Maximum Cliques in Protein Structure Comparison

Noël Malod-Dognin, Rumen Andonov, Nicola Yanev

Maximum Cliques in Protein Structure Comparison

N. Malod-Dognin[1], R. Andonov[2*], and N. Yanev[34]

[1] South-West University "Neofit Rilski", Blagoevgrad, Bulgaria.
[2] INRIA Rennes - Bretagne Atlantique and University of Rennes 1, France.
[3] Faculty of Mathematics and Informatics, University of Sofia, Bulgaria.
[4] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences.
nmaloddg@swu.bg, randonov@irisa.fr, choby@math.bas.bg

Abstract. Computing the similarity between two protein structures is a crucial task in molecular biology, and has been extensively investigated. Many protein structure comparison methods can be modeled as maximum clique problems in specific k-partite graphs, referred here as alignment graphs. In this paper, we propose a new protein structure comparison method based on internal distances (DAST), which main characteristic is that it generates alignments having RMSD smaller than any previously given threshold. DAST is posed as a maximum clique problem in an alignment graph, and in order to compute DAST's alignments, we also design an algorithm (ACF) for solving such maximum clique problems. We compare ACF with one of the fastest clique finder, recently conceived by Östergård. On a popular benchmark (the Skolnick set) we observe that ACF is about 20 times faster in average than the Östergård's algorithm. We then successfully use DAST's alignments to obtain automatic classification in very good agreement with SCOP.

Key words: protein structure comparison, maximum clique problem, k-partite graphs, combinatorial optimization, branch and bound.

1 Introduction

A fruitful assumption in molecular biology is that proteins of similar three-dimensional (3D) structures are likely to share a common function and in most cases derive from a same ancestor. Understanding and computing the protein structures similarities is one of the keys for developing protein based medical treatments, and thus it has been extensively investigated [1, 2]. Evaluating the similarity of two protein structures can be done by finding an optimal (according to some criterions) order-preserving matching (also called alignment) between their components. In this paper, we propose a new protein structure comparison method based on internal distances (DAST). Its main characteristic is to generate alignments having RMSD smaller than any previously given threshold. We show that finding such alignments is equivalent to solving maximum clique problems in specific k-partite graphs referred here as alignment graphs. These graphs could be very large (more than 25000 vertices and 3×10^7 edges) when comparing real proteins. Even very recent general clique finders [3, 4] are oriented to notably

---

* Corresponding author.

smaller instances and are not able to solve problems of such size (the available code of [4] is limited to graphs with up to 1000 vertices).

For solving the maximum clique problem in this context we conceive an algorithm, denoted by *ACF* (for Alignment Clique Finder), which profits from the particular structure of the alignment graphs. We furthermore compare ACF to an efficient general clique solver [5] and the obtained results clearly demonstrate the usefulness of our dedicated algorithm. In addition, we show that the scores obtained by DAST allow to obtain an automatic classification in agreement with SCOP [6]. The main focus here is on designing an algorithm able to generate alignments with guaranteed small RMSD. Evaluating the quality of these alignments and its comparison with other structure alignment methods is beyond the scope of this paper and is a subject of our coming research.

Strickland et al. [7] also exploit the properties of the maximum cliques in protein-based alignment graphs. However, their approach considerably differs from ours: the alignment graphs are defined in a different manner (see section 1.3) and the authors in [7] concentrate on specialized preprocessing techniques in order to accelerate the solution of another optimization problem–Contact Map Overlap Maximization. The maximum cliques instances that are solved in [7] are much smaller than ours.

1.1   The maximum clique problem

We usually denote an undirected graph by $G = (V, E)$, where V is the set of vertices and E is the set of edges. Two vertices i and j are said to be adjacent if they are connected by an edge of E. A clique of a graph is a subset of its vertex set, such that any two vertices in it are adjacent.

Definition 1. *The **maximum clique problem** (also called maximum cardinality clique problem) is to find a largest, in terms of vertices, clique of an arbitrary undirected graph G, which will be denoted by $MCC(G)$.*

The maximum clique problem is one of the first problem shown to be NP-complete [8] and it has been studied extensively in literature. Interested readers can refer to [9] for a detailed state of the art about the maximum clique problem. It can be easily proven that solving this problem in the context of k-partite graphs does not reduce its complexity.

1.2   Alignment graphs

In this paper, we focus on grid alike graphs, which we define as follows.

Definition 2. *A $m \times n$ **alignment graph** $G = (V, E)$ is a graph in which the vertex set V is depicted by a (m-rows) $\times$ (n-columns) array T, where each cell $T[i][k]$ contains at most one vertex i.k from V (note that for both arrays and vertices, the first index stands for the row number, and the second for the column number). Two vertices i.k and j.l can be connected by an edge $(i.k, j.l) \in E$ only if $i < j$ and $k < l$. An example of such alignment graph is given in Fig 2a.*

It is easily seen that the m rows form a m-partition of the alignment graph G, and that the n columns also form a n-partition. In the rest of this paper we will use the following

notations. A successor of a vertex $i.k \in V$ is an element of the set $\Gamma^+(i.k) = \{j.l \in V$ s.t. $(i.k, j.l) \in E, i < j$ and $k < l\}$. $V^{i.k}$ is the subset of V restricted to vertices in rows $j, i \leq j \leq m$, and in columns $l, k \leq l \leq n$. Note that $\Gamma^+(i.k) \subset V^{i+1.k+1}$. $G^{i.k}$ is the subgraph of G induced by the vertices in $V^{i.k}$. The cardinality of a vertex set U is $|U|$.

1.3 Relations with protein structure similarity

In graph-theoretic language, two proteins P_1 and P_2 can be represented by two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ where the sets of vertices V_1 and V_2 stand for residues/SSE, while edges depict contacts/relationships between them. The similarity between P_1 and P_2 can be estimated by finding the longest alignment between the elements of V_1 and V_2. In our approach, this is modeled by an alignment graph $G = (V, E)$ of size $|V_1| \times |V_2|$, where each row corresponds to an element of V_1 and each column corresponds to an element of V_2. A vertex $i.k$ is in V (i.e. matching $i \leftrightarrow k$ is possible), only if elements $i \in V_1$ and $k \in V_2$ are compatible. An edge $(i.k, j.l)$ is in E if and only if : (i) $i < j$ and $k < l$, for order preserving, and (ii) matching $i \leftrightarrow k$ is compatible with matching $j \leftrightarrow l$. A feasible alignment of P_1 and P_2 is then a clique in G, and the longest alignment corresponds to a maximum clique in G.

At least two protein structure similarity related problems from the literature can be converted into clique problems in alignment graphs : the secondary structure alignment in VAST[10], and the Contact Map Overlap Maximization problem (CMO)[11].

VAST, or Vector Alignment Search Tool, is a software for aligning protein 3D structures largely used in the National Center for Biotechnology Information [5]. In VAST, V_1 and V_2 contain 3D vectors representing the secondary structure elements (SSE) of P_1 and P_2. Matching $i \leftrightarrow k$ is possible if vectors i and k have similar norms and correspond either both to α-helices or both to β-strands. Finally, matching $i \leftrightarrow k$ is compatible with matching $j \leftrightarrow l$ only if the couple of vectors (i, j) from P_1 can be well superimposed in 3D-space with the couple of vectors (k, l) from P_2.

CMO is one of the most reliable and robust measures of protein structure similarity. Comparisons are done by aligning the residues (amino-acids) of two proteins in a way that maximizes the number of common contacts (when two residues that are close in 3D space are matched with two residues that are also close in 3D space). We have already dealt with CMO, but not using cliques [12]. The above definition of the alignment graph is inspired by the one we used and proved to be very successful in the case of CMO. There is a multitude of other alignment methods and they differ mainly by the nature of the elements of V_1 and V_2, and by the compatibility definitions between elements and between pairs of matched elements. One essential difference between our approach and the one used in [7] resides in the definition itself of the alignment graph. Every vertex in the so-called specially defined graph from [7] corresponds to an overlap of an edge/contact from P_1 with an edge/contact from P_2 and hence the graph size is $|E_1| \times |E_2|$, versus $|V_1| \times |V_2|$ in our definition.

---

[5] http://www.ncbi.nlm.nih.gov/Structure/VAST/vast.shtml

1.4 DAST: an improvement of CMO based on internal distances

The objective in CMO is to maximize the number of common contacts. It has been shown that this objective finds a good global similarity score which can be successfully used for classification of structures [13, 12]. However, such a strategy also introduces some "errors" in the structure based alignment–like aligning two residues that are close in 3D space with two residues that are remote, as illustrated in Fig 1. These errors could potentially yield alignments with big root mean square deviations (RMSD) which is not desirable for structures comparison. To avoid such problems we propose DAST (Distance-based Alignment Search Tool)–an alignment method based on internal distances which is modeled in an alignment graph. In DAST, two proteins P_1 and P_2 are represented by their ordered sets of residues V_1 and V_2. Two residues $i \in V_1$ and $k \in V_2$ are compatible if they come from the same kind of secondary structure elements (i.e. i and k both come from α-helices, or from β-strands) or if both come from loops. Let us denote by d_{ij} (resp. $d_{k.l}$) the euclidean distance between the α-carbons of residues i and j (resp. k and l). Matching $i \leftrightarrow k$ is compatible with matching $j \leftrightarrow l$ only if $|d_{ij} - d_{kl}| \leq \tau$, where τ is a distance threshold. The longest alignment in terms of residues, in which each couple of residues from P_1 is matched with a couple of residues from P_2 having similar distance relations, corresponds to a maximum clique in the alignment graph G. For example the clique $(2.1), (3.2), (4.3)$ in Fig 2a is generated by aligning residues $2, 3, 4$ from P_1 (rows) with residues $1, 2, 3$ from P_2 (columns).

Fig. 1. An optimal CMO matching.
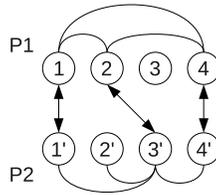
Two proteins (P_1 and P_2) are represented by their contact map graphs where the vertices corresponds to the residues and where edges connect residues in contacts (i.e. close). The matching "$1 \leftrightarrow 1', 2 \leftrightarrow 3', 4 \leftrightarrow 4'$", represented by the arrows, yields two common contacts which is the maximum for the considered case. However, it also matches residues 1 and 4 from P_1 which are in contacts with residues $1'$ and $4'$ in P_2 which are remote.

Given a set of n deviations $S = \{s_1, s_2, \ldots, s_n\}$, its Root Mean Square Deviation ($RMSD$) is : $RMSD(S) = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} s_i^2}$. For assessing the quality of an alignment, the biologists use two different $RMSD$ measures which differ on the deviations they take into account. The first one is the $RMSD$ of superimposed coordinates ($RMSD_c$). After superimposing the two protein structures, the measured deviations are the euclidean distances between the matched amino-acid d_{ik}, for all matching pairs $i \leftrightarrow k$. The second one is the $RMSD$ of internal distances ($RMSD_d$). The measured deviations are $|d_{ij} - d_{kl}|$,

for all couples of matching pairs "$i \leftrightarrow k, j \leftrightarrow l$". Let us denote by P the later set and by N_m its cardinality. We therefore have that $RMSD_d = \sqrt{\frac{1}{N_m} \times \sum_{(ij,jk) \in P} (|d_{ij} - d_{kl}|^2)}$ and since $|d_{ij} - d_{kl}| \leq \tau$ holds for all matching pairs "$i \leftrightarrow k, j \leftrightarrow l$", the alignments generated by DAST are characterized by the desired property $RMSD_d \leq \tau$.

2   Branch and Bound approach

We have been inspired by [5] to propose our own algorithm which is more suitable for solving the maximum clique problem in the previously defined $m \times n$ alignment graph $G = (V, E)$. Let *Best* be the biggest clique found so far (first it is set to $\emptyset$), and $|\overline{MCC}(G)|$ be an over-estimation of $|MCC(G)|$. By definition, $V^{i+1.k+1} \subset V^{i.k+1} \subset V^{i.k}$, and similarly $V^{i+1.k+1} \subset V^{i+1.k} \subset V^{i.k}$. >From these inclusions and from definition2, it is easily seen that for any $G^{i.k}$, $MCC(G^{i.k})$ is the biggest clique among $MCC(G^{i+1.k})$, $MCC(G^{i.k+1})$ and $MCC(G^{i+1.k+1}) \bigcup \{i.k\}$, but for the latter only if vertex $i.k$ is adjacent to all vertices in $MCC(G^{i+1.k+1})$. Let C be a $(m+1) \times (n+1)$ array where $C[i][k] = |\overline{MCC}(G^{i.k})|$ (values in row $m+1$ or column $n+1$ are equal to 0). For reasoning purpose, let assume that the upper-bounds in C are exact. If a vertex $i.k$ is adjacent to all vertices in $MCC(G^{i+1.k+1})$, then $C[i][k] = 1 + C[i+1][k+1]$, else $C[i][k] = \max(C[i][k+1], C[i+1][k])$. We can deduce that a vertex $i.k$ cannot be in a clique in $G^{i.k}$ which is bigger than *Best* if $C[i+1][k+1] < |Best|$, and this reasoning still holds if values in C are upper estimations. Another important inclusion is $\Gamma^+(i.k) \subset V^{i+1.k+1}$. Even if $C[i+1][k+1] \geq |Best|$, if $|\overline{MCC}(\Gamma^+(i.k))| < |Best|$ then $i.k$ cannot be in a clique in $G^{i.k}$ bigger than *Best*.

Fig. 2. A 4×4 alignment graph and the visiting order of its array T

A) An exemple of 4x4 alignment graph.

B) Visiting order or array T.
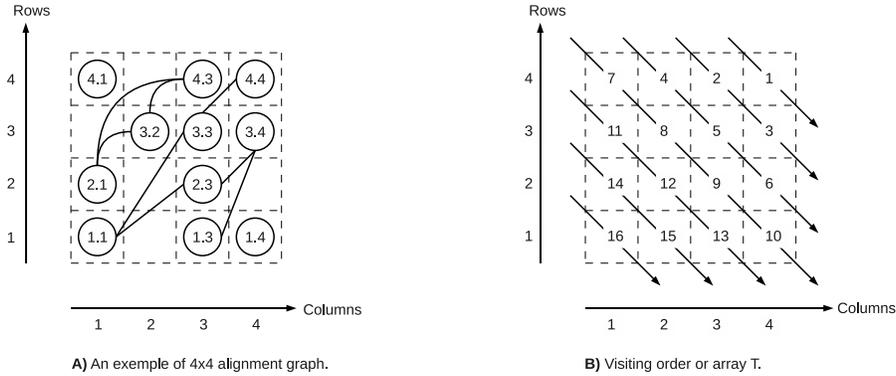
Our main clique cardinality estimator is constructed and used according to these properties. A function, Find_clique(G), will visit the cells of T according to north-west to south-est diagonals, from diagonal "$i + k = m + n$" to diagonal "$i + k = 2$"

as illustrated in Fig 2b. For each cell $T[i][k]$ containing a vertex $i.k \in V$, it may call Extend_clique($\{i.k\}$, $\Gamma^+(i.k)$), a function which tries to extend the clique $\{i.k\}$ with vertices in $\Gamma^+(i.k)$ in order to obtain a clique bigger than *Best* (which cannot be bigger than |Best| +1). If such a clique is found, *Best* is updated. However, Find_clique() will call Extend_clique() only if two conditions are satisfied : (i) $C[i+1][k+1] = |Best|$ and (ii) $|\overline{MCC}(\Gamma^+(i.k))| \geq |Best|$. After the call to Extend_clique(), $C[i][k]$ is set to $|Best|$. For all other cells $T[i][k]$, $C[i][k]$ is set to $\max(C[i][k+1], C[i+1][k])$ if $i.k \notin V$, or to $1 + C[i+1][k+1])$ if $i.k \in V$. Note that the order used for visiting the cells in T guaranties that when computing the value of $C[i][k]$, the values of $C[i+1][k]$, $C[i][k+1]$ and $C[i+1][k+1]$ are already computed.

Array C can also be used in function Extend_clique() to fasten the maximum clique search. This function is a branch a bound (B&B) search using the following branching rules. Each node of the B&B tree is characterized by a couple (*Cli*, *Cand*) where *Cli* is the clique under construction and *Cand* is the set of candidate vertices to be added to *Cli*. Each call to Extend_clique($\{i.k\}$, $\Gamma^+(i.k)$) create a new B&B tree which root node is ($\{i.k\}$, $\Gamma^+(i.k)$). The successors of a B&B node (*Cli*,*Cand*) are the nodes $(Cli \bigcup \{i'.k'\}, Cand \bigcap \Gamma^+(i'.k'))$, for all vertices $i'.k' \in Cand$. Branching follows lexicographic increasing order (row first). According to the branching rules, for any given B&B node (*Cli*, *Cand*) the following cutting rules holds : (i) if $|Cli| + |Cand| \leq |Best|$ then the current branch cannot lead to a clique bigger than |Best| and can be fathomed, (ii) if $|\overline{MCC}(Cand)| \leq |Best| - |Cli|$, then the current branch cannot lead to a clique bigger than |Best|, and (iii) if $|\overline{MCC}(Cand \bigcap \Gamma^+(i.k))| \leq |Best| - |Cli| - 1$, then branching on $i.k$ cannot lead to a clique bigger than |Best|. For any set *Cand* and any vertex $i.k$, $Cand \bigcap \Gamma^+(i.k) \subset \Gamma^+(i.k)$, and $\Gamma^+(i.k) \subset G^{i+1.k+1}$. From these inclusions we can deduce two way of over-estimating $|MCC(Cand \bigcap \Gamma^+(i.k))|$. First, by using $C[i+1][k+1]$ which over-estimate $|MCC(G^{i+1.k+1})|$ and second, by over-estimating $|MCC(\Gamma^+(i.k))|$. All values $|\overline{MCC}(\Gamma^+(i.k))|$ are computed once for all in Find_clique() and thus, only $|\overline{MCC}(Cand)|$ needs to be computed in each B&B node.

3 Maximum clique cardinality estimators

Even if the described functions depend on array C, they also use another upper-estimator of the cardinality of a maximum clique in an alignment graph. By using the properties of alignment graphs, we developed the following estimators.

3.1 Minimum number of rows and columns

Definition 2 implies that there is no edge between vertices from the same row or the same column. This means that in a $m \times n$ alignment graph, $|MCC(G)| \leq \min(m,n)$. If the numbers of rows and columns are not computed at the creation of the alignment graph, they can be computed in $O(|V|)$.

3.2 Longest increasing subset of vertices

Definition 3. *An **increasing subset of vertices** in an alignment graph $G = \{V, E\}$ is an ordered subset $\{i_1.k_1, i_2.k_2, \ldots, i_t.k_t\}$ of V, such that $\forall j \in [1, t-1]$, $i_j < i_{j+1}$, $k_j < k_{j+1}$. $LIS(G)$ is the longest, in terms of vertices, increasing subset of vertices of G.*

Since any two vertices in a clique are adjacent, definition 2 implies that a clique in G is an increasing subset of vertices. However, an increasing subset of vertices is not necessarily a clique (since vertices are not necessarily adjacent), and thus $|MCC(G)| \leq |LIS(G)|$. In a $m \times n$ alignment graph $G = (V, E)$, $LIS(G)$ can be computed in $O(n \times m)$ times by dynamic programming. However, it is possible by using the longest increasing subsequence to solve $LIS(G)$ in $O(|V| \times \ln(|V|))$ times which is more suited in the case of sparse graph like in our protein structure comparison experiments.

Definition 4. *The **longest increasing subsequence** of an arbitrary finite sequence of integers $S = \text{``}i_i, i_2, \ldots, i_n\text{''}$ is the longest subsequence $S' = \text{``}i'_i, i'_2, \ldots, i'_t\text{''}$ of S respecting the original order of S, and such that for all $j \in [1, t], i'_j < i'_{j+1}$. By example, the longest increasing subsequence of "1,5,2,3" is "1,2,3".*

For any given alignment graph $G = \{V, E\}$, we can easily reorder the vertex set V, first by increasing order of columns, and second by decreasing order of rows. Let's denote by V' this reordered vertex set. Then we can create an integer sequence S corresponding to the row indices of vertices in V'. For example, by using the alignment graph presented in Fig2a, the reordered vertex set V' is {4.1, 2.1, 1.1, 3.2, 4.3, 3.3, 2.3, 1.3, 4.4, 3.4, 1.4}, and the corresponding sequence of row indices S is "4, 2, 1, 3, 4, 3, 2, 1, 4, 3, 1". An increasing subsequence of S will pick at most one number from a column, and thus an increasing subsequence is longest if and only if it covers a maximal number of increasing rows. This proves that solving the longest increasing subsequence in S is equivalent to solving the longest increasing subset of vertices in G. Note that the longest increasing subsequence problem is solvable in time $O(l \times \ln(l))$ [14], where l denotes the length of the input sequence. In our case, this corresponds to $O(|V| \times \ln(|V|))$.

3.3 Longest increasing path

Definition 5. *An **increasing path** in an alignment $G = \{V, E\}$ is an increasing subset of vertex $\{i_1.k_1, i_2.k_2, \ldots, i_t.k_t\}$ such that $\forall j \in [1, t-1]$, $(i_j.k_j, i_{j+1}.k_{j+1}) \in E$. The longest increasing path in G is denoted by $LIP(G)$*

As the increasing path take into account edges between consecutive vertices, $|LIP(G)|$, should better estimate $MCC(G)|$. $|LIP(G)|$ can be computed in $O(|V|^2)$ by the following recurrence. Let $DP[i][k]$ be the length of the longest increasing path in $G^{i.k}$ containing vertex $i.k$. $DP[i][k] = 1 + \max_{i'.k' \in \Gamma^+ i.k}(DP[i'][k'])$. The sum over all $\Gamma^+(i.k))$ is done in $O(|E|)$ time complexity, and finding the maximum over all $DP[i][k]$ is done in $O(|V|)$. This results in a $O(|V| + |E|)$ time complexity for computing $|LIP(G)|$.

Any of the previously defined estimators can be used as bound generator in our B&B, and without them our algorithm is about 2.21 times slower than the Östergård's

one. Experimentally, the longest increasing subset of vertices (solved using the longest increasing subsequence) exhibits the best performances, allowing our algorithm to be about 20 times faster than the Östergård's one, and is the bound generator that we used for obtaining the optimal alignments presented in the next section.

4   Results

All results presented in this section come from real protein structure comparison instances. Our algorithm, denoted by *ACF* (for Alignment Clique Finder), has been implemented in C and was tested in the context of DAST. *ACF* will be compared to the fast clique finder (denoted by here *Östergård*) which has been proposed in [5] and which code is publicly available.

4.1   Residues alignment

In this section we compare *ACF* to *Östergård* in the context of residue alignments in DAST. Computations were done on a PC with an Intel Core2 processor at 3Ghz, and for both algorithms the computation time was bounded to 5 hours per instance. Secondary structures assignments were done by KAKSI [15], and the threshold distance τ was set to 3Å. The protein structures come from the well known Skolnick set, described in [16]. It contains 40 protein chains having from 90 to 256 residues, classified in SCOP[6] (v1.73) into five families. Amongst the 780 corresponding alignment instances, 164 align protein chains from the same family and will be called "similar". The 616 other instances align protein chains from different families and thus will be called "dissimilar". Characteristics of the corresponding alignment graphs are presented in **table 1**.

Table 1. DAST alignment graphs characteristics

		array size	|V|	|E|	density	|MCC|
similar	min	97×97	4018	106373	8.32%	45
instances	max	256×255	25706	31726150	15.44%	233
dissimilar	min	97×104	1581	77164	5.76%	12
instances	max	256×191	21244	16839653	14.13%	48

All alignment graphs from DAST have small edge density (less than 16%). Similar instances are characterized by bigger maximum cliques than the dissimilar instances.

Table 2 compares the number of instances solved by each algorithm on Skolnick set. Note that when an instance is solved, the B&B algorithm finds both the optimal score (maximum clique cardinality), as well as the corresponding residues alignment. *ACF* solved 155 from 164 similar instances, while *Östergård* solved 128 instances. *ACF* was able to solve all 616 dissimilar instances, while *Östergård* solved 545 instances only. Thus, on this popular benchmark set, *ACF* clearly outperforms *Östergård* in terms of number of solved instances.

Table 2. Number of solved instances comparison

	Östergård	ACF
Similar instances (164)	128	**155**
Dissimilar instances (616)	545	**616**
Total (780)	673	**771**

On the Skolnick set *ACF* solves 21% more similar instances and 13% more dissimilar instances than *Östergård* when the running time was upper-bounded by 5 hours per instance.

Figure 3 compares the running time of *ACF* to the one of *Östergård* on the set of 673 instances solved by both algorithms (all instances solved by *Östergård* were also solved by *ACF*). For all but one instances, *ACF* is significantly faster than *Östergård*. More precisely, *ACF* needed 12 hs. 29 min. 56 sec. to solve all these 673 instances, while *Östergård* needed 260 hs. 10 min. 10 sec. Thus, on the Skolnick set, *ACF* is about 20 times faster in average than *Östergård*, (up to 4029 times for some instances).

Fig. 3. Running time comparison on Skolnick set
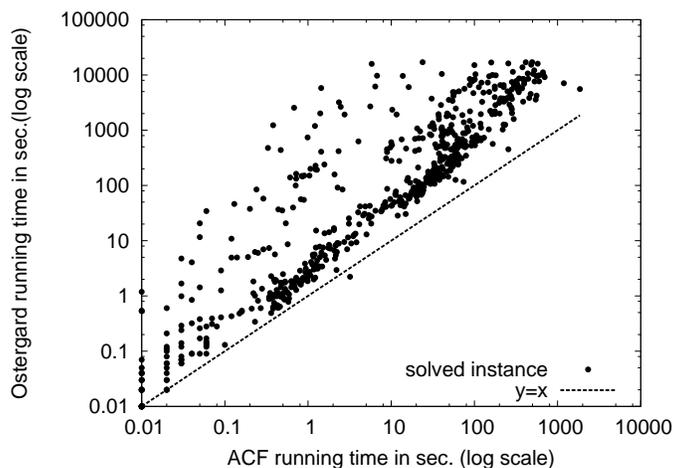
ACF versus *Östergård* running time comparison on the set of the 673 Skolnick instances solved by both algorithms. The *ACF* time is presented on the x-axis, while the one of *Östergård* is on the y-axis. For all instances except one, *ACF* is faster than *Östergård*.

4.2 Comparison between DAST's and CMO's alignments

In order to compare the alignments of DAST to the ones of CMO[12], we extracted from the Skolnick set 10 instances that are optimally solved by both methods (see table 3). The five "similar" instances compare protein structures coming from the same

SCOP family, while the five "dissimilar" instances compare protein structures coming from different SCOP families. The distance threshold of DAST was set to 3 Å(which corresponds to the desired $RMSD_d$ of alignments), while the contact threshold of CMO was set to 7.5 Å(optimal value according to [13]).

Table 3 compares the obtained alignments, both in terms of length (percentage of aligned amino-acids) and in terms of $RMSD_d$. The alignments of CMO for similar proteins are very good : they are both long and possess small $RMSD_d$ values. However, for dissimilar proteins, the alignments of CMO possess very bad $RMSD_d$ values, which means that they do not correspond to common substructures. On the other hand, for both similar and dissimilar proteins, the alignments of DAST always possess small $RMSD_d$ values (smaller than the perviously fixed threshold). DAST's alignments are shorter than the ones of CMO, but their lengths better reflect the similarity between two proteins, since the alignments between similar proteins are always much longer than the alignments between dissimilar proteins. Note that this property does not hold for CMO's alignments.

Table 3. CMO vs DAST alignments

	Instance	Length (AA %)		$RMSD_d$ (Å)	
		CMO	DAST	CMO	DAST
similar instances	1amkA–1aw2A	**97.4 %**	78.9 %	1.39	**0.68**
	1amkA–1htiA	**99.0 %**	81.8 %	1.24	**0.74**
	1qmpA–1qmpB	**99.2 %**	90.8 %	**0.22**	**0.22**
	1ninA–1plaA	**96.0 %**	57.4 %	1.42	**0.96**
	1tmhA–1treA	**99.8 %**	91.6 %	0.90	**0.44**
dissimilar instances	1amkA–1b00A	**63.5 %**	21.7 %	5.62	**1.23**
	1amkA–1dpsA	**78.0 %**	15.3 %	13.01	**1.06**
	1b9bA–1dbwA	**68.3 %**	24.4 %	6.02	**1.11**
	1qmpA–2pltA	**83.3 %**	15.0 %	7.36	**1.18**
	1rn1A–1b71A	**70.5 %**	17.6 %	11.22	**0.82**

Similar instances compare proteins coming from the same SCOP family, while dissimilar instances compare proteins coming from different SCOP families. The distance threshold of DAST was set to 3 Å, while the contact threshold of CMO was set to 7.5 Å. Columns 3 and 4 compare the length of the alignments (in percentage of aligned amino-acids), while columns 5 and 6 compare the $RMSD_d$ of the alignments. DAST's alignments always possess good (small) $RMSD_d$ values, but are shorter than CMO's ones.

4.3 Automatic classification

In this section, we test the possibility to obtain good automatic classifications based on DAST's alignments. For this purpose we used the following protocol : on the Skolnick set, the runs of DAST were limited to 5 hours per instance. The similarity score between two proteins P_1 and P_2 (having respectively $|V_1|$ and $|V_2|$ amino-acids) was defined as $SIM(P_1,P_2) = \dfrac{2 \times N_m}{|V_1| + |V_2|}$, where N_m is the number of aligned amino-acids (i.e. the size

of the biggest clique found by DAST). These scores were given to CHAVL [17], an unsupervised ascendant classification tool based on likelihood maximization, and the obtained classification was compared to SCOP classification [6], which is a curated classification of the protein structures.

Table 4 presents the obtained classification. It is very similar to the one of SCOP, except that the protein chain "1ntrA" is not classified with the other members of its SCOP family. We detected that this error was provoked by Kaksi's secondary structure assignment of 1ntrA, which is not in agreement with the one used in SCOP.

Table 4. DAST classification of the Skolnick set

DAST class	SCOP Family	Proteins
1	CheY-related	1b00A, 1dbwA, 1natA, 3chyA 1qmp(A,B,C,D), 4tmy(A,B)
2	CheY-related	1ntrA
3	Plastocyanin /azurin-like	1bawA, 1byo(A,B), 1kdiA, 1ninA 1plaA, 2b3iA, 2pcyA, 2pltA
4	Triosephosphate isomerase (TIM)	1amkA, 1aw2A, 1b9bA, 1btmA, 1htiA 1tmhA, 1treA, 1triA, 1ydvA, 3ypiA, 8timA
5	Ferritin	1b71A, 1bcfA, 1dpsA, 1fhaA, 1ierA, 1rcdA
6	Fungal ribonucleases	1rn1(A,B,C)

The classification returned by CHAVL based on similarity score found by DAST, is very similar to the SCOP classification, except for the protein chain 1ntrA (class 2) which is not recognized as a CheY-related protein.

5 Conclusion and future work

In this paper we introduce a novel protein structure comparison approach DAST, for Distance-based Alignment Search Tool. For any fixed threshold τ, it finds the longest alignment in which each couple of pairs of matched residues shares the same distance relation (+/- τ), and thus the RMSD of the alignment is $\leq \tau$. This property is not guaranteed by the CMO approach, which inspired initially DAST. From computation standpoint, DAST requires solving the maximum clique problem in a specific k-partite graph. By exploiting the peculiar structure of this graph, we design a new maximum clique solver which significantly outperforms one of the best general maximum clique solver. Our solver was successfully integrated into DAST and will be freely available soon. We are currently studying the quality of DAST alignments from practical viewpoint and compare the obtained results with other structure comparison methods.

Acknowledgments

References

1. Godzik, A.: The structural alignment between two proteins: Is there a unique answer? Protein Science (7) (1996) 1325–1338
2. Sierk, M., Kleywegt, G.: Déjà vu all over again: Finding and analyzing protein structure similarities. Structure **12**(12) (2004) 2103–2111
3. Konc, J., Janezic, D.: An efficient branch-and-bound algorithm for finding a maximum clique. Discrete Mathematics and Theoretical Computer Science. **58** (2003) 220
4. Tomita, E., Seki, T.: An improved branch and bound algorithm for the maximum clique problem. Communications in Mathematical and in Computer Chemistry / MATCH. **58** (2007) 569–590
5. Östergård, P.R.J.: A fast algorithm for the maximum clique problem. Discrete Applied Mathematics. **120**(1-3) (2002) 197–207
6. Andreeva, A., Howorth, D., Chandonia, J.M., Brenner, S., Hubbard, T., Chothia, C., Murzin., A.: Data growth and its impact on the SCOP database: new developments. Nucl. Acids Res. **36** (11 2007) 419–425
7. Strickland, D., Barnes, E., Sokol, J.: Optimal protein structure alignment using maximum cliques. Oper. Res. **53**(3) (2005) 389–402
8. Karp, R.: Reducibility among combinatorial problems. Complexity of Computer Computations. **6** (06 1972) 85–103
9. Bomze, I., Budinich, M., Pardalos, P., Pelillo, M.: The maximum clique problem. Handbook of Combinatorial Optimization. (1999)
10. Gibrat, J.F., Madej, T., Bryant, S.: Surprising similarities in structure comparison. Current Opinion in Structural Biology. **6** (06 1996) 377–385
11. Godzik, A., Skolnick, J.: Flexible algorithm for direct multiple alignment of protein structures and sequences. CABIOS **10** (1994) 587–596
12. Andonov, R., Yanev, N., Malod-Dognin, N.: An efficient lagrangian relaxation for the contact map overlap problem. In: WABI '08: Proceedings of the 8th international workshop on Algorithms in Bioinformatics, Springer-Verlag (2008) 162–173
13. Caprara, A., Carr, R., Israil, S., Lancia, G., Walenz, B.: 1001 optimal PDB structure alignments: integer programming methods for finding the maximum contact map overlap. J. Comput. Biol. **11**(1) (2004) 27–52
14. Fredman., M.: On computing the length of longest increasing subsequences. Discrete Mathematics. **11** (1 1975) 29–35
15. Martin, J., Letellier, G., Marin, A., Taly, J.F., de Brevern, A., Gibrat., J.F.: Protein secondary structure assignment revisited: a detailed analysis of different assignment methods. BMC Structural Biology. **5** (2005) 17
16. Lancia, G., Carr, R., Walenz, B., Istrail, S.: 101 optimal pdb structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In: RECOMB '01: Proceedings of the fifth annual international conference on Computational biology. (2001) 193–202
17. Lerman, I.: Likelihood linkage analysis (lla) classification method (around an example treated by hand). Biochimie **75**(5) (1993) 379–397