A Scalable Algorithm for Clustering Protein Sequences *

Valerie Guralnik
Department of Computer Science and Engineering/Army HPC Research Center
University of Minnesota
guralnik@cs.umn.edu

George Karypis
Department of Computer Science and Engineering/Army HPC Research Center
University of Minnesota
karypis@cs.umn.edu

ABSTRACT

The enormous growth of public sequence databases and continuing addition of fully sequenced genomes has created many challenges in developing novel and scalable computational techniques for searching, comparing, and analyzing these databases. Over the years, many methods have been developed for clustering proteins according to their sequence similarity. However, most of these methods tend to have a computational complexity that is at least quadratic on the number of sequences. In this paper we present an entirely different approach to protein clustering that does not require an all-against-all analysis and uses a near-linear complexity K-means based clustering algorithm. Our experimental evaluation on three different data sets containing up to 43,569 protein sequences, show that this approach leads to reasonably good clusters.

1. INTRODUCTION

In recent years, as a result of many large scale genome-wide sequencing projects, we have witnessed an exponential increase in the number of DNA and protein sequences that have become available in public databases. This ever-expanding amount of biological sequence information allows us to get a better understanding of how various species have evolved and how each specie's genome functions. This information promises to revolutionize many aspects of our everyday life, from science to medicine to manufacturing. One of the key facts of genetics is that genes or proteins that share the same primary structure (*i.e.*, DNA or protein sequence) are extremely good indicators of functional similarity as well. This has lead to the development of a variety of computational techniques for analyzing the various genes or proteins based on their sequence similarity [6].

The enormous growth of public sequence databases and continuing addition of fully sequenced genomes has created many challenges in developing novel and scalable computational techniques for searching, comparing, and analyzing these databases. Large scale protein sequence comparison is increasingly becoming an effective way to extract useful biological information from genome sequences. Due to the increasing sizes of protein databases, such methods besides being accurate, they have to be both computationally efficient as well as require minimal user intervention. Accurate prediction of protein function necessitates the clustering of proteins into paralogous groups within genomes and into orthologous groups across genomes.

Over the years, many methods have been developed for clustering proteins according to their sequence similarity. However, most of these methods tend to have a computational complexity that is at least quadratic on the number of sequences, as they require an all-against-all initial analysis, and at the same time require significant user intervention [19; 5].

In this paper we present an entirely different approach to protein clustering that does not require an all-against-all analysis and uses a near-linear complexity K-means based clustering algorithm. The key idea of our approach is to find a set of features that capture the sequential nature of the various proteins, project each protein into a new space whose dimensions are these features, and then use a traditional vector-space K-means based clustering algorithm to find the protein clusters. Our approach was inspired by research in document clustering that showed that high quality clusters can be obtained when each document is represented using a "bag of words". Clustering the documents based solely on their similarity with respect to these words, generates clustering solutions which are equally good to methods that try to take into account phrase, paragraph, and document structure. In light of this example, our algorithm can be thought of as first discovering the "words" (*i.e.*, features) of the sequences, and then clustering the sequences based on the words that they have. Our experimental evaluation on three different data sets containing up to 43,569 protein sequences, show that this approach appears promising and leads to reasonably good clusters.

The rest of this paper is organized as follows. Section 2 provides brief overview of clustering algorithms, Section 3 describes the proposed approach, which is experimentally evaluated in Section 4. Finally, Section 5 provides some concluding remarks.

2. BACKGROUND

Clustering is the task of groupping together the objects into meaningful subclasses. Agglomerative hierarchical clustering and K-means are two techniques that are commonly used for clustering. A widely known study, discussed in [9],

*This work was supported by NSF CCR-9972519, EIA-9986042, ACI-9982274, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by the Minnesota Supercomputing Institute.

indicated that agglomerative hierarchical clustering is superior to K-means, although slower. K-means is used because of its efficiency and agglomerative hierarchical clustering is used because of its quality.

Hierarchical techniques produce a nested sequence of partitions, with a single all-inclusive cluster at the top and singleton clusters of individual points at the bottom. Each intermediate level can be viewed as combining two clusters from the next lower level (or splitting a cluster from the next higher level). Agglomerative hierarchical algorithms start with all the data points as a separate cluster. Each step of the algorithm involves merging two clusters that are most similar. After each merge, the total number of clusters decreases by one. These steps can be repeated until the desired number of clusters is obtained or the distance between two closest clusters is above a certain threshold distance.

In contrast to hierarchical techniques, partitional clustering techniques create a one-level (un-nested) partitioning of the data points. Partitional clustering attempts to break a data set into K clusters such that the partition optimizes a given criterion. Centroid-based approaches, as typified by K-means try to assign objects to clusters such the mean square distance of objects to the centroid of the assigned cluster is minimized. Centroid-based techniques are suitable only for data in metric spaces (e.g. Euclidean space) in which it is possible to compute centroid for a given set of points. Medoid-based methods, work with similarity data, i.e. data in arbitrary similarity space. These techniques try to find representative points (medoids) so as to minimize the sum of the distances of points from their closest medoid.

One of the key steps in all clustering algorithms is the method used to compute the similarity between the objects being clustered. In the context of protein sequences the similarity between two sequences is commonly computed using either the scores of the optimal global sequence alignment [11], the score of an optimal local sequence alignment [16], or the BLAST [2] similarity score. The global and local sequence alignment scores are calculated by aligning the protein by well-known dynamic programming algorithms, using either the PAM[4; 14] or BLOSUM [8] amino acid substitution matrices.

Over the years hierarchical clustering algorithms have been extensively used for clustering protein sequences, a fact primarily motivated by evolution considerations (phylogenetic trees) [13]. Unfortunately, one limitation of using these hierarchical approaches is that when the dynamic programming algorithms are used to compute the similarity, their complexity is $O(n^2m^2 + n^2 \log n)$, where n is the number of protein sequences and m is the average length of each protein, which prohibitively high. In the case in which the similarity is computed using the BLAST score, the complexity reduces to $O(n^2m + n^2 \log n)$, which even though it is smaller by a factor of m, it is still quite high, limiting the applicability of these techniques for large-scale cross-species studies.

3. FEATURE-BASED CLUSTERING

The high computational requirements of the heirarchical clustering algorithms are due to both the fact that they need to compute the pairwise similarity between all the protein sequences and because the similarity computations have a complexity that is either linear or quadratic to the length of the sequences involved. To address these high computational requirements, we explore an alternate approach for clustering sequences that (i) does not use dynamic programming or BLAST to compute the similarity, and (ii) it uses a K-means algorithm whose complexity is near-linear to the number of sequences.

The key idea of our approach is to find a set of features that capture the sequential nature of the various proteins, project each protein into a new space whose dimensions are these features, and then use a traditional vector-space K-means-based clustering algorithm [18] to find the clusters of proteins in this transformed space.

In the remaining of this section we describe the various algorithms and issues associated with each one of these three steps.

3.1 Finding the Feature Space

An essential part of the proposed approach is finding the set of features that will form the basis of the transform space. In particular, these features must satisfy the following properties:

1. The features should capture the sequential relations between the different amino-acids that are present in the protein sequences. This is particularly important, since the proposed clustering algorithm will cluster the proteins based solely on their similarity with respect to these features.

2. The features should be present in a nontrivial number of proteins. This is because, in general, rare features do not improve the overall clustering, as they are useful only in comparing a small set of proteins.

3. The feature space should be complete, in the sense that all such interesting features should be contained in the transformed space.

Our algorithm achieves these goals by using as features all the amino-acid subsequences whose length is between l_{min} and l_{max} that satisfy a given minimum support constraint. These frequent amino-acid subsequences, often called *motifs*, can be either computed using a variety of sequential pattern discovery algorithms [1; 17; 20; 10; 7] (by constraining them to find sequences of consequtive amino-acids), or for small values of l_{max}, can be computed quite fast in a brute-force manner (by using $O(20^{l_{max}})$ memory).

3.2 Projecting in to the Feature Space

The critical step in our approach is that of representing each protein into the newly discovered space of sequential features. If N is the dimensionality of the feature space, a straighforward way of achieving this is to represent each protein as an N-dimensional vector of zeros and ones, with ones corresponding to all the features that are supported by that particular protein.

Unfortunately, this representation can potentially lead to poor clustering results. This is because, the different features that are supported by a particular sequence may be highly dependent which can substantially distort the similarity measure that is used in the transformed space. For instance, if a particular motif w of length l, with $l > l_{min}$ is supported by a particular sequence, then all of its subsequences of length greater than l_{min} will also be supported as well. As a result, when we compare two sequences that

Seq. ID.	Sequence
S_1	AQVHGHKKSVDAM
S_2	AQVHKKSGSDGLP
S_3	AQVHAHVAQIKDP
S_4	AQVHDALGPHKKS
S_5	DALGPAQVHMHKKS
S_6	AQIKDDALGPAQP
S_7	KKSPQIKDQVG
S_8	QIKDALGMAQVHP

Pattern	Support
ALG	4
AQV	6
DAL	4
HKK	4
IKD	4
KKS	5
QIK	4
QVH	6
AQVH	6
DALG	4
HKKS	4
QIKD	4

Globally Selected Features

Seq.ID.	KKS	AQVH	DALG	QIKD
S_1	x	x		
S_2	x	x		
S_3		x		x
S_4	x	x	x	
S_5	x	x	x	
S_6			x	x
S_7	x			x
S_8		x	x	x

Locally Selected Features

Seq.ID.	ALG	KKS	AQVH	DALG	HKKS	QIKD
S_1			x		x	
S_2		x	x			
S_3			x			x
S_4			x	x	x	
S_5			x	x	x	
S_6				x		x
S_7		x				x
S_8	x		x			x

Figure 1: Feature Selection Example

both have w, their similarity will be distorted by the different subsequences of w that they also share. Similar problem occurs when two motifs partially overlap as well. For example, consider the following scenario. Let's assume that we have database of amino-acid sequences, which is shown in Figure 1 together with all frequent motifs of of lengths 3 and 4, having support of 50%. Let's concentrate on the first two sequences S_1 and S_2 and the two discovered motifs AQVH and HKKS. Saying that both proteins subscribe to both patterns will mean that there are two similarity of regions of length 4 between them, while if we computed the alignment of those proteins we would find that there is only one region of length 4 where both proteins align (either AQVH or HKKS). Therefore, it is important to represent each sequence in a way such that the dimensions that they are using are as independent of each other as possible. We implemented two different approaches to address this problem, that are described in the rest of this section.

3.2.1 *Global Approach*

One way of addressing the above problem is to prune the feature space by selecting only a set of independent features, prior to projection. In particular, we say that two motifs are *dependent* if and only if

1. Either one is the prefix of the other or one is a subsequence of the other, and
2. Their intersection of their respective supporting sets is non-trivial.

hese conditions essentially call two motifs that draw support from the same region of the sequence to be dependent. Coming back to the example from Figure 1, let's assume that the intersection of two motifs supporting set is non-trivial if its cardinality is at least two thirds of smallest support of the motif. Under this condition one possible set of independent motifs is KKS, AQVH, DALG, QIKD, as shown in Figure 1.

Using the definition of indepedence, we can then use a greedy algorithm to select a maximal set of independent features, and restrict the space to only this set. However, there are several problems with this approach. First, computation of the pairwise intersection of the supporting sets for each motif is computationally expensive. Second, the resulting space will either be over-pruned or under-pruned. Thus in our example, motifs AQVH and HKKS are found dependent (the number of proteins that support both of them is 4). As a result all the sequences supporting both of these motifs subscribe to only AQVH. However, almost all of the sequences that support both motifs have two regions of similarity of length 4. Hence, we are presented with over-pruned space. Ideally we would like for S_1 to subscribe to both motifs, and for S_2 to subscribe to only one of them. On the other hand, motifs DALG and QIKD are found independent (the number of proteins that support both of them is 2). As a result the sequences S_6 and S_8 have two regions of similarity of length 4 QIKD and DALG which is not correct. As we can see, over-running of the space contradicts the required property of completeness. Under-pruning does not solve the problem of having redundant features.

3.2.2 *Local Approach*

In order to correct the problem of the global approach, we developed a method for selecting a set of independent features that is done locally, on a per protein basis. In this approach, for each protein we first find the set of features that it supports, and from this set we select a maximal set of independent features. In this context, two features are considered to be independent, if they are supported by non-overlapping segments of the underlying protein sequence. The advantage of this approach is that it allows us to subscribe each protein to as many independent features as possible (regardless of the features selected by other proteins), and at the same time, the process of feature selection is very fast. One potential problem with this approach is that sequences that share a large number of motifs, may actually end up having low similarity, because the independent sets they selected, have little overlap. One way of addressing this problem is to select the locally independent features using the same greedy strategy, so that we will increase the likelyhood that if two proteins share a number of motifs, then a considerable number of them will be selected by both of them—ensuring that if two proteins are similar in the original space, will also be similar in the transformed space. There are many such greedy strategies that can be followed. One way to select a feature out of set of dependent patterns is to select a more frequent pattern, or pattern that has more items. An example of locally selected features is presented in Figure 1, in which the selection strategy gave preference to the longer pattern.

3.3 Clustering in the Feature Space

Once the proteins have been projected into the feature space, we use an efficient vector-space clustering algorithm based on K-means [18] to find k clusters. The basic K-means clustering technique is presented below.

1. Select k points as the initial centroids.
2. Assign all points to the closest centroid.
3. Recompute the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids do not change.

In this algorithm, each protein is represented as a vector in the feature-space, and the similarity between two proteins is computed using the cosine similarity function, commonly used in the context of information retrieval [12]. Moreover, since our motif discovery algorithm does not use any amino-acid scoring matrices, to account for frequently occuring low complexity motifs, we scale each of the features following the *inverse-document-frequency* methodoly, again inspired by research in information retrieval. In this approach, if a particular feature appears in m out of n proteins, its weight is multiplied by $\log(n/m)$. The effect of this scaling is that infrequently occuring features are given higher weight that features that occur in almost every protein.

4. EXPERIMENTAL EVALUATION

To evaluate the performance of the proposed clustering algorithm we generated three different data sets, DS1, DS2, and DS3, containing protein sequences from the SWISS-PROT [3] public protein sequence database. Each one of the data sets contains proteins from 20 different protein families. DS1 contains 4,775 sequences, DS2 contains 5,288, and DS3 contains 43,569 sequences. For each of the data sets, we found frequent motifs of length 3 through length 6. The minimum support used for each data set was equal to 25% of the size of the smallest class. In all of our experiments, we used the local scheme for selecting independent dimensions during projection, and these dimensions were selected by giving preference to the longest motifs.

Data Set	Feature-Based K-means	K-medoid
DS1	1.43	2.12
DS2	1.51	2.19

Table 1: Comparison of Entropy Measure

We evaluated the quality of the resulting clustering solution using both internal and external metrics. The interal metric as based on the average similarity between the protein sequences in the frequent feature space, whereas the external similarity was based on computing the entropy of the class distribution of the proteins assigned to each cluster. The entropy was calculated as follows. Let CS be a clustering solution. For each cluster j, we first compute the distribution of the proteins that it contains for each class i, *i.e.*, p_{ij} is equal to the probability a randomly drawn protein from cluster j to be of class i. Then using this class distribution, the entropy of each cluster j is calculated using the formula $E_j = -\sum_i p_{ij} log(p_{ij})$ [15]. The total entropy for a set of clusters is calculated as the sum of the entropies for each cluster weighted by the size of each cluster: $E_{cs} = \sum_{j=1}^{m} \frac{n_j * E_j}{n}$, where n_j is the size of cluster j, m is the number of clusters, and n is the total number of proteins in that data set.

Figure 2, 3, and 4 show the 20-way clustering solution produced by our algorithm on the DS1, DS2, and DS3 data sets, respectively. For DS1, a total of 13,331 frequent motifs of length 3–6 were discovered, out of which 11,780 were kept after independent motifs were selected locally. In the case of DS2, the initial and final number of motifs were 19,129 and 14,139, respectively, and in the case of DS3 they were 22,672 and 21,223. Also, each sequence subscribed to an average of 71, 76, and 81 features for DS1, DS2, and DS3, respectively. The first three columns of each table shows the number of proteins assigned to each cluster, the average pairwise protein similarity between the proteins in each cluster, and the entropy of each cluster, respectively. For each of the clusters, the remaining 20 columns of each row, show the class distribution of the proteins that were assigned to that particular cluster.

Looking at the various clustering solutions we can see that the proposed algorithm was able to produce, in general, clusters that primarily contained proteins from either one or two protein families. Furthermore, 14 functional classes are clearly distinguishable in both DS1 and DS2, and 13 are distinguishable in DS3. The members of remaining functional classes were also mostly kept together, however they were clustered together with members of other functional classes. The overall quality of the clustering solution produced by our algorithm, as measured by entropy, was 1.43, 1.51, and 1.67, for DS1, DS2, and DS3, respectively.

A common characteristic of the clustering solutions for all three data sets was the fact that one or two of the clusters

Feature-based clustering solution

No. of Seqs	Clust. Sim.	Clust. Entropy	Functional Classes																			
			F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
232	0.69	0.00	232	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
229	0.66	0.04	0	228	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
207	0.45	0.00	0	0	207	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
220	0.62	0.00	0	0	0	220	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
211	0.5	0.00	0	0	0	0	211	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
196	0.43	0.00	0	0	0	0	0	196	0	0	0	0	0	0	0	0	0	0	0	0	0	0
191	0.48	0.00	0	0	0	0	0	0	191	0	0	0	0	0	0	0	0	0	0	0	0	0
185	0.34	0.00	0	0	0	0	0	0	0	185	0	0	0	0	0	0	0	0	0	0	0	0
200	0.25	0.62	0	0	0	0	0	17	0	2	178	1	0	0	0	0	0	0	0	0	0	0
171	0.38	0.27	0	0	0	0	0	0	0	0	0	163	8	0	0	0	0	0	0	0	0	0
154	0.51	0.06	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	1	0	0	0	0
180	0.31	0.94	0	3	0	0	1	0	1	0	0	0	37	138	0	0	0	0	0	0	0	0
183	0.5	1.45	0	0	0	0	0	1	0	0	0	43	0	104	0	0	35	0	0	0	0	0
122	0.41	0.50	0	0	0	0	9	0	0	0	0	0	0	0	111	0	0	2	0	0	0	0
181	0.27	1.27	0	1	0	0	0	0	0	0	0	0	0	0	96	0	0	77	0	0	2	5
83	0.48	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0	83	0	0	0	0	0	0
56	0.8	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0
128	0.38	1.12	0	0	0	0	2	0	1	0	0	0	0	0	0	0	77	48	0	0	0	0
770	0.14	3.44	0	3	2	1	15	5	27	14	13	12	26	0	14	68	56	56	96	96	142	124
876	0.13	3.60	0	1	23	16	12	16	17	33	47	18	22	1	12	84	25	51	146	143	106	103

K-medoid clustering solution

No. of Seqs	Clust. Sim.	Clust. Entropy	Functional Classes																			
			F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
83	0.248	2.213	53	3	0	2	2	1	5	3	4	1	2	0	0	1	0	2	0	0	3	1
521	0.212	2.619	101	230	15	0	2	6	0	21	14	68	0	0	3	1	3	12	5	20	3	17
143	0.254	1.417	0	0	110	0	0	10	0	2	2	0	0	1	0	3	0	5	1	0	7	2
124	0.218	1.646	10	0	90	0	1	0	1	1	5	0	0	0	0	3	0	5	3	0	1	4
262	0.301	0.874	1	1	0	232	3	1	1	10	1	3	1	1	0	1	0	2	0	4	0	0
294	0.24	1.52	0	0	10	0	223	6	0	3	10	19	6	0	0	0	1	4	0	1	1	10
515	0.191	2.844	42	0	2	0	2	80	222	6	18	25	5	1	27	6	3	38	8	22	1	7
188	0.192	1.862	0	0	0	0	0	0	1	116	4	0	1	0	0	1	1	5	25	16	18	0
52	0.241	0.468	0	0	0	0	0	2	0	48	0	0	0	0	0	0	0	0	2	0	0	0
175	0.212	0.717	0	0	0	0	0	21	0	0	150	0	0	0	0	1	0	0	0	1	2	0
151	0.21	1.819	0	0	0	0	0	0	0	0	0	96	0	0	1	2	4	7	3	8	5	25
416	0.2	2.581	0	0	0	0	8	9	2	5	6	6	215	7	3	8	0	35	19	51	23	19
302	0.223	1.364	9	0	0	0	0	31	0	0	0	0	0	230	1	5	0	11	2	10	3	0
314	0.191	2.353	18	2	4	3	9	13	3	5	6	15	14	3	198	5	4	5	0	2	2	3
326	0.192	2.225	0	0	0	0	0	0	0	1	1	1	0	0	0	172	10	22	18	32	36	33
41	0.302	0.608	0	0	1	0	0	2	0	1	0	0	0	0	0	0	37	0	0	0	0	0
254	0.191	2.823	0	0	0	0	0	33	1	1	4	3	1	0	0	7	96	22	15	22	25	24
263	0.188	2.624	0	0	0	0	0	1	0	5	4	0	0	0	0	4	86	15	23	28	59	38
203	0.183	2.469	0	0	0	0	0	19	1	1	9	0	1	0	0	10	0	26	97	11	21	7
148	0.182	2.578	0	0	0	0	0	0	0	6	0	0	0	0	0	5	4	19	21	11	40	42

Figure 2: Clustering solution for DS1

Feature-based clustering solution

No. of	Clust.	Clust.	Functional Classes																			
Seqs	Sim.	Entropy	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
266	0.45	0.00	266	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
243	0.69	0.00	0	243	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
236	0.47	0.00	0	0	236	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
229	0.6	0.00	0	0	0	229	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
225	0.82	0.00	0	0	0	0	225	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
227	0.41	0.00	0	0	0	0	0	227	0	0	0	0	0	0	0	0	0	0	0	0	0	0
207	0.48	0.00	0	0	0	0	0	0	207	0	0	0	0	0	0	0	0	0	0	0	0	0
246	0.57	0.72	0	0	0	0	0	0	0	197	0	0	0	0	0	0	49	0	0	0	0	0
93	0.49	1.39	0	0	0	0	0	0	0	59	7	2	0	1	0	0	0	0	24	0	0	0
190	0.49	0.00	0	0	0	0	0	0	0	0	190	0	0	0	0	0	0	0	0	0	0	0
184	0.56	0.00	0	0	0	0	0	0	0	0	0	184	0	0	0	0	0	0	0	0	0	0
175	0.3	0.05	0	0	0	0	0	0	0	0	0	0	174	0	0	0	0	0	0	1	0	0
193	0.64	0.00	0	0	0	0	0	0	0	0	0	0	0	193	0	0	0	0	0	0	0	0
156	0.58	0.00	0	0	0	0	0	0	0	0	0	0	0	0	156	0	0	0	0	0	0	0
169	0.24	1.77	0	0	0	6	0	2	1	0	0	9	0	4	2	111	27	5	2	0	0	0
85	0.65	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0	85	0	0	0	0	0	0
115	0.32	1.16	0	26	0	0	0	0	0	0	0	0	0	0	0	0	82	2	4	0	1	0
132	0.31	2.13	0	0	0	0	24	15	0	0	1	0	0	0	2	0	18	62	10	0	0	0
646	0.14	3.51	0	0	3	24	1	3	15	0	15	16	33	22	33	19	41	94	55	139	74	59
1271	0.11	3.70	2	3	31	7	3	26	40	0	50	59	50	52	77	56	54	91	162	119	177	212

K-medoid clustering solution

No. of	Clust.	Clust.	Functional Classes																			
Seqs	Sim.	Entropy	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
205	0.29	0.392	194	0	0	0	0	0	0	0	0	0	2	0	0	0	2	6	0	1	0	0
267	0.29	1.147	0	206	0	0	0	0	8	0	0	0	3	0	0	0	4	7	0	39	0	0
267	0.255	0.791	2	2	237	0	0	0	0	0	1	1	0	0	14	2	2	4	2	0	0	0
261	0.233	2.347	0	0	0	142	0	2	7	1	0	4	40	0	1	4	5	12	16	19	2	6
400	0.194	2.956	1	5	13	102	2	0	1	1	76	19	11	3	0	6	10	3	19	12	109	7
269	0.352	0.928	0	0	0	0	232	0	0	0	0	2	2	0	0	6	0	7	12	5	0	3
175	0.22	1.541	4	0	0	0	0	118	0	0	39	1	2	0	1	2	1	1	2	1	2	1
177	0.214	1.409	11	6	0	0	0	128	0	0	25	0	0	3	0	0	0	2	0	1	0	1
487	0.19	2.762	9	38	1	0	0	1	206	9	82	6	28	2	1	3	3	10	5	5	22	56
201	0.215	2.686	6	5	3	21	9	14	5	107	4	3	1	4	0	3	2	4	4	6	0	0
216	0.211	2.801	30	3	15	0	8	6	4	104	9	9	1	2	2	1	10	4	4	1	3	0
546	0.192	2.892	0	0	0	0	0	0	10	0	5	196	45	5	2	54	18	34	32	10	41	94
458	0.217	2.354	0	0	0	0	1	2	4	3	2	5	11	219	0	115	6	13	41	10	16	10
405	0.199	2.237	1	4	1	1	0	1	1	26	15	1	31	3	244	2	9	17	1	0	14	33
320	0.181	3.208	4	0	0	0	0	1	4	0	0	22	25	5	5	41	13	30	89	17	25	39
147	0.194	2.084	0	0	0	0	0	0	0	4	0	0	8	0	0	20	68	38	0	6	0	3
147	0.21	2.754	6	3	0	0	1	0	4	0	5	0	11	0	0	10	62	2	19	0	18	6
111	0.204	1.907	0	0	0	0	0	0	0	0	0	1	8	1	0	0	49	40	1	9	0	2
81	0.189	2.672	0	0	0	0	0	0	6	1	0	0	18	25	0	2	1	14	6	7	0	1
148	0.194	1.456	0	0	0	0	0	0	3	0	0	0	10	0	0	0	6	6	4	110	0	9

Figure 3: Clustering solution for DS2

No. of	Clust.	Clust.	Functional Classes																			
Seqs	Sim.	Entropy	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
2466	0.33	0.01	2463	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2169	0.15	1.39	11	1578	0	0	15	5	193	2	0	1	3	1	267	0	4	2	0	1	80	6
3581	0.67	0	0	0	3581	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
582	0.67	0.02	0	0	581	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1291	0.52	0	0	0	0	1291	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1522	0.53	0	0	0	0	0	1522	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1573	0.37	0.75	0	0	1	0	0	1240	0	0	0	0	0	0	332	0	0	0	0	0	0	0
1159	0.35	0.02	0	0	1	0	0	0	1157	0	0	0	0	0	0	0	0	0	0	0	0	1
1773	0.43	0	0	0	0	0	0	0	0	1773	0	0	0	0	0	0	0	0	0	0	0	0
1712	0.39	0.02	0	0	0	0	4	0	0	0	1708	0	0	0	0	0	0	0	0	0	0	0
1219	0.2	0.5	0	0	0	0	0	0	1	72	0	1123	1	1	0	0	0	1	10	1	8	1
718	0.49	0.06	0	0	0	0	0	0	0	0	0	0	714	0	0	1	1	0	1	0	1	0
1005	0.25	0.67	0	0	0	0	0	0	0	3	0	91	882	0	0	0	2	26	0	0	0	1
1708	0.22	1.07	0	0	0	0	0	19	4	1	0	2	0	1182	479	1	0	0	0	0	20	0
802	0.54	0.04	0	0	0	0	0	0	1	0	0	0	0	0	799	0	0	2	0	0	0	0
1050	0.29	1.7	0	0	0	0	320	0	2	0	40	0	0	2	506	0	0	0	0	178	0	2
1129	0.22	1.54	177	0	0	0	0	0	20	0	243	1	0	2	1	676	0	1	0	0	1	7
2916	0.13	2.26	8	4	0	0	11	1	46	184	56	7	229	21	30	2	1595	283	8	400	25	6
534	0.68	0.02	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	533	0	0	0	0
14660	0.12	3.65	84	721	21	125	702	66	68	66	159	3161	650	1055	211	958	665	676	1905	956	1080	1331

Figure 4: Feature-based clustering solution for DS3

tend to be somewhat larger than the rest, and were both *loose* (as measured by the average pairwise similarity) and contained proteins from different families. In analyzing the reason for this behaviour, we discovered that the proteins that were in these clusters contained motifs that were of length either 3 or 4, indicating that the proteins in them did not share some of the longer conserved motifs that did the rest of the proteins. One way of addressing this limitation of our approach is to use amino-acid substitution matrices or amino-acid similarity matrices to define equivalent classes of motifs.

The entropy measure of clustering solution generated by our approach was compared against the entropy measure of clustering solution generated by K-medoid algorithm. Only two data sets DS1 and DS2 were used in this comparison, due to the need to compute all-against-all similarity matrix for each of the data sets. The computation of such matrix for each DS1 and DS2 took over three days. Because data set DS3 contained roughly ten times more data-sequences than either DS1 and DS2, the computation of the similarity matrix for this data set would have taken a prohibitively large amount of time.

Table 1 shows the comparison of entropy results for both data sets. From this table it can be observed that our algorithm outperformed the K-medoid. Figure 2 and 3 compares the 20-way clustering solutions produced by our approach and K-medoid algorithm on DS1 and DS2 respectively . A common characteristic of those clustering solutions is that the groups of proteins that could not be correctly clustered by our approach also did not cluster well by K-medoid. In addition, the functional classes which were clearly distinguishable in the feature based clustering solution were not clustered as well by K-medoid approach.

5. CONCLUSION

In this paper we presented a new approach to protein clustering that uses a near-linear complexity K-means based clustering algorithm. Our approach is based on projecting the proteins onto space of frequent motifs and using K-means based clustering algorithm to find protein clusters in that space. Our experimental evaluation shows that this approach appears promising and leads to reasonably good clusters. However, some of the resulting clusters contain proteins from different families. The proteins in those families do not share long conserved motifs. We believe that the performance of our approach can be further improved by using amino-acid similarity matrices to define equivalent classes of motifs.

Even though the proposed technique appears promissing, by representing each sequence as a "bag of motifs" it looses some of the important information, such as the order of the motifs and their position in each sequence. To further improve performance of clustering algorithm, we would like to develop a centroid-based approach. Preliminary research has shown that such approach is indeed feasible in case of iterative partitional algorithm, in which the centroids are updated incrementally.

6. REFERENCES

[1] R. Aggrawal and R. Srikant. Mining sequential patterns. In *Proc. of the Int'l Conference on Data Engineering (ICDE)*, Taipei, Taiwan, 1996.

[2] S. F. Altschul, T. L. Madden, A. A. Schafer, A. A. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[3] A. Bairoch and B. Boeckmann. The swiss-prot protein sequence data bank. *Nucleic Acids Research*, 19:2247–2249, 1991.

[4] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–352, 1978.

[5] A. J. Enright, I. Iliopoulos, N. Kyrpides, and C. A. Ouzounis. Protein interaction maps for complete genomes based on gene fusion events. *Nature*, 402:86–90, 1999.

[6] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Press Syndicate of the University of Cambridge, New Your, NY, 1997.

[7] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In *Proc. 2000 Intl. Conference on KDD*, 2000.

[8] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Academy Science*, 89(10):915–919, 1992.

[9] A.K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[10] Mahesh V. Joshi, George Karypis, and Vipin Kumar. Universal formulation of sequential patterns. Technical report, University of Minnesota, Department of Computer Science, Minneapolis, 1999.

[11] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Molecular Biology*, 48:443–453, 1970.

[12] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[13] D. Sankoff. Minimal mutation trees of sequences. *SIAM Jouranl of Applied Mathematics*, 28:35–42, 1975.

[14] R. M. Schwartz and M. O. Dayhoff. Matrices for detecting distant relantionships. *Atlas of Protein Sequences*, pages 353–358, 1979.

[15] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 1948.

[16] T. F. Smith and M. S. Waterman. Comparison of biosequences. *Advanced Applied Mathematics*, 2:482–489, 1981.

[17] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, 1996.

[18] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[19] R. L. Tatusov, E. V. Koonin, and Lipman D.J. A genomic perspective on protein families. *Science*, 278:631–637, 1997.

[20] M.J. Zaki. Efficient enumeration of frequent sequences. In *7th International Conference on Information and Knowledge Management*, 1998.