# Domain Discovery Method for Topological Profile Searches in Protein Structures

**Juris Viksna**[1,2*] **David Gilbert**[2] **Gilleain Torrance**[2]
jviksna@cclu.lv drg@dcs.gla.ac.uk maclean@dcs.gla.ac.uk

[1] Institute of Mathematics and Computer Science, University of Latvia, Rainis boulevard 29, Riga LV-1459, Latvia
[2] Bioinformatics Research Centre, Department of Computing Science, A416 Davidson Building, University of Glasgow, Glasgow G12 8QQ, UK

**Abstract**

We describe a method for automated domain discovery for topological profile searches in protein structures. The method is used in a system *TOPStructure* for fast prediction of CATH classification for protein structures (given as PDB files). It is important for profile searches in multi-domain proteins, for which the profile method by itself tends to perform poorly. We also present an $O(C(n)k + nk^2)$ time algorithm for this problem, compared to the $O(C(n)k + (nk)^2)$ time used by a trivial algorithm (where $n$ is the length of the structure, $k$ is the number of profiles and $C(n)$ is the time needed to check for a presence of a given motif in a structure of length $n$). This method has been developed and is currently used for TOPS representations of protein structures and prediction of CATH classification, but may be applied to other graph-based representations of protein or RNA structures and/or other prediction problems. A protein structure prediction system incorporating the domain discovery method is available at `http://bioinf.mii.lu.lv/tops/`.

**Keywords:** protein topology, graph-based representation of structures, structure comparison

## 1 Introduction

This is a continuation of the authors' previous work on protein structure comparison/classification using high-level topological representations of protein structure (TOPS diagrams). Whilst in principle such representations inevitably lead to some loss of information, when compared to "complete" structural data given by atomic coordinates, they have an advantage that structure comparisons can be done comparatively fast and still can lead to reasonably good results.

The TOPS representations of structures can be successfully used for fast automated prediction of protein structure classification from given PDB files. In current implementations this means prediction of CATH homologous superfamilies [7], however the method can be applied also for other classifications (e.g. SCOP). Basically there are two approaches that have been developed for this task.

The simplest is *Best Motif search* that uses a similar approach at the structural level to that used for sequences in the PROSITE database [5]. It is based on a pre-computed database of motifs for proteins with known structures, i.e. structural patterns which (ideally) are associated with some protein function. A given structure then is searched for the presence of motifs from the database, and if some of these are found to be present, the corresponding homologous superfamilies are predicted.

*Profile search* is a related method which tries to overcome some weaknesses arising from using single motifs. Instead of a single motif it assigns a pre-computed profile to each group of proteins, such profile indicating which motifs must be present and which must be absent in this particular group. When a new structure is compared to these profiles, it is searched for all possible motifs and the results of these searches are compared to all known profiles.

*Supported by Wellcome Trust International Research Award 060332/Z/00/Z

In general profile search tends to produce much better results than a simple best motif search, however it is very sensitive to the way the structure has been split into domains. The reason is that if the submitted domain is larger than the ones on which the profile is based, it may contain motifs, which according to the profile should be absent. As a result, if the profiles are based on CATH homologous superfamilies, then the proteins submitted for analysis need to be split into domains in a very similar way to how this is done in CATH. Unfortunately CATH claims that for 47% of proteins domain assignment is still done manually thus making such splitting very difficult. Our current approach is not to split submitted structures into domains at all, however this means that profile search is likely to give poor results for all multi domain proteins.

Two "heuristic" approaches to deal with this problem have been proposed in [8]. Here we describe a more powerful method which effectively searches for all "potential" domains in a given structure and report the ones which give the highest matching scores. To do this we propose a sweep-line type algorithm that performs a domain search against a single profile in $O(C(n) + nk^2)$ time, which is an improvement over a straightforward $O(C(n)k + (nk)^2)$ algorithm. Here $n$ is the length of the structure (number of secondary structure elements), $k$ is the number of profiles and $C(n)$ is the (average) time needed to check for a presence of a given motif in a structure of length $n$. Besides giving prediction of structure classification the method also gives the boundaries of potential domains.

## 2 TOPS Representation of Protein Structure

TOPS diagrams are high-level topology-based representations of protein structures, containing information about secondary structure elements (SSEs) and relations between them (see [2, 10, 11]). They can be regarded as formalizations of so called TOPS cartoons, which have been used by biologists for some time. Technically a TOPS diagram can be considered as a vertex-ordered graph with several types of vertices (currently there are four of them - up- or down- oriented strands and up- or down- oriented helices) and several types of edges (currently parallel or antiparallel H-bonds and left or right oriented chiralities), for more information see [2]. The database of TOPS diagrams is currently available at `http://www.tops.leeds.ac.uk/` and contains TOPS diagrams for all protein domains classified in CATH. It also includes software for diagram generation from PDB files as well as comparison service that allows comparing submitted structure with structures in database.

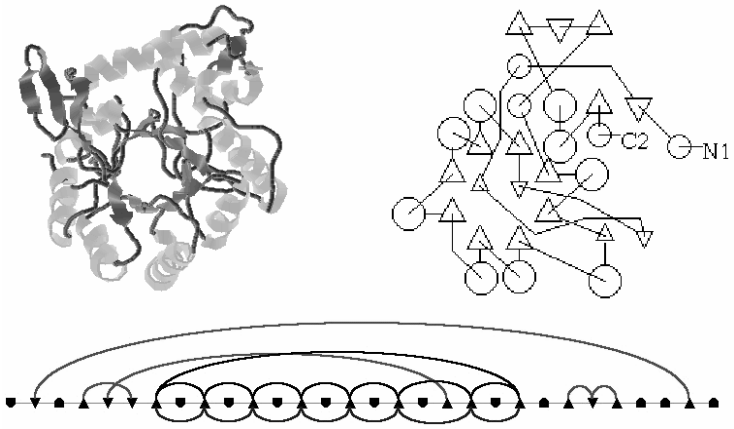


Figure 1: Rasmol picture, TOPS cartoon and TOPS diagram for 1ak5 protein.

In most cases TOPS diagrams are not sensitive to the way how the protein is split into parts, i.e. a diagram for a part of the protein will be a subgraph of the diagram for the whole protein. There are a few exceptions and in some proteins orientation of some SSEs will be different in TOPS diagram for the whole protein and in the diagram for a particular domain. This can have negative implications for the prediction of classification (see Section 7); however the number of such cases is comparatively small.

Structure comparisons at the TOPS level are done by looking for a maximal common substructure for two diagrams (i.e. we have to solve the maximal common subgraph problem for vertex-ordered graphs), allowing us to compare two structures one to another or to pre-compute a motif for given group of structures. To check whether a given motif is present in a particular structure one has to solve the subgraph isomorphism problem for vertex-ordered graphs. The subgraph isomorphism algorithm is also used as a sub-routine in the computation of maximal sub-structures. For more details see [8].

## 3 Searching for the Best Motifs

This is an approach similar to that used for sequences in PROSITE database [4] (`http://ca.expasy.org/prosite/`). Submitted structures are compared against a database of already pre-computed TOPS motifs. The motif database contains common motifs (technically these are the largest common subgraphs) for all CATH homologous superfamilies (proteins sharing CATH numbers up to the H-level). Motifs defined in this way may not be unique (in which case we can choose either an arbitrary one, or, ideally, find all of them and select the one with the largest quality value), however non-unique motifs are rare for real data. Then the *quality* $q$ of a motif is estimated as the ratio *size of the homologous superfamily/number of positive matches for the common motif* (sometimes this is also called 'Positive Predictive Value'). If the value of $q$ is large (close to 1), the motif can be reliably used to predict that a given structure belongs to a particular homologous superfamily.

On a test set containing 28053 protein domains divided into 1225 homologous superfamilies; "perfect" predictions (with $q = 1$) were obtained for 25% of all groups with the best results for CATH classes 2 (mainly $\beta$) and 3 ($\alpha$-$\beta$), having $q = 1$ for correspondingly 33% and 36% of homologous superfamilies (see [8]). The percentage of correct predictions for some other $q$ values is shown in Table 1. Since the motifs are largely based on hydrogen bounds between SSEs, better performance for $\beta$ and $\alpha$-$\beta$ classes could be expected. One can also notice that by allowing lower quality the percentage of correct predictions does not increase too much (e.g. for all classes there are 25% correct predictions with $q = 1$ and 32% correct predictions with $q = 0.2$). The current version of *TOPStructure* database includes motifs with $q \geq 0.05$, allowing the prediction of 434 homologous superfamilies.

## 4 Searching Profiles

For mainly $\alpha$ domains, or for ones with few SSEs, a search for best motifs does not produce particularly good results. If there are very few SSEs then there is very little that can be done. For larger structures one of the noticeable problems is that the largest common subgraph approach does not differentiate between two homologous superfamilies if the motif of one of them is a subgraph of another (which often turns out to be the case). While in principle it could be possible to consider more general motifs that capture this aspect by having "negative" edges – i.e. those that necessarily must be absent to match this motif – such an approach is computationally unrealistic. The difficulty here is that the complexity is exponential with the number of edges in a motif. Although the current motifs have comparatively few edges (usually slightly more than vertices, being constrained by limited number of possible spatial contacts in 3d space), "negative" edges will lead to complete graphs with a number of edges quadratic to the number of vertices. In practice the computation of such motifs will be unfeasible. In this respect the situation with graph-based motifs turns out to be different than it is for sequence motifs; inclusion of sequence elements that "must be absent" in sequence motifs does not make finding such motifs computationally more difficult.

Instead of modifying the notion of motifs, our profile search uses "negative" motifs (i.e. those that necessarily must be absent in a particular homologous superfamily) to capture some of the information about missing edges. First common motifs for all homologous superfamilies are automatically discovered. Then, for each group a *profile* comprising two sets of motifs is computed. One of these

sets contains all motifs that are present in all domains in this group (positive motifs), and the other contains all motifs that are absent in all domains in this group (negative motifs). A protein domain has a given profile $P$ if it contains as subgraphs all positive motifs from $P$ and does not contain as a subgraph any negative motif from $P$.

The method at first may appear not to be particularly efficient, since the number of motifs you need to check for is the same as the number of homologous superfamilies. However, to compare a domain to a profile we only need to perform checking for subgraph isomorphisms for all profile motifs, and for real motifs this still can be done reasonably fast (actually the time required is similar to what is needed for a Best Motif search, except that the number of motifs is about 3 times larger). At the same time, while comparison of a domain with one generalized motif (which also contains some information about missing edges) probably can be done faster, there is no known reasonably efficient way in which such a super-motif could be constructed.

The profile method was tested against the CATH protein classification, similarly as was done in the case of single motifs. Initially, single motifs were constructed for all homologous superfamilies. Then a profile, containing all these motifs (either positive or negative) for each group was computed. Finally the quality $q$ of profile was computed as the ratio *size of the homologous superfamily /number of domains having group's profile*. In comparison to single motif approach, the profile method led to a considerable increase to the number of groups with $q = 1$. Overall it gave "perfect" profiles for 53% of CATH homologous superfamilies. This ratio increases to 74% for groups in class 3 and to 63% for groups in class 2. This permits reliable predictions for about 70% of structures with a significant amount of $\beta$ strands. The percentage of correct predictions for other quality values is given in Table 1. Checking which profiles (currently there are 1225 of them) are matched by a given structure is fast - the overhead comparing to Best Motif search is 10-20 milliseconds on 2.7 GHz workstation (the total time of search including the generation of TOPS diagram from PDB file is about 100-200 milliseconds; see [9]).

Table 1: The results of best motif search and profile methods when tested against CATH. The percentage of groups having prediction qualities of 1, 0.8, 0.5, 0.2 are shown.

| | Best motif search | | | | Profile search | | | |
|---|---|---|---|---|---|---|---|---|
| Prediction quality | 1.0 | 0.8 | 0.5 | 0.2 | 1.0 | 0.8 | 0.5 | 0.2 |
| Class 1 | 7 | 7 | 9 | 11 | 24 | 28 | 32 | 40 |
| Class 2 | 33 | 35 | 38 | 42 | 63 | 68 | 74 | 79 |
| Class 3 | 36 | 39 | 42 | 49 | 74 | 77 | 83 | 91 |
| Class 4 | 0 | 0 | 0 | 0 | 8 | 10 | 13 | 22 |
| All classes | 25 | 27 | 29 | 32 | 53 | 57 | 63 | 68 |

It is interesting that a similar fast prediction of CATH classification very recently has been reported by CATH group itself ([4]). Their GRATH program also works with graph-based structure representations and are claimed to have even higher prediction accuracy than our profile method (90% versus 74%). At the same time it is somewhat slower and uses manually selected representatives for homologous superfamilies, which is likely to give a noticeable advantage in comparison to our fully automatic approach.

One of the largest problems with the profile method is dealing with multiple domain chains. The results are dependent on how the chain is split into domains. For example, consider two profiles $P = \{\{A\}, \{B\}\}$ and $Q = \{\{B\}, \{A\}\}$, meaning that structures corresponding to profile $P$ must contain motif A and must not contain motif $B$, similarly structures corresponding to profile $Q$ must contain motif $B$ and must not contain motif $A$, each of them characterizing a group of domains. An un-split domain, which contains a component from each of these groups will match both motifs $A$ and $B$, thus it will not correspond to profiles $P$ or $Q$ (since each profile requires that one of these motifs must not be matched).

If the method is used for the prediction of CATH numbers, this implies that proteins submitted for analysis need to be split into domains very similarly to how this is done in CATH. Unfortunately CATH claims that for 47% of proteins domain assignment still is done manually thus making such splitting difficult. The current version of *TOPStructure* does not attempt any domain splitting at all, however this means that one has to expect quite poor prediction results for multi-domain proteins.

## 5 Some Definitions

We will define more formally some notions discussed above. Whilst in principle the objects we are dealing with are graphs and basically we are interested in subgraph isomorphisms, for the aims of this paper we will assume a more simplistic approach. Namely that we are given a set of motifs (without specifying what they are) and that each structure (TOPS diagram) is defined by specifying the regions where these motifs are present.

Let $M$ be a fixed finite set of motifs. A structure $S$ is characterized by its length $l(S)$ (a positive integer). If $S$ is a structure then $S_{a,b}$, where $1 \leq a \leq b \leq l(S)$, is also a structure with $l(S_{a,b}) = b-a+1$. An incidence function $I(S,m)$ for a given structure $S$ and a motif $m$ produces a set $T$ of substructures $S_{x,y}$ and satisfies the following properties:

- if $S_{x,y} \in T$ then there is no $y' > y$ and $x' < x$ such that $S_{x',y'} \in T$,
- if $S_{x,y} \in T$ then $S_{x,y} \in I(S_{a,b}, m)$ for all $a \leq x$ and $b \geq y$.

Basically this means that we characterize each structure by the number of SSEs and that we can obtain substructures by taking substrings of SSEs. The incidence function $I$ describes for each motif $m$ in which sub-structures the motif can be found in $S$. The first of the requirements above means that if a motif is found in two sub-structures, one of which is completely contained in another, we are interested only in the smallest sub-structure. The second requirement means that if $S'$ is a sub-structure of $S$ and a motif is present in some sub-structure of $S'$, then it must be present also in some sub-structure of $S$.

A profile for a set of structures $\{S_1, \ldots, S_k\}$ is a pair of sets $M_+$ and $M_-$ of motifs, such that $m \in M_+$ iff $I(S_i, m) \neq \emptyset$ for all $i = 1, \ldots, k$ and $m \in M_-$ iff $I(S_i, m) = \emptyset$ for all $i = 1, \ldots, k$. (Informally $M_+$ contains all motifs that are present in at least on sub-structure for each of $S_i$ -s; $M_-$ contains all motifs that are absent in all sub-structures of $S_i$ -s.)

A structure $S$ corresponds to profile $(M_+, M_-)$ if $I(S,m) \neq \emptyset$ for all $m \in M_+$ and $I(S,m) = \emptyset$ for all $m \in M_-$.

Additionally a score $sc(m)$ is assigned to each motif $m$ (we can assume that it is a positive integer). Similarly to each profile $(M_+, M_-)$ there is assigned a score $sc(M_+, M_-)$. In practice these scores are taken to be equal correspondingly to the quality of motifs and profiles.

In this setting the best motif search can be defined as follows:

**Best Motif Search.** For a given structure $S$, set of motifs $M$ and a given threshold $t \in \boldsymbol{Z}_+$: compute $I(S,m)$ for all $m \in M$ and output all motifs $x$ with $I(S,x) \neq \emptyset$ and $sc(x) \geq t$.

Similarly the profile search can be defined as follows.

**Profile Search.** For a given structure $S$, set of motifs $M$, set of profiles $P$ and a given threshold $t \in \boldsymbol{Z}_+$: compute $I(S,m)$ for all $m \in M$, then for each profile $(M_+, M_-) \in P$ check whether $S$ corresponds to $(M_+, M_-)$ and output all profiles $X$, such that $S$ corresponds to $X$ and $sc(X) \geq t$.

For a given structure $S$ and a motif $m$ the incidence function $I(S,m)$ can be computed by using a slightly modified version of subgraph isomorphism algorithm (described in [8]). Of course the complexity of checking for subgraph isomorphism in principle is exponential from the size of $S$; however when the algorithm is applied to TOPS diagrams it is sufficiently fast and in practice roughly in time roughly linear on the size of $S$. Let denote by $C(n)$.the (average) time needed for computing of $I(S,m)$ for the structure $S$ of length $n$.

If $n$ is the length of the structure $S$ and $k$ is the number of motifs or profiles (in practice the number of motifs and number of profiles is almost the same) then the best motif search can be performed in $O(C(n)k + k)$ time and profile search in time $O(C(n)k + k^2)$. The running time of the profile method can also be further improved (by a linear factor) by using decision trees for checking to which profiles the structure corresponds (see [9]).

## 6 Automated Domain Discovery in Multi-Domain Structures

Some heuristic methods to deal with multi-domain proteins in profile search were proposed in [9]. One of the approaches was to compute the so-called error tolerance for each profile and to allow up to this number of motifs from $M_-$ be present in the structure for it still to correspond to the profile. Here we propose a more general method which effectively searches for all "potential" domains in a given structure and report the ones which give the highest matching scores. The problem can be stated as follows:

**Profile Search with domain prediction.** For a given structure S, set of motifs $M$, set of profiles $P$ and a given threshold $t \in \boldsymbol{Z}_+$: compute $I(S, m)$ for all $m \in M$, then for each profile $(M_+, M_-) \in P$ find all sub-structures $S_{a,b}$ of $S$ that correspond to $(M_+, M_-)$ and output all profiles $X$ and sub-structures $S_{a,b}$, such that $S_{a,b}$ corresponds to $X$ and $sc(X) \geq t$.

The basic solution of the problem is very straightforward: for a given structure $S$ just run the profile search for all sub-structures $S_{a,b}$. This will give a $O(C(n)k + (nk)^2)$ time algorithm (there are $O(n^2)$ sub-structures $S_{a,b}$ to be searched; note that the incidence function $I(S, m)$ for all motifs $m$ still needs to be computed just once). In principle such a running time might be acceptable; however the slowdown compared to the Profile Search method can be as large as $n^2$. In the current implementation (running on a 2.7 GHz processor) a profile search (not including the computation of incidence function values) tends to run in time up to 10-20 milliseconds. When $n$ is close to 100 the running times for domain prediction will increase to 2-3 minutes, i.e. it will be a quite significant increase of workload for a server. Below we propose a sweep-line type algorithm for the same problem which runs in $O(C(n)k + nk^2)$ time. Assuming that $C(n)$ is roughly linear, this will reduce the running time by a factor $n$, i.e. we can expect a search to be performed in less than 1-2 seconds. This estimate is also confirmed by experiments.

The main idea of algorithm is as follows. Let assume that we are given a structure $S$ with $l(S) = n$, a set $M$ of $k$ motifs $M$ and a set of $k$ profiles $P$. Algorithm uses 4 arrays: *Structure1* and *Structure2* with length $n$ and *Profile* and *Motifs* with length $k$.

Informally *Structure1* contains the information of the places where the motifs are found in a given structure. The elements of *Structure1* are linked lists, which can contain the pairs in the form $<m, 1>$, where $m$ is a motif and $l$ is the length of $m$. The presence of a pair $<m, 1>$ in the list *Structure1[i]* indicates that a motif $m$ with the length $l$ starts at the $i$-th position of the structure $S$.

*Structure2* contains information where motifs found in the current sub-structure terminate (its elements similarly are linked lists). The presence of a pair $<m, s>$ in the list *Structure2[i]* indicates that a motif $m$ starts at the $s$-th position and terminates at the $i$-th position of the structure $S$.

*Profile* describes the current profile and *Motifs* holds the endpoints of the motifs found in current sub-structure.

Additionally there are 2 pointers *Start* and *End* to array *Structure1* and 2 counters *Pos* and *Neg*. *Start* and *End* point correspondingly to the first and the last element of sub-structure that currently is being considered, counters indicate how many of positive or negative motifs are still missing.

The first array *Structure1* is initialized according to the values $I(S, m)$, each element *Structure1[i]* is assigned a linked list containing all motifs that appear in sub-structure $S_{i,n}$ and the length of these motifs. Then the algorithm performs $k$ iterations, in each of them checking for sub-structures of $S$ that correspond to the $i$-th profile. Each iteration begins with initialization of array *Profile*, its values are

correspondingly set to 1, -1 and 0 for positive, negative and un-used motifs. Array *Motifs* is initialized with 0s. *Pos* is set to the number of "1"s in array *Profile* and *Neg* is set to 0. *Start* and *End* are positioned on the first element of array *Structure1*.

Algorithm 1. In structure $S$ find a sub-structure corresponding to profile $(M_+, M_-)$

**Input:** Structure $S$, set of motifs $M$ and a profile $(M_+, M_-)$,
Size $n$ of structure $S$ and number $k$ of motifs in $M$.
**Output:** Substructure $S_{a,b}$ corresponding to profile $(M_+, M_-)$

**begin**
  Compute $I(S,m)$ for all $m \in M$ and for each $S_{x,y} \in I(S,m)$ found **1**
    add pair $<m, y-x>$ to linked list $Structure1[x]$
  Initialize $Structure2[x]$ to empty lists for all $x$
  Initialize $Profile[x] \leftarrow 1$ for $x \in M_+$ and $Profile[x] \leftarrow -1$ for $x \in M_-$
  Initialize $Motifs[x] \leftarrow 0$ for all $x$
  $Pos \leftarrow (number\ of\ \text{"1"}\ in\ Profile)$; $Neg \leftarrow 0$
  $Start \leftarrow 1$; $End \leftarrow 1$
  **foreach** $<m,l>$ in list $Structure1[1]$ **do** **2**
    Add $<m,1>$ to linked list $Structure2[l+1]$
  **foreach** $<m,l>$ in list $Structure1[1]$ **do if** $l = 0$ **then**
    **if** $Profile[m] = 1$ **then**
      $Motifs[m] \leftarrow 1$; $Pos \leftarrow Pos - 1$
    **if** $Profile[m] = -1$ *and* $Motifs[m] = 0$ **then**
      $Motifs[m] \leftarrow 1$; $Neg \leftarrow Neg + 1$
  **while** $Pos > 0$ *or* $Neg > 0$ **do** **3**
    **if** $Start = n$ **then return** *No substructures found*
    **if** $End = n$ *and* $Pos > 0$ **then return** *No substructures found*
    **if** $Pos > 0$ *or* $(Neg > 0$ *and* $Start = End)$ **then** **4**
      $End \leftarrow End + 1$
      **foreach** $<m,l>$ in list $Structure1[End]$ **do**
        Add $<m, End>$ to linked list $Structure2[l + End]$
      **foreach** $<m,s>$ in list $Structure2[End]$ **do if** $End - s > Start$ **then**
        **if** $0 < Motifs[m] < End - s$ **then**
          $Motifs[m] \leftarrow End - s$
        **if** $Profile[m] = 1$ *and*$Motifs[m] = 0$ **then**
          $Motifs[m] \leftarrow End - s$; $Pos \leftarrow Pos - 1$
        **if** $Profile[m] = -1$ *and* $Motifs[m] = 0$ **then**
          $Motifs[m] \leftarrow End - s$; $Neg \leftarrow Neg + 1$
    **if** $Neg > 0$ **then** **5**
      **foreach** $<m,s>$ in list $Structure2[Start]$ **do if** $Motifs[m] = Start$ **then**
        $Motifs[m] \leftarrow 0$
        **if** $Profile[m] = 1$ **then**
          $Pos \leftarrow Pos + 1$
        **if** $Profile[m] = -1$ **then**
          $Neg \leftarrow Neg - 1$
      $Start \leftarrow Start + 1$
    **End**
    **return** $Start, End$
**End**

Then, perform the following steps while pointers have not reached the end of the structure:

- if $Pos = 0$ and $Neg = 0$, then *Start* and *End* are reported as the first and the last elements of the domain found and the algorithm terminates;
- if $Pos > 0$ pointer *End* is advanced and the information about the current sub-structure is re-computed;
- if $Neg > 0$ pointer *Start* is advanced and the information about the current sub-structure is re-computed.

If there is no sub-structure found that corresponds to the profile and pointers have reached the end of the structure then the algorithm terminates with failure.

A formal description of algorithm is given below, for simplicity it is assumed that only one profile is checked.

In step 1 array initialization is performed in time $O(C(n)k + n + k)$. Since the computation of $I(S, m)$ has to be done only once for all profiles, then $k$ iterations (one for each profile) of this part of the algorithm can be performed in time $O(C(n)k + nk + k^2)$.

In step 2 it is checked which motifs are present in a (trivial) domain that consists only of the first structure element. Each of the two **foreach** loops can be performed no more that $k$ times, thus step 2 will require time $O(k)$.

The **while** loop 3 is performed up to $2n$ times. Step 4 advances the pointer *End* to the end of the domain; step 5 advances the pointer *Start* to the beginning of the domain; in both cases the motifs present in the domain are re-computed. This re-computation could be done in time $O(k)$, since each of the **foreach** loops will be performed no more than $k$ times. This gives a total time complexity $O(nk)$ for steps 3,4 and 5. The total running time of steps 2-5 will be $O(k) + O(nk) = O(nk)$.

The algorithm needs to be performed $k$ times (once for each profile). This requires time $O(C(n)k + nk + k^2)$ for all iterations of step 1 and $O(nk^2)$ for all iterations of steps 2-5. Thus we obtain an $O(C(n)k + nk^2)$ algorithm for the domain discovery problem.

## 7 Results Obtained by Domain Discovery Method

The implementation of the algorithm was sufficiently efficient for a web-server application. The running times on a (2.7 GHz processor) for profile searches with domain prediction typically were below 500 milliseconds. For comparison, a simple profile search requires 100-200 milliseconds (most of which is spent on TOPS diagram generation from a PDB file and computations of incidence function for motifs).

The algorithm was tested on a set of all PDB files containing CATH domains with "perfect" Profile search predictions (i.e. domains that can be predicted by Profile search with $q = 1$). The set contained 3676 PDB files, about 40% of these corresponding to multiple-domain proteins.

By definition, the domain discovery algorithm was expected to identify correctly all the domains in all these proteins (but probably giving also some incorrect predictions). In practice the algorithm failed to find correct predictions in 47 cases (about 1.2% of the total number). The reason for this is the problem already mentioned in Section 2 that in few cases the program for generating TOPS diagrams produces incompatible diagrams depending on whether it is run for un-split proteins or for individual domains. Fortunately the number of such cases is small.

The tests also showed that for some proteins there are a noticeable number of spurious predictions. In general the correct predictions were among these with the highest scores (as scores are used the qualities q of profiles defined in Section 4; however, as these are not based on the analysis of multiple domain proteins, the intuitive meaning that they represent the likelihood that the obtained prediction will be correct is largely lost - in particular one can also have several different predictions that "are 100% correct"). At the same time, the number of spurious predictions tends to increase when the size of the predicted domains is smaller. Ideally one might wish that, if several predictions are produced, the correct ones are ranked as high as possible.

To evaluate the impact of this we tested two ranking schemes:

- with prediction results sorted simply in decreasing order of the scores (since in this test most of the correct predictions have the highest possible score, on average the position of the right prediction will be roughly equal to half of the number of spurious results with the highest possible score); and
- with prediction results first sorted in decreasing order of the size of domains and then, within each size group, in decreasing order of scores.

The results of these tests are summarized in Table 2.

Table 2: The results of Profile Search with domain prediction.

| Positions | Number of correct matches in these positions (results sorted by scores) | Number of correct matches in these positions (results first sorted by domain sizes, then by scores) |
|---|---|---|
| 1 | 2356 | 887 |
| 2 | 290 | 837 |
| 3 | 244 | 610 |
| 4 | 148 | 464 |
| 5 | 92 | 313 |
| 6-10 | 275 | 413 |
| 11-20 | 124 | 98 |
| 21-50 | 71 | 7 |
| larger than 50 | 29 | 0 |
| maximal position | 152 | 23 |

The results generally show that the domain prediction method works reasonably well - for more than 2/3 of the considered proteins the correct prediction was in the first position in the list of the results (with results sorted by decreasing scores). Moreover, in most cases sorting just by the scores guarantees that the correct prediction is closer to the top of the list of the results, thus in most cases this might be the preferred way to view the results. However, the maximal positions where the correct predictions can be found in the list of the results are smaller, if the sorting is done by the size of the domains first. This might be the preferred way to view the results, if the number of highly scored predictions is large.

A small challenging problem is devising a more advanced result ranking scheme (incorporating both the currently used quality-based score, the size of the predicted domain, as well as possibly some other parameters - e.g. an average size on domains on which a particular profile was based) that will rank the correct predictions as close as possible to the top of the list of the results. A possible approach that we have tried is to consider predictions as points in $N$-dimensional space (with $N$ different scores used as coordinates) and find a hyperplane that in the "best way" separates correct and incorrect predictions. A distance to this hyperplane (i.e. a linear combination of $N$ initial scores) then can be used as a new score for the ranking of the results. We tried to construct such weighted score by a minimum squared-error technique using the initial quality-based score and size of predicted domains as parameters. However, whilst the obtained weighted score reduced the maximal ranking position for correct predictions from 152 to 28, in all aspects it performed worse than our second ranking approach: sorting predictions in decreasing order of the size of predicted domains and then, within each size group, in decreasing order of scores. This suggests that a good ranking scheme probably needs to be based on more complicated functions than just linear combinations of initial scores.

## 8 Software Availability

An automated domain discovery method (as well as *Best Motif search* and *Profile search* methods) is incorporated in *TOPStructure* web service providing CATH homologous superfamily prediction for submitted structure files (in PDB format), and is available at `http://bioinf.mii.lu.lv/tops/`.

## Acknowledgments

The work of Juris Viksna on this project was supported by a Wellcome Trust International Research Development Award 060332/Z/00/Z.

## References

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

[2] Flores, T.P.J., Moss, D.M., and Thornton, J.M., An algorithm for automatically generating protein topology cartoons, *Protein Engineering*, 7:31–37, 1994.

[3] Gilbert, D., Westhead, D.R., Nagano, N., Thornton, J.M., Motif-based searching in tops protein topology databases, *Bioinformatics*, 15:317–326, 1999.

[4] Harrison, A., Pearl, F., Sillitoe, I. Slidel, T., Mott, R., Thornton, J., and Orengo, C., Recognizing the fold of a protein structure, *Bioinformatics*, 19:1748–1759, 2003.

[5] Hofmann, K., Bucher, P., Falquet, L., and Bairoch, A., The PROSITE database, its status in 1999, *Nucleic Acids Res.*, 27:215–219, 1999.

[6] Kabsch, W. and Sander, C., Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features, *Biopolymers*, 22:2577–2637, 1983.

[7] Orengo, C.A., Michie, A.D., Jones, S., and Swindelis, M.B., CATH - a hierarchic classification of protein domain structures, *Structure* 5:1093–1108, 1997.

[8] Vīksna, J. and Gilbert, D., Pattern matching and pattern discovery algorithms for protein topologies, *Proc. of WABI 2001*, LNCS 2149:98–111, 2001.

[9] Vīksna, J., Gilbert, D., and Torrance, G., Protein structure comparison based on profiles of topological motifs: A feasible way to deal with information from negative examples, *Proc. of German Bioinformatics Conference*, 159–165, 2003.

[10] Westhead, D.R., Hatton, D.C., and Thornton, J.M., An atlas of protein topology cartoons available on the world wide web, *Trends in Biochemical Sciences*, 23:35–36, 1998.

[11] Westhead, D.R., Slidel, T.W.F., Flores, T.P.J., and Thornton, J.M., Protein structural topology: Automated analysis and diagrammatic representation, *Protein Science*, 8:897–904, 1999.