcal{Q}$, and a distance threshold $\epsilon > 0$.

Output: A subsequence $\mathcal{S}$ of $\mathcal{P}$, a rigid transformation T, and a bijection f from $\mathcal{S}$ to a subsequence of $\mathcal{Q}$ such that $|\mathcal{S}| = |sLCP(\mathcal{P}, \mathcal{Q}, \epsilon)|$ and $D(T(\mathcal{S}), f(\mathcal{S})) \le \epsilon$.

Unless otherwise stated, we let n and m denote the lengths of $\mathcal{P}$ and $\mathcal{Q}$, respectively, and assume $n \ge m$ without loss of generality. We consider the nontrivial case $|sLCP(\mathcal{P}, \mathcal{Q}, \epsilon)| \ge 2$, since instances with optimum size at most one can be solved immediately. For the time-complexity analysis, we use the real RAM model of computation, in which basic arithmetic operations on real numbers are exact and take constant time.

3 Exact Algorithms

In this section, we discuss exact algorithms for the Protein Structure Alignment problem. As noted in the introduction, the exact algorithm of Ambühl et al. [2], previously reported to run in $O(n^{32})$ time, was once regarded as the fastest exact method for this problem. However, the stated running-time bound does not necessarily follow from the reported analysis. Thus, a reliable theoretical baseline with a provable polynomial-time bound is needed.

We show that the algorithm of Choi and Goyal [9], originally developed for point pattern matching, can be adapted to the sequential setting to solve the problem in $O(n^{14})$ time. We first describe this adaptation and then improve it by exploiting the Bounded Density property (Property 1), obtaining an exact algorithm with running time $O(\Delta_\epsilon n^{13} \log n)$.

3.1 A Baseline Exact Algorithm: Adapting Choi and Goyal's Framework

Let $\mathfrak{T}$ denote the space of all rigid motions in $\mathbb{R}^3$. We identify $\mathfrak{T}$ with $\mathbb{R}^3 \times SO(3)$, corresponding to translations and rotations, respectively. For two points $p, q \in \mathbb{R}^3$ and a threshold $\epsilon > 0$, define

$$\text{score}(p, q, \epsilon) = \begin{cases} 1 & \text{if } \|p - q\| \leq \epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

Given two proteins $\mathcal{P} = (p_1, \ldots, p_n)$ and $\mathcal{Q} = (q_1, \ldots, q_m)$, two rigid motions $T_1, T_2 \in \mathfrak{T}$ are said to be equivalent if $\text{score}(T_1(p_i), q_j, \epsilon) = \text{score}(T_2(p_i), q_j, \epsilon)$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$. The induced partition of $\mathfrak{T}$ is denoted by $\Pi(\mathcal{P}, \mathcal{Q}, \epsilon)$, and each equivalence class is called a cell. Although such a cell need not be connected, it is characterized by an $n \times m$ binary score table whose (i, j)-entry is $\text{score}(T(p_i), q_j, \epsilon)$ for any T in the cell.

We use the following theorem of Choi and Goyal [9].

▶ **Theorem 2** ([9]). *For any cell $F \in \Pi(\mathcal{P}, \mathcal{Q}, \epsilon)$, there exists a set C of $k \leq 6$ contact constraints whose satisfying rigid motions form $2^{O(1)}$ connected components, at least one of which lies in the closure of F.*

This theorem implies that it suffices to enumerate all sets of at most six contact constraints and examine representative rigid motions obtained from them. Here, a contact constraint corresponds to a pair (p_i, q_j) whose distance becomes exactly ϵ after applying the rigid motion to p_i. The resulting adaptation gives the following algorithm.

Step 1. **Enumerate constraints:** Each contact constraint corresponds to a pair $(p_i, q_j) \in \mathcal{P} \times \mathcal{Q}$ for which $T(p_i)$ lies on the boundary of the ϵ-ball centered at q_j. We enumerate all combinations of $k \leq 6$ such constraints, giving $O(n^6 m^6)$ combinations.

Step 2. **Compute transformations:** For each combination, compute representative rigid motions satisfying the corresponding constraints. Since the dimension is fixed to three and the number of constraints is at most six, this can be done in constant time under the real RAM model, for example by using the method of Basu et al. [4].

Step 3. **Dynamic programming:** For each representative rigid motion T, construct the $n \times m$ score table in $O(nm)$ time. We then compute the maximum number of matched residue pairs that preserve the sequence order under this fixed transformation, which we call the sequential alignment size for T. This value is obtained by the dynamic programming recurrence of Akutsu [1]: $A(i, j) = \max\{A(i-1, j), A(i, j-1), A(i-1, j-1) + \text{score}(T(p_i), q_j, \epsilon)\}$, where $A(i, j) = 0$ if $i = 0$ or $j = 0$.

Step 4. **Output recovery:** Choose the rigid motion maximizing $A(n, m)$ and recover the optimal subsequence $\mathcal{S}$ and bijection f by standard DP backtracking.

For each of the $O(n^6 m^6)$ combinations, the dynamic programming step takes $O(nm)$ time. Thus, since $n \geq m$, we obtain the following theorem.

▶ **Theorem 3.** *The Protein Structure Alignment problem is solvable in $O(n^{14})$ time.*

3.2 An Improved Exact Algorithm

We now improve the baseline exact algorithm by avoiding the dense $n \times m$ score table for each candidate transformation. For a fixed rigid transformation T, define the active pair set as $E_T = \{(i, j) \mid D(T(p_i), q_j) \leq \epsilon\}$. Only pairs in E_T have score one and can contribute to the dynamic programming value; all other entries of the score table are zero. The baseline algorithm constructs the full score table, whereas our algorithm enumerates only E_T.

Algorithm 1 Improved Exact Algorithm for Protein Structure Alignment.

Input : Two proteins $\mathcal{P}$ ($|\mathcal{P}| = n$) and $\mathcal{Q}$ ($|\mathcal{Q}| = m$), and a distance threshold $\epsilon > 0$
Output : A subsequence $\mathcal{S}$ of $\mathcal{P}$, a rigid transformation T, and a bijection f

$T \leftarrow$ identity transformation, $s_{\max} \leftarrow 0, E_{\max} \leftarrow \emptyset$;
$\mathfrak{T} \leftarrow$ set of candidate transformations ; `// Steps 1 and 2 in Sec. 3.1`
$H \leftarrow$ map grouping each $q_j \in \mathcal{Q}$ by grid key $(\lfloor x_{q_j}/\epsilon \rfloor, \lfloor y_{q_j}/\epsilon \rfloor, \lfloor z_{q_j}/\epsilon \rfloor)$;
for $T_c \in \mathfrak{T}$ **do**
  $E_{T_c} \leftarrow \emptyset$;
  for $i = 1$ **to** n **do**
    $p'_i \leftarrow T_c(p_i)$;
    for each $q_j \in$ GetFrom27NeighborCells(H, p'_i, ϵ) **do**
      if $D(p'_i, q_j) \leq \epsilon$ **then**
        $E_{T_c} \leftarrow E_{T_c} \cup \{(i, j)\}$; `// Active pair`
  $s_c \leftarrow$ CalculateAlignmentScore(E_{T_c}) ; `// Sparse DP in [19]`
  if $s_c > s_{\max}$ **then**
    $s_{\max} \leftarrow s_c, T \leftarrow T_c, E_{\max} \leftarrow E_{T_c}$;
Recover $\mathcal{S}$ and f from $E_{\max}$ by sparse DP backtracking ; `// Step 4 in Sec. 3.1`
return $\mathcal{S}$, T, f ;

Our improvement is based on the Bounded Density property (Property 1). Since T preserves inter-residue distances, any ball of radius ϵ contains at most $O(\Delta_\epsilon)$ points of $T(\mathcal{P})$. Therefore, over all points of $\mathcal{Q}$, the number of active pairs is $|E_T| = O(\Delta_\epsilon m)$. Algorithm 1 exploits this sparsity using two procedures, which we describe below.

First, GetFrom27NeighborCells is a spatial lookup procedure for retrieving candidates near a transformed point. We partition $\mathcal{Q}$ into grid cells of side length ϵ and store each point q_j in a map H indexed by its grid key. Given a point $p' = T(p_i)$, GetFrom27NeighborCells(H, p', ϵ) returns all points of $\mathcal{Q}$ stored in the cell containing p' and its 26 neighboring cells. Any point within distance ϵ from p' must lie in one of these 27 cells, so checking these candidates and applying the distance test exactly recovers all pairs $(i, j) \in E_T$. Using a balanced map such as a B-tree [5], H is constructed in $O(m \log m)$ time, and all active pairs for a fixed T are retrieved in $O(\Delta_\epsilon n \log m + |E_T|)$ time because each cell contains $O(\Delta_\epsilon)$ points. Since $|E_T| = O(\Delta_\epsilon m)$ and $n \geq m$, this is bounded by $O(\Delta_\epsilon n \log m)$.

Second, CalculateAlignmentScore is the sparse dynamic programming procedure applied to the active pair set. Given E_T, it computes the maximum number of active pairs that can be selected while preserving the order of both sequences; equivalently, it computes the sequential alignment size for the fixed transformation T. We implement this procedure using the sparse DP framework of Tsukahara and Shibuya [19], which runs in $O(|E_T| \log m)$ time. Since $|E_T| = O(\Delta_\epsilon m)$ and $n \geq m$, this is bounded by $O(\Delta_\epsilon n \log m)$.

Thus, for each candidate transformation, Algorithm 1 replaces the dense $O(nm)$ score-table construction and DP with sparse pair retrieval and sparse DP in $O(\Delta_\epsilon n \log m)$ time, up to lower-order terms. Applying this procedure to all $O(n^6 m^6)$ candidate transformations and using $n \geq m$, we obtain the following theorem.

▶ **Theorem 4.** *The Protein Structure Alignment problem is solvable in $O(\Delta_\epsilon n^{13} \log n)$ time.*

4 Approximation Algorithms

The exact algorithm in the previous section, even with the improved $O(\Delta_\epsilon n^{13} \log n)$ running time, is still computationally prohibitive for large protein structures. To obtain theoretically guaranteed solutions more efficiently, Tsukahara and Shibuya [19] corrected and extended the approximation framework of Li and Ng [14]. Their result gives the following guarantee.

▶ **Theorem 5** ([19]). *Given two proteins $\mathcal{P}$ and $\mathcal{Q}$, a distance threshold $\epsilon > 0$, and an approximation factor $0 < \delta \leq 3$, there exists an algorithm that computes a subsequence $\hat{\mathcal{S}}$ of $\mathcal{P}$, a rigid transformation $\hat{T}$, and a bijection $\hat{f}$ from $\hat{\mathcal{S}}$ to a subsequence of $\mathcal{Q}$ such that $|\hat{\mathcal{S}}| \geq |sLCP(\mathcal{P}, \mathcal{Q}, \epsilon)|$ and $D(\hat{T}(\hat{\mathcal{S}}), \hat{f}(\hat{\mathcal{S}})) \leq (1+\delta)\epsilon$ in $O(\Delta_\epsilon n^7 \log n/\delta^5)$ time, where $n = \max(|\mathcal{P}|, |\mathcal{Q}|)$.*

Although this is much faster than exact algorithms, its practical scalability is still limited. As shown in Section 5, even the fastest existing guaranteed algorithm becomes costly around $n = 120$ to 130, which is near the most common size range in the PDB. We therefore propose an improved approximation algorithm based on a pruning technique. Our method preserves the guarantee of Theorem 5 while substantially improving practical efficiency. We first review the radial-pair framework underlying Li and Ng [14] and Tsukahara and Shibuya [19], and then present our improvement.

4.1 The Approximation Framework

The framework relies on the notion of a *radial pair*. For a point sequence $\mathcal{P}$, a point $p' \in \mathcal{P}$ is a radial point of $p \in \mathcal{P}$ if it is farthest from p among all points in $\mathcal{P}$. The pair (p, p') is called a radial pair of $\mathcal{P}$. Note that (p', p) is not necessarily a radial pair of $\mathcal{P}$. The following lemma of Li et al. [12] is the geometric basis of the approximation framework.

▶ **Lemma 6** ([12]). *Let $\mathcal{P}$ be a point sequence, T_1 and T_2 be rigid transformations, and (p, p') be a radial pair of $\mathcal{P}$. If $D(T_1(p), T_2(p)) \leq \gamma$ and $D(T_1(p'), T_2(p')) \leq \gamma$, then there exists a rotation R around the axis, namely the line through $T_1(p)$ and $T_1(p')$, such that $D(R(T_1(\tilde{p})), T_2(\tilde{p})) \leq 3\gamma$ for all $\tilde{p} \in \mathcal{P}$.*

Using this geometric property, Tsukahara and Shibuya [19] obtained the following subroutine for the Protein Structure Alignment problem.

▶ **Lemma 7** ([19]). *Given two proteins $\mathcal{P}$ and $\mathcal{Q}$, a distance threshold $\epsilon > 0$, an approximation factor $0 < \delta \leq 3$, a radial pair (p, p') of an optimal $sLCP(\mathcal{P}, \mathcal{Q}, \epsilon)$, and a pair (q, q') in $\mathcal{Q}$ such that $f(p) = q$ and $f(p') = q'$ for an optimal bijection f, there exists an $O(\Delta_\epsilon n^3 \log n/\delta^5)$-time algorithm that computes a subsequence $\hat{\mathcal{S}}$ of $\mathcal{P}$, a rigid transformation $\hat{T}$, and a bijection $\hat{f}$ satisfying the guarantee of Theorem 5, where $n = \max(|\mathcal{P}|, |\mathcal{Q}|)$.*

Proof. Let $\mathcal{S}^* = sLCP(\mathcal{P}, \mathcal{Q}, \epsilon)$, and let T^* and f^* be an optimal rigid transformation and bijection for $\mathcal{S}^*$. Thus, $D(T^*(\mathcal{S}^*), f^*(\mathcal{S}^*)) \leq \epsilon$. By assumption, (p, p') is a radial pair of $\mathcal{S}^*$ and $f^*(p) = q$, $f^*(p') = q'$.

The algorithm of Tsukahara and Shibuya [19], based on the sampling framework of Li and Ng [14], enumerates $O(1/\delta^5)$ candidate transformations determined by the fixed pairs (p, p') and (q, q'). Among these candidates, there is a transformation T' such that the images of the two radial points are sufficiently close to their positions under the optimal alignment. By Lemma 6, this implies that there exists a rotation R around the axis $l(T'(p), T'(p'))$ such that, for every $\tilde{p} \in \mathcal{S}^*$, the transformed point $R(T'(\tilde{p}))$ is within distance $\delta\epsilon$ of $T^*(\tilde{p})$. Therefore, for every $\tilde{p} \in \mathcal{S}^*$, $D(R(T'(\tilde{p})), f^*(\tilde{p})) \leq D(R(T'(\tilde{p})), T^*(\tilde{p})) + D(T^*(\tilde{p}), f^*(\tilde{p})) \leq \delta\epsilon + \epsilon = (1+\delta)\epsilon$.

Algorithm 2 Baseline Approximation Algorithm for Protein Structure Alignment [19].

Input : Two proteins $\mathcal{P}$ ($|\mathcal{P}| = n$) and $\mathcal{Q}$ ($|\mathcal{Q}| = m$), a distance threshold $\epsilon > 0$, and an approximation factor $0 < \delta \leq 3$

Output : A subsequence $\hat{\mathcal{S}}$ of $\mathcal{P}$, a rigid transformation $\hat{T}$, and a bijection $\hat{f}$

1 $\hat{\mathcal{S}} \leftarrow (), \hat{T} \leftarrow$ identity transformation, $\hat{f} \leftarrow$ empty bijection;
2 **for** $(p_{i_1}, p_{i_2}) \in \mathcal{P} \times \mathcal{P}$ *with* $i_1 \neq i_2$ **do**
3 **for** $(q_{j_1}, q_{j_2}) \in \mathcal{Q} \times \mathcal{Q}$ *with* $j_1 \neq j_2$ **do**
4 Calculate $\mathcal{S}_c, T_c, f_c$ from $(\mathcal{P}, \mathcal{Q}, \epsilon, \delta, (p_{i_1}, p_{i_2}), (q_{j_1}, q_{j_2}))$; `// Lemma 7`
5 **if** $|\mathcal{S}_c| > |\hat{\mathcal{S}}|$ **then**
6 $\hat{\mathcal{S}} \leftarrow \mathcal{S}_c, \hat{T} \leftarrow T_c, \hat{f} \leftarrow f_c$;
7 **return** $\hat{\mathcal{S}}$, $\hat{T}$, $\hat{f}$;

Hence, during the enumeration, the algorithm necessarily considers a transformation under which all points of the optimal subsequence $\mathcal{S}^*$ can be matched to $\mathcal{Q}$ within distance $(1+\delta)\epsilon$ while preserving the sequence order. The dynamic programming step for that transformation therefore returns a subsequence of size at least $|\mathcal{S}^*|$.

For each of the $O(1/\delta^5)$ sampled candidates, the best order-preserving matching can be computed in $O(\Delta_\epsilon n^3 \log n)$ time by the sparse dynamic programming procedure of Tsukahara and Shibuya [19]. Thus the total running time is $O(\Delta_\epsilon n^3 \log n/\delta^5)$. ◀

Lemma 7 gives the desired approximation once a radial pair of the optimal solution and its image pair in $\mathcal{Q}$ are fixed. Since these pairs are unknown, the baseline framework enumerates all $O(n^2)$ candidate pairs in $\mathcal{P}$ and all $O(m^2)$ image pairs in $\mathcal{Q}$, applies the fixed-pair procedure of Lemma 7 to each quadruple, and returns the largest solution. This procedure is summarized in Algorithm 2.

The enumeration considers $O(n^2m^2)$ quadruples, each requiring $O(\Delta_\epsilon n^3 \log n/\delta^5)$ time. Thus, for $n \geq m$, the total running time is $O(\Delta_\epsilon n^7 \log n/\delta^5)$. Since one quadruple matches the optimal radial pair and image pair, Theorem 5 follows.

4.2 An Improved Approximation Algorithm

The baseline framework in Algorithm 2 has strong theoretical guarantees, but each call to the fixed-pair procedure is performed on the full point sequences $\mathcal{P}$ and $\mathcal{Q}$. A closer look at the proof of Lemma 7 shows that, for the quadruple corresponding to an optimal solution, retaining only the optimal subsequence $\mathcal{S}^* = sLCP(\mathcal{P}, \mathcal{Q}, \epsilon)$ and its image $f^*(\mathcal{S}^*)$ is sufficient to obtain an approximate solution satisfying Theorem 5, where f^* is an optimal bijection. We therefore prune the input point sequences passed to each fixed-pair call, while ensuring that at least one optimal quadruple still retains both $\mathcal{S}^*$ and $f^*(\mathcal{S}^*)$. This leads to the improved approximation algorithm presented in Algorithm 3, which preserves the guarantee of Theorem 5.

We first justify the suffix pruning in line 4.

▶ **Lemma 8.** *Let $\mathcal{S}^* = sLCP(\mathcal{P}, \mathcal{Q}, \epsilon)$, and let p_i be the first point of $\mathcal{S}^*$ in the sequence order of $\mathcal{P}$. If all candidate pairs $(p_i, \tilde{p})$ with $\tilde{p} \in \mathcal{S}^*$ and all possible image pairs in $\mathcal{Q}$ are examined, then the search finds an approximate solution satisfying Theorem 5.*

Algorithm 3 Improved Approximation Algorithm for Protein Structure Alignment.

Input : Two proteins $\mathcal{P}$ ($|\mathcal{P}| = n$) and $\mathcal{Q}$ ($|\mathcal{Q}| = m$), a distance threshold $\epsilon > 0$, and an approximation factor $0 < \delta \leq 3$

Output : A subsequence $\hat{\mathcal{S}}$ of $\mathcal{P}$, a rigid transformation $\hat{T}$, and a bijection $\hat{f}$

```
1  Ŝ ← (), T̂ ← identity transformation, f̂ ← empty bijection;
2  for i_1 = 1 to n do
3      for i_2 = i_1 + 1 to n do
4          P' ← (p_{i_1}, p_{i_1+1}, ..., p_n) ;                          // Lemma 8
5          P' ← filter(P', condition: D(p_{i_1}, ·) ≤ D(p_{i_1}, p_{i_2})) ;   // Lemma 9
6          for (q_{j_1}, q_{j_2}) ∈ Q × Q with j_1 ≠ j_2 do
7              Q' ← filter(Q, condition: D(q_{j_1}, ·) ≤ D(p_{i_1}, p_{i_2}) + 2ε) ;   // Lemma 10
8              Calculate S_c, T_c, f_c from (P', Q', ε, δ, (p_{i_1}, p_{i_2}), (q_{j_1}, q_{j_2})) ;   // Lemma 7
9              if |S_c| > |Ŝ| then
10                 Ŝ ← S_c, T̂ ← T_c, f̂ ← f_c;
11 return Ŝ, T̂, f̂;
```

Proof. Since $p_i \in \mathcal{S}^*$, there exists a radial pair (p_i, p') of $\mathcal{S}^*$ by definition. Because p_i is the first point of $\mathcal{S}^*$, the point p' is contained in the suffix $(p_i, p_{i+1}, \ldots, p_n)$. Let $q = f^*(p_i)$ and $q' = f^*(p')$. Since all image pairs in $\mathcal{Q}$ are examined, the pair (q, q') is considered. Thus, the search includes a quadruple satisfying Lemma 7, and the corresponding fixed-pair call returns an approximate solution satisfying Theorem 5. ◀

Next, we justify the distance filter in line 5.

▶ **Lemma 9.** *If (p, p') is a radial pair of $\mathcal{S}^*$, then no point $p'' \in \mathcal{S}^*$ has $D(p, p'') > D(p, p')$.*

Proof. By definition, p' is a farthest point from p in $\mathcal{S}^*$. ◀

Therefore, when (p_{i_1}, p_{i_2}) is the radial pair used by an optimal quadruple, line 5 removes only points that cannot belong to $\mathcal{S}^*$.

Finally, we justify the pruning of $\mathcal{Q}$ in line 7.

▶ **Lemma 10.** *Let (p, p') be a radial pair of $\mathcal{S}^*$, and let f^* be an optimal bijection. If $f^*(p) = q$, then every point $q'' \in f^*(\mathcal{S}^*)$ satisfies $D(q, q'') \leq D(p, p') + 2\epsilon$.*

Proof. Let T^* be an optimal rigid transformation, and $p'' \in \mathcal{S}^*$ satisfy $f^*(p'') = q''$. Since (p, p') is a radial pair of $\mathcal{S}^*$, we have $D(p, p'') \leq D(p, p')$. Moreover, $D(T^*(p), q) \leq \epsilon$ and $D(T^*(p''), q'') \leq \epsilon$. Because T^* is rigid, $D(T^*(p), T^*(p'')) = D(p, p'')$. By the triangle inequality, $D(q, q'') \leq D(q, T^*(p)) + D(T^*(p), T^*(p'')) + D(T^*(p''), q'') \leq D(p, p') + 2\epsilon$. ◀

Combining Lemmas 8–10, consider the iteration in which p_{i_1} is the first point of $\mathcal{S}^*$, p_{i_2} is a radial point of p_{i_1} in $\mathcal{S}^*$, and $(q_{j_1}, q_{j_2}) = (f^*(p_{i_1}), f^*(p_{i_2}))$. In this iteration, the reduced sequences still contain $\mathcal{S}^*$ and $f^*(\mathcal{S}^*)$, so the fixed-pair call satisfies Lemma 7. Therefore, Algorithm 3 preserves the guarantee of Theorem 5, while reducing the practical search space.

In the next section, we evaluate these approximation algorithms under the original threshold ϵ of the Protein Structure Alignment problem, counting only alignments with $D(T(\mathcal{S}), f(\mathcal{S})) \leq \epsilon$. Thus, when needed, an algorithm with approximation factor δ is run with threshold $\epsilon/(1 + \delta)$; this is consistent with the evaluation protocol of [13, 19]. In our implementation of the proposed pruning framework, we use AlignFastLCS rather than AlignFastLIS as the underlying algorithm from [19], since AlignFastLCS was generally shown to be faster in their experiments.

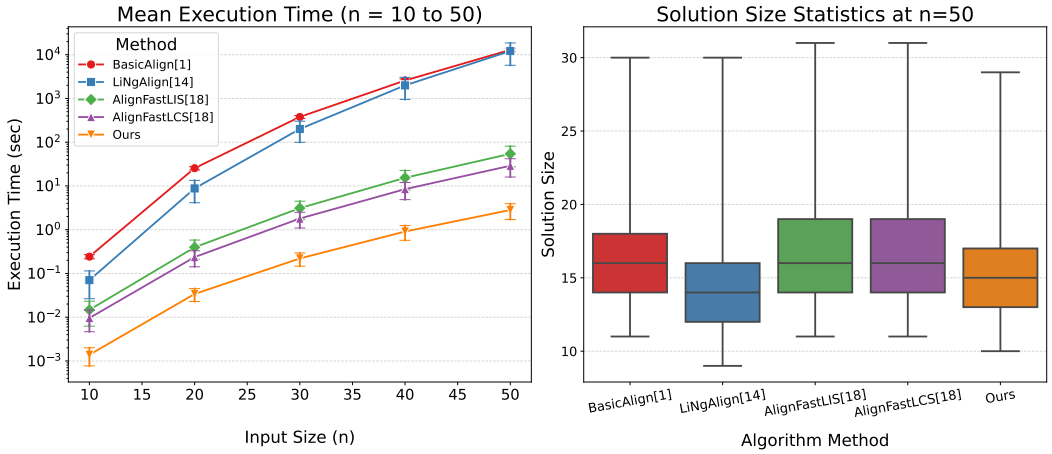

Figure 2 Results of Experiment 1 (equal lengths). **(Left)** Mean execution time for input sizes $n \in \{10, 20, 30, 40, 50\}$. The vertical axis is logarithmic in seconds, and error bars show the standard deviation over 10^3 trials. **(Right)** Distribution of alignment sizes at $n = 50$. Each box plot shows the median, interquartile range, and whiskers over 10^3 trials.

Table 1 Detailed results of Experiment 1, showing average execution time (s) and mean alignment size for $n = 10$ to 50.

	BasicAlign[1]		LiNgAlign[14]		AlignFastLIS[19]		AlignFastLCS[19]		Ours	
$n^{*)}$	Time	Size	Time	Size	Time	Size	Time	Size	Time	Size
10	0.24261	6.47	0.07036	6.57	0.01473	6.61	0.00958	6.61	0.00139	6.24
20	25.4969	10.08	8.78990	9.89	0.39586	10.26	0.23665	10.26	0.03394	9.50
30	379.052	12.60	200.972	11.89	3.11385	12.81	1.80055	12.81	0.22019	11.74
40	2578.34	14.84	1993.46	13.65	15.3602	15.07	8.45435	15.07	0.90889	13.80
50	12921.2	16.46	12157.9	14.66	54.3501	16.69	28.9667	16.69	2.82026	15.26

*) n is the length of each input protein, where $n = m$.

5 Experiments and Results

To evaluate the practical performance of our approach, we implemented the proposed algorithm in `C++` and compared it with baseline approximation algorithms from previous studies [1, 14, 19]. The experiments used protein structures obtained from the Protein Data Bank (PDB) [6] on May 9, 2026. Our implementation is available at `https://github.com/masat03110/ProteinStructureAlignmentPlus`.

All experiments were conducted on the SHIROKANE supercomputer at the Human Genome Center, The University of Tokyo. Each node is equipped with AMD EPYC 7713 CPUs and 512 GB of RAM, running CentOS Stream 8. The source code was compiled with GCC 13.2.0 using the `-std=c++20` and `-O3` flags. Each task was executed as a single-core job with 2.1 GB of memory using the `qsub` job scheduler.

5.1 Design and Execution

We conducted two experiments under the same basic settings: distance threshold $\epsilon = 2.0$ Å, approximation factor $\delta = 3$, and equal input sizes ($n = m$). For each parameter setting, we performed 10^3 trials. In each trial, we sampled random consecutive segments of Cα atoms from PDB proteins and measured both execution time and alignment size.

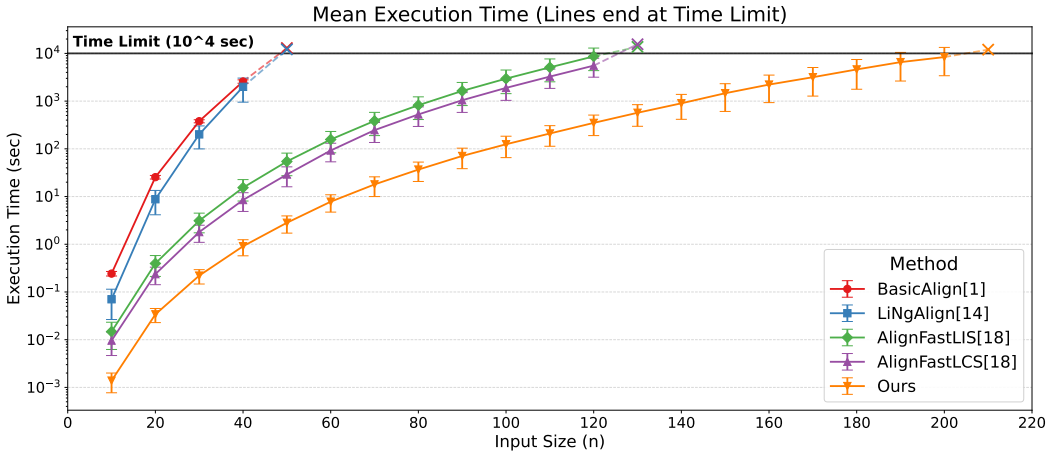

Figure 3 Results of Experiment 2 (scalability under a time limit). The graph shows mean execution time as a function of input size n, with the vertical axis on a logarithmic scale in seconds. Each method was evaluated while increasing n until its mean execution time exceeded 10^4 seconds. Dashed lines with an 'x' mark indicate where evaluation of a method was stopped under this rule. Error bars show the standard deviation over 10^3 trials.

Table 2 Detailed results of Experiment 2 for selected input sizes, showing average execution time (s) and mean alignment size.

	BasicAlign[1]		LiNgAlign[14]		AlignFastLIS[19]		AlignFastLCS[19]		Ours	
n	Time	Size	Time	Size	Time	Size	Time	Size	Time	Size
120	–*)	–*)	–*)	–*)	8647.04	25.61	5540.95	25.61	349.624	23.39
130	–*)	–*)	–*)	–*)	13687.5	26.64	15548.6	26.64	569.733	24.29
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
200	–*)	–*)	–*)	–*)	–*)	–*)	–*)	–*)	8375.27	29.67
210	–*)	–*)	–*)	–*)	–*)	–*)	–*)	–*)	11986.1	30.44

*) "–" indicates that the method was not evaluated because its mean execution time had already exceeded 10^4 seconds at a smaller input size.

Experiment 1: Equal Lengths. All methods were evaluated for input sizes $n \in \{10, 20, 30, 40, 50\}$. The average execution times and alignment sizes are shown in Figure 2 and Table 1.

Experiment 2: Scalability under a Time Limit. We evaluated scalability for input sizes from $n = 10$ to 220 in increments of 10, increasing n until the mean execution time exceeded 10^4 seconds. Once the mean execution time of a method exceeded 10^4 seconds at some input size, we did not evaluate that method for larger inputs. The results are shown in Figure 3 and Table 2.

5.2 Observations and Analysis

In Experiment 1, our method achieved a substantial speedup over the baseline algorithms while maintaining comparable alignment quality. As shown in Table 1, at $n = 50$ our method was more than 10 times faster than AlignFastLCS, the fastest baseline, and over 4500 times faster than BasicAlign. Although its mean alignment size was slightly smaller than those

of BasicAlign, AlignFastLIS, and AlignFastLCS, the difference was modest. The box plots in Figure 2 (Right) also show that, at $n = 50$, the alignment sizes produced by our method remain close to those of the high-accuracy baselines. This is consistent with the fact that our pruning preserves the same theoretical guarantee as the framework of Tsukahara and Shibuya [19].

Experiment 2 further demonstrates the scalability advantage of our method. As shown in Figure 3 and Table 2, AlignFastLIS and AlignFastLCS become impractical around $n = 120$–130: at $n = 130$, they require 13687.5 and 15548.6 seconds on average, respectively. In contrast, our method finishes in 349.624 seconds at $n = 120$ and 569.733 seconds at $n = 130$, corresponding to 5.83 and 9.50 minutes. Thus, at $n = 130$, our method achieves over a 24-fold speedup over the fastest baseline. Moreover, our method remains applicable to larger inputs, processing $n = 200$ and $n = 210$ in 8375.27 and 11986.1 seconds, or 2.33 and 3.33 hours on average, respectively.

These improvements are important in light of the PDB length distribution discussed in the introduction. As shown in Figure 1, the highest protein entity-length bin is (120, 130] residues, and the 25%, 50%, and 75% percentiles correspond to $L = 130$, $L = 210$, and $L = 340$, respectively. Under our experimental setting, the baseline methods become too slow around the first quartile of this distribution, whereas our method reaches approximately the median length. These results indicate that the proposed pruning substantially expands the practically accessible range of guaranteed protein structure alignment from roughly the shortest 25% of PDB entities to instances representative of nearly 50% of them.

6 Conclusion and Future Work

In this paper, we studied the Protein Structure Alignment problem through the sequential LCP formulation under the bottleneck distance. On the theoretical side, we adapted the framework of Choi and Goyal and improved the resulting exact algorithm from $O(n^{14})$ time to $O(\Delta_\epsilon n^{13} \log n)$ time by exploiting the bounded-density property of protein structures. On the practical side, we proposed a filtering technique for the guaranteed approximation framework of Li and Ng and Tsukahara and Shibuya. The proposed method preserves the theoretical approximation guarantee while substantially reducing the input size of each fixed-pair computation.

Experiments on PDB data demonstrated that our method maintains solution quality comparable to existing guaranteed algorithms for small instances with $n = 10, \ldots, 50$, while achieving much better scalability. In particular, our method finished in 5.83 and 9.50 minutes on average at $n = 120$ and $n = 130$, respectively, and achieved over a 24-fold speedup at $n = 130$, the most frequent size range in the PDB. Moreover, it processed instances with $n = 200$ and $n = 210$ in 2.33 and 3.33 hours on average, respectively, thereby reaching nearly the median length of PDB entities and covering nearly 50% of them. These results suggest that the proposed pruning makes theoretically guaranteed protein structure alignment more practical for real biological datasets.

Several directions remain for future work. First, the enumeration over candidate radial pairs and image pairs is highly parallelizable, and parallel implementation may further extend the applicable input size. Second, randomized sampling or heuristic prioritization of promising candidate pairs may provide additional speedups, especially when one can trade strict worst-case guarantees for practical efficiency. Finally, integrating the proposed approach with existing structural search pipelines may make it useful for broader database-scale analyses in structural bioinformatics.

References

1 Tatsuya Akutsu. Protein structure alignment using dynamic programing and iterative improvement. *IEICE Transactions on Information*, E79-D(12):1629–1636, 1996. URL: https://globals.ieice.org/en_transactions/information/10.1587/e79-d_12_1629/_p.

2 Christoph Ambühl, Samarjit Chakraborty, and Bernd Gärtner. Computing largest common point sets under approximate congruence. In Mike S. Paterson, editor, *Algorithms - ESA 2000*, pages 52–64. Springer Berlin Heidelberg, 2000. doi:10.1007/3-540-45253-2_6.

3 K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987. doi:10.1109/TPAMI.1987.4767965.

4 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer, 2 edition, 2006. doi:10.1007/3-540-33099-2.

5 Rudolf Bayer. Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta Informatica*, 1(4):290–306, December 1972. doi:10.1007/BF00289509.

6 Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, January 2000. doi:10.1093/nar/28.1.235.

7 Nathalie S. Boutonnet, Marianne J. Rooman, Maria-Elena Ochagavia, Jean Richelle, and Shoshana J. Wodak. Optimal protein structure alignments by multiple linkage clustering: application to distantly related proteins. *Protein Engineering, Design and Selection*, 8(7):647–662, July 1995. doi:10.1093/protein/8.7.647.

8 Samarjit Chakraborty and Somenath Biswas. Approximation algorithms for 3-d common substructure identification in drug and protein molecules. In Frank Dehne, Jörg-Rüdiger Sack, Arvind Gupta, and Roberto Tamassia, editors, *Algorithms and Data Structures*, pages 253–264. Springer Berlin Heidelberg, 1999. doi:10.1007/3-540-48447-7_26.

9 Vicky Choi and Navin Goyal. A combinatorial shape matching algorithm for rigid protein docking. In Suleyman Cenk Sahinalp, S. Muthukrishnan, and Ugur Dogrusoz, editors, *Combinatorial Pattern Matching*, pages 285–296, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-27801-6_21.

10 Vicky Choi and Navin Goyal. An efficient approximation algorithm for point pattern matching under noise. In José R. Correa, Alejandro Hevia, and Marcos Kiwi, editors, *LATIN 2006: Theoretical Informatics*, pages 298–310. Springer Berlin Heidelberg, 2006. doi:10.1007/11682462_30.

11 Lars Holm, Christos Ouzounis, Chris Sander, Georg Tuparev, and Gert Vriend. A database of protein structure families with common folding motifs. *Protein Science*, 1(12):1691–1698, 1992. doi:10.1002/pro.5560011217.

12 Shuai Cheng Li, Dongbo Bu, Jinbo Xu, and Ming Li. Finding largest well-predicted subset of protein structure models. In Paolo Ferragina and Gad M. Landau, editors, *Combinatorial Pattern Matching*, pages 44–55, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-69068-9_7.

13 Shuai Cheng Li, Dongbo Bu, Jinbo Xu, and Ming Li. Finding nearly optimal gdt scores. *Journal of Computational Biology*, 18(5):693–704, 2011. doi:10.1089/cmb.2010.0123.

14 Shuai Cheng Li and Yen Kaow Ng. On protein structure alignment under distance constraint. *Theoretical Computer Science*, 412(32):4187–4199, 2011. Algorithms and Computation. doi:10.1016/j.tcs.2010.11.045.

15 John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, Burkhard Rost, and Anna Tramontano. Critical assessment of methods of protein structure prediction - round viii. *Proteins*, 77(Suppl 9):1–4, 2009. doi:10.1002/prot.22589.

16 John Moult, Jan T. Pedersen, Richard Judson, and Krzysztof Fidelis. A large-scale experiment to assess protein structure prediction methods. *Proteins: Structure, Function, and Bioinformatics*, 23(3):ii–iv, 1995. doi:10.1002/prot.340230303.

17 Aleksandar Poleksic. On complexity of protein structure alignment problem under distance constraint. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(2):511–516, 2012. doi:10.1109/TCBB.2011.133.

18 Masahito Tsukahara and Tetsuo Shibuya. ProteinStructureAlignmentPlus. Software, swhId: swh:1:dir:8bdeadc1f4304d35619d3600f97069c17bc38192 (visited on 2026-08-10). URL: https://github.com/masat03110/ProteinStructureAlignmentPlus.

19 Masahito Tsukahara and Tetsuo Shibuya. Efficient and accurate approximation algorithms for protein structure alignment. In *Bioinformatics Research and Applications: 21st International Symposium, ISBRA 2025, Helsinki, Finland, August 3–5, 2025, Proceedings, Part I*, pages 122–134, Berlin, Heidelberg, 2025. Springer-Verlag. doi:10.1007/978-981-95-0698-9_11.

20 Gert Vriend and Chris Sander. Detection of common three-dimensional substructures in proteins. *Proteins*, 11(1):52–58, 1991. doi:10.1002/prot.340110107.

21 Jinbo Xu, Feng Jiao, and Bonnie Berger. A parameterized algorithm for protein structure alignment. *Journal of Computational Biology*, 14(5):564–577, 2007. PMID: 17683261. doi:10.1089/cmb.2007.R003.

22 Rongqing Yuan, Jing Zhang, Andriy Kryshtafovych, R. Dustin Schaeffer, Jian Zhou, Qian Cong, and Nick V. Grishin. Casp16 protein monomer structure prediction assessment. *Proteins: Structure, Function, and Bioinformatics*, 94(1):86–105, 2026. doi:10.1002/prot.70031.

23 A. Zemla, C. Venclovas, J. Moult, and K. Fidelis. Processing and analysis of casp3 protein structure predictions. *Proteins*, 37(Suppl 3):22–29, 1999. doi:10.1002/(sici)1097-0134(1999)37:3+<22::aid-prot5>3.3.co;2-n.

24 Adam Zemla. Lga: a method for finding 3d similarities in protein structures. *Nucleic Acids Research*, 31(13):3370–3374, July 2003. doi:10.1093/nar/gkg571.

25 Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004. doi:10.1002/prot.20264.