# Parallel seed-based approach to protein structure similarity detection

Guillaume Chapuis, Mathilde Le Boudic-Jamin, Rumen Andonov, Hristo Djidjev, Dominique Lavenier

# Parallel seed-based approach to protein structure similarity detection

Guillaume Chapuis[1], Mathilde Le Boudic - Jamin[1], Rumen Andonov[1], Hristo Djidjev[2], Dominique Lavenier[1]

[1]INRIA/ IRISA Rennes, GenScale, Rennes, France
{guillaume.chapuis,mathilde.le_boudic-jamin,
rumen.andonov,dominique.lavenier}@irisa.fr
[2]Los Alamos National Laboratory, Los Alamos NM, USA
djidjev@lanl.gov

**Abstract.** Finding similarities between protein structures is a crucial task in molecular biology. Many tools exist for finding an optimal alignment between two proteins. These tools, however, only find one alignment even when multiple similar regions exist. We propose a new parallel heuristic-based approach to structural similarity detection between proteins that discovers multiple pairs of similar regions. We prove that returned alignments have $RMSDc$ and $RMSDd$ lower than a given threshold. Computational complexity is addressed by taking advantage of both fine- and coarse-grain paralellism.

**Keywords:** protein structure comparison, parallel computing, seed-based heuristic, alignment graph.

## 1 Introduction

A protein's three dimensional structure tends to be better evolutionarily preserved than its sequence. Therefore, finding structural similarities between two proteins can give insights into whether these proteins share a common function or whether they are evolutionarily related. Structural similarities between two proteins are expressed by a one-to-one mapping (also called alignment) of their three dimensional representations. The quality of these alignments is crucial to correctly estimate protein functions and protein relations. Detecting the longest alignment, when comparing protein structures, is frequently modeled as finding the maximum clique [8,6], or enumerating all maximal cliques [3,10]. Both problems are NP-hard. In these approaches, cliques are looked for in so-called product (or alignment) graphs, where each edge corresponds to matching of similar internal distances (up to a user-defined threshold $\tau$). All edges in the target cliques satisfy this condition, but exactly this requirement leads to solving NP-hard problems.

Here, we relax this condition and accept cliques such that edges correspond to matching of similar internal distances up to $2\tau$. For this relaxed problem we propose a polynomial algorithm and its efficient parallel implementation comparing

two protein structures that guarantees to return alignments with both $RMSDc$ and $RMSDd$ less than a given threshold value, if such alignments exist. This methodology also offers the possibility to return more than one alignment for a single pair of proteins to address cases where two proteins share more than a single similar region. Our approach takes advantage of internal distance similarities among both proteins to search for an optimal transformation to superimpose their structures. To the best of our knowledge, our tool is unique in the capacity to generate multiple alignments with "good" $RMSDc$ and $RMSDd$ values. Thanks to this property, the tool is able to detect structural repetitions within a single protein and between related proteins. We do not require vertices in the alignment graph to be ordered which make our algorithm suitable for detecting similar domains when comparing multiple domained proteins. Computational complexity is addressed by extensive use of parallel computing techniques.

### 1.1 Alignment graphs

Undirected graphs $G = (V, E)$ are represented by a set $V$ of vertices and a set $E$ of edges between these vertices. In this paper, we focus on a subset consisting of grid-like graphs, referred to as alignment graphs.

An $m \times n$ *alignment graph* $G = (V, E)$ is a graph in which the vertex set $V$ is depicted by an $m \times n$ array $T$ , where each cell $T[i][k]$ contains at most one vertex $(i, k)$ from $V$. An example of such an alignment graph for protein comparison is given in Fig. 1.

The matching of two proteins $P_1$ and $P_2$ can be solved by analyzing an alignment graph $G = (V, E)$, where $V = \{(v_1, v_2) | v_1 \in V_1, v_2 \in V_2\}$ and $V_1$ (resp. $V_2$) is the set of atoms of interest in protein $P_1$ (resp. protein $P_2$). A vertex $(i, k)$ is present in $V$ only if atoms $i \in V_1$ and $k \in V_2$ are compatible. An example of incompatibility could be different electrostatic properties of the two atoms. An edge $((i, k), (j, l))$ is in $E$ if and only if the distance between atoms $i$ and $j$ in protein $P_1$, $d(i, j)$, is similar to the distance between atoms $k$ and $l$ in protein $P_2$, $d(k, l)$. In our case, these distances are considered similar if $|d(i, j) - d(k, l)| < \tau$, where $\tau$ is a given threshold.

### 1.2 Relation to protein structure comparison

In an alignment graph between two proteins $P_1$ and $P_2$, a subgraph with high density of edges denotes similar regions in both proteins. Finding similarities between two proteins can therefore be performed by searching the corresponding alignment graph for subgraphs with high edge density. The highest possible edge density is found in a clique, a subset of vertices that are all connected to each other.

DAST [8], for Distance-based Alignment Search Tool, aims at finding the maximal clique in an alignment graph. DAST uses alignment graphs where rows (resp. columns) represent an ordered set of atoms $V_1$ (resp. $V_2$) from protein $P_1$ (resp. protein $P_2$). A vertex $(i, j)$ is present in the graph if and only if residues $i$ and $j$ belong to similar secondary structures in both proteins. An edge is

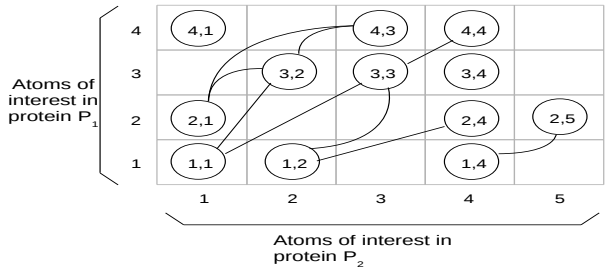


**Fig. 1.** Example of an alignment graph used here to compare the structures of two proteins. The presence of an edge between vertex $(1,1)$ and vertex $(3,2)$ means that the distance between atoms 1 and 2 of protein 1 is similar to the distance between atoms 1 and 3 of protein 2.

present between vertex $(i,j)$ and vertex $(k,l)$ if and only if $|d(i,j) - d(k,l)| < \tau$, where $\tau$ is a given threshold. By construction, alignments returned by DAST are guaranteed to have associated RMSDd strictly less than $\tau$.

### 1.3 Measures for protein alignments

Many measures have been proposed to assess the quality of a protein alignment. These measures include additive scores based on the distance between aligned residues such as the TM-score [13] or the STRUCTAL score [11] and Root Mean Square Deviation (RMSD) based scores, such as RMSD100, SAS and GSAS [5]. Given a set of $n$ deviations $S = s_1, s_2, ..., s_n$, its Root Mean Square Deviation is: $RMSD(S) = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} s_i^2}$. Two different RMSD measures are used for protein structure comparison: $RMSDc$, which takes into account deviations consisting of the euclidean distances between matched residues after optimal superposition of the two structures; $RMSDd$, which takes into account deviations consisting of absolute differences of internal distances within the matched structures. The measured deviations are $|d(i,j) - d(k,l)|$, for all couples of matching pairs "$i \leftrightarrow k, j \leftrightarrow l$". Let $P$ be the latter set and $N_m$, its cardinality. We have that $RMSDd = \sqrt{\frac{1}{N_m} * \sum_{(ij,kl) \in P} (|d(i,j) - d(k,l)|^2)}$.

## 2 Methods

### 2.1 Our approach

Looking for the maximal clique in a graph is a NP-complete problem [4]. Being an exact solver, DAST faces prohibitively long run times for some instances. We propose a polynomial approach to protein structure comparison that guarantees to return alignments with the following properties $RMSDd < 2\tau$ and

$RMSDc < \tau$, if such exist. Our approach offers the possibility to return an arbitrary number of distinct alignments. Returning multiple similar regions can prove useful, for instance, when looking for a structural pattern that may be present more than once in a protein or when comparing highly flexible proteins. However, enumerating multiple similar regions requires a more systematic approach than those developped in other existing heuristic-based tools. The computational burden associated with such a systematic approach can nevertheless be addressed by making use of multiple levels of parallelism.

Our method is inspired by the maximal clique search implemented in DAST. Instead of testing the presence of all edges among a subset of vertices ad in DAST, we only test the presence of edges between every vertex of the subset and an initial 3-clique, referred to as seed. The correctness of the resulting algorithm follows from geometric arguments, namely that the position of any 3-dimensional solid object is determined by the positions of three of its points that are not collinear.

### 2.2 Overview of the algorithm

The algorithm consists of the following three steps:

- Seeds in the alignment graph are enumerated. In our case, a seed is a set of three points in the alignment graph that correspond to two triangles (one in each protein) with similar internal distances. This step is detailed in section 2.3.
- Each seed is then extended. Extending a seed consists in adding all pairs of atoms, for which distances to the seed are similar in both proteins, to the set of three pairs of atoms that make up the seed. Seed extension is detailed in section 2.4.
- Each seed extension is filtered. Extension filtering is detailed in section 2.5 and consists in removing pairs of atoms that do not match correctly.

Filtered extensions are then ranked according to their size - number of aligned pairs of atoms - and a user-defined number of best matches are returned. This process is explained in section 2.7. The overall worst-case complexity of this algorithm is $O(|V| * |E|^{3/2})$.

### 2.3 Seed enumeration

A seed consists of three pairs of atoms that form similar triangles in both proteins. A triangle $IJK$ in protein $P_1$ is considered similar to a triangle $I'J'K'$ in protein $P_2$ if the following conditions are met: $|d(I,J) - d(I',J')| < \tau$, $|d(I,K) - d(I',K')| < \tau$ and $|d(J,K) - d(J',K')| < \tau$. Here, $d$ denotes the euclidean distance and $\tau$ is a user-defined threshold parameter. The default value for $\tau$ is 2.0 Ångströms.

In the alignment graph terminology, these conditions for a seed $(v_i = (I,I'), v_j = (J,J'), v_k = (K,K'))$ in graph $G(V,E)$ translate to the following: $(v_i,v_j) \in E$, $(v_i,v_k) \in E$ and $(v_j,v_k) \in E$.

A seed thus corresponds to a 3-clique in the alignment graph; i.e., three vertices that are connected to each other. Enumerating all the seeds is therefeore equivalent to enumerating every 3-clique in the input alignment graph.

Not all 3-cliques, however, are relevant. Suitable 3-cliques are composed of triangles for which a unique transformation can be found to optimally superimpose them. Namely, 3-cliques composed of triangles that appear to be too "flat" will not yield a useful transformation. We thus ensure that the triangles in both proteins defined by a potential seed are not composed of aligned points (or points which are close to being aligned). The worst-case complexity of this step is $O(|E|^{3/2})$ using, e.g., the algorithms from [9].

### 2.4 Seed extension

Extending a seed consists in finding the set of vertices that correspond to pairs of atoms that potentially match well (see section 2.5 for details) when the two triangles defined by the seed are optimally superimposed. Finding a superset of pairs of atoms that match well is performed by triangulation with the three pairs of atoms composing the seed. The computational complexity associated to this step is $O(|V|)$.

### 2.5 Extension filtering

In order to remove issues with symmetry (where the atoms in the extending pair are roughly symmetrical with respect to the plane determined by the seed atoms), we implemented a method to filter seed extensions. This method consists in computing the optimal transformation $T$ to superimpose the triangle from the seed corresponding to the first protein onto the triangle corresponding to the second. The optimal transformation is obtained using the fast, quaternion-based method of [7]. For each pair of atoms (L,L') composing the extension of a seed, we compute the euclidean distance between $T(L)$ and $L'$. If the distance is greater than a given threshold $\tau$, the pair is removed from the extension. The complexity of this step is $O(|V|)$ per seed.

### 2.6 Guarantees on resulting alignments' $RMSD$ scores

By construction, the filtering method ensures that the RMSD for a resulting alignment is less than $\tau$: the distance between two aligned residues after superimposition of the two structures is guaranteed to be less than $\tau$.

Internal distances between any additional pair of atoms and the seed is also guaranteed, by construction to be less than $\tau$. Concerning internal distances between two additional pairs of atoms, we ensure that in the worst possible case, the difference is $2 * \tau$, see Fig 2. The worst possible case happens when two additional pairs of atoms $v_l = (L, L')$ and $v_m = (M, M')$, added to the extension of a seed $(v_i, v_j, v_k)$, have atoms $L$, $L'$, $M$ and $M'$ aligned, after superimposition, and atoms from one protein lie within the segment defined by the two other atoms. In such a case, the filtering step ensures that $d(L, L') < \tau$ and $d(M, M') < \tau$ ; it follows that $|d(L, M) - d(L', M')| < 2 * \tau$.

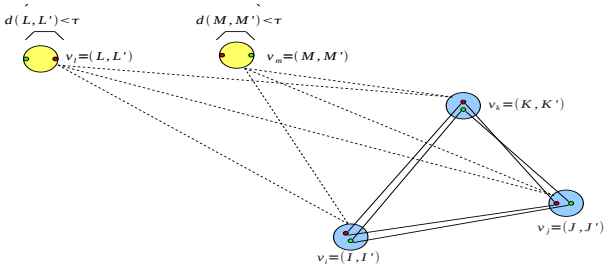


**Fig. 2.** Illustration of the guarantee on the similarity of internal distances between two pairs of atoms $v_l = (L, L')$ and $v_m = (M, M')$, here represented in yellow, added to a seed $(v_i, v_j, v_k)$ represented in blue. Dashed lines represent internal distances, the similarity of which is tested in the alignment graph.

### 2.7 Result ranking

When comparing two proteins, we face a double objective: finding alignments that are both long and have low $RMSD$ scores. The methodology described in section 2.5 ensures that any returned alignment will have a $RMSDd$ lower or equal to twice a user-defined parameter $\tau$. We can therefore leave the responsability to the user to define a threshold for $RMSD$ scores of interest. However, ranking alignments that conform to this $RMSD$ threshold simply based on their lengths is not an acceptable solution. In a given alignment graph, several seeds may lead to very similar transformations and thus very similar alignments. The purpose of returning multiple alignments for a single comparison is to find distinct similar regions in both proteins. Therefore, when two alignments are considered similar, we discard the shorter of the two.

Two alignments are considered similar, when they share defined number of pairs of atoms. This number can be adjusted depending on the expected length of the alignments or even set to a percentage of the smaller of the two compared alignments. This methodology of ranking results ensures that no two returned alignments match the same region in the first protein to the same region in the second protein.

## 3 Parallelism

### 3.1 Overview of the implemented parallelism

The overall complexity of our algorithm being $O(|V| * |E|^{3/2})$, handling large protein comparison with a decent level of precision - i.e., using alignment graphs with a large number of edges - can prove time-consuming. Our approach is however parallelizable at multiple levels.

Fig. 3 shows an overview of our parallel implementation. Multiple seeds are treated simultaneously to form a coarse-grain level of parallelism, while a finer grain parallelism is used when extending a single seed.

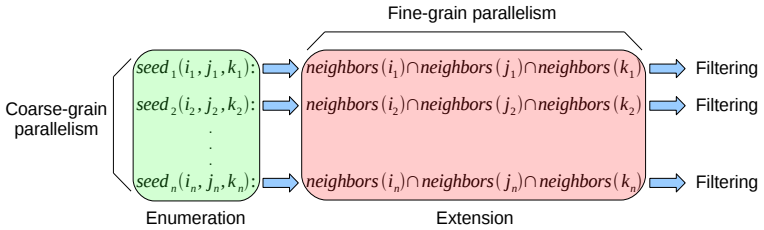


**Fig. 3.** Overview of the implemented parallelism.

### 3.2 Coarse-grain parallelism

Computations for enumerating seeds - see section 2.3, extending seeds - see section 2.4, and filtering the resulting extensions - see section 2.5, are independent processes, which can be performed in parallel. A user-defined number of threads can be spawned to handle, in parallel, computations for the various seeds present in the graph. This parallelism is implemented using the openMP standard [2].

Threads, however need to share their results to populate a global list of results. Inserting new entries in this global-result list would prove rather inefficient, because thread safety would need to be ensured by using locks around accesses to this result list. With such locks, threads would often stall whenever inserting a new alignment and the time lost on these accesses would only increase with the number of threads in use. In order to avoid any bottleneck when inserting a new alignment in the result list, each thread has its own private list. These lists are merged at the end of the computations to form a global result list. This method prevents the need for a synchronization mechanism and allows threads to be completely independent.

However, using this method can, in some cases, increase the total amount of computations. Whenever a seed extension is smaller than the smallest alignment present in the result list, it is discarded, thus avoiding the cost of a filtering step. Since each thread has its own result list, the minimal size required for the thread to consider filtering an extension is only a lower bound of the global minimal size found so far by all threads. Sharing only this global minimal size among threads is not a suitable solution, because no guarantee could be made on the distinctness of two alignments from different threads. Therefore, smaller similar regions would be wrongly discarded.

### 3.3 Fine-grain parallelism

Seed extension makes extensive use of set interection operations. In order to speed up these particular operations, we implemented a bit vector representation of the neighbors set of each vertex of the alignment graph. These bit vectors represent the neighbors in the alignment graph of each vertex (cf. figure 4). For a vertex $v_i$, a bit is set at position $j$ if and only if vertices $v_i$ and $v_j$ are connected in the alignment graph.

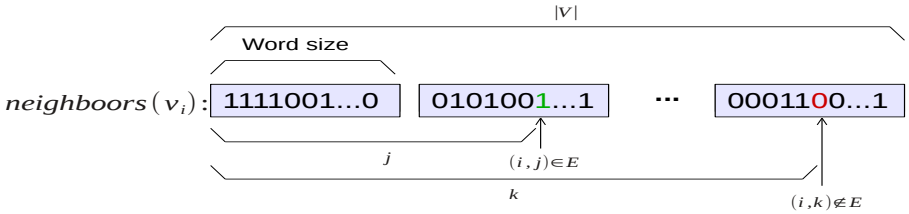


**Fig. 4.** Bit vector representation of the neighbors of vertex $v_i$ in an alignment graph $G(V, E)$. In this example, $v_j$ unlike $v_k$ is a neighbor of $v_i$.

This bit vector representation of the neighbors sets allows bit parallel computations of set intersection. A simple logic *and* operation over every word element of the two sets yields the intersection.

Intersection operations also benefit from SSE[1] instructions. A number of atomic operations equal to the size of the SSE registers available on the machine (typically 128 or 256) can be computed simultaneously. However, this sparse approach to computing set intersections increases the number of atomic operations to perform. Namely, vertices, which are not neighbors of any of the two vertices for which the intersection is computed, will induce atomic operations. Such vertices would not be considered in a traditional approach to set intersection. This sparse approach is still faster in our case because alignment graphs tend to be dense enough. The size of the resulting intersections is required for the rest of our algorithm. Knowing the size of an intersection allows us to discard seeds, when larger results have already been found. Computing the size of a sparse set is not as trivial as it is with a dense set. In order to compute the size of a sparse set, we use a built-in population count instruction (POPCNT) available in SSE4. This operation returns, in constant time, the number of bits set in a single machine word. For architectures without a built-in population count instruction, a slower alternative is provided.

## 4 Results and perspectives

In order to test the capacity of our approach to detect multiple regions of interest, we considered two proteins (PDB IDs 4clna and 2bbma). These proteins are each composed of two similar domains - named A and B (resp. C and D) for the first protein (resp. second protein), separated by a flexible bridge. Existing approaches, such as PAUL [12] and ones based on contact map overlap (CMO) [1], tend to match both proteins integrally, yielding larger alignments but poorer RMSD scores. TM_align [14], the reference tool for protein comparison, only matches domain A onto domain C. The four top results of our tool correspond to all four possible combinations of domain matching. Our tool was run using 12 cores of an Intel(R) Xeon(R) CPU E5645 @ 2.40GHz and the distance threshold

[1] Streaming SIMD Extensions

was set to 7 Ångströms and to 2 Ångströms in the alignment graph. Scores corresponding to these alignments are displayed in Table 1.

| | **CMO** | **PAUL** | **TMAlign** | **AC** | **BD** | **AD** | **BC** |
|---|---|---|---|---|---|---|---|
| **# of aligned residues** | 148 | 148 | 79 | 72 | 70 | 66 | 64 |
| **% of aligned residues** | 100 | 100 | 53.4 | 48.7 | 47.3 | 44.6 | 43.2 |
| **RMSDc** | 14.781 | 14.781 | 2.935 | 2.048 | 1.731 | *1.592* | 2.210 |
| **RMSDd** | 10.838 | 10.838 | 2.627 | 1.797 | 1.475 | *1.414* | 1.770 |
| **TM_score** | 0.161 | 0.161 | *0.422* | 0.411 | *0.422* | 0.405 | 0.358 |

**Table 1.** Details of the alignments returned by other tools - columns 2 through 4 - and our method - columns 5 through 8. Best scores are in italics.

In order to test our coarse-grain parallel implementation, we compare run times obtained with various numbers of threads on a single artificially large instance. Any instance can be made artificially large by allowing a large number of vertices and edges when creating the alignment graph. The input alignment graph for this instance contains 15024 vertices for 9565358 edges. Computations were run using a varying number of cores of an Intel(R) Xeon(R) CPU E5645 @ 2.40GHz. Table 2 shows run times and speedups with respect to the number of CPU cores. The gain in terms of speedup becomes less significant beyond 12 cores. Note that similar results - both in terms of length and $RMSD$ scores - can be obtained in less than 30 seconds with a sparser alignment graph.

| # of cores | 1 | 2 | 3 | 4 | 6 | 8 | 12 | 16 | 20 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|
| Run time (s) | 6479 | 3696 | 2494 | 1932 | 1374 | 1072 | 781 | 723 | 676 | 643 |
| Speedup | 1 | 1.8 | 2.6 | 3.4 | 4.7 | 6.0 | 8.3 | 9.0 | 9.6 | 10.1 |

**Table 2.** Run times and speedups for varying # of cores.

Fig. 5 shows run times for graphs with a varying number of edges and the same number of vertices - 21904. Computations were run using 12 cores of an Intel(R) Xeon(R) CPU E5645 @ 2.40GHz. Input alignment graphs were all generated from the same two proteins and different parameters to allow a varying number of edges.

This approach could be used to find similarities between RNA structures. However, such structures can be much larger than proteins. Therefore, future work includes further optimizations to allow larger alignment graphs to be computed.

## References

1. Rumen Andonov, Noël Malod-Dognin, and Nicola Yanev. Maximum contact map overlap revisited. *Journal of Computational Biology*, 18(1):27–41, 2011.

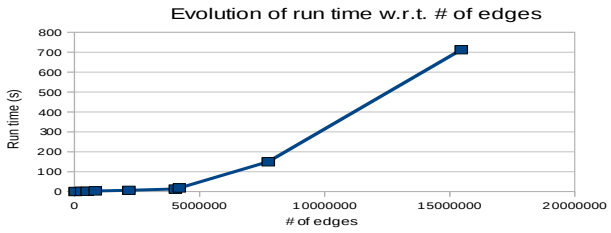


**Fig. 5.** Evolution of run times with respect to # of edges in the alignment graph.


2. Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.
3. Jean-Francois Gibrat, Thomas Madej, and Stephen H Bryant. Surprising similarities in structure comparison. *Current opinion in structural biology*, 6(3):377–385, 1996.
4. Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.
5. Rachel Kolodny, Patrice Koehl, and Michael Levitt. Comprehensive evaluation of protein structure alignment methods: scoring by geometric measures. *Journal of molecular biology*, 346(4):1173–1188, 2005.
6. Janez Konc and Dušanka Janežič. Probis algorithm for detection of structurally similar protein binding sites by local structural alignment. *Bioinformatics*, 26(9):1160–1168, 2010.
7. Pu Liu, Dimitris K Agrafiotis, and Douglas L Theobald. Fast determination of the optimal rotational matrix for macromolecular superpositions. *Journal of computational chemistry*, 31(7):1561–1563, 2010.
8. Noël Malod-Dognin, Rumen Andonov, and Nicola Yanev. Maximum cliques in protein structure comparison. In *Experimental Algorithms*, pages 106–117. Springer, 2010.
9. Thomas Schank and Dorothea Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *Experimental and Efficient Algorithms*, pages 606–609. Springer, 2005.
10. Stefan Schmitt, Daniel Kuhn, Gerhard Klebe, et al. A new method to detect related function among proteins independent of sequence and fold homology. *Journal of molecular biology*, 323(2):387–406, 2002.
11. S Subbiah, DV Laurents, and M Levitt. Structural similarity of dna-binding domains of bacteriophage repressors and the globin core. *Current Biology*, 3(3):141–148, 1993.
12. Inken Wohlers, Lars Petzold, Francisco Domingues, and Gunnar Klau. Paul: Protein structural alignment using integer linear programming and lagrangian relaxation. *BMC Bioinformatics*, 10(Suppl 13):P2, 2009.
13. 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.
14. Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309, 2005.