Using Dominances for Solving the Protein Family Identification Problem

Noël Malod-Dognin, Mathilde Le Boudic-Jamin, Pritish Kamath, Rumen Andonov

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Using Dominances for Solving the Protein Family Identification Problem

Noël Malod-Dognin — Mathilde Le Boudic-Jamin — Pritish Kamath — Rumen Andonov

N° 7688

Juillet 2011

_ Computational Biology and Bioinformatics _

apport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7688--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Using Dominances for Solving the Protein Family Identification Problem

Noël Malod-Dognin*, Mathilde Le Boudic-Jamin†, Pritish Kamath‡, Rumen Andonov†

Theme : Computational Biology and Bioinformatics
Computational Sciences for Biology, Medicine and the Environment
Équipes-Projets ABS et Symbiose

Rapport de recherche n° 7688 — Juillet 2011 — 15 pages

Abstract: Identification of protein families is a computational biology challenge that needs efficient and reliable methods. Here we introduce the concept of dominance and propose a novel combined approach based on Distance Alignment Search Tool (DAST), which contains an exact algorithm with bounds. Our experiments show that this method successfully finds the most similar proteins in a set without solving all instances.

Key-words: Protein structure comparison, classification, bounds, dominance

* A.B.S., INRIA Sophia Antipolis - Méditerranée, France.
† Symbiose, INRIA Rennes - Bretagne Atlantique, France.
‡ Computer Science and Engineering Department, Indian Institute of Technology, India.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Utilisation de la dominance pour l'identification de familles protéiques

Résumé : L'identification des familles protéique est un challenge de la biologie computationnelle qui nécessite des méthodes efficaces et robustes. Nous introduisons ici le concept de dominance entre instance de comparaison de structures prot'eiques, et proposons une nouvelle approche basée sur DAST (Distance Alignment Search Tool), un algorithme exact auquel nous rajoutons des bornes. Les résultats obtenus montrent que notre méthode résout correctement le problème de l'identification des familles protéique sans avoir besoin de résoudre toutes les instances de comparaison de structures.

Mots-clés : Comparaison de structure protéique, classification, bornes, dominance

1 Introduction

The 3D structure of macro-molecules underpins all biological functions. Similarities between protein structures may come from evolutionary relationships [18, 12], and similar protein structures relate to similar functions. Thus, the protein structure comparison is a key tool in structural biology, whose primary goal is to understand function through structure. During the last decades, many protein structure comparison approaches have been proposed, each aiming at quantifying the intuitive notion of structural similarity. Most of the proposed methods are either based on optimal rigid-body superimposition (like VAST[7] or STRUCTAL[6]), whose computation is based on the least Root Mean Square Deviation of residue coordinates ($RMSDc$) as first defined by Kabsch[10], or on the comparison of the internal distances between the residues (like DALI[9]), CMO[8] or DAST[14]). The main challenge in protein structure comparison is to design efficient algorithms, since the comparison of two protein structures is often NP-complete, as first shown in [13].

1.1 DAST

DAST (Distance-based Alignment Search Tool) is a protein structure comparison method based on internal distances [14]. In DAST, two proteins p_1 and p_2 are represented by their ordered sets of residues N_1 and N_2. The matching between residues $i \in N_1$ and $k \in N_2$, denoted by $i \leftrightarrow k$, is allowed only if i and k come from the same kind of secondary structure (i.e. if either i and k both come from α-helices, or both come from β-strands, or both come from loops). By assumption, for any pair of residues i and j from the same protein we know the euclidean distance between their α-carbons (denoted here by d_{ij}).

Definition 1. *Pair (i, j) from p_1 is compatible with pair (k, l) from p_2 if and only if: (1) $i < j$ and $k < l$ (order preserving) and ; (2) $|d_{ij} - d_{kl}| \leq \tau$, where τ is a given distance threshold (isometric relationship).*

If (i, j) is compatible with (k, l) we are allowed to match (align) $i \leftrightarrow k$ and $j \leftrightarrow l$ (i.e. they form a matching pair). An optimal structural alignment is given by the longest sequence of matching pairs "$i_1 \leftrightarrow k_1$, $i_2 \leftrightarrow k_2$, ..., $i_t \leftrightarrow k_t$" in which any two matching pairs are compatible. As shown in section 3.2, DAST is equivalent to solving a maximum clique problem and is thus is an NP-Complete problem [11]. Set ncr (number of common residues) to denote the length of the optimal alignment between p_1 and p_2. Two similarity scores can be used:

$$S^G_{DAST}(p_1, p_2) = \frac{2 \times ncr}{|N_1| + |N_2|} \quad \text{and} \quad S^L_{DAST}(p_1, p_2) = \frac{ncr}{\min(|N_1|, |N_2|)}. \quad (1)$$

The first score, $S^G_{DAST}(p_1, p_2)$, is normalized according to the mean of two proteins and is oriented to detect global similarity between p_1 and p_2. The second one, $S^L_{DAST}(p_1, p_2)$, is normalized according to the smallest between the two proteins and is more suitable to detect local similarity.

Conceptually, DAST is inspired from the CMO approach which has been largely studied in the literature [8, 4, 1]. Both approaches aim to maximize the number of compatible matching pairs. However, the CMO compatibility does not consider the isometric condition from definition 1. As a consequence, the

underlying optimization problems, the behavior of the corresponding solvers, as well as the characteristics of the provided alignments differ in both approaches. The most interesting feature of DAST is that it always returns an alignment (matching) of good quality, i.e. having a Root Mean Squared Deviation of internal distances ($RMSD_d$) which is less or equal than the given threshold τ (see [14]). This property is not guaranteed in CMO. On the other hand, a very efficient exact algorithm for finding the longest augmented path (the optimization problems in CMO) has been recently proposed in [1]. The associate solver, A_purva, is significantly faster than the nowadays maximum cardinality clique solvers. These issues will be discussed and illustrated in the computational results section.

1.2 The protein family identification problem

The exponential growth of the number of known protein structures in the Protein Data Bank [3] over the past decade led to the problem of protein classification. We mean here how to automatically insert new protein structures into an already existing classified database such as CATH[17] or SCOP[2]. The problem of determining in which classes new structures belong, referred here as the Protein Family Identification Problem , can be defined as follows.

Definition 2. *Given a set of to-be-classified query protein structures* $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$*, a set of classified target protein structures* $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$*, and a protein structure similarity function* $S : \mathcal{Q} \times \mathcal{P} \rightarrow \mathcal{R}^+$*, the* ***Protein Family Identification Problem (FIP)*** *consists in classifying each query structure* $q_i \in \mathcal{Q}$ *in the class of it's nearest neighbor* NN_i *which is defined as* $NN_i = \underset{p_j \in \mathcal{P}}{\operatorname{argmax}} S(q_i, p_j)$.

There are computational pitfalls in the FIP . The number of similarity scores $S(q_i, p_j)$ that need to be computed is $|\mathcal{Q}| \times |\mathcal{P}|$, where $|\mathcal{P}|$ can be very large (there are currently 152920 classified protein structures in the expert classification CATH). Moreover, computing a single similarity score is often equivalent to solving a NP-hard problem (ex: DALI, DAST, CMO, VAST, etc...). Depending on how these NP-hard problems are solved, two cases are possible. First, if the solver is a heuristic (ex: DALI, VAST), then the similarity scores are only approximated, and thus the resulting classification is not optimal (according to the similarity function). Second, if the solver is exact (ex CMO, DAST), and because of the NP-hardness of the problem, some instances cannot be optimally solved within reasonable time, leading to either sub-optimal or to missing similarity scores, both implying that the obtained classification is not optimal, or cannot be computed if too many similarity scores are missing.

In this paper, we propose a notion of dominance between the protein structure comparison instances that allows the computation of optimal protein structure classifications without optimally solving all the comparison instances, and thus reduces the effect of the NP-Hardness of the similarity score. As presented in section 2, using dominance supposes to compute both upper and lower bound on the similarity score. In section 3.3, we propose an efficient bounding strategy for DAST.

2 Dominance

The idea behind the dominance is that in FIP problem, given a query protein structure $q \in \mathcal{Q}$ and a classified set of target protein structures $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, we are interested only in finding the nearest neighbor of q in $\mathcal{P}$, i.e. $NN_q = \underset{p_j \in \mathcal{P}}{\operatorname{argmax}} S(q, p_j)$. If a protein structure p_j can be proved not to be NN_q before $S(q, p_j)$ is optimally computed, then spending more time on proceeding $S(q, p_j)$ is useless.

Let us suppose that the solving process can be stopped with a time limit t, and can then return both a lower-bound $S_{min}(q, p_i)$ and an upper-bound $S_{max}(q, p_i)$ on the similarity score $S(q, p_i)$, i.e $S_{min}(q, p_i) \leq S(q, p_i) \leq S_{max}(q, p_i)$ (if the instance is optimally solved, then $S_{min}(q, p_i) = S(q, p_i) = S_{max}(q, p_i)$).

Definition 3. *Given a query $q \in \mathcal{Q}$ and two target protein structures $p_1 \in \mathcal{P}$ and $p_2 \in \mathcal{P}$, the instance (q, p_2)* ***dominates*** *the instance (q, p_1) if $S_{min}(q, p_2) \geq S_{max}(q, p_1)$.*

If the instance (q, p_2) dominates the instance (q, p_1), then $S(q, p_2) \geq S(q, p_1)$. Thus, p_1 is not the nearest neighbor of q and there is no need to continue computing $S(q, p_1)$. Moreover, if one instance (q, p_i) dominates all the other instances (q, p_j), $p_j \in \mathcal{P}$, then $NN_q = p_i$, and the entire procedure can be stopped (including the instance (q, p_i)).

From now on, we use the dominances to fasten the FIP as follows.

1. All instances (q_i, p_j), $q_i \in \mathcal{Q}$, $p_j \in \mathcal{P}$ are put in a queue, and a time limit argument t is set to a small value.

2. For each instances (q_i, p_j) in the queue, the similarity $S(q_i, p_j)$ is evaluated (by computing $S_{min}(q_i, p_j)$ and $S_{max}(q_i, p_j)$) within the time limit t.

3. All dominated instances are removed from the queue. If an instance (q_i, p_j) dominates all the other instances (q_i, p_k), $p_k \in \mathcal{P}$, then the nearest neighbor of q_i is set to p_j, and the instance (q_i, p_j) is also removed from the queue.

4. If the queue is empty, then the nearest neighbors of all the queries have been found and the FIP is optimally solved. Otherwise, the time limit t is increased, and steps 2 to 4 are repeated until the queue is empty.

3 Modifying DAST for using dominance

Using dominances supposes that the solution process can return upper and lower-bounds on the similarity score. Unfortunately, as presented in [14], DAST does not possess such features. This section presents how these bounds were added into DAST. First, in sections 3.2, we briefly recall DAST principle. Then, in section 3.3, we present our bounding strategy.

3.1 Notation and definitions

Let us first introduce some notations and definitions coming from [1] and [14].

Definition 4. *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 the figure 1:**Right**.*

Definition 5. *Given graph $G = (V, E)$, a **clique** is a subset S of V such that for any two vertices $u \in S$ and $v \in S$, $u \neq v$, u and v are connected by an edge (u, v) in E.*

Definition 6. *The **maximum clique problem** consists in finding in a graph $G = V, E$ a largest (in terms of vertices) clique, denoted by $MCC(G)$. The maximum clique problem is one of the first shown to be NP-complete [11].*

In a $n \times m$ alignment graph $G = (V, E)$, the subset of V restricted to the vertices in the rows $j > i$ and in the columns $l > k$ is denoted by $V^{i.k}$. Similarly, $\hat{V}^{i.k}$ is the subset of V restricted to the vertices in the rows j, $0 \leq j \leq i$ and in the columns l, $0 \leq l \leq k$. The subgraph of G induced by the vertices in $V^{i.k}$ is denoted by $G^{i.k}$, and the subgraph of G induced by the vertices in $\hat{V}^{i.k}$ is denoted by $\hat{G}^{i.k}$. The vertex $j.l$ is a successor of the vertex $i.k$ if $i < j$, $k < l$ and edge $(i.k, j.l)$ is in E, and the set of successors of a vertex $i.k$ is denoted by $\Gamma^+(i.k)$. The vertex $a.b$ is a predecessor of the vertex $i.k$ if $a < i$, $b < k$ and edge $(a.b, i.k)$ is in E, and the set of predecessor of a vertex $i.k$ is denoted by $\Gamma^-(i.k)$. The maximum clique in a graph G is denoted by $MCC(G)$, and its cardinality is denoted by $|MCC(G)|$. An upper-bound on $|MCC(G)|$ is denoted by $|\overline{MCC}(G)|$.

Definition 7. *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.*

Definition 8. *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, in terms of vertices, increasing path in G is denoted by $LIP(G)$*

Lemma 1: $|MCC(G)| \leq |LIP(G)| \leq |LIS(G)|$. **Proof:** Since any two vertices in a clique are adjacent, definition 4 implies that a clique in G is both an increasing subset of vertices and an increasing path, thus $|MCC(G)| \leq |LIP(G)|$. Moreover, $LIP(G)$ is by definition an increasing subset of vertices, which implies that $|LIP(G)| \leq |LIS(G)|$.

3.2 Maximum clique formulation of DAST

DAST is rephrased as a maximum clique problem in an alignment graph as follows. Let G be a $|N_1| \times |N_2|$ alignment graph, where each row corresponds to a residue of N_1 and each column corresponds to a residue of N_2. A vertex $i.k$ is in V only if residues $i \in N_1$ and $k \in N_2$ both come from the same kind of secondary structure (i.e. if matching $i \leftrightarrow k$ is possible). An edge $(i.k, j.l)$ is in E if and only if (i) $i < j$ and $k < l$, for order preserving, and (ii) if $|d_{ij} - d_{kl}| \leq \tau$.

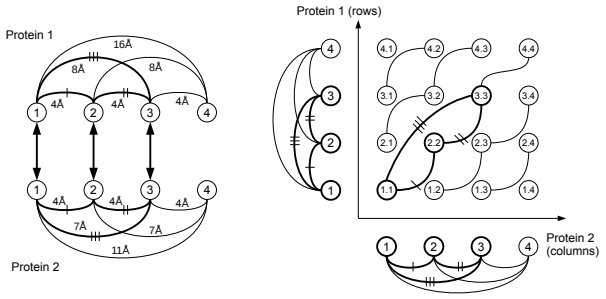


Figure 1: **Left:** An optimal matching (represented by the arrows) between protein 1 and 2, when using a distance threshold of 1Å. **Right:** the corresponding maximum clique in the $|N_1| \times |N_2|$ alignment graph.

As illustrated in figure 1, an optimal matching between two protein structures p_1 and p_2 corresponds to a maximum clique in G. For example the maximum clique $\{(1.1), (2.2), (3.3)\}$ in figure 1:Right corresponds to the optimal matching between residues $(1, 2, 3)$ from p_1 and residues $(1, 2, 3)$ from p_2.

3.3 Bounding strategy

Both DAST similarity scores 1 use ncr, the number of common residues, which is equal to $|MCC(G)|$. Thus, bounding DAST score is equivalent to provide a lower and upper-bound on the cardinality of the maximum clique in G, when the time limit t is reached. The case of the lower-bound is trivial, since the best clique found so far, $Best$, is by definition a lower-bound of ncr. Computing an efficient upper-bound on $|MCC(G)|$ is less straightforward.

3.3.1 Intermediate state of execution

As explained in [14], the maximum clique solver of DAST visits the vertices of V in decreasing order of column (first) and decreasing order of row (second). For each visited vertex $i.k$, the clique solver computes (or upper-bounds) the maximum clique in $G^{i.k}$. If a clique larger than the current best one ($Best$) is found, then $Best$ is updated. Finally, an array C is used to store, in each entry $C[i][k]$, the upper-estimated size of the maximum clique in $G^{i.k}$. This array is later used to fasten the maximum clique computation starting from a vertex having lower row and column indexes, and this implies that computation of $C[i][k]$ requires that $C[i+1][j]$, $C[i+1][j+1]$ and $C[i][j+1]$ are already computed. Since the evaluation of a given cell $T[i][k]$ (i.e. the computation of $C[i][k]$) only occurs after $C[i+1][j]$, $C[i+1][j+1]$ and $C[i][j+1]$ have been computed, an intermediate state of execution of DAST can be represented as in figure 2. In such intermediate state, the cells of T can be split in the following way.

- Cells in which $C[i][k]$ has already been evaluated will be referred to as *evaluated cells*. They correspond to the horizontally striped area in Figure 2:Left. The current best clique, $Best$, has been found in this set.

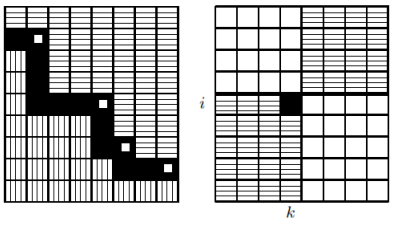


Figure 2: **Left:** An intermediate state of DAST computation; **Right:** $G_L^{i.k}$

- Cells for which $C[i][k]$ have not yet been evaluated will be referred to as *unevaluated cells*.
- Unevaluated cells which are adjacent to an evaluated cell (either side-wise or diagonally) will be referred to as *boundary cells*, and are shown in black in Figure 2:Left. The unevaluated cells for which $C[i][k]$ can be computed (i.e. for which $C[i+1][j]$, $C[i+1][j+1]$ and $C[i][j+1]$ are already computed), belong by definition to the subset of the boundary cells, and are marked with a white square in Figure 2:Left. Finally, unevaluated cells that are not boundary cells are shown as vertically striped.

3.3.2 Upper bounding strategy

It is important to remember that in an intermediate state of execution, even if the possible contribution of a given vertex $i.k$ into a maximum clique of G is completely unknown if $i.k$ lies in the unevaluated region, the contribution of $i.k$ in a maximum clique in $G^{i.k}$ is tightly estimated by $C[i][k]$ if the vertex lies in the evaluated region. Here, we propose an upper-bound on $|MCC(G)|$ that takes advantages of this property.

We define $G_L^{i.k}$, hereafter referred to as *local induced subgraph*, as the subgraph of G induced by the vertex set $V_L^{i.k} = V^{i+1.k+1} \cup \hat{V}^{i.k}$. Figure 2:Right, describes $G_L^{i.k}$ where $i.k$ is the black square.

Lemma 2: For any unevaluated cell $T[i][k]$ and any evaluated cell $T[j][l]$ such that $i < j$ and $k < l$, there exists a boundary cell $T[p][q]$ such that $i \leq p < j$ and $k \leq q < l$.

Proof: Consider the rectangle $R \subset T$ induced by cells $T[i][k]$ and $T[j][l]$ ($R = \{T[a][b]$ such that $i \leq a \leq j$ and $k \leq b \leq l\}$). By definition, R contains both unevaluated (at least $T[i][k]$) and evaluated cells (at least $T[j][l]$), so there exists an unevaluated cell $T[p][q] \in R$, which is adjacent to an evaluated cell, and since $T[p][q] \in R$, then $i \leq p < j$ and $k \leq q < l$.

Lemma 3: $MCC(G) = \max_{i.k|T[i][k] \text{ is boundary}} MCC(G_L^{i.k})$, and thus

$$\overline{MCC}(G) = \max_{i.k|T[i][k] \text{ is boundary}} \overline{MCC}(G_L^{i.k})$$

Proof: Proving Lemma 3 is equivalent to proving that the maximum clique lies in one of the *local induced subgraphs* of G that is induced by a *boundary cell*. Toward this goal, we will assume that we are in an intermediate state of execution, which implies that $T[1][1]$ is an unevaluated cell and that $T[m][n]$

has been evaluated (where m = number of rows in G, n = number of columns in G).

Any clique K in an alignment graph G is an increasing subset of vertices, namely, $K = \{i_1.k_1, i_2.k_2, \ldots, i_{|K|}.k_{|K|}\}$, where $i_l < i_{l+1}$ and $k_l < k_{l+1}$ for all $1 \leq l < |K|$. To prove that K lies completely inside one locally induced subgraph, we instead prove that $K' = \{i_0.k_0, i_1.k_1, \ldots, i_{|K|}.k_{|K|}, i_{|K|+1}.k_{|K|+1}\}$ lies completely inside one locally induced subgraph, where $i_0.k_0 = 1.1$ and $i_{|K|+1}.k_{|K|+1} = m.n$. Since K' intersects with both the *evaluated* and the *unevaluated* region, there exists l, such that vertices $i_0.k_0, i_1.k_1, \ldots, i_l.k_l$ lie in the unevaluated region and vertices $i_{l+1}.k_{l+1}, \ldots, i_{|K|}.k_{|K|}, i_{|K|+1}.k_{|K|+1}$ lie in the evaluated region. By invoking Lemma 2 with $i = i_l, k = k_l, j = i_{l+1}, l = k_{l+1}$, we obtain that there exists a boundary cell $T[p][q]$ such that $i_l \leq p < i_{l+1}$ and $k_l \leq q < k_{l+1}$. Thus, K' and hence the clique K lies entirely in the *local induced subgraph* induced by the boundary cell $T[p][q]$.

Since any clique in $G_L^{i.k}$ implicitly defines a clique over $\hat{V}^{i.k}$ (that is in the unevaluated region) and another clique over $V^{i+1.k+1}$ (that is in the evaluated region), $MCC(G_L^{i.k}) \leq MCC(\hat{G}^{i.k}) + MCC(G^{i+1.k+1})$. Then, $|MCC(G)|$ is upper-bounded by:

$$\overline{MCC}(G) = \max_{i.k|T[i][k] \text{ is boundary}} \overline{MCC}(\hat{G}^{i.k}) + \overline{MCC}(G^{i+1.k+1}), \qquad (2)$$

where $\overline{MCC}(G^{i+1.k+1})$ is tightly estimated by $C[i+1][k+1]$, and where $\overline{MCC}(\hat{G}^{i.k})$ is estimated in a preprocessing step by using the longest increasing path in $\hat{G}^{i.k}$ (i.e. $\overline{MCC}(\hat{G}^{i.k}) = |LIP(\hat{G}^{i.k})|$).

Computing all $|LIP(\hat{G}^{i.k})|$) can be done in $O(n^2 \times m^2)$ time using the algorithm presented in [14]. Moreover, there are no more than $n+m$ boundary cells in T. The global upper-bound $\overline{MCC}(G)$ can be either computed once when the time limit is attained, or also can be maintained at each execution step for the need of branch and bounds strategy.

4 Computational Results

All presented results were obtained on a cluster under Linux RedHat Enterprise 5 architecture 64 Bits, 64 GB RAM, 2.8GHZ Intel Xeon. The efficiency of the dominance strategy for solving FIP was evaluated through two benchmarks. We tackled the FIP problem using the following protocol: any of the proteins has been considered as a query, then removed from the dataset and compared with the remaining proteins in order to find its family based on its nearest neighbor.

First, we used Skolnick set, described in [4]. It is a popular benchmark that contains 40 protein domains having from 97 to 256 residues and classified in SCOP (v1.73) into five families. The second benchmark comes from 3D SHape Recognition Contest 2010 (SHREC'10) [16] and consists of 50 query protein structures and 1000 target protein structures, all classified into 100 superfamilies in the CATH classification. The goal of this contest was to identify the family of each query. Identifying the 50 queries implies solving 50000 comparison instances. The best results have been obtained by the structure comparison tool A_purva [1]. We will us it to compare with the results of DAST.

		$DAST_a$			$DAST_b$			
Step	T(s)	#in	#sol_in	#uns_ins	#ins	#dom_ins	#ins_left	#ass_q
1	2	1560	506	1054	1560	1383	137	29/40
2	300	1054	767	287	137	122	15	37/40
3	3600	287	266	21	15	15	0	40/40

Table 1: Three steps of the FIP computation over the Skolnick set. We use the following abbreviations: # ins –number of instances proceeded for the given lapse of time T (in seconds); # sol_ins –number of solved instances; # uns_ins –number of unsolved (unclassified) instances; # dom_ins –number of dominated instances when using $DAST_b$ and dominance; # ins_left –number of instances to reload; # ass_q –number of assigned (classified) queries.

4.1 DAST on Skolnick set

4.1.1 Running time comparison

Computing the upper-bound at each intermediate state slowdowns the solving process. $DAST_a$ (without bounds computation) solves more instances than $DAST_b$ (with upper-bounds computation) when both methods are given the same distance threshold and the same time limit. For example, for a threshold of 3 Å and when the running time was bounded by 2 seconds per instance $DAST_a$ solved 506 instances, versus 338 for $DAST_b$ (i.e. $DAST_a$ is about 1.5 times faster than $DAST_b$). However, as we will see below, the advantages of $DAST_b$ for solving FIP using the dominances recompense notably this slowdown.

4.1.2 Solving FIP without dominance

Classifying all proteins from the Skolnick set without dominances (i.e. using $DAST_a$) requires solving 1560 instances. As shown in Table 1 when the running time was bounded by 2 seconds per instance, 1054 instances remained unsolved and none of the query could be assigned. Table 1 presents the evolution of the number of solved instances by $DAST_a$ with different time limits. Even with the larger time limit that we used (one hour per instance), 21 instances remained unsolved. The whole computation time was about 15 days, and all of the 29 queries that could be classified were correctly classified.

4.1.3 Solving FIP with dominance

As mentioned above, when the execution time was bounded by two seconds per instance, $DAST_b$ solved only 338 over 1560 instances. However, by applying the dominance relation, 29 queries where correctly assigned into their family by the nearest neighbor measure. Only 137 instances required further processing in order to complete the analysis. These instances were then reloaded with a larger time limit and this process was repeated until the family identification was fully completed. Table 1 details the 3 steps that were needed for Skolnick set. The entire computation time was about 5 hours and 45 minutes. So we observe that $DAST_b$ is significantly faster than $DAST_a$ when solving FIP. Moreover, using the dominance relations guarantees that the exact nearest neighbor is found **without solving all instances**, which is not true for $DAST_a$, neither for any algorithm that does not provide bounds.

4.2 DAST$_b$ versus A_purva on SHREC'10 set

A_purva is an exact solver based on Contact Map Overlap maximization (CMO) similarity measure [1, 15][1]. It has been shown to be both efficient (notably faster than the previous exact algorithms), and reliable (providing accurate upper and lower bounds of the solution). A_purva is based on an integer programming formulation of CMO, and it converges to the optimal solution using a branch and bound strategy. At each node, A_purva provides two numbers derived from a Lagrangian relaxation: a lowerbound LB and an upperbound UB of the maximum number of common contacts (ncc). When an instance is optimally solved, the relation $LB = UB$ holds. Otherwise, $UB > LB$ and the so called relative gap value $RG = (UB - LB)/UB$ gives the precision of the results. This property is very useful in the context of large-scale database comparisons where the execution time is usually bounded. The above properties make A_purva applicable for large-scale protein comparison and classification. Since it is an exact solver, it is often used to evaluate the quality of various heuristic approaches [5, 19].

In this section, we compare DAST$_b$ with A_purva when solving FIP on SHREC'10 set. Both tools provide the best local matching (alignment) between proteins p_1 and p_2 in their corresponding feasible sets (compatible matching pairs), and according to their specific objective functions–maximum number of isometric pairs of amino-acids for DAST and, respectively, maximum number of common contacts for A_purva. On SHREC'10 dataset DAST$_b$ was more precise than A_purva. This can be explained by the isometric constraint in definition 1. Table 2 presents the different steps of this process. A_purva assigned all the 50 queries, 46 of them were correctly predicted according to the CATH classification. However, A_purva failed for the queries 1tteA02, 1wwjA00, 1jftA01 and 3bioA02. DAST$_b$ also assigned 50 queries and correctly predicted 49 of them (DAST failed for 3bioA02). Furthermore, A_purva was significantly faster than DAST$_b$ (the corresponding total running time on SHREC'10 benchmark dataset was 28 hours versus 60 days).

Neither of the methods correctly classified the query 3bioA02. A_purva and DAST found two nearest neighbors from different families with similarity scores of 0.6059 and 0.2 respectively. These low values of DAST similarity score indicate that there is no true nearest neighbor for it in SHREC'10 data set. We contacted an expert from the domain[2] who confirmed the CATH classification of 3bioA02 and suggested us to study its similarity with protein 1f06. As a consequence, we observed that adding the domain 1f06A02 to SHREC'10 dataset allows to assign the query 3bioA02 correctly (i.e. SHREC'10 data set is not enough representative).

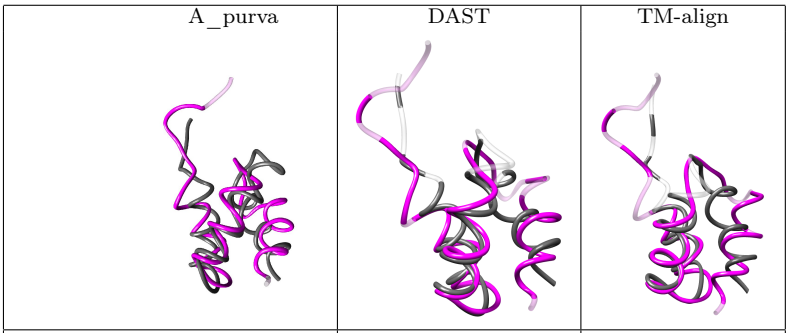
Figure 3 visualizes the alignments provided by A_purva and DAST for the query 1tteA02 which was wrongly predicted by A_purva but was correctly identified by DAST with 1ixrB03 as its nearest neighbor. Aiming to maximize the number of common contacts, A_purva matched non-isometric residues in the middle loop and at both ends. On the contrary, DAST matched closed substructures only (here the three helices of 1tteA02) and ignored the middle loop and extremities. For comparison purpose we also present here the alignment

[1] A_purva is available at http://apurva.genouest.org

[2] Alexey Murzin from the Laboratory of Molecular Biology, Cambridge

Method	Step	Time limit (s)	#Instances	#Dominated	#Left	#Assigned queries
DAST$_b$	1	2	50000	41399	8551	12/50
	2	300	8551	7894	619	19/50
	3	3600	619	548	40	45/50
	4	7200	40	12	28	46/50
	5	60000	28	28	0	50/50
A_purva	1	2	50000	49721	229	43/50
	2	10	229	227	2	48/50
	3	50	2	2	0	50/50

Table 2: Number of dominated instances, of instances to reload and of assigned queries at each of the three steps of the FIP computation over the SHREC'10 set when using dominance, for both DAST$_b$ and A_purva



	A_purva	DAST	TM-align
len_align	53	43	45
cRMSD	5.35	2.37	2.71
TM-score	0.43	0.51	0.53

Figure 3: The instance 1ixrB03-1tteA02 aligned by A_purva (left), DAST (center) and TM-align (right). The parameters for DAST were $\tau = 5.0$Å and sse1 (filter 1). The length of the associated alignment, (len_align), as well as the corresponding RMSDc and TM-score are given. We observe that A_purva matches as much as possible residues, while DAST and TM-align focuse on the local similar structures only.

given by TM-align–well know protein structure comparator [20]. The TM-align alignment is very similar the DAST alignment.

Towards a combined tool These results led us to propose a combined strategy for protein family identification. It uses the normalized $RMSDc$, defined as $NRMSDc = \frac{RMSDc}{\text{length(Query)}}$. First we ran A_purva on SHREC'10 set and computed the corresponding $NRMSDc$ values. We observed that they were higher than 0.1 only for four instances (query,NN)–an indication for a strong deviation between the corresponding structures. For all other instances the $NRMSDc$ value was obviously smaller, less than 0.05. We also realized that these four instances correspond to the four wrongly predicted by A_purva couples (query-NN). Then DAST$_b$ was executed for these four queries only (it required computing new 4000 instances). This combined strategy achieved an accuracy of 50/50 correctly assigned queries (better than any of DAST or A_purva results) but for much less computational time than DAST running time.

5 Conclusion and future work

In this paper we enrich the local structure comparator tool DAST with bounds. This permits to use it in the context of a new dominance relation. The last one is very useful for the protein family identification problem since avoids solving all instances. Moreover, this relation is applicable to any NP-complete comparison methods and can be used for solving the FIP or for clustering large sets of protein structures.

6 Acknowledgements

Thanks to Inken Wohlers from CWI for discussions and suggestions. R. Andonov is supported by BioWIC ANR-08-SEGI-005 project. All computations were done on the Ouest-genopole bioinformatics platform (http://genouest.org).

References

[1] Rumen Andonov, Noël Malod-Dognin, and Nicola Yanev. Maximum contact map overlap revisited. *J Comput Biol.*, 18(1):27–41, 2011.

[2] A. Andreeva, D. Howorth, J-M. Chandonia, S.E. Brenner, T.J.P. Hubbard, C. Chothia, and A.G. Murzin. Data growth and its impact on the scop database: new developments. *Nucl. Acids Res.*, 36:419–425, 11 2007.

[3] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[4] A. Caprara, R. Carr, S. Israil, G. Lancia, and B. Walenz. 1001 optimal pdb structure alignments: integer programming methods for finding the maximum contact map overlap. *J. Comput. Biol.*, 11(1):27–52, 2004.

[5] P Di Lena, P Fariselli, L Margara, M Vassura, and R Casadio. Fast overlapping of protein contact maps by alignment of eigenvectors. *Bioinformatics*, 26(18):2250–2258, 2010.

[6] M. Gerstein and M. Levitt. Using iterative dynamic programming to obtain accurate pair-wise and multiple alignments of protein structures. *ISMB-96 Proceedings*, pages 59–67, 1996.

[7] J-F. Gibrat, T. Madej, and S.H. Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology.*, 6:377–385, 06 1996.

[8] A. Godzik and J. Skolnick. Flexible algorithm for direct multiple alignment of protein structures and seequences. *CABIOS*, 10:587–596, 1994.

[9] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology.*, 223:123–138, 1993.

[10] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828, sep 1978.

[11] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations.*, 6:85–103, 06 1972.

[12] P. Koehl. Protein structure similarities. *Curr Opin Struct Biol*, 11(3):348–353, 2001.

[13] R.H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is np-complete. *Protein Engineering.*, 7(9):1059–1068, 1994.

[14] N. Malod-Dognin, R. Andonov, and N. Yanev. Maximum clique in protein structure comparison. *SEA 2010*, pages 106–117, 2010.

[15] Noël Malod-Dognin, Nicola Yanev, and Rumen Andonov. Comparing protein 3d structures using a_purva. Rapport de recherche RR-7464, INRIA, 2010.

[16] L. Mavridis, V. Venkatraman, D. W. Ritchie, N. Morikawa, R. Andonov, A. Cornu, N. Malod-Dognin, J. Nicolas, M. Temerinac-Ott, M. Reisert, and H. Burkhardtand A. Axenopoulos. Shrec-10 track: Protein models. *3DOR: Eurographics Workshop on 3D Object Retrieval*, pages 117–124, 2010.

[17] C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. Cath - a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[18] C.A. Orengo and J.M. Thornton. Protein families and their evolution - a structural perspective. *Annual Review of Biochemistry*, 74(1):867–900, 2005.

[19] Shibberu Y. and Holder A. A spectral approach to protein structure alignment. *IEEE/ACM Trans Comput Biol Bioinform.*, 8(4):867 – 875, 2011.

[20] Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic Acids Research*, 33:2302–2309, 2005.

Contents

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399