Motif Finding and Multiple Alignment through Vector-Space Embedding of Protein Sequences

Arnab Bhattacharya, Tamer Kahveci, Ambuj K. Singh
Department of Computer Science
University of California, Santa Barbara, CA 93106
{arnab,tamer,ambuj}@cs.ucsb.edu

Keywords: motif, multiple alignment, vector space, dependency graph

1 Algorithm

We introduce a vector-space embedding of protein sequences which will allow us to find the motifs in a set of proteins. Our method can also be used for the multiple alignment of more than two proteins. It is superior to the existing methods that depend on the order of proteins since we consider all the proteins at once.
Our motif finding method consists of the following steps:

1. Protein subsequences are mapped to points in a multi-dimensional space.
2. Spatially tight clusters of these points are found such that most or all of these proteins are represented in each cluster.
3. A *dependency graph* is constructed with the clusters as vertices and directed edges between the *non-conflicting* vertices.
4. The longest path in the graph is chosen as the motif.

Our multiple alignment algorithm adds another step:

5. Use the motif found in step 4 as the backbone of the alignment and recursively invoke the motif finding algorithm for each unaligned region.

We will now explain each of these steps in more detail.
Step 1: A window of length w is slid along the protein sequence. Each positioning of the window produces a subsequence of w residues. A *score vector* is computed for each such subsequence as the concatenation of the score vectors of each residue. Each row of a score matrix is considered to be the score vector of the amino acid for that row. A score vector maps to a point in a multi-dimensional space. A protein of length n will thus have $n - w + 1$ points in the vector space. We typically choose $w = 3$. We refer the reader to [1] for further details on vector space embedding.
Step 2: All the clusters of points in the vector space are identified. A cluster is defined as a set of points (at most one from each protein) which are within a radius of r from each other and which represents at least p % of the total number of proteins. Here, r and p are the *distance* and the *membership* (the percentage of proteins in the cluster) thresholds respectively. Typically, $r =$ 1-2 % of the dimensions of the search space and $p =$ 80-100 %.
Step 3: A directed dependency graph is built on the clusters as follows. Each cluster is considered to be a vertex in the graph. A directed edge from vertex i to vertex j is added if all of the protein residue positions in i are strictly less than those in j. Such vertex pairs are called non-conflicting. A weight is assigned to each vertex based on how tight it is. A vertex (i.e., a cluster) with less inter-point distances gets a larger weight. Also, a vertex with a higher membership gets a larger weight. A weight is assigned to each edge as well. Each edge corresponds to a pair of points from each protein. The edges get a higher weight if the differences between the residue positions of each pair of points are 1) small and 2) not much varied. These conditions demote the number of gaps.

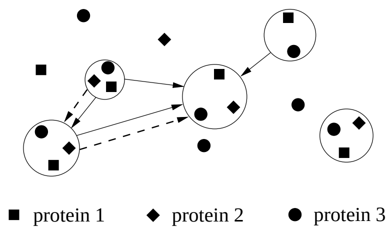

Figure 1: An illustration of the points of three proteins in a two-dimensional space. The circles show the clusters (nodes of the dependency graph). The arrows show the edges of the dependency graph. The dashed arrows show the largest weighted path.

Table 1: The motifs that we find for five proteins from the *FMN-linked oxidoreductases* superfamily for $w = 3$. The bold letters show the backbone. The residue positions for the motifs are (7, 141, 220, 270, 288) for `1al7:-`, (81, 225, 284, 346, 348) for `1d3g:A`, (9, 55, 108, 116, 118) for `1huv:A`, (7, 97, 131, 178, 184) for `1icp:A`, and (113, 184, 252, 336, 365) for `1lco:B`.

PDB id	Motifs										
1al7:-	···	V**N**E	···	L**V**R	···	L**Q**T	···	L**E**E	···	G**V**R	···
1d3g:A	···	Y**K**M	···	L**V**K	···	L**S**T	···	L**E**A		**L**L	···
1huv:A	···	V**E**D	···	L**V**D	···	L**S**T	···	I**E**D		**L**A	···
1icp:A	···	V**E**E	···	I**V**D	···	I**S**C	···	I**E**A	···	G**V**E	···
1lco:B	···	L**K**S	···	M**L**G	···	Y**Q**L	···	I**E**E	···	G**V**S	···

Step 4: Once the directed graph is built, each path on the dependency graph defines a motif. This is because each vertex corresponds to a set of matching residues from the proteins, and the directed edges of a path ensures that these matching residues do not conflict. We find the largest weighted path in the graph. The weight of a path is defined as the sum of the weights of its vertices and edges. Figure 1 illustrates the algorithm developed so far in two-dimensions.
Step 5: The motif found in Step 4 defines the backbone of the multiple alignment. We partition the sequences by clipping the proteins from the residues in the motif. This produces sets of subsequences whose end points are the two consecutive motif residues. Each of these sets are then realigned using Steps 1 to 4 recursively until the length of the subsequences drop below a threshold. These subsequences are then aligned using multiple alignment.

2 Results

In order to demonstrate the effectiveness of our method, we ran it on a number of proteins. Table 1 shows the motifs found for the proteins `1al7:-`, `1d3g:A`, `1huv:A`, `1icp:A` and `1lco:B` of the *FMN-linked oxidoreductases* superfamily for $w = 3$. In this example, we find five motifs which are shown in bold letters. The letters next to the bold ones are also similar with a high probability since they are in the same window as the bold letters. For multiple alignment, the subsequences between consecutive bold letters are aligned similarly.

References

[1] A.Bhattacharya, T. Can, T. Kahveci, A.K. Singh, and Y.-F. Wang. ProGreSS: Simultaneous Searching of Protein Databases by Sequence and Structure. In *PSB*, pages 264–275, 2004.