Substructure Clustering on Sequential 3d Object Datasets

Zhenqiang Tan
School of Computing
National University of Singapore
Singapore
tanzhenq@comp.nus.edu.sg

Anthony K. H. Tung
School of Computing
National University of Singapore
Singapore
atung@comp.nus.edu.sg

Abstract

In this paper, we will look at substructure clustering of sequential 3d objects. A sequential 3d object is a set of points located in a three dimensional space that are linked up to form a sequence. Given a set of sequential 3d objects, our aim is to find significantly large substructures which are present in many of the sequential 3d objects. Unlike traditional subspace clustering methods in which objects are compared based on values in the same dimension, the matching dimensions between two 3d sequential objects are affected by both the translation and rotation of the objects and are thus not well defined. Instead, similarity between the objects are judge by computing a structural distance measurement call $rmsd$($Root\ Mean\ Square\ Distance$) which require proper alignment (including translation and rotation) of the objects. As the computation of $rmsd$ is expensive, we proposed a new measure call ald($Angel\ Length\ Distance$) which is shown experiemntally to approximate $rmsd$. Based on ald, we define a new clustering model called $sCluster$ and devise an algorithm for discovering all maximum $sCluster$ in a 3d sequential dataset. Experiments are conducted to illustrate the efficiency and effectiveness of our algorithm.

1 Introduction

In this paper, we will look at substructure clustering of sequential 3d objects. A sequential 3d object is a set of points located in a three dimensional space that are linked up to form a sequence. Given a set of sequential 3d objects, our aim is to find significantly large substructures which are present in many of the sequential 3d objects.

The problem of finding structural patterns in large data sets has received much attention in data mining community recently. Algorithms had been developed for discovering frequent structure patterns from various datasets including sequences, trees and graphs. There also work on similar structural queries which search for similar or inexact structures in the structural sequences of proteins and other biomolecules with the form of 3d (x,y,z) coordinates[7] . A method called geometric hashing for comparison of 3d geometric structures was proposed in[3] which argue that finding clusters of molecules satisfying a required property is more important than finding precise structural difference between pair of molecules.

The objective of our work is different from these existing methods in that we focus on finding all clusters of the substructures based on similarity in the sequential 3d object data set. Sequential 3d objects appear in many real applications and finding out all frequent substructures in the 3d object dataset is a common and meaningful problem. The following are two of such examples.

1. **Identifying common substructures and drug design:** Structure patterns can be used for characterizing families of proteins which are defined to be set of functional or structurally related proteins. The discovery of such patterns can help to understand the working of living organisms [1]. Some protein structure motifs cannot be detected by conducting amino acid sequence alignment, where the identity among their respective amino acid sequence are less than 25% but their 3d structures are quite similar [6]. Clustering substructures on the sequential linked C_α atoms help to explore motifs on remote homologous proteins. Figure 1 plots two protein structures which have similar fragments marked as *Local Similar Substructure*.

2. **Find coherent movement of moving objects:** In these kind of applications, those objects which have coherent movement during a given time period would be found. This is useful for detecting militarily gathering, terriorist group movement etc..

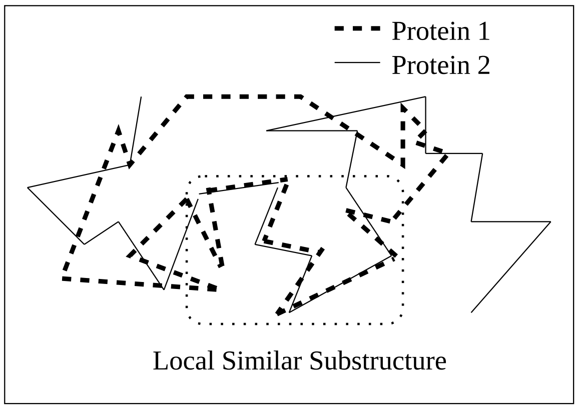


Figure 1. Similar Protein Substructures

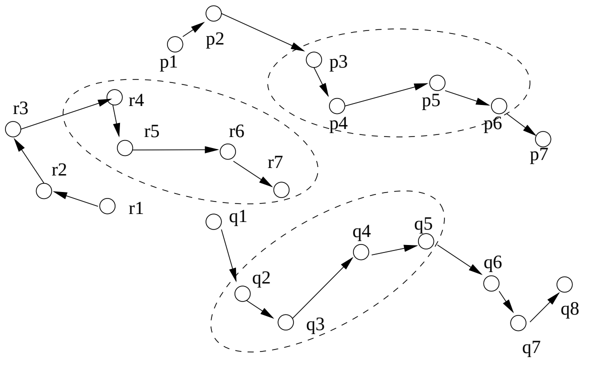


Figure 2. Example of similar sequential substructures

In this paper, our purpose is to find all the groups or clusters which contain similar substructure occurrences in sequential 3d object data sets. There are two major challenges in finding these clusters in sequential 3d object datasets. First, unlike subspace clustering methods[8, 11, 4, 10] in which objects are compared based on values in the same dimension, the matching dimensions [1] between two 3d sequential objects are affected by both the translation and rotation of the objects and are thus not well defined. For example, in Figure 2, $P[3:6]$, $Q[2:5]$ and $R[4:7]$ are similar substructures which are on different dimension groups. Existing subspace clustering techniques are unable to detect such matching dimensions automatically through the proper rotations and translations. Second, traditional similarity measurement for 3d object sequences like $rmsd(Root\ Mean\ Square\ Distance)$ is computationally intensive which will reduce the scalabitity of our algorithm.

Our contributions:

1. We first present a 3d objects distance measurement called $Angle\ Length\ Distance(ald)$, for two 3d objects instead of using the traditional structural distance measurement, $rmsd$. Experiments shows that ald is effective and the computational cost is much lower than $Root\ Mean\ Square\ Distance\ (rmsd)$.

2. Based on this new distance measurement, a general model $sCluser$ is designed for defining substructure clusters.

3. Furthermore, we develop a modified apriori algorithm for mining all maximal $sCluster$s in a given sequential 3d object dataset without false positive and negative. The $sCluster$ can be incrementally enlarged when new structures are added into the dataset.

[1]Note that we are actually matching two sets of points in a three dimensional space. However, since we are still computing the difference between matching points just like we compute the difference for matching dimensions, the term ”dimension/s” is used here to draw an analogy to subspace clustering

The rest of this paper is organized as follows. In section 2, we briefly review the related work. In section 3, a new distance measure called ald(Angle Length Distance) is defined, and the analogy between ald and $rmsd$ is experimentally studied. Section 4 describes the $sCluster$ model. Section 5 proposes the modified-apriori algorithm for finding all $sCluster$ in sequential 3d object datasets. Performance studies is done in section 6. Section 7 concludes this paper and outlines some issues for future research.

2 Related Work

A platform [7] called *AnMol* is proposed for supporting similarity search over structural data of large biomolecules. It represents the structural information using one or more vectors. A new distance measurement based on the vectors, AnMol distance is given out to avoid complex $rmsd$ computation. Experiments show that AnMol distance is very different from $rmsd$. However, it reveals the similarity from the graph prospective.

J. Yang et al. [11] define δ-cluster as a cluster of points/objects that have coherent behaviors rather than points/objects that are physically close to each other. The main objective is to capture a set of objects exhibit strong coherence on the set of dimensions/attributes despite the fact that each object may bear a nonzero bias/offset.

The $pCluster$ model [8] is proposed as a generalization of subspace clustering to capture not only the closeness of objects but also the similarity of patterns exhibited by the objects. They focused on the pattern similarity during clustering where most of the traditional value-similarity-based subspace clustering approaches are a special case in the proposed $pCluster$ model. They designed a depth-first search algorithm to deterministically find all similar patterns. In this model, the value could be shifted on the same dimension groups. However the model and the algorithm will not discover patterns that appear in different dimensions.

Work in [12] aim to find sequential frequent patterns with noisy data. A compatibility matrix is introduced as a means to provide the connection from the observation to the underlying true value. The author uses a novel statistical sampling method and a border collapsing algorithm to discover long patterns in minimum number of scans of sequence databases with sufficiently high confidence. It mainly focus on finding the frequently occurring sequences but not sequential substructure patterns.

An algorithm [9], *CloseGraph*, is developed for mining closed frequent graph patterns in large graph datasets. A frequent pattern is closed if there exists no proper super-pattern with the same support in the dataset. By this means, it significantly reduces the search space. In addition, some new methods are invented to prune search space with small additional cost. Overall, CloseGraph is mean for finding frequent graph patterns, not 3d structures.

3 Accumulated Distance Measurement

Traditionally $rmsd$ is used to evaluate the similarity between two 3d structures. To find the optimal matching from one structure to the other, optimal vertexes alignments, rotations and translations should be identified before calculating the distance.

Definition 3.1 $rmsd(S[1:n], P[1:n])$

$$= min\{\sqrt{\textstyle\sum_{i=1}^{n} |T(S[i]) - P[i]|^2}\}$$

where S and P are two n-vertex 3d objects, and the minimum is taken from all isometric transformation (rotations and translation) T in 3d.

In our case, the two 3d structures are sequential so that the alignment is determined by the order of the vertexes. The complexity for computing $rmsd$ on two sequential structures is $O(n)$. We use the algorithm in [2] for implementing the $rmsd$ calculation.

Problem 3.1 Sequential 3d Object Alignment
Given two sequential 3d objects $S[1:n]$ and $P[1:n]$ and a distance threshold ε, the problem is to find all longest similar substructure pairs which have distance less than ε. (Here if $(S[b_s : e_s], P[b_p : e_p])$ is longest similar substructure pair, then there does not exist i, j, where $i + j > 0$, to have the distance between $S[b_s - i : e_s + j]$ and $P[b_p - i : e_p + j]$ be less than ε)

To extend the substructure alignment, we have to re-calculate the optimal rotation and transformation for computing the new $rmsd$. This measurement could not be incrementally calculated. As such the computational complexity of solving the above problem based on $rmsd$ 3.1 is $O(n^3)$. In real life datasets like protein 3d-structure, the typical number of vertexes in each sequential structure can range from hundreds to thousands. Computing $rmsd$ will lead to inefficient performance.

To go around this problem, we defined a class of distance measurement called $Accumulated\ Distance\ Measurement$ as below:

Definition 3.2 $Accumulated\ Distance\ Measure(adm)$
For any two sequential 3d objects, $S[1:n]$ and $P[1:n]$, if distance measurement D satisfy the property
$$D(S[1:n], P[1:n]) = D(S[1:i], P[1:i]) + D(S[i+1:n], P[i+1:n])$$
then we say D is an adm.

For any adm, the distance between two structures is equal to the sum of the distances between their corresponding substructuress. Thus the distance could be incrementally computed. In the next section, we will propose an algorithm call 1 for solving Problem 3.1. For clustering 3d sequential objects, we define $Angle\ Length\ Distance(ald)$ as an adm for evaluating the similarity between 3d sequential structures.

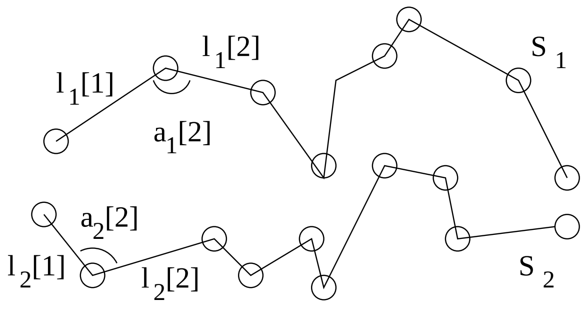


Figure 3. Sample of Angle Length Distance

Definition 3.3 *Angle Length Distance(ald)*
$$ald(S1[1:n], S2[1:n]) = \sum_{i=1}^{n-2} \frac{|a_1[i] - a_2[i]|}{a_1[i] + a_2[i]} + \sum_{i=1}^{n-1} \frac{|l_1[i] - l_2[i]|}{l_1[i] + l_2[i]}$$

where $a_j[i]$ denotes the angle between the edge $i-1$ and the edge i in S_j, and $l_j[i]$ denotes the length of the edge i in S_j.

In Figure 3, S_1 and S_2 are two sequential structures. In S_1, $a_1[2]$ is the angle between the first edge and the second edge, and $l_1[1]$ and $l_1[2]$ are the lengths of the first edge and the second edge respectively. Similarly we have $a_2[2]$, $l_2[1]$ and $l_2[2]$. The ald is the summary of the normalized angle difference and edge length difference.

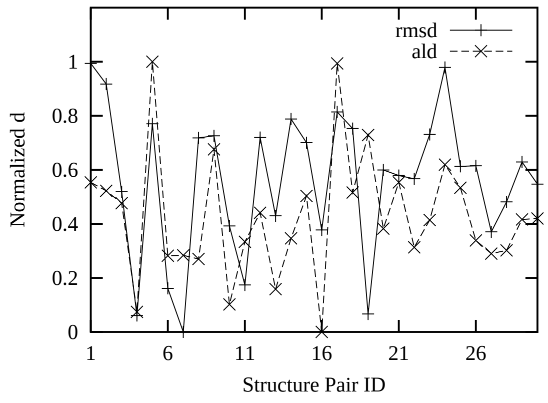


Figure 4. ald **vs** $rmsd$ $(length = 20)$

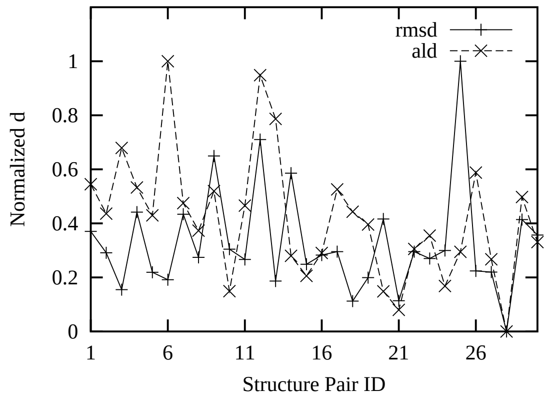


Figure 5. ald **vs** $rmsd$ $(length = 40)$

As a similarity measurement, ald is a simple but effective. To study the effectiveness of ald, we download 200 protein structures from the PDB website and link the $C\alpha$ atoms in each protein to form a 3d sequential structure. A set of experiments are designed to compare the proposed ald to $rmsd$. Figure 4 and 5 illustrate the comparison results on 20-vertex substructure pairs and 40-vertex substructure pairs. We compute the ald and $rmsd$ for 30 randomly selected pairs of 20-vertex and 40-vertex substructures, normalizing the distance as follow: below:

$$normalized\ d = \frac{d - minimum\ distance}{maximum\ distance - minimum\ distance}$$

Both Figure 4 and 5 show that the trend of ald curve is similar with that of $rmsd$. Since we are looking for a distance measurement to determine whether two structures are similar, we believe that ald is a sufficiently good replacement for $rmsd$. In the later section, we will provide further evidence for this by showing some interesting clustering results.

4 The $sCluster$ Model

We next define the $sCluster$ model and provide a formal problem statement.

4.1 Definitions and Problem Statement

Definition 4.1 $sObject(n)$
A 3d sequential structure,$sObject(n)$ is a sequence of n vertexes where each vertex is a point in a 3d space.

Definition 4.2 $SubObject$
Given a $sObject(k)$, $P[1:k]$, and $sObject(n)$, $S[1:n]$ where $k \le n$, if there exists i such that

$$S[i] = P[1],\ ...,\ S[i+k-1] = P[k]$$

then we call P a SubObject of S.

The $SubObject$s of $S[1:n]$ include all $sObject$s of the form $S[i,j]$, where $1 \le i \le j \le n$. In total, a $sObject$ $S[1:n]$ has $\frac{n(n-1)}{2}$ $SubObject$s.

Definition 4.3 $sCluster(D, \varepsilon, w, u)$
A set of u 3d sequential structures is a $sCluster(D, \varepsilon, w, u)$ if

1. *Each sequential structures in the set are of length w*
2. *Given the adm, D, any two sObjects, S and P in the set satisfy $D(S,P) \le \varepsilon$.*

For example, give a set of $sObject$ $\{S_1, ..., S_{10}\}$, if we have $ald(S_i, S_j) \le 7$ where $1 \le i < j \le 10$, then $\{S_1, ..., S_{10}\}$ is a $sCluster(ald, 7, 50, 10)$. Also we say it is a $sCluster$ with maximum distance of 7.

Property 4.1 *Let D be an adm. For $sObject(n)$, $S[1:n]$ and $P[1:n]$, if $D(S[1:n], P[1:n]) \le \varepsilon$, then for any $SubObject$ of S, $S[i:j]$ and the $SubObject$ of P, $P[i:j]$, $D(S[i:j], P[i:j]) \le \varepsilon$ holds.*

Proof: According to definition of adm,
$D(S[i:j], P[i:j])$=$D(S[1:n], P[1:n])-D(S[1:i-1], P[1:i-1])-D(S[j+1:n], P[j+1:n])$. Since

$D(S[1:n], P[1:n]) \leq \varepsilon$, $D(S[1:i-1], P[1:i-1]) \geq 0$ and $D(S[j+1:n], P[j+1:n]) \geq 0$, thus we have $D(S[i:j], P[i:j]) \leq \varepsilon$.

For example, if $D(S[1:50], P[1:50]) \leq 7$, then we know $D(S[10:40], P[10:40]) \leq 7$.

Definition 4.4 *Maximum sCluster*
Let D be an adm and T be a dataset of n $sObjects$. Let $C = \{S_1[b_1 : b_1 + w - 1], ..., S_u[b_u : b_u + w - 1]\}$ be a $sCluster(D, \varepsilon, w, u)$ where $\{S_1, ..., S_u\} \subseteq T$. We say C is a $maximum\ sCluster$ of T if and only if

1. *There does not exist another $sCluster(D, \varepsilon, w, u')$, $C' = \{S'_1[b'_1 : b'_1+w-1], ..., S'_{u'}[b'_{u'} : b'_{u'}+w \;\; -1]\}$, such that $\{S'_1, ..., S'_{u'}\} \subseteq T$ and $C \subset C'$.*

2. *There does not exist another $sCluster(D, \varepsilon, w', u)$, $C'' = \{S_1[b_1 - r : b_1 + w - 1 + s], ..., S_u[b_u - r : b_u+w-1+s]\}$, where $r+s > 0$ and $w+r+s = w'$.*

In Figure 6, T is a data set of 5 $sObject$s. If $C = \{S_2[20 : 35], S_4[25 : 40]\}$ is a $maximum\ sCluster$ then the first criteria in Definition 4.4 state that $C' = \{S_2[20 : 35], S_4[25 : 40], S_5[7 : 22]\}$ is not an $sCluster$ on T. Furthermore, criteria 2 gurantees that $C'' = \{S_2[15 : 40], S_4[20 : 45]\}$ is not an $sCluster$ on T as well.

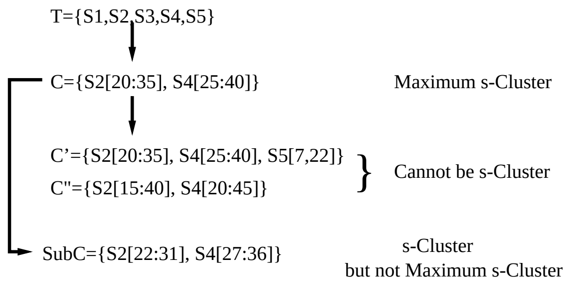

T={S1,S2,S3,S4,S5}
C={S2[20:35], S4[25:40]}
Maximum s-Cluster
C'={S2[20:35], S4[25:40], S5[7,22]}
C"={S2[15:40], S4[20:45]}
Cannot be s-Cluster
SubC={S2[22:31], S4[27:36]}
s-Cluster
but not Maximum s-Cluster

Figure 6. Sample of $Maximum\ sCluster$

Problem 4.1 *Finding $Maximum\ sCluster$*
Given a dataset T of $sObject$, a maximum distance threshold, ε, a minimum length threshold, w and a minimum cardinality threshodl u, our problem is to find all maximum $sCluster(D, \varepsilon, l, t)$ on T such that $l \geq w$ and $t \geq u$.

Definition 4.5 *Synchronized $sObject$ Set (SSS)*
Given two sets of $sObjects$ $C' = \{S_1[b'_1 : e'_1], ..., S_n[b'_n : e'_n]\}$ and $C = \{S_1[b_1 : e_1], ..., S_n[b_n : e_n]\}$, we say C' is the SSS of C with offset of $[b'_1 - b_1, e'_1 - e_1]$ if and only if for any $i, j \in [1 : n]$,

$$b_i - b'_i = b_j \;\; - b'_j \text{ and } e_i - e'_i = e_j \;\; - e'_j$$

We denote C' as $SSS(C, [b'_1 - b_1, e'_1 - e_1])$.

In Figure 6, $C = \{S_2[20 : 35], S_4[25 : 40]\}$ is an SSS of $SubC = \{S_2[22 : 31], S_4[27 : 36]\}$ with offset of $[-2, 4]$, i.e. $SSS(SubC, [-2, 4])$. On the other hand, $SubC$ is also an SSS of C with offset of $[2, -4]$, i.e. $SSS(C, [2, -4])$.

Definition 4.6 *Longest Synchronized Similar SubObject Pair (LSSSP)*
Given an adm D, a distance threshold ε and two $sObjects$ $S[b_s : e_s]$ and $P[b_p : e_p]$, if

$$\begin{cases} D(S[i:j], P[i':j']) \leq \varepsilon \\ D(S[i-r : j+s], P[i'-r : j'+s]) > \varepsilon \\ r+s > 0 \\ b_s \leq i-r \leq j+s \leq e_s \\ b_p \leq i'-r \leq j'+s \leq e_p \end{cases}$$

then we say $(S[i:j], P[i':j']) \in LSSSP(S, P)$.

$LSSSP(S, P)$ is equivalent to the set of all $maximum\ sCluster$s on $\{S, P\}$ with respect to the given adm and distance threshold. For example, given $\varepsilon = 7$, if $D(S[20 : 35], P[25 : 40]) = 6$, $D(S[19 : 35], P[24 : 40]) = 8$ and $D(S[20 : 36], P[25 : 41]) = 7.5$, then $\{S[20 : 35], P[25 : 40]\}$ is a $maximum\ sCluster$ on $\{S, P\}$ as well as a member of $LSSSP(S, P)$.

Lemma 4.1 *Let D be an adm. Given*
$C' = \{S_1[b'_1 : e'_1], ..., S_n[b'_n : e'_n]\}$ and
$C = \{S_1[b_1 : e_1], ..., S_n[b_n : e_n]\}$
where $C' = SSS(C, [b, e])$, assume $S_1[b'_i : e'_i]$ is a $SubObject$ of $S_1[b_i : e_i]$ for $1 \leq i \leq n$. If C is an $sCluster$ with maximum distance threshold of ε, then C' is also an $sCluster$ with maximum distance threshold of ε.

Proof: Since C is an $sCluster$, for any $1 \leq i < j \leq n$, we have $D(S_i[b_i : e_i], S_j[b_j : e_j]) \leq \varepsilon$. According to the definition of adm, we know $D(S_i[b'_i : e'_i], S_j[b'_j : e'_j]) \leq \varepsilon$. So C' is also an $sCluster$ with maximum threshold of ε.

This lemma shows that the set of synchronized $SubObject$s of an $sCluster$ is also a $sCluster$. This combinations lead to a large number of $sCluster$s. Mining maximum $sCluster$ can avoid producing this large number of redundant output.

Lemma 4.2 *Let D be an adm. Given*

$$C' = \{S_1[b'_1 : e'_1], ..., S_n[b'_n : e'_n]\}$$
$$C = \{S_1[b_1 : e_1], ..., S_n[b_n : e_n]\}$$

where $C' = SSS(C, [b, e])$, and both

$$C + \{P[b_p : e_p]\} \textit{ and } C' + \{Q[b_q : e_q]\}$$

are $sClusters$ with maximum distance threshold of ε. Let

$$[br_i : er_i] = [b_i : e_i] \cap [b_i' : e_i'] \; i \in [1 : \quad n]$$

If there exists $\{P[br_p : er_p], Q[br_q : er_q]\}$ which is

$$\begin{cases} an\ SSS\ of\ \{P[b_p : e_p], Q[b_q : e_q]\} \\ an\ sCluster\ with\ maximum\ threshold\ of\ \varepsilon \end{cases}$$

Let

$$\begin{cases} [b_p' : e_p'] = [br_p : er_p] \cap [b_p - b_1 + br_1 : e_p - e_1 + er_1] \\ [b_q' : e_q'] = [br_q : er_q] \cap [b_p - b_1' + br_1 : e_p - e_1' + er_1] \end{cases}$$

Then

$$SSS(C', [b_p - b_p', e_p - e_p']) + \{P[b_p' : e_p'], Q[b_q' : e_q']\}$$

is an $sCluster$ with maximum distance threshold of ε.

Proof: According to Lemma 4.1, since the $sObjects$ in $SSS(C', [b_p - b_p', e_p - e_p']) + \{P[b_p' : e_p']\}$ are $SubObjects$ of the $sObjects$ in $C + \{P[b_p, e_p]\}$, we have

$$\begin{cases} D(S_i[b_i' + b : e_i' + e], S_j[b_j' + b : e_j' + e]) \le \varepsilon \\ D(S_i[b_i' + b : e_i' + e], P[b_p' : e_p']) \le \varepsilon \\ where\ b = b_p - b_p'\ and\ e = e_p - e_p' \end{cases}$$

Similarly we know

$$D(S_i[b_i' + b_p - b_p' : e_i' + e_p - e_p'], Q[b_1' : e_q']) \le \varepsilon$$

Furthermore, because $\{P[b_p' : e_p'], Q[b_q' : e_q']\}$ is an SSS and the two $sObjects$ are $SubObjects$ of $P[br_p : er_p], Q[br_q : er_q]$ respectively, thus it is an $sCluster$ with maximum threshold ε also.

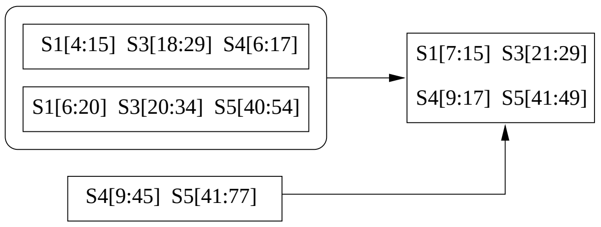


Figure 7. Sample of Lemma 4.2

In Figure 7, $\{S_1[4 : 15], S_3[18 : 29], S_4[6 : 15]\}$ and $\{S_1[6 : 20], S_3[20 : 34], S_4[40 : 54]\}$ imply that $\{S_1[6 : 15], S_3[20 : 29], S_4[8 : 17]\}$ and $\{S_1[6 : 15], S_3[20 : 29], S_5[40 : 49]\}$ are $sClusters$ with maximum distance threshold of ε. By combining them with the $sCluster$ of $\{S_4[9 : 45], S_5[41 : 77]\}$, we have a new $sCluster$, $\{S_1[7 : 15], S_3[21 : 29], S_4[9 : 17], S_5[41 : 49]\}$.

```
Algorithm 1 $LSSSP(\{S, P\}, \varepsilon, w)$
Input: two sequences $S, P$,
    maximum distance $\varepsilon$,
    minimum length $w$
Output: $Maximum\ sCluster(\{S, P\}, \varepsilon, w, 2)$

1. for i=0 to $m - 1$ /*compute distance matrix*/
a. for j=i to $m - 1$
   i. $d[i, j] = d[j, i] = ald(S[i], P[j])$
2. $result \leftarrow \emptyset$
3. for j=0 to $m - L - 2$ /*search in up-triangle*/
a. for i=0 to $m - 1 - j$ /*compute distance-sum*/
   sum[i]=$\sum_{i=0}^{i} d(i, j + i)$
b. Search in each diagonal
 i. $\varepsilon' \leftarrow \varepsilon$
 ii.for i=0 to $m - 1 - j - w$
   1)Binary search $k$ where
     $sum[k] = max\{sum[u]\ where\ sum[u] \le \varepsilon$
                 $and\ 0 \le u \le m - 1 - j - w\}$
   2)$\varepsilon' \leftarrow \varepsilon' + d[i, j + i]$
   3)If $k \ge w$
     then $result \leftarrow result + \{S[i : k], P[j : j + k]\}$
4. Search LSPs in down-triangle similar with step 3
5. Return $result$
```

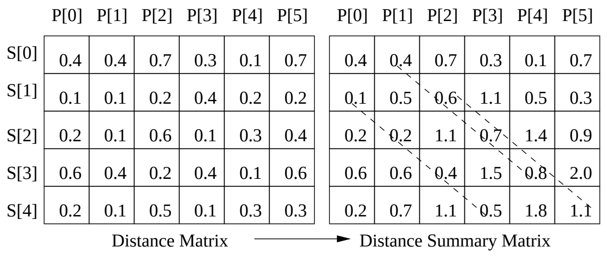
Figure 8. Algorithm for mining $LSSSP$

5 Algorithm

In this section, we will look at our algorithm for finding $sCluster$. Our approach detects clusters which are made up of 3d sequential substructures whose pairwise distance is no more than user-specified ε. Our algorithm, a modified apriori mining process, can find all qualified $sClusters$ without loss. Mining $LSSSP$ from pairwise $sObjects$ is the basis for generating $sClusters$. To find the $LSSSP$ of two $sObjects$ is equivalent to finding $sCluster$ on the set of two $sObjects$. Given two $sObjects$ S and P, we assume that S and P have the same length. The algorithm for computing $sCluster(D, \{S, P\}, \varepsilon, w, 2)$ is as shown in Algorithm 1:

In Figure 9, $d[i, j]$ in the distance matrix represents the distance between the $S[i]$ and $P[j]$. We convert it to distance summary matrix by summing up the distances by each diagonal. Assuming that the maximum distance threshold ε is 1.0, we see there are three $LSSSPs$, $\{S[1 : 4], P[0 : 3]\}, \{S[0 : 3], P[1 : 4]\},\ and\ \{S[1 : 4], P[2 : 5]\}$.

Having generate the initial $LSSSPs$, we can gradually generate all the $sClusters$ with larger cardinality using a modified-apriori mining algorithm shown in Al-



Distance Matrix
Distance Summary Matrix

	P[0]	P[1]	P[2]	P[3]	P[4]	P[5]
S[0]	0.4	0.4	0.7	0.3	0.1	0.7
S[1]	0.1	0.1	0.2	0.4	0.2	0.2
S[2]	0.2	0.1	0.6	0.1	0.3	0.4
S[3]	0.6	0.4	0.2	0.4	0.1	0.6
S[4]	0.2	0.1	0.5	0.1	0.3	0.3

	P[0]	P[1]	P[2]	P[3]	P[4]	P[5]
S[0]	0.4	0.4	0.7	0.3	0.1	0.7
S[1]	0.1	0.5	0.6	1.1	0.5	0.3
S[2]	0.2	0.2	1.1	0.7	1.4	0.9
S[3]	0.6	0.6	0.4	1.5	0.8	2.0
S[4]	0.2	0.7	1.1	0.5	1.8	1.1

Figure 9. Sample of pairwise sCluster

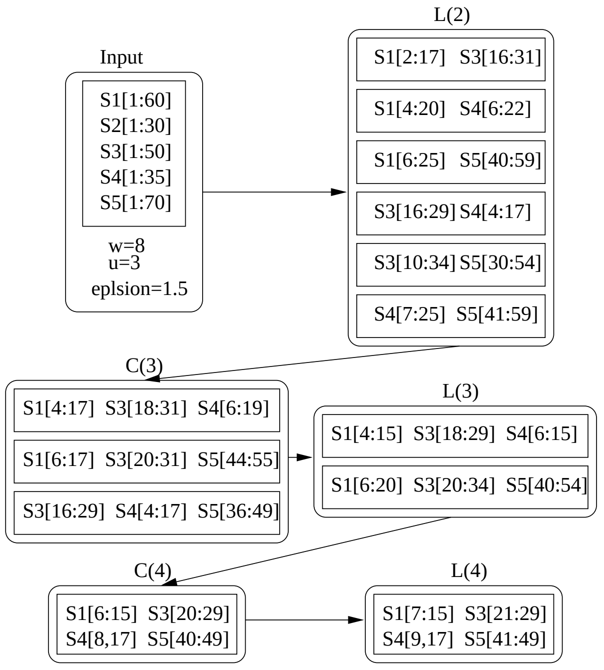


Figure 10. Sample of step 3 in Mining Maximum $sCluster$

gorithm 2. The algorithm is essentially a level by level algorithm in which candidate $sCluster$ containing $k+1$ $sObjects$ are generate at level $k+1$ using $sCluster$ that are found in the k level. Each of these candidates will then go through a verification process in Step 3.2 to test whether they are actually a $sCluster$. If this is so, they will be added to the list of $sCluster$s found at level $k+1$ which will in turn generate candidate $sCluster$s at the $k+2$ level. We will illutrate the algorithm using an example.

In Figure 10, we have a input of 5 $sObject$s, and our aim is to look for all maximum $sCluster$ with maximum distance threshold $\varepsilon = 1.5$, minimum length $w = 8$ and minimum cardinality $u = 3$. After computing $LSSSP$, we have $L(2)$ which includes 6 $sCluster$s. In the $sCluster$s, $\{S_1[2 : 17], S_3[6 : 31]\}$ and $\{S_1[4 :$

Algorithm 2 *Mining Maximum* $sCluster(T, \varepsilon, w, u)$

```
Input: a set of n sequence $T = \{S_1, S_2, ..., S_n\}$
        maximum distance $\varepsilon$,
        minimum length $w$
        minimum cluster cardinality $m$
Output: $maximum\ sCluster(D, \{S_1, ..., S_n\}, \varepsilon, w, m)$

1. Generate all $LSSSPs$
$L(2) \leftarrow \bigcup_{1 \le i < j \le n} LSSSP(S_i, S_j)$
for i = 1 to n
for j=i to n
   $L(k) \leftarrow L(k) + LSSSP(\{S_i, S_j\}, \varepsilon, L)$
2. k=2
3. while($L(k) \neq \emptyset$)
3.1 Generate candidates C(k+1) from $L(k)$
a. $C(k+1) \leftarrow \emptyset$
b. For any $E_1, E_2 \in L(k)$:
  $E_1 = \{S_1[b_1 : e_1], ..., S_{k-1}[b_{k-1} : e_{k-1}], S_x[b_x : e_x]\}$
  $E_2 = \{S_2[b'_2 : e'_2], ..., S_{k-1}[b'_{k-1} : e'_{k-1}], S_y[b_y : e_y]\}$
  i. If $\{S_1[b_1 : e_1], ..., S_{k-1}[b_{k-1} : e_{k-1}]\}$ is an $SSS$ of
       $\{S_2[b'_2 : e'_2], ..., S_{k-1}[b'_{k-1} : e'_{k-1}], S_y[b_y : e_y]\}$
    Then
      1)$[br_i, er_i] \leftarrow [b_i, e_i] \bigcap [b'_i, e'_i]$ , $i \in [1 : k-1]$
      2)If $er_1 - br_1 \ge w$
        Then
          a)$newCandidate \leftarrow$
            $\{S_1[br_1 : er_1], ..., S_{k-1}[br_{k-1} : er_{k-1}],$
            $S_x[b_x + br_1 - b1 : e_x + e_1 - er_1],$
            $S_y[b_y + br_1 - b1 : e_y + e_1 - er_1]\}$
          b)$C(k+1) \leftarrow C(k+1) + \{newCandidate\}$
3.2 Look-up $L(2)$ to refine each candidate in $C(k+1)$
a. $L(k+1) \leftarrow \emptyset$
b. For $E = \{S_1[br'_1 : er'_1], ..., S_{k-1}[br'_{k-1} : er'_{k-1}],$
            $S_x[br'_x : er'_x], S_y[br'_y : er'_y]\} \in C(k+1)$
    i. If there exists $\{S_x[b'_x : e'_x], S_y[b'_y : e'_y]\} \in L(2)$
       which is an $SSS$ of $\{S_x[br_x : er_x], S_y[br_y : er_y]\}$
    Then
      1)$[br'_x : er'_x] \leftarrow [br_x : er_x] \bigcap [b'_x : e'_x]$
      2)$[br'_y : er'_y] \leftarrow [br_y : er_y] \bigcap [b'_y : e'_y]$
      3)If $er'_x - br_x \ge w$
        Then
          a)$newCluster \leftarrow$
            $\{S_1[br_1 + br'_x - br_x : er_i + er'_x - er_x], ...,$
            $S_{k-1}[br_{k-1} + br'_x - br_x : er_{k-1} + er'_x - er_x],$
            $S_x[b'_x : e'_x], S_y[b'_y : e'_y]\}$
          b)$L(k+1) \leftarrow L(k+1) + \{newCluster\}$
4. Mark non-maximum $sCluster$
a) For each $B \in L(i)$, $i \in [2 : k]$
   If there exists $B' \in L(k+1)$ and $B \subset B'$
   Then mark $B$ as non-maximum sCluster
5. Return $\bigcup_{i=1}^{k} \{x : x \in L(i)$
          and $x$ is not a non-maximum $sCluster$
```

Figure 11. Algorithm for mining maximum $sCluster$

$20], S_4[6:22]\}$, we see

$$\{S_1[2:17]\} = SSS(\{S_1[4:20]\}, [-2,-3])$$

and in $L(2)$, we know $\{S_3[16:29], S_4[4:17]\}$ is an $sCluster$. It restricts the candidate $\{S_1[4:17], S_3[18:31], S_4[6:19]\}$ to $\{S_1[4:15], S_3[18:29], S_4[6:15]\}$. Similarly, we produce $C(3), C(4), L(3)$ and $L(4)$.

Property 5.1 *Algorithm 2 generates all qualified maximum $sClusters$ without false positive and negative.*

Proof: Firstly, L(2) includes all $LSSSP$s between any two $sObjects$. In step 3, for any longest $sCluster$ $B = \{S_1[b_1:e_1], S_2[b_2:e_2], S_3[b_3:e_3]$, there must exist $P = \{S_1[bp_1:ep_1], S_2[bp_2:ep_2]$, $Q = \{S_2[bq_2:eq_2], S_3[bq_3:eq_3]$ and $R = \{S_1[br_1:er_1], S_3[br_3:er_3]$ where

$$\begin{cases} [b_1:e_1] \subseteq [bp_1:ep_1] \; and \; [b_1:e_1] \subseteq [br_1:er_1] \\ [b_2:e_2] \subseteq [bp_2:ep_2] \; and \; [b_2:e_2] \subseteq [bq_2:eq_2] \\ [b_3:e_3] \subseteq [bp_3:ep_3] \; and \; [b_3:e_3] \subseteq [br_3:er_3] \end{cases}$$

According to Lemma 4.2, step 3 in Algorithm 2 will generate B and put it into $L(3)$ without loss. Similarly, we can deduct that all the longest $sCluster$s with k substructures will be in $L(k)$ without loss. In step 4, those longest $sCluster$ which could be covered by bigger $sCluster$ are marked as non-maximum $sCluster$. Thus, we know all the maximum $sCluster$s are found without false positive and negative. Furthermore, if we retaing the clustering results $\bigcup_{i=1}^{k} L(i)$, we can incrementally create new $sCluster$s when new sequential 3d objects are added.

6 Experiments

We implement the $sCluster$ algorithm in C. All our experiments are done on a PC with a Pentium 4 1.6Ghz CPU, 256MB of SDRAM and a 7200rpm 20GB harddisk running Windows XP. Since there are no other algorithm that discover $sCluster$, we design an alternative algorithm based on $rmsd$, named $rmsd$-$based\ Clustering$. Because $rmsd$ cannot be incrementally calculated, we adopt a step-by-step extension approach to explore the $LSSSP$. The extension will be proceeded when the current substructure pair is qualified but will terminated when the $rmsd$ is larger than the given threshold. After computing $LSSSP$, the method for generating and refining candidates and clusters is similar with that in $sCluster$.

Synthetic data sets and a set of real protein structures are used in our experiments. In the synthetic data sets, the value of the coordinates are randomly generated and ranging from 0-100. 4 synthetic 500-$sObject$ data sets are generated. The numbers of vertexes in the 4 data sets are 50, 100, 150 and 200 respectively. If we use $sObject$s less than 500, we randomly select required $sObject$s from these 4 data sets. The real data set consists of 200 protein 3d structures downloaded from PDB website where every two protein have similarity of amino acid sequence lower than 25%. On average, each protein has 227 $C\alpha$ atoms (vertexes). We truncate the long protein in order to investigate performance on shorter structure data sets.

7 Performance Analysis

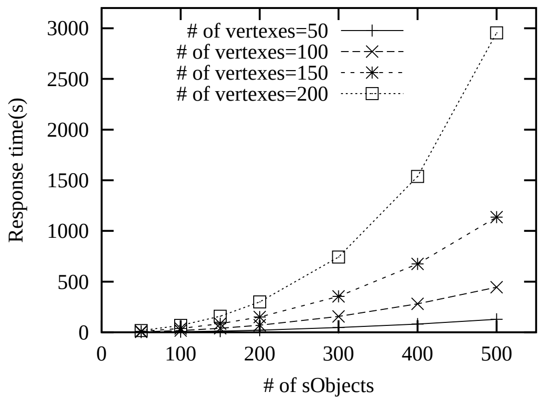


Figure 12. Response time vs. # of $sObject$s in synthetic data set

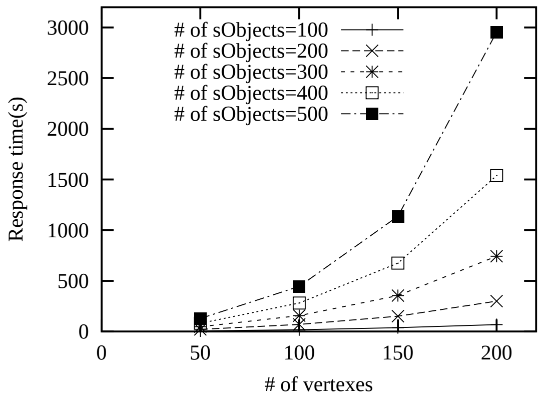


Figure 13. Response time vs. # of vertexes in synthetic data set

We evaluate the response time of the $sCluster$ algorithm as we increase the number of $sObject$s and average number of vertexes. Figure 12 and 13 presented the average response time of our algorithm on various synthetic datasets. We set the minimum distance threshold $\varepsilon = 4.5$, minimum length $w = 20$, and minimum cluster cardinality $u = 2$. The response time has a superlinear growth with respect to both the increase in the number of $sObject$s and the number of vertexes. The reason is that each ald between any two pair of vertexes is calculated during the computation of $LSSSP$.

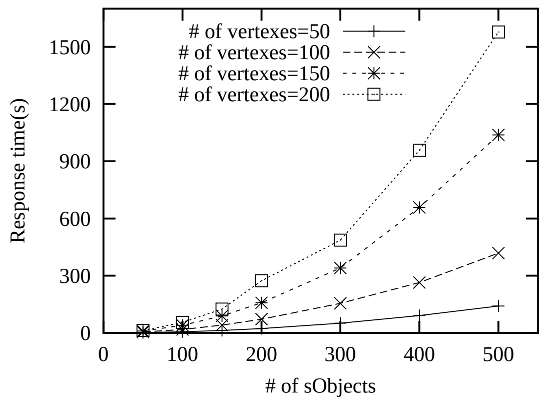


Figure 14. Response time vs. # of $sObject$s in protein dataset

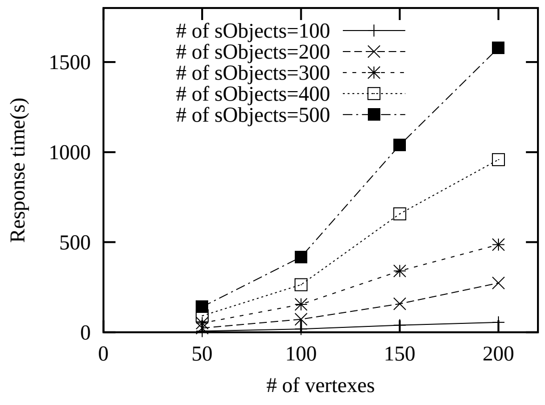


Figure 15. Response time vs. # of vertexes in protein set

Figure 14 and 15 showed the response time versus the dataset size and the average number of vertexes in the protein dataset. Since the edge length between two neighboring $C\alpha$ atoms varies slightly compared to that in the randomized data set, we invoke the clustering algorithm with a smaller minimum distance threshold ε as 1.7, and keep minimum length w as 20 and minimum cardinality u at 4. The observation here is similar to that for the synthetic datasets: the response time grows quickly with the data set size and average $sObject$ length.

The number of $sCluster$s being output are given in Figure 16 and 17. The number of $sCluster$ grows significantly with the average number of vertexes because the $LSSSP$s includes all the possible $SubObject$ pairs.

We investigated the distance threshold using 200 protein structures, each of which having 200 $C\alpha$ in average. In Figure 18, we see the response time of minimum length $w = 20$ increases significantly when $\varepsilon = 1.8$. This is because the growth of $sCluster$s leads to a superlinear increasing of the time for generating and refining candidates. In Figure 19, the growth of minimum length w leads to a significant increase in response time especially when the maximum distance threshold ε is large.

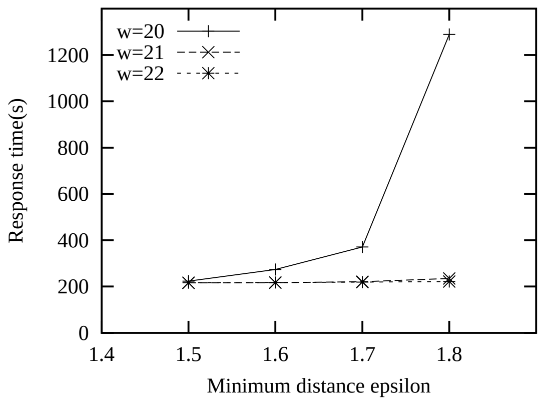


Figure 18. Response time vs. ε

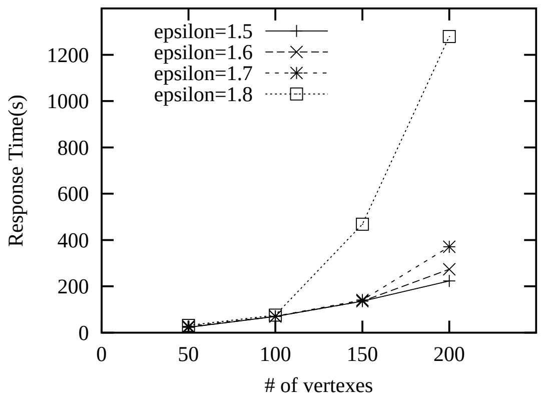


Figure 19. Response time vs. $\# \ of \ vertexes$

Next, we studied the effect of the minimum length

vertexes	u=2	u=3	u=4	u=5	u=6	time
50	1318	98				128
100	38592	17532	1084	5		444
150	117896	78177	7754	116		1136
200	246384	225588	36795	1734	8	2954

Figure 16. sClusters in synthetic data (# of sObject=200 $\varepsilon = 4.5$)

vertexes	u=2	u=3	u=4	u=5	u=6	u=7	time(s)
50	317	37	1				140
100	9602	6252	925	22			418
150	30683	40077	34100	49110	18982	3	1038
200	48409	71108	58240	72670	32792	157	1578

Figure 17. sClusters in real data (# of sObject=200 $\varepsilon = 1.7$)

threshold by invoking the algorithm with minimum length $w = 20$ and minimum cardinality $u = 4$. Figure 20 showed that the response time decreases drastically with the minimum length because the number of $LSSSP$ decreases significantly with the minimum length.

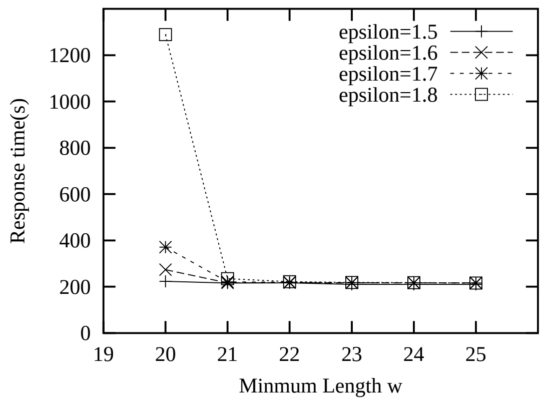


Figure 20. Response time vs. w

To illustrate the effectiveness of $sCluster$, we randomly select an $sCluster$ from the above experiment results on the real protein structure datasets, and plot them in Figure 21. The results are interesting. Each two of the four proteins, *Pdb1bpm*, *Pdb1ayl*, *Pdb1aln* and *Pdb1amx*, have amino acid sequence homology less than 25% so they are ignored by most of the existing protein motif mining methods. However, our $sCluster$ method find that their four substructures *Pdb1bpm[224:243]*, *Pdb1ayl[226:245]*, *Pdb1aln[67:86]* and *Pdb1amx[104:123]* share similar 3d shapes. This will be very interesting to biologists and drug-designers.

Finally, we compare the $sCluster$ algorithm with the $rmsd$-$based\ Clustering$ algorithm. Because the range of $rmsd$ value and the range of ald value are different, we set maximum distance threshold for $rmsd$ such that it generates almost as many clusters as the $sCluster$ algorithm. Due to the high computational complexity of the alternative algorithm which hardly produces results for large datasets in reasonably acceptable time, we use a small portion of the real protein data set for performance comparison. In the experiment for varying $sObject$ length, Figure 22 shows that $sCluster$ outperforms the $rmsd$-$based\ Clustering$ by a magnitude of 3. Likewise, when the number of $sObject$s is varied, Figure 23 also shows that $sCluster$ is faster than $rmsd$ Clustering by a magnitude of 3.

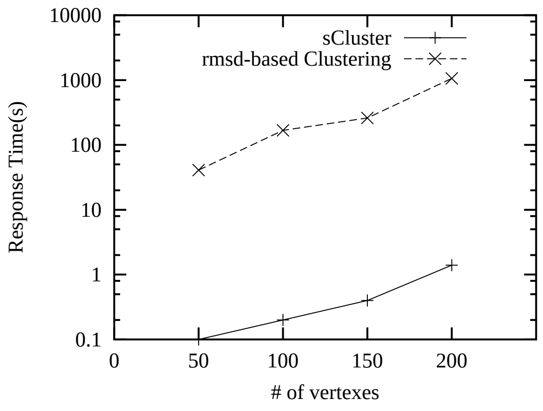


Figure 22. $sCluster\ vs.\ rmsd$-$based\ Clustering\ on\ various\ \#\ of\ vertexes$

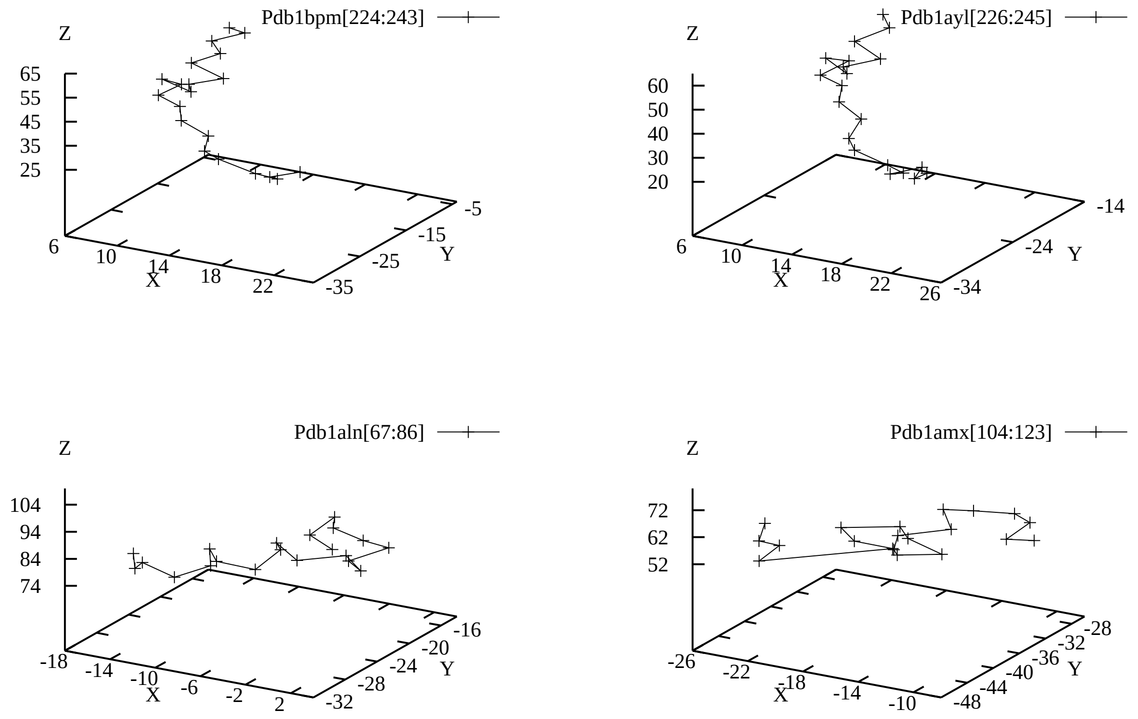


Figure 21. Result sample on real data set

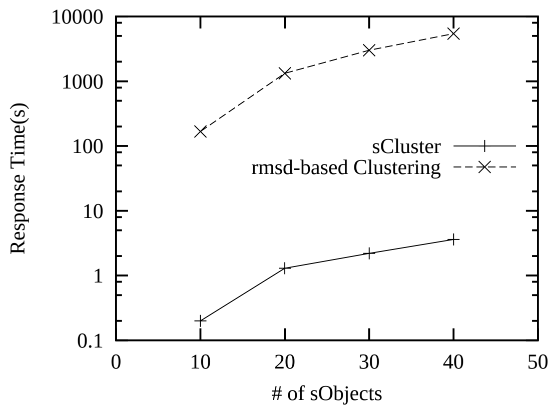


Figure 23. $sCluster\ vs.\ rmsd$-$based\ Clustering\ on\ various\ \#\ of\ sObjects$

8 Conclusions and Future Work

In this paper, we look at substructure clustering of of sequential 3d objects such as a set of consecutive C_α atoms in protein structures. Then we propose an $sCluster$ model to support these applications. An effective and efficient distance measurement, ald, is designed to evaluate the distance between sequential 3d objects. In addition, we devise a fast algorithm for discovering longest synchronized similar subobject pairs between two $sObject$s. Finally we develop a modified-apriori algorithm which efficiently find all maximum $sCluster$.

The applications of $sCluster$ model ranges widely from bioinformatics, biopharmaceutical research and moving-object group detection. In the future, the $sCluster$ could be improved to discover similar 3d objects rather than only sequential 3d structures. We can also extend the $sCluster$ algorithm by producing centroid for each cluster, leveraging on the centroid for speeding up the clustering process, and visualizing the centroid for better interpretation.

Acknowledgment: We would like to thank Guoguan Lu for his program to compute $rmsd$ on sequential 3d objects. And we also appreciate Xia Cao and Zeyar Aung for their helpful suggestions.

References

[1] I. Eidhammer, I. Jonassen, and W. R. Taylor. Structure comparison and structure patterns. *Journal of Computational Biology*, 7:685–716, 2000.

[2] L. G. Top: A new method for protein structure comparisons and similarity searches. *Journal of Appl. Cryst.*, 33:176–183, 2000.

[3] Y. Lamdan and H. Wolfson. Geometric hashing: A general and efficient model based recognition scheme. In *International Conference on Computer Vision*, pages 213–29, 1998.

[4] H. S. Nagesh, S. Goil, and A. N. Choudhary. A scalable parallel subspace clustering algorithm for massive data sets. In *International Conference on Parallel Processing*, 2000.

[5] P. Patel, E.Keogh, J.Lin, and S.Lonardi. Mining motifs in massive time series databases. In *ICDM*, 2002.

[6] B. Rost and C. Sander. Pitfalls of protein sequence analysis. *Current Opinion in Biotechnology*, 5:372–380, 1994.

[7] S. Srinivasa and S. Kumar. A platform based on the multi-dimensional data model for analysis of biomolecular structure. In *VLDB*, 2003.

[8] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data set. In *SIGMOD*, pages 394–405, 2002.

[9] X. Yan and J. Han. Closegraph:mining closed frequent graph patterns. In *KDD*, 2003.

[10] J. Yang and W. Wang. Cluseq: efficient and effective sequence clustering. In *ICDE*, 2003.

[11] J. Yang, W. Wang, H. Wang, and P. S. Yu. delta-cluster: Capturing subspace correlation in a large data set. In *ICDE*, 2002.

[12] J. Yang, W. Wang, P. S. Yu, and J. Han. Mining long sequential patterns in a noisy environment. In *SIGMOD Conference*, 2002.