Pattern matching and pattern discovery algorithms for protein topologies

Juris Vīksna[1*], David Gilbert[2]

[1] Institute of Mathematics and Computer Science, The University of Latvia,
Rainis boulevard 29, Rīga LV – 1459, Latvia
juris@cclu.lv
[2] Department of Computing, City University,
Northampton square, London EC1V 0HB, UK
drg@soi.city.ac.uk

Abstract. We describe algorithms for pattern matching and pattern learning in TOPS diagrams (formal descriptions of protein topologies). These problems can be reduced to checking for subgraph isomorphism and finding maximal common subgraphs in a restricted class of ordered graphs. We have developed a subgraph isomorphism algorithm for ordered graphs, which performs well on the given set of data. The maximal common subgraph problem then is solved by repeated subgraph extension and checking for isomorphisms. Despite the apparent inefficiency such approach gives an algorithm with time complexity proportional to the number of graphs in the input set and is still practical on the given set of data. As a result we obtain fast methods which can be used for building a database of protein topological motifs, and for the comparison of a given protein of known secondary structure against a motif database.

1 Biological motivation

Once the structure of a protein has been determined, the next task for biologist is to find hypotheses about its function. One possible approach is pairwise comparison of the structure with the structures of proteins whose functions are already known. There are already several tools that allow such comparisons, for example DALI [7] (http://www.ebi.ac.uk/dali/) or CATH [11] (http://www.biochem.ucl.ac.uk/bsm/cath/). However there are two weaknesses with such approach. Firstly, as the number of proteins with given structure is growing the time needed to do such comparisons is also growing. Currently there are about 15000 protein structure descriptions deposited in the Protein

* Supported by a Wellcome Trust International Research Award

Data Bank [1] (http://www.rcsb.org/pdb/), but in the future this number may grow significantly. Secondly, even if a similarity with one or more proteins has been found, it may not be apparent whether this may also imply functional similarity, especially if the similarity is not very strong.

Another possibility is to try to use a similar approach at a structure level to that used for sequences in PROSITE database [6] (http://ca.expasy.org/prosite/). That is pre-compute a database of motifs for proteins with known structures – i.e. structural patterns which are associated with some particular protein function. This effectively requires comnputing the maximal common substructure for a set of structures. One such approach is that of CORA [10], based on multiple structural alignments of protein sequences for given CATH families.

Both of these approaches have been successfully used for protein comparison on the sequence level. The main difficulty in adapting them to the structural level is the complexity of the necessary algorithms – whilst exact sequence comparison algorithms work in linear time, exact structure comparison algorithms may require exponential time and the situation only gets worse with algorithms for finding maximal common substructures. Another aspect of the problem is that it is far from clear which is the best way to define structure similarity. There are many possible approaches, which require different algorithmic methods and are likely to produce different results.

Our work is aimed at the development of efficient comparison and maximal common substructure algorithms using TOPS diagrams for structural topology descriptions, defining structure similarity in a natural way that arises from such formalisation, and at the evaluation of usefulness of such approach. The drawback of such an approach is that TOPS diagrams are not very rich in information; however it has the advantage that it is still possible to define practical algorithms for this level of abstraction.

2 TOPS diagrams

At a comparatively simple level protein structures can be described using TOPS cartoons (see [4], [13] and [14]). A sample cartoon for 2bopA0 is shown in Figure 1 (for comparison a Rasmol-style picture is given in Figure 2). The cartoon shows the secondary structure elements (SSE's) – β-strands (depicted by triangles) and α-helices (depicted by circles), how they are connected in a sequence from amino to carboxyl terminus, their relative spatial positions and orientations. Such representations have been used by biologists for some time. However the graphical images do not explicitly represent all the topological information implied by such description and there

are no very strict rules governing the appearance of a TOPS cartoon for a given protein.

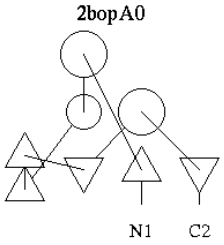

Fig. 1. Cartoon of 2bopA0

Fig. 2. Rasmol diagram of 2bopA0

TOPS *diagrams*, developed by Gilbert et al (see [5]), are a more formal description of protein structural topology and are based on TOPS cartoons. Instead of representing spatial positions by element positions in a plane, a TOPS diagram contains information about the grouping of β-strands in β-sheets (two adjacent elements in β-sheet are connected by an "H-bond", which can be either parallel or anti-parallel) and some information about relative orientation of elements (any two SSE's can be connected by either left or right chirality). Note that in the topological sense we reduce the set of atomic hydrogen bonds between a pair of strands to a single H-bond relationship between the strands. In principle chiralities can be defined between any two SSE's, however only a subset of "most important" chiralities is included in TOPS diagrams; this subset roughly corresponds to the implicit position information in TOPS cartoons.

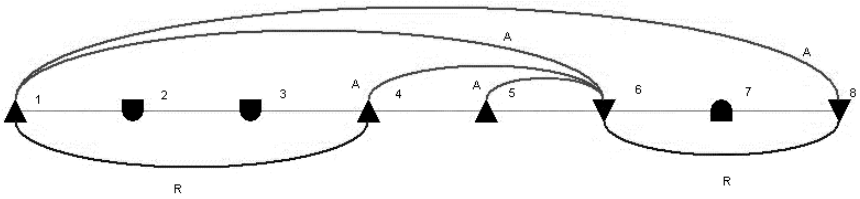

Fig. 3. TOPS diagram of 2bopA0

A TOPS diagram can be regarded as a graph with four different types of vertices (corresponding to up- or down- oriented strands and up- or down-oriented helices) and four different types of edges (corresponding to parallel or antiparallel H-bonds and left or right oriented chiralities). Besides that, the

corresponding graph is ordered – each vertex is assigned a unique number from 1 to *n*, where *n* is the total number of vertices. In the figure above the ordering is also indicated by placing the vertices in the order of increasing numbers (looking from left to right).

3 Pattern matching and pattern discovery in TOPS

If we describe protein secondary structure by TOPS diagrams, a natural way to characterise the similarity of two proteins is by using patterns. In general, we can define patterns using the same type of graphs as for TOPS diagrams. We say that a given pattern matches a given TOPS diagram if and only if the corresponding pattern graph is a subgraph of the corresponding TOPS diagram graph. Here we assume that subgraph relation also preserves the order of vertices – i.e. there is a mapping *F* of pattern graph vertices to target graph vertices such that for any pair of vertices *v* and *w* in pattern graph:

- if the number of *v* is larger than the number of *w*, then also the number of *F(v)* is larger than the number of *F(w)*, and
- if there is an edge between *v* and *w*, then there is an edge (of the same type) between *F(v)* and *F(w)*.

Figure 4 shows one of the possible patterns that matches the diagram for 2bopA0 by mapping vertices with numbers 1, 2, 3, 4, 5, 6 corresponding to vertices with numbers 1, 2, 4, 6, 7, 8.

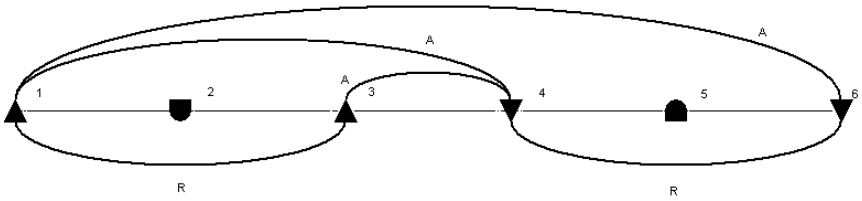

Fig. 4. TOPS pattern

In practice, however, it might be useful to make the pattern definition more complicated. There might be reasons to require that "close" vertices in pattern (i.e. vertices with close numbers) are to be mapped in "close" vertices in target diagram (for some natural notion of "close"). Alternatively it might be useful to require that target graph does not contain extra edges between vertices to which pattern graph vertices are mapped (in this case pattern graph must be induced subgraph of target graph).

If we want to compare a target TOPS diagram to a set of diagrams we can do this by pairwise comparison between the target and each of the comparison set; each such comparison can be made by finding a largest common pattern for two diagrams and assigning a similarity measure based on the size of pattern and the sizes of the two diagrams. Alternatively, if we want to use a motif-based approach, we can find the largest common patterns for a given set of proteins, consider these patterns as motifs, and check whether a pattern for some motif matches the diagram of a target protein. In practice the definition of a motif may be more complicated – for example, it may include several patterns and/or some additional information.

Several algorithms of protein comparison based on the notion of patterns have already been developed and implemented by David Gilbert and the system is available at http://www3.ebi.ac.uk/tops/. It permits searching for proteins that match a given pattern, or to perform pattern based comparison of TOPS descriptions of proteins. Our current task is to implement the more efficient algorithms that we describe here. This will permit the fast generation of motif database, which we plan to make available on the web.

4 Experimental results

4.1 Methodology and databases

In experiments that we have performed to date we have tried to estimate the usefulness of the pattern-based protein motifs, i.e. what is the probability that the fact that a protein matches a given motif implies that protein has also some real similarity with other proteins characterised by the same motif. To do this, we have tried to compare our approach against the existing CATH protein classification database. CATH [11] is a hierarchical classification of protein domain structures, which clusters proteins at four major levels – Class (C), Architecture (A), Topology (T) and Homologous superfamily (H). There are four different C classes – mainly alpha (class 1), mainly beta (class 2), alpha-beta (class 3) and low secondary structure content (class 4). In most cases C classes are assigned automatically. The architecture level describes the overall shape of the domain structure according to orientations of the secondary structures; classes in this level are assigned manually. Classes in the topology level depend on both the overall shape and connectivity of the secondary structures and are assigned automatically by the SSAP algorithm. Classes in the homologous superfamily level group together protein domains which are thought to share a common ancestor and can therefore be described as homologous. They are assigned automatically from the results of sequence comparisons and structure comparisons (using SSAP).

Our comparisons are based on the assumption that identical CATH numbers will also imply some similarity of the TOPS diagrams for the corresponding proteins. The TOPS Atlas database [13], containing 2853 domains and based on clustering structures from the protein data bank [1] using the standard single linkage clustering algorithm at 95% sequence similarity, was selected as the data set for this investigation. Structures with identical CATH numbers (to a given level) have been placed in one group and a maximal common pattern for this group has been computed. Then the pattern was matched against all structures in the selected subset and the quality q of the pattern, corresponding to positive predictive value, computed as follows:

q = *number of proteins in a given group / number of successful matches*.

Thus, $q = 1$ corresponds to a "good" pattern (no false positives) and the value of q is lower for less good patterns.

4.2 Results

The experiments were performed using CATH number identity at levels A, T and H. The CATH number identity at the A level was clearly insufficient to guaranty any similarity at TOPS diagram level; somewhat more surprising was the fact that identity at the T (topological) level still produced noticeably weaker results than identity at H level. Results for the latter are shown in Figure 5.

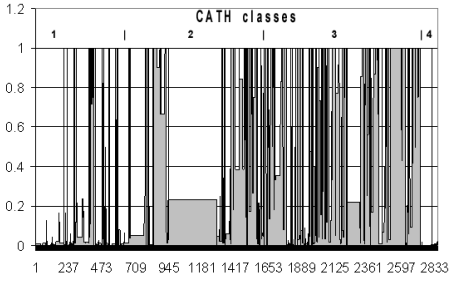

Fig. 5. Quality of TOPS patterns at CATH H level

Here the values of q for all domains from the data set (in lexicographical order by CATH numbers) are shown. The first 527 structures correspond to CATH class 1 (mainly alpha), the next 1048 to class 2 (mainly beta), the following 1151 to class 3 (alpha-beta) and the last 124 to class 4 (weak secondary structure contents). As can be expected q values are small for class 4, since there is very little secondary structure information and also for class 1, since in mainly alpha domains there are few H-bonds and the

corresponding TOPS diagrams contain little information about topology. Better *q* values can be observed for classes 2 and 3.

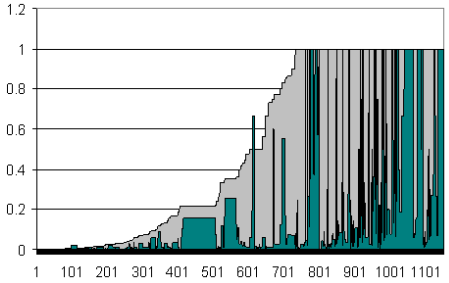

Fig. 6. Quality of TOPS patterns for CATH class 3

Figure 6 shows *q* values (in light-grey) for class 3. Here the proteins have been reordered according to increasing *q* values. As can be seen, in about 36% of cases *q* value is 1, i.e. the CATH number is uniquely defined by a TOPS pattern. Also, there are not many proteins with *q* values close, but less than 1. Therefore, if a pattern has been proven to be "good" for known proteins, it is likely that it will remain "good" for new, as yet unclassified, proteins. For comparison the figure also contains values (in dark-grey) where *q* values have been computed using only secondary structure sequence patterns instead of complete TOPS diagrams. This demonstrates that "good" sequence patterns only exist for approximately 8% of structures. The "superiority" of sequence patterns for one group is caused by different definitions of the largest pattern.

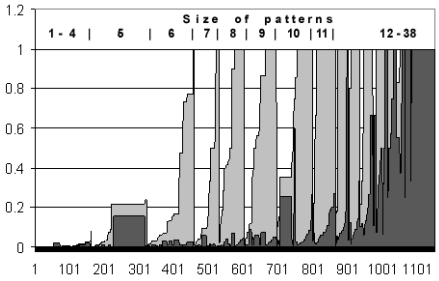

Fig. 7. Quality of TOPS patterns for CATH class 3 ordered by the size of patterns

Figure 7 contains the same data as Figure 6, but initially ordered by pattern size as computed by the number of SSE's in the pattern, and then by *q* values. It can be seen that we start to get good *q* values when the number of SSE's reaches 7 or 8 (proteins with numbers from 459 or 531 on horizontal axis), and that *q* values are good in most cases when number of SSE's reaches 11 (proteins with numbers from 800 on horizontal axis). Therefore, if a protein

contains 7 or more SSE's, there is a good chance that it will have "good" pattern, and, if it contains 11 or more SSE's then in most cases it will have "good" pattern.

Thus, the results obtained as far suggest that a database of pattern motifs could be quite useful for comparison of those proteins that have sufficiently rich secondary structure content and especially for proteins with a large number of strands. This is not the largest subgroup of all proteins; however for this subgroup there are good chances that comparison with TOPS motifs will give biologically useful information. Of course, TOPS diagrams contain limited information about secondary structure; thus we can expect that motifs based on richer secondary structure models may give better results. At the same time the TOPS formalism has advantage that all computations can be performed comparatively quickly. The exact computation times are very dependent on the given data, but in general it can be assumed that the comparison of a given protein against a database of about 1000 motifs requires less than 0.1 second on an ordinary 600 MHz PC workstation. The discovery of motifs and associated evaluation via pattern matching over the TOPS Atlas has been done in about 2 hours on the same equipment.

5 TOPS patterns and related graph problems

The basic problems that arise in TOPS formalism, namely, pattern matching and pattern discovery, can be easily reduced to that of subgraph isomorphism and maximal common subgraph problems in ordered graphs. We consider the following types of vertex ordered labelled graphs.

5.1 Definitions

A given graph $G = (V,E)$ is *vertex ordered* if there is a one to one mapping between the set of numbers $\{1, 2,\ldots,|V|\}$ and set of vertices V. Let us call the number that corresponds to $v \in V$ the *vertex position* and denote it by $p(v)$.

We consider undirected graphs, thus we can assume that edges are defined by ordered pairs (v,w) where $p(v) < p(w)$.

Given vertex ordered graph, we define the *edge order* in the following way:

$p((v_1,w_1)) < p((v_2,w_2))$ if and only if $p(v_1) < p(v_2)$, or $p(v_1) = p(v_2)$ and $p(w_1) < p(w_2)$,

and assign to edges numbers $\{1, 2,\ldots,|E|\}$ according to this order. Let call these numbers *edge positions* and denote them by $p(e)$.

A graph $G = (V,E)$ is *vertex (edge) labelled* with set of labels S, if there is given a function $l_v: V \to S$ ($l_e: E \to S$ for edge labels). We denote the label of a vertex $v \in V$ by $l(v)$ and the label of an edge $e \in E$ by $l(e)$.

For given vertex ordered and vertex and edge labelled graphs $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ we say that G_1 is isomorphic to a subgraph of G_2, if there is an injective mapping I from V_1 to V_2, such that:

- for all $v,w \in V_1$, if $p(v) < p(w)$, then $p(I(v)) < p(I(v))$,
- for all $v,w \in V_1$, if $(v,w) \in E_1$, then $(I(v),I(w)) \in E_2$,
- for all $v \in V_1$ we have $l(v) = l(I(v))$,
- for all $(v,w) \in E_1$ we have $l((v,w)) = l((I(v),I(w))$.

Since each edge is uniquely determined by two vertices we can extend isomorphism mapping I also to edges by defining $I((v,w)) = (I(v),I(w))$. Then I preserves edge order, similarly as it preserves vertex order, i.e. for all $e_1,e_2 \in E_1$, if $p(e_1) < p(e_2)$, then $p(I(e_1)) < p(I(e_2))$.

5.2 Graph representation of TOPS diagrams

We can consider a TOPS diagram as a vertex ordered and vertex and edge labelled graph with the set of vertex labels $S_V = \{e+,e–,h+,h–\}$ (up- or down-oriented strand or up- or down- oriented helix) and the set of edge labels $S_E = \{P,A,L,R,PL,PR,AL,AR\}$ (parallel or antiparallel H-bonds or left or right oriented chiralities or a combination of H-bonds and chiralities). In practice, P edges are only permitted between $e+$ and $e+$ or $e–$ and $e–$ vertices, and A edges are allowed only between $e+$ and $e–$ or $e–$ and $e+$ vertices, but here for us this is not essential.

For practical purposes it is also worth noting the complexity of graphs that have to be dealt with in TOPS formalism – the maximal number of vertices is around 50 and the number of edges is comparatively small and similar to the number of vertices.

Let P be a TOPS pattern and let D_1 and D_2 be TOPS diagrams and let $G(P)$, $G(D_1)$ and $G(D_2)$ be the graphs corresponding to these patterns or diagrams. Then the problem of checking whether TOPS pattern P will match diagram D_1 is equivalent to checking whether $G(P)$ is isomorphic to a subgraph of $G(D_1)$.

Similarly, the problem of finding a largest common pattern P of D_1 and D_2 is equivalent with finding a largest common subgraph $G(P)$ of $G(D_1)$ and $G(D_2)$.

5.3 Complexity and relation to other work

First, it is easy to see that the subgraph isomorphism problem for vertex ordered graphs remains **NP**-complete, since the maximal clique problem is **NP** complete, and this is not altered by vertex ordering. Also, the relatively small number of edges cannot be exploited to obtain polynomial algorithms, since in [3] and [15] similar graph structures are considered that are even simpler (the vertex degree is 0 or 1) and for such graphs the subgraph isomorphism problem is proven to be **NP** -complete. In [3] an algorithm is

given that is polynomial with the respect to the number of overlapping edges, however in TOPS this number tends to be quite large.

There are several "good" non-polynomial algorithms for subgraph isomorphism problem, the most popular being by Ullmann [12] and McGregor [9]. Although these are not straightforwardly adaptable to vertex ordered graphs, the vertex ordering seems to be the property that could considerably improve the algorithm efficiency. Our algorithm can be regarded as a variant of method based on constraint satisfaction [9]; however there is an additional mechanism of re-computing constraints which is periodically invoked.

A very similar class of graphs has also been considered by I.Koch, T.Legauer and E.Wanke in [8] where the authors describe a maximal common subgraph algorithm based on searching for maximal cliques in a vertex product graph. This method seems to be applicable also for TOPS; however it is only practical for finding maximal common subgraphs for two graphs and is not directly useful for finding motifs for larger sets of proteins.

6 Subgraph isomorphism algorithm for ordered graphs

We have developed a subgraph isomorphism algorithm that exploits the fact that the graphs are vertex oriented. Initially, let us assume that we are dealing with graphs that are connected and do not contain isolated vertices (this set is also the most important in practice). Then an isomorphism mapping I is uniquely determined by defining the mapping of edges.

The algorithm tries to match edges in the increasing order of edge positions and backtracks if for some edge match can not be found. Since the graphs are ordered, the positions in the target graph to which a given edge may be mapped and which have to be checked can only increase. Two additional ideas are used to make this process more efficient. Firstly, we assign a number of additional labels to vertices and edges. Secondly, if an edge e can not be mapped according to the existing mapping for previous edges, then the next place where this edge can be mapped according to the labels is found, and the minimal match positions of previous edges are advanced in order to be compatible with the minimal position of e.

6.1 Labelling

By definition vertices and edges are already assigned labels l_v and l_e correspondingly that must be preserved by isomorphism mapping. Additionally we use an another kind of label for both vertices and edges, which we call *Index.*

Vertex index *Index(v)* is a 16-tuple of integers (containing twice as many elements as there are edge labels). The *i*-th element of *Index(v)* is the number of edges *(x,v)* with $l_e((x,v))$ equal to the *i*-th possible value of l_e (according to some initially fixed order of labels). Similarly, the *(k+i)*-th element of *Index(v)* is the number of edges *(v,x)* with $l_e((v,x))$ equal to the *i*-th possible value of l_e. Thus, the value *Index(v)* encodes the numbers of "incoming" and "outgoing" edges of all possible types for a given vertex *v*.

Edge index *Index(e)* for edge *e = (v,w)* is a 4-tuple of integers $<S_1,S_2,E_1,E_2>$, where S_1 is the number of edges *(v,x)* with *p(x) < p(w)*, *S2* is the number of edges *(v,x)* with *p(x) > p(w)*, E_1 is the number of edges *(y,w)* with *p(y) < p(w)*, and E_2 is the number of edges *(y,w)* with *p(y) > p(w)*. The edge index describes how many "shorter" or "longer" other edges are connected to the endpoints of a given edge.

For both vertices and edges we define *Index(x)* ≤ *Index(y)* if the inequality holds between the all corresponding pairs of 16-tuples (or 4-tuples). It is easy to see that for isomorphism mapping *I* for any vertex or edge *x* we must have *Index(x)* ≤ *Index(I(x))*.

6.2 Algorithm

We assume that graphs are given as arrays *PV*, *PE*, *TV* and *TE*, where *PV* is an array of vertices in the pattern graph with *PV[i]* being the vertex *v* with *p(v) = i*, *PE* is an array of edges in the pattern graph with *PE[i]* being the edge *e* with *p(e) = i*, and *TV* and *TE* are similar arrays for the target graph.

For an edge of pattern graph *e* list *Matches(e)* contains all possible positions (in increasing order) in target graph to which e can be matched according to vertex and edge labels and *Index*-es. By *Matches(e)[i]* we denote the *i*-th element from this list. The number *Next(e)* is the first position in *Matches(e)* list to which it still may be worth to try to match the edge. Initially for all edges we have *Next(e)* = 1.

For vertex *v* the number *Pos(v)* is the position in target graph to which vertex *v* is matched. *Pos(v)* = 0 for vertices for which the match still is not found.

The main loop of the algorithm is as follows.

procedure *SubgraphIsomorphismInOrderedGraphs(PV,PE,TV,TE)*
  for all vertices *e* in *PE* **do**
    Compute the list *Matches(e); Next(e)* ← 1
    if *Matches(e)* = ∅ **then return** *Not Isomorphic*
  end for
  for all vertices *v* in *PV* **do** *Pos(v)* ← 0
  k ← 1
  while *k* ≤ |*PE*| **do**

edge = (v,w) ← *PE[k]*
if *Next(edge)* > |*Matches(edge)*| **then return** *Not Isomorphic*
Find the smallest *i* ≥ *Next(edge)* such that for the target graph edge *(vt,wt)* = *Matches(edge)[i]* and for both vertices *v* and *w* either *Pos(v)* = 0 or *Pos(v)* = *vt* and either *Pos(w)* = 0 or *Pos(w)* = *wt* (and *p((vt,wt))* > *Matches(PE[k–1])(Next(PE[k–1])*, if *k* > 1)
if such an *i* is found **then**
Next(edge) ← *i; Pos(v)* ← *vt; Pos(w)* ← *wt; k* ← *k* + 1
else
Find the smallest *j* ≥ *Next(edge)* such that (if *k* > 1) for the target graph edge *(vt, wt)* = *Matches(edge)[j]* we have *p((vt,wt))* > *Matches(PE[k–1])[Next(PE[k–1])]* (take *j* ← *Next(edge), if k* = 1)
Next(edge) ← *j*
for all edges *e* in *PE* with *p(e)* < *k* **do** *Moved(e)* ← **false**
(vt,wt) ← *Matches(edge)[j]*
AdvanceEdgeMatchPositions(v,vt)
Set *k* to be the smallest value for which there is an edge *e* = *(v2, w2)* with *p(e)* = *k* and either *Pos(v2)* = 0 or *Pos(w2)* = 0
end if
end while
return *Pos* (array of vertex mappings)
end procedure

Starting from the first edge the algorithm tries to find matches for all edges in increasing order and returns an array *Pos* of vertex mappings, if it succeeds. If for some edge a match consistent with matches for previous edges can not be found a procedure *AdvanceEdgeMatchPositions* is invoked, which tries to increase the values *Next(e)* for some of already matched edges and the matching process is continued starting from the first edge for which the value *Next(e)* has been changed.

Procedure *AdvanceEdgeMatchPositions* uses a variant of depth first search to find edges for which *Next(e)* can be increased. Alternative strategies are possible,

procedure *AdvanceEdgeMatchPositions(v,vt)*
PatternVertexStack ← **empty**; *Push(PatternVertexStack,v)*
TargetVertexStack ← **empty**; *Push(TargetVertexStack,vt)*
while *PatternVertexStack* ≠ ∅ **do**
pvert ← *Pop(PatternVertexStack); tvert* ← *Pop(TargetVertexStack)*
for all edges *e* with *p(e)* < *k*, *Moved(e)* = **false** and with one endpoint *pvert* **do**

Moved(e) ← **true**
Find the smallest $i \geq Next(e)$ such that for *(vt2, wt2) = Matches(e)[i]* we have $wt2 \geq tvert$ (or $vt2 \geq tvert$, if *pvert* is the "rightmost" endpoint of *e*)
if such an *i* is found **then**
Next(e) ← *i*
Let *newpvert* be the other endpoint of *e*
newtvert ← *vt2* (or *newtvert* ← *wt2*, if *pvert* is the"rightmost" endpoint of *e*)
if *Pos(newpvert)* ≠ 0 **then**
Pos(newpvert) = 0
Push(PatternVertexStack,newpvert)
Push(TargetVertexStack,newtvert)
end if
else return *Not Isomorphic*
end for
end while
end procedure

6.3 Correctness

The informal motivation why the algorithm correctly finds an isomorphic subgraph (or gives the answer that no isomorphic subgraph exists) is the following. First, as already noted above, for connected oriented graphs the isomorphism mapping is completely defined by defining the mapping for edges. For an isomorphism mapping it is sufficient to satisfy the labelling constraints on edge endpoints, preserve edge order and connectivity. If the *AdvanceEdgeMatchPositions* procedure is not used, the algorithm simply performs an exhaustive search of all mappings satisfying these constraints and either finds one, or returns an answer that no such mapping exists. If the *AdvanceEdgeMatchPositions* procedure is included, then when invoked it receives a vertex *v* in pattern graph and the first vertex *vt* in target to which *v* may be mapped according to search performed so far. The constraints on edge mappings are then narrowed down to be consistent with the mapping requirement for vertex *v*.

6.4 General case of disconnected graphs

To deal with graphs that may be disconnected (but do not have isolated vertices) we additionally have to check that the vertex positions are preserved by isomorphism mapping, i.e., for vertices *v* and *w* in pattern graph with $p(v) < p(w)$ we must have $p(I(v)) < p(I(w))$. If we have isolated vertices, we additionally have to check that the sequence of vertices between *v* and *w* is a

substring of the sequence of vertices between *I(v)* and *I(w)*. This additional checking can be easily incorporated into the algorithm.

7 Maximal common subgraph problem

The subgraph isomorphism algorithm is very fast for graphs corresponding to TOPS diagrams. This permits finding maximal common subgraphs by repeated extension and checking for subgraph isomorphism.

In order to find the maximal common subgraph for a given set of graphs we basically use an exhaustive search. Starting a the simple (one vertex) pattern graph, we check for subgraph isomorphism against all graphs in a given set and in the case of success attempt to extend the already matched pattern graph in all possible ways. Some restrictions on the number of different types of edges and vertices can be deduced from the given set of target graphs and are used by the algorithm. Apart from that, the previous successful match may be used to deduce information about extensions which are more likely to be successful in the next match. In general this does not prune the search space but may help to discover large common subgraphs earlier. There is also a greater probability that the largest common subgraph is found within a given time limit, even when the search has not been completed.

The advantage of this approach is that we obtain an algorithm with time complexity that is linear with respect to the number of graphs in a given set. Since there are likely to be more restrictions on the pattern for larger sets, often the most difficult cases arise for sets containing only one graph – however in this case we can simply return the given graph as the maximal common subgraph. Other methods that are known (for example as described in [8]) may be more efficient for sets containing a small number (basically just two) of graphs, but in general cannot be used to find the exact answer to the problem for larger sets.

Experiments suggest that this approach is still practical for TOPS diagrams. As mentioned above in the results section, all motifs in the Atlas for the CATH level H have been found by using repeated pattern matching and extension in 2 hours on an ordinary PC workstation. However it seems that the size of TOPS diagrams is quite close to the limit up to which such a maximal common subgraph algorithm can be successfully used, thus our solution may be quite problem specific. At the same time we expect that the subgraph isomorphism algorithm may be adapted also for considerably larger structures and may be useful for the other problems in bioinformatics.

References

1. Berman, H.M., Westbrook, J., Feng., Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The Protein Data Bank. Nucleic Acids Research 28 (2000) 235–242.
2. Bron, C., Kerbosch, J.: Algorithm 457: Finding all cliques of an undirected graph. Communications of ACM 16 (1973) 575–577.
3. Evans, P.A.: Finding common subsequences with arcs and pseudoknots. Proceedings of Combinatorial Pattern Matching 1999, LNCS 1645 (1999) 270–280.
4. Flores, T.P.J., Moss, D.M., Thornton, J.M.: An algorithm for automatically generating protein topology cartoons. Protein Engineering 7 (1994) 31–37.
5. Gilbert, D., Westhead, D.R., Nagano, N., Thornton, J.M.: Motif-based searching in tops protein topology databases. Bioinformatics 15 (1999) 317–326.
6. Hofmann, K., Bucher, P., Falquet, L., Bairoch, A.: The PROSITE database, its status in 1999. Nucleic Acids Research 27 (1999) 215–219.
7. Holm, L., Park, J.: DaliLite workbench for protein structure comparison. Bioinformatics 16 (2000) 566–567.
8. Koch, I., Lengauer, T., Wanke, E.: An algorithm for finding maximal common subtopologies in a set of protein structures. Journal of Computational Biology 3 (1996) 289–306.
9. McGregor, J.J.: Relational consistency algorithms and their application in finding subgraph and graph isomorphisms. Information Science 19 (1979) 229–250.
10. Orengo, C.A.: CORA – topological fingerprints for protein structural families. Protein Science 8 (1999) 699–715.
11. Orengo, C.A., Michie, A.D., Jones, S., Swindelis, M.B.: CATH – a hierarchic classification of protein domain structures. Structure 5 (1997) 1093–1108.
12. Ullmann, J.R.: An algorithm for subgraph isomorphism. Journal of the ACM 23 (1976) 31–42.
13. Westhead, D.R., Hatton, D.C., Thornton, J.M.: An atlas of protein topology cartoons available on the World Wide Web. Trends in Biochemical Sciences 23 (1998) 35–36.
14. Westhead, D.R., Slidel, T.W.F., Flores, T.P.J., Thornton, J.M.: Protein structural topology: automated analysis and diagrammatic representation. Protein Science 8 (1999) 897–904.
15. Zhang, K., Wang, L., Ma, B.: Computing similarity between RNA structures. Proceedings of Combinatorial Pattern Matching 1999, LNCS 1645 (1999) 281–293.