A New Approach to Protein Structure Mining and Alignment*

Hongyuan Li, Keith Marsolo, Srinivasan Parthasarathy and Dmitrii Polshakov †
The Ohio State University
Columbus, Ohio, USA
li.274@osu.edu, {marsolo,srini}@cse.ohio-state.edu, dpolshak@chemistry.ohio-state.edu

ABSTRACT

One of the largest areas of focus in bioinformatic and data mining research has been on the protein domain. These research efforts have included protein structure prediction, folding pathway prediction, sequence alignment, *ab initio* simulation, structure alignment, substructure detection and many others. In this work, we deal with substructure detection and sequence alignment. Substructure detection is generally defined as the mining of a molecule's 3D structure in order to find interesting/frequent domains. Sequence alignment involves determining the similarity of two (or more) protein molecules based on the how well their amino acid sequences "match." There are potential pitfalls when trying solve both of these problems, however. In the case of substructure mining, focusing solely on structural information can lead to the discovery of biologically irrelevant substructures. With sequence alignment, the alignment results can vary greatly, depending on the substitution matrix used. In this paper we describe a method that combines the benefits of both substructure mining and sequence alignment in an attempt to determine the similarity between protein molecules. In the absence of biological information, our work will quickly and efficiently mine a protein molecule in order to determine frequent local structures. With the addition of biological sequence information, however, our algorithm provides a way to align proteins with similar local structures and sequence, yielding a global alignment between molecules. We present a novel structure mining/alignment algorithm as well as some additional work into a new clustering metric for amino acids based on several different physiochemical properties. This metric is used with our alignment algorithm in order to provide a mechanism for globally aligning protein molecules.

General Terms

Algorithms, Experimentation

*Work funded in part by NSF Career Grant IIS-0347662 (SP,KM)

†Department of Chemistry (HL, DP), Department of Computer Science and Engineering (KM, SP)
Note: Authors are listed in alphabetical order. Each contributed equally to this work.

Keywords

Protein structure alignment, sequence alignment, substructure discovery, multi-scale analysis

1. INTRODUCTION:

With the ever-increasing power and storage capacities of computers comes the ability to process larger amounts of information. Through endeavors such as the Human Genome Project [36] and the Sloan Digital Sky Survey [41], the amount of potential data has increased exponentially, to the point where new techniques are needed to analyze and comprehend it. One of the fastest-growing areas in computer science is that of data mining, or the process of deriving useful relationships and patterns from large stores of data. Data mining has been increasingly applied to problems in the scientific domain, especially bioinformatics, which involves the application of data mining techniques to biological datasets.

One of the richest research areas in bioinformatics has been in the protein domain. Proteins are often studied because they play an important role in a countless number of biological processes, yet there is still a great deal about proteins that is not understood. For instance, a protein can fold spontaneously and reproducibly into a three-dimensional structure when placed into aqueous solution. This transformation occurs in a fraction of a second, yet researchers still have not been able to determine the exact sequence of steps that cause a protein to fold. It is known that a protein's amino acid sequence uniquely determines its three-dimensional structure and that this structure influences the protein's biological function. Thus, if two proteins share a similar structure, they may have a similar biological function. While researchers have found that sequence influences structure, they have not yet determined the exact nature of the link between the two.

Substructure detection involves the mining of a protein's three-dimensional graph in order to find "interesting" (or possibly just frequent) structural motifs [4–6, 9, 12, 18, 25, 30, 33, 43]. By determining whether an previously unclassified protein contains certain structural motifs, one can make inferences as to the role it might play biologically. The problem with substructure detection algorithms is that the analysis methods are often quite complicated and require large amounts of time, memory, and computational resources to execute. With protein sequence mining, there has been a great deal of success in determining the similarity between proteins based on their amino acid sequence, yet through

evolution, it is possible for a protein's sequence to mutate. These mutations may not have any influence over a protein's structure or function, yet may lead to false notions of similarity between molecules. As a result, one would like to create a program that is able to combine the best of both worlds: have the ability to find interesting structural motifs within a protein, and then, using those motifs and a protein's amino acid sequence, construct an alignment between proteins that can be used to determine the similarity between molecules. In this paper we present work that is able to provide such functionality. By adding domain-specific extensions to a previously developed substructure mining algorithm [7, 8, 29, 37] our work makes the following contributions to research in the protein domain:

1. The ability to quickly and efficiently find local substructures within a protein molecule.
2. With the inclusion of biological sequence information, the ability to align local substructures to determine a global alignment between protein molecules.
3. The incorporation of a new classification for amino acids based on physio-chemical properties that allows for partial matching and partial alignment between molecules.

2. RELATED WORK AND BACKGROUND

2.1 Sequence Alignment

The idea of using alignment to determine protein similarity is not a new one. Programs like BLAST [15] and its refinements PSI-BLAST and gapped BLAST [16] have been used to align proteins based on their amino acid sequence. With the completion of the Human Genome Project [36] and other genome mapping initiatives, there is obviously a great need for such an alignment method. However, the number of new protein structures is growing enormously as well. The Protein Data Bank (PDB) [3] currently holds over 22,000 protein structures and is growing by almost 4,000 structures every year.

Much information about the structure of proteins can be found in the Structural Classification of Proteins (SCOP) database [34]. The SCOP database provides "a detailed and comprehensive description of the structural and evolutionary relationships of proteins," including information on a protein's secondary and tertiary structure. This information is derived by the visual inspection of the proteins in the PDB. The SCOP database is arranged into four different hierarchical levels: Class, Fold, Superfamily and Family. Proteins in the same Class share similar secondary structure information, while proteins within the same Fold have similar secondary structures that are arranged in the same topological configuration. Proteins within the same Superfamily show clear structural homology and proteins within the same Family exhibit a great deal of sequence similarity and are thought to be evolutionarily related.

2.2 Structure Alignment

There have been a number of methods proposed to compare protein structures. Some methods compare the secondary structures of the proteins; others try to align proteins based simply on their backbone configuration. A number of public tools exist that provide some type of alignment/similarity function, including DALI, STRUCTAL and LOCK. A brief description of each follows.

DALI [21] is based on the alignment of two-dimensional distance matrices, with the matrix values representing the distances between the C_α atoms of a protein. The algorithm attempts to find patterns of similar distances within two matrices. These patterns are combined with the intention of maximizing the number of atoms and minimizing the root mean square distance (RMSD) between them. DALI also uses a Monte Carlo optimization [32] to prevent the algorithm from quickly reaching a local minimum.

The STRUCTAL [17] algorithm uses an iterative dynamic programming [2] approach to align two proteins. The principal behind the algorithm is to minimize the RMSD between two protein backbones. First, the distance between all C_α carbons is computed. These distances are converted into a scoring matrix. Standard dynamic programming is employed to compute the optimal alignment of the two proteins. Since the solution to this algorithm depends heavily on the starting alignments of the two proteins, several different starting configurations are used.

LOCK [39] attempts to align proteins by using hierarchical structure superposition. A protein is decomposed into its secondary structures, which are represented as a series of vectors. A scoring matrix is created based on the vectors of the two proteins being aligned. Dynamic programming is then used to find the best local alignment between the vectors. Next, the algorithm attempts to iteratively minimize the RMSD between pairs of nearest atoms. Finally, a core of well-aligned atoms is created and the algorithm attempts to minimize the RMSD of the core.

2.3 Substructure Analysis

Discovering important structures in molecular datasets has been the focus of many recent research efforts in scientific data analysis [4–6, 9, 12, 18, 25, 30, 33, 43]. These efforts have targeted substructure analysis in small molecules, material defect analysis in molecular dynamics simulations, and more recently in macromolecules such as proteins and nucleic acids [21, 24, 46].

Several methods for secondary level motif finding in proteins have been proposed in the past. An algorithm based on subgraph isomorphism was proposed in [33]; it searches for an exact match of a specific pattern in a database. The search for distantly related proteins using a graph to represent the helices and strands was proposed in [25]. An approach based on maximally common substructures between two proteins was proposed in [18]; it also highlights areas of structural overlap. SUBDUE [9] is an approach based on Minimum Description Length for finding patterns in proteins. Another graph based method for structure discovery, based on geometric hashing, was presented in [43]. Recent work on graph data mining is also related to this effort [9, 18, 25–27, 33, 43, 45].

2.4 MotifMiner Toolkit

Our own attempts at substructure detection have resulted in the development of an extensible prototype toolkit, MotifMiner [7,8,29,37], that detects frequently occurring structural motifs. We have conducted a fairly in-depth evaluation of MotifMiner on various datasets, from pharmaceutical data [7], to tRNA data, to protein data (from the

PDB) [7,37] to data obtained from molecular dynamics simulations [6]. As stated previously, MotifMiner was designed to be extensible and here we present several extensions to the toolkit, some domain-neutral, others targeted specifically to proteins.

3. ALGORITHMS

The work described here is based on the MotifMiner project introduced in Section 2.4. A discussion of the general algorithm can be found in Section 3.1. Several basic extensions to the MotifMiner toolkit have been implemented and are presented in Section 3.2. A number of domain-specific constraints for the mining and alignment of proteins have also been developed. They are discussed in Section 3.3.

3.1 Background

MotifMiner represents the interaction between a pair of nodes A_i and A_j, as a *mining bond*. A node can be an atom, an amino acid, a secondary structure, etc., depending on the resolution desired. A mining bond $M(A_iA_j)$ is a 3-tuple of the form:

$$M(A_iA_j) = \{A_itype, A_jtype, AttributeSet(A_i, A_j)\}$$

The information contained in $AttributeSet(A_i, A_j)$ can vary depending on the resolution of the structure being represented. For instance, if the resolution of the structure is at the atomic level, $AttributeSet(A_i, A_j)$ could contain the distances between atoms A_i and A_j. At the secondary structure level, $AttributeSet(A_i, A_j)$ might contain the secondary structure type (α-helix or β-sheet), the number of residues within the secondary structure and so forth. Using the above definition, a *k-nodeset* is a substructure containing k connected (within a user-specified range) nodes, and is represented as:

$$X = \{\mathbf{S_X}, A_1, A_2, \ldots, A_k\},$$

where A_i is the i^{th} node and $\mathbf{S_X}$ is the set of mining bonds describing the nodeset. By defining pairs of nodes with mining bonds, the graph is completely represented, such that two nodesets X and Y are considered to be the same substructure if $\mathbf{S_X} = \mathbf{S_Y}$. Since we only deal with atoms in this work, we will refer to nodesets as *atomsets*. In addition, atomsets with a similar set of mining bonds are said to belong to the same *atomset family*, or *motif*.

Additionally, MotifMiner uses the following principles to generate frequent substructures: 1) *Range pruning* to limit the search for viable strongly connected sub-structures, 2) *Candidate pruning* [1], for pruning the search space of possible frequent structures, 3) *Recursive Fuzzy Hashing* for rapid matching of structures (to determine frequency of occurrence), and finally 4) *Distance Binning and Resolution* to work in conjunction with recursive fuzzy hashing to deal with noise in the input data.

Range pruning and candidate pruning reduce the candidate search space, thereby reducing the memory footprint and significantly improving the scalability of the algorithm. The biological motivation behind range pruning is that even though molecules are made up of atoms that interact with one another, there is only a finite distance over which such an interaction can occur. At a larger distance, the interaction between two atoms is essentially negligible and two atoms can be considered independent. As a result, by having a user-specified range parameter, it is possible to cut down on the number of potential atomsets.

1. Prune *infrequent* atoms (*1*-atomsets)
2. Generate candidate *2*-atomsets from *frequent* atoms
3. Prune *infrequent 2*-atomsets
4. $k = 3$
5. **while** (| *frequent* k-atomsets $|> 0$)
6. Generate candidate k-atomsets from *frequent (k-1)*-atomsets
7. Prune *infrequent* k-atomsets
8. $k = k + 1$

Figure 1: Local substructure discovery algorithm

Recursive fuzzy hashing, which is similar in principle to geometric hashing [28, 44], was designed to efficiently handle noise effects in data [37]. The idea behind distance binning and resolution is the data mining principle of discretization [13]. The raw Euclidean distance between two atoms is discretized by binning; This task is accomplished by choosing a *resolution* value and dividing the inter-atom distance into equi-width bins based on this value, represented efficiently as bits in the mining bond. Binning of the data simplifies calculations and helps MotifMiner handle *minor* fluctuations in distance.

As shown in Figure 1, atomsets of size *(i+1)* are derived by combining two frequent atomsets of size i that differ by one atom. Once an atomset has been generated, its frequency is determined using the following metrics:

- *atomsetSupport*- The number of atomsets in the atomset family.
- *coverRate*- The percentage of molecules that contain at least one atomset from the atomset family.

The minimum support thresholds for both parameters can be specified by the user.

3.2 Basic Extensions

In this section we present several basic improvements to the original MotifMiner algorithm. These extensions are domain-independent and can be applied to bio-molecular data of any type. These extensions were borne out of experimental testing of the original MotifMiner algorithm. They represent ways to improve both the running time and the quality of the results.

3.2.1 Variable Resolution

In the original version of MotifMiner, the resolution parameter is used to handle noise in the input data. One drawback with resolution in the original version is that the parameter is not flexible and cannot handle modulation in structure. The differences between two substructures can be very small in the short range, but as the substructures become larger, those differences are accumulated and magnified. Thus, the resolution is now variable. As the distance between two atoms increases, so does the resolution. This allows for the identification of larger similar structures. Figure 2 shows an example of this principle. With a sliding resolution, it is possible to identify an α-helix and a smaller helix from a helix-bundle as similar. Without variable resolution, the

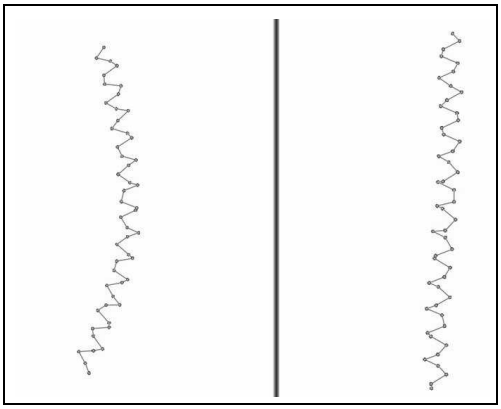


Figure 2: The twisted α-helices found in 1A02_J and 1A02_F (subunits of nuclear transcription complex).

```
1. Generate candidate atomset A
2. If atomCount_A ≤ minLinkage then
3.   ∀ atoms i ∈ A
4.     ∀ atoms j ∈ A (j ≠ i)
5.       if dist(atom i, atom j) > Range then
6.         discard A
7. Else
8.   ∀ atoms i ∈ A
9.     ∀ atoms j ∈ A (j ≠ i)
10.      count = count + 1
11.      if count < minLinkage then
12.        discard A
```

Figure 3: Local Structure Linkage Algorithm

bend in helix on left would have prevented it from being matched with the helix on the right.

3.2.2 Boundary Conditions

Another potential problem when dealing with noise in the input data is the handling of boundary conditions. For instance, if the range is specified to be 5Å, and the distance between two atoms i and j is 4.99Å, a mining bond will be created between the atoms. If the distance between them is 5.01Å, however, no bond will be created, which can cause problems when trying to determine substructure frequency. As a result, a mining bond will be created when the distance between two atoms is just over the range value.

3.2.3 Local Structure Linkage

The notion behind Local Structure Linkage is that an atomset should contain a minimum number of "close" points. In this case, "close" means that the distance between two atoms is less than the user-specified *Range* value. The minimum number of points is a user-specified parameter designated *minLinkage*. In most experiments, *minLinkage* was set to four.

The Local Structure Linkage algorithm is presented in Figure 3. The effect of the algorithm is to ensure that every atom in an atomset is within a distance *Range* of at least *minLinkage* atoms. Additionally, Local Structure Linkage makes use of another parameter: *initialRange*, where *initialRange* $\leq$ *Range* (see Figure 4). The effect of *initialRange* is to guarantee that each atom has at least one close neighbor and to eliminate many meaningless substructures. The results of an experiment testing the *minLinkage* algorithm are shown in Figure 5. Twenty-three molecules from each of the four major SCOP [34] fold classes (5-α, 7-β, 6-$\alpha+\beta$, 5-$\alpha\backslash\beta$) were mined looking for substructures with a minimum coverage of three molecules (a *coverRate* of 13%). Several substructures were found and are presented in Figure 5.

```
1. ∀ atomsets A with atomCount = 2:
2.   for atoms i, j ∈ A (j ≠ i)
3.     if dist(atom i, atom j) > initialRange then
4.       discard A
```

Figure 4: Effect of *initialRange* Parameter

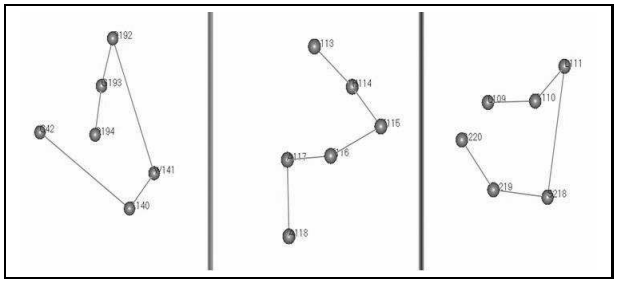


Figure 5: Examples of 6-atom local structures. Left, an unspecified local structure. Middle, partial α-helix. Right, partial anti-parallel β-sheet. Each of these three substructures occurs more than 100 times among the 23 molecules.

3.3 Domain Constraints

One of the most important contributions of this work is the incorporation of domain constraints into the original MotifMiner algorithm. Recall that MotifMiner was intended to be a general framework that could be used across multiple domains. By incorporating domain constraints, [35,40], it is possible to increase the utility of the original framework. Such constraints enable the researcher to interrogate the data while incorporating specific domain knowledge in the process. We have identified several such constraints which are described below.

3.3.1 Abstraction Using α-carbons

All proteins contain a backbone that is formed by peptide bonds. This substructure is very frequent and generates a number of trivial atomset families. Using the α-carbon of the amino acid as an abstraction of the peptide bond is a good way to reduce the number of atoms that need to be examined and enhance the speed of the algorithm. We do lose information about the chemical linkage of the peptide bond with such an abstraction, but compensate for the loss by including information about the amino acid sequence.

3.3.2 Sequence-based Pruning of Motifs

We also incorporate domain constraints that integrate sequence information. This has useful possibilities for the alignment of two proteins and also results in the detection of biologically relevant motifs. Specifically, the algorithm has an option wherein candidate atomsets can be pruned based

on the relative sequence ordering of the atomsets. In terms of implementation, this constraint is relatively straightforward. In addition to the mining bond information, information about the sequence order is maintained in the *Attribute-Set* parameter of the mining bond. When two substructures are compared, we first compare the relative sequence ordering and then match the substructures. We demonstrate the use of this constraint in our global alignment algorithm (Section 3.4).

3.3.3 Approximate Matching of Amino-Acids Based on Physio-Chemical Properties

As an additional extension, we now support approximate sequence matches where a particular amino acid is replaced by a label that represents amino acids which share similar physio-chemical properties (hydrophobicity, helical propensity, etc.). To implement this feature, we used a multi-dimension description of the amino acid space that included a large number (243) of physio-chemical properties that were collected from a number of different sources. In addition, we extended that list of physio-chemical properties with properties obtained from quantum chemical calculations. We used the Gaussian 98[1] program to compute properties such as ground state energy, dipole moment, and vibration frequency for all 20 amino acids.

To reduce the inherent redundancy in the physio-chemical property space, we relied on the technique of multi-scale analysis [42]. This method involves the multi-dimensional scaling of the high-dimension physio-chemical property space to a lower dimensional space using a PCA-style reduction [23]. We found that the first five Eigenvectors sufficed to capture more than 95% of the total inertia of the data. Figure 6 shows the projection of the amino acids on the first two principal-component dimensions (left), and the first and third principal-component dimension (right). After computing the eigenvectors, we used the K-means clustering algorithm [31] to group amino acids by Euclidean distance in 5D space. The result of clustering is shown in Table 1.

Some of the clusters in Table 1 are similar to the results found in [42]. When K=4, for example, residues I, V, L, F, and M fall into the same cluster. This cluster consists of hydrophobic amino acids. The cluster of amino acids W, Y and C consists of polar residues. At high K values (K >5) this cluster separates into two, one of which contains just the aromatic residues W and Y. Another noticeable cluster found in several different levels contains the small residues N, D, S, T, G and P. As shown in Table 1, residues G and P always fall into the same cluster. This result agrees with experimental observation, as it is known that residues G and P play an important role in the determining the 3D architecture of a protein [38]. They are frequently located in the linkage between secondary structures; for example, between two α-helices or between an α-helix and a β-sheet.

To model the cost of a replacement we use the following principle, depending on whether a coarse-grained or fine-grained cost model is desired. For a coarse-grained level, the cost of a replacement is 1 if two amino acids are in different clusters and 0 if they are in the same cluster. At a more fine-grained level we simply use the distances between amino acids in the scaled dimensional space to quantify the cost of replacement. A user-specified threshold determines whether a replacement or a series of replacements in a structure is acceptable or not.

[1] http://www.gaussian.com

1. Generate i-atomsets using the local substructure discovery algorithm and a *coverRate* of 100%
2. **If** there exists any *ambiguity* among atomsets **then**
3. Increment i, go to step 1 and repeat until the *ambiguity* is resolved.
4. **Else**
5. Begin alignment.

Figure 7: Ideal Alignment Preprocessing

1. Generate i-atomsets using the local substructure discovery algorithm and a *coverRate* of 100%.
2. **If** there exists any *ambiguity* among atomsets **then**
3. **If** *out-of-memory* **then**
4. Force alignment.
5. **Else**
6. Increment i, go to step 1 and repeat until the *ambiguity* is resolved or *out-of-memory*.
7. **Else**
8. Begin alignment.

Figure 8: Modified Alignment Preprocessing

3.4 Global Alignment

The most significant contribution of this work is the development of a global alignment method that aligns protein molecules based on their structure as well as their sequence. The alignment algorithm works by generating frequent local substructures and then, starting with the largest local structures discovered, attempts to assemble an alignment between two molecules.

Alignment Preprocessing

Before the global alignment of two molecules can occur, several preprocessing steps must be taken, starting with the generation of local substructures. A high-level presentation of the preprocessing steps is given in Figure 7.

In the ideal case, the substructure generation algorithm would be able to execute as shown in Figure 7. In practice, however, the number of frequent atomsets is usually very large, often to the point where they do not all fit into memory. When this occurs, we say that the local substructure discovery algorithm is *out-of-memory*. As a result, we must modify the preprocessing steps we take before the alignment can begin. The modified algorithm is shown in Figure 8.

In the algorithms presented in Figures 7 and 8, the term *ambiguity* is used to denote when there are atomsets from each molecule that belong to the same atomset family but do not contain exactly the same types of atoms (this can occur due to recursive fuzzy hashing). Thus, it is possible for a single atom in an atomset to align with multiple atoms in the other atomsets in the family. For example, given the 4-atomset families shown in the top table of Table 2, there is ambiguity when aligning atom D. It can align with either atom D' or atom E.

In order to solve this problem, the substructures at the next level are generated to see if they resolve the ambiguity. Since

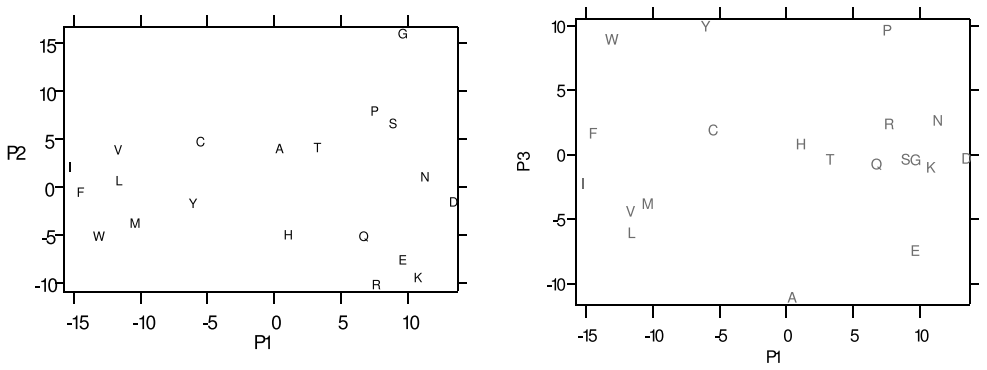


Figure 6: Multi-scale projection of amino acids on Components 1 and 2 (left) and Components 1 and 3 (right).

K	Clusters
2	{A,R,N,D,Q,E,G,H,K,P,S,T}{C,I,L,M,F,W,Y,V}
3	{A,R,Q,E,H,K}{N,D,G,P,S,T}{C,I,L,M,F,W,Y,V}
4	{A,R,Q,E,H,K}{N,D,G,P,S,T}{C,W,Y}{I,L,M,F,V}
5	{A,E}{R,Q,H,K}{N,D,G,P,S,T}{C,I,L,M,F,V}{W,Y}
6	{A,E}{R,Q,H,K}{N,D,S,T}{C,I,L,M,F,V}{G,P}{W,Y}
7	{A,E}{R,Q,H,K}{N,D,S,T}{C}{G,P}{I,L,M,F,V}{W,Y}
8	{A,E}{R,Q,H,K}{N,D,S,T}{C}{G,P}{I,M,V}{L,F}{W,Y}

Table 1: K-Means clustering of amino acids based on multi-dimensional scaling

Family 1	Family 2
ABCD	ABCD
A'B'C'D'	A'B'C'E'
Level 4	

Family 1
ABCDF
A'B'C'D'F'
Level 5

Table 2: Ambiguity between Families. In the 4-atomsets, atom D can possibly align with atoms D' and E. By growing the atomsets to the next level, the ambiguity is resolved.

the atom sequences of Family 2 differ, it will not be expanded at the next level. The atomsets of Family 1 *will* be used in the next level, however. After the next step of the the substructure discovery phase, suppose Family 1 now contains the atom sequences shown in the bottom table of Table 2. There is no longer any ambiguity. D can align with D' and F will align with F'. In this manner, the ambiguity is resolved. As shown in Figure 8, the alignment preprocessing algorithm runs until there are no ambiguous substructures between the molecules or until the program runs out of memory, whichever comes first. Once this point is reached, the assembly of the alignment between the molecules can begin.

Initial Alignment Assembly

When the alignment preprocessing algorithm finishes, we are left with two possible cases. In the first case, the algorithm is able to finish without any ambiguity among the atomsets. When this occurs, all of the atomsets at the highest level reached by the substructure generation algorithm (i.e. the largest frequent substructures discovered) are used as the basis for the starting alignment. This is considered to be *normal assembly*. The other case occurs when the algorithm runs out of memory before resolving all of the ambiguities between atomsets (i.e. atomsets contained in the same atomset family do not have exactly the same atom types). When this occurs, the algorithm is said to start with *forced assembly*. Before assembling the alignment, the algorithm attempts to find the atomset families that have the fewest conflicts with the other atomset families at the same level (in this case, the highest level reached by the substructure generation algorithm before running out of memory). The total number of conflicts is defined as the number of sequence conflicts between atomsets in the same family. For example, suppose an atomset family contained the 3-atomsets ABC and ABD. This family would contain one conflict: the conflict between atoms C and D. Given the family of 3-atomsets ABD and AEF, there would be two conflicts: atoms BD and atoms EF. The algorithm attempts to find the atomset families with the smallest number of conflicts and use them as the starting alignment.

Alignment Assembly

Once the initial alignment has been determined, the alignment assembly can begin. Suppose that the largest substructures found by the initial alignment algorithm are of size *n*. The assembly algorithm examines the atomsets at level *n-1* and determines whether there are any conflicts (using the measure of conflict defined above) between those atomsets. If there are, the alignments with fewer conflicts are given a higher priority. Any candidate atomset (i.e. non-conflicting or a conflicting with a high priority) at this new level is

1. Determine the initial (existing) alignment from level-n atomsets.
2. $n = n - 1$.
3. **while** $n > minLevel$
4. Determine conflicts among level-n atomsets.
5. Remove any atomset that conflicts with the existing alignment.
6. Merge any remaining atomsets with the existing alignment. The resulting set becomes the new existing alignment.
7. $n = n - 1$
8. **Return** the set of atomsets as the global alignment.

Figure 9: Alignment Assembly Algorithm

checked against the existing alignment (i.e. the larger atomsets). If there is any conflict between the candidate atomset and an atomset in the existing alignment, the candidate atomset is removed. Once this step has completed, all of the remaining candidate atomsets are added to the existing alignment. The algorithm then examines the atomsets at the next lower level. These steps repeat until the algorithm reaches a lower limit of potential atomset size that is specified by the user. A pseudo-code description of the algorithm is shown in Figure 9.

4. VALIDATION

In the following examples, we present the results of several experiments that serve as a preliminary validation of our global alignment algorithm.

4.1 Alignment of FHA Domains

The FHA domain is a phospho-protein binding domain. It was originally identified using sequence alignment [20]. However, FHA domains have very few conserved residues (only three residues are completely conserved) and sequence alignment only detected the core region. Later, after the structures of FHA domains were solved, the full domain was demonstrated to cover a much larger region than the core region. We used our global alignment to align the proteins Rad53 and Chk2. The aligned result is very similar to those obtained through manual alignment [14]. The results are shown in Figure 10.

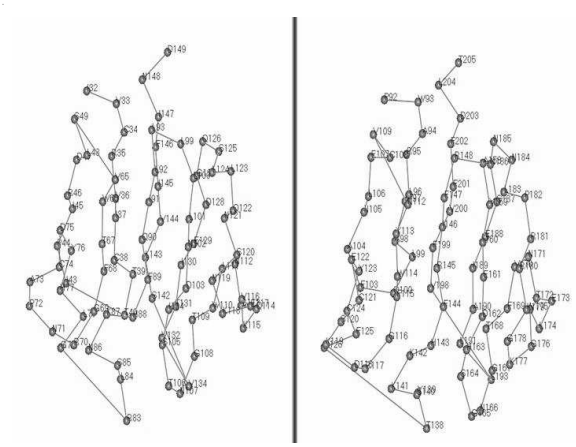
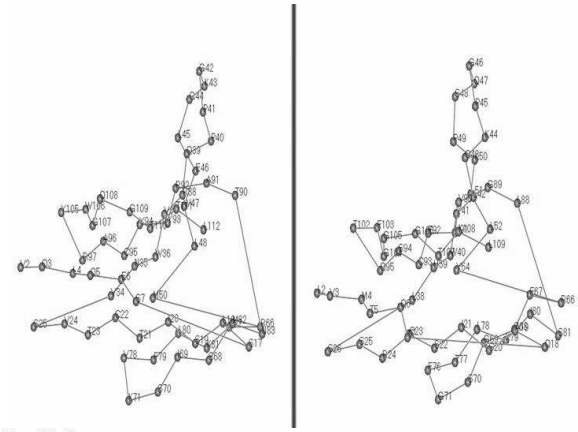


Figure 10: Structural alignment of two FHA domains. FHA1 of Rad53 (left) and FHA of Chk2 (right)



```
74_pdb1a2y_B
2-VQLQES^SLSITCTVS^VNWVRQPPGKGLEWL_M^RLSISK^VFLKMN^TARYYCAR^YWGQGTTL-112
113_pdb1a4j_L
2-LVMTQT^QASISCRSS^LHWYLQKPGQSPKLL_Y^RFSGSG^FTLKIS^LGVYFCSQ^TFGGGTKL-109
```

Figure 11: Sequence alignment of pdb1a2y_B and pdb1a4j_L "_" indicates one space, "^" indicates more than 2 spaces

4.2 Alignment of Sequentially Distinct Proteins

Pair-wise structural alignment generates a number of possible sequence alignments that are very hard to align using just a scoring matrix. To give one example, we found that proteins pdb1a2y_B and pdb1a4j_L give an alignment of 49 α-carbons. These corresponding residues have a very similar substructure (Figure 11, top). The resulting sequence alignment of the substructure is shown in Figure 11, bottom. We attempted to align these proteins based on sequence information only, using the scoring matrices BLOSUM 62 [19] and PAM 250 [10, 11]. Neither scoring matrix was able to give a clear result, however (Results omitted due to space constraints).

Sequence alignment has two major difficulties: How to choose scoring matrix and how to estimate gap cost. These two problems no longer exist in structure-aided sequence alignment. We can give solid parameters to control the similarities of the structures and if there is any gap in the sequence, it is omitted by structure alignment. Thus, structure is more conserved than sequence since all amino acids share the backbone structure.

4.3 Alignment using Physio-Chemical Properties

Since the structural alignment algorithm in this paper uses α-carbons only, the side chain information is ignored to some extent. In many proteins, however, the side chain plays an important role in their activity. Adding amino acid constraints helps defray this loss and such constraints will also help identify residues that contribute more than just backbone linking in structure. Fewer α-carbons need to be aligned which usually speeds up the program.

As a final example, in Figure 12 we show the results of align-

Cluster	# Proteins	Function
1	23	Antibodies
2	11	Hydrolases (mainly serine proteases)
3	7	Transferases (mainly kinases)
4	6	Oxygen Transport (including heme proteins)
5	5	Oxidoreductase (Dehydrogenease)
5	5	Haloperoxidase

Table 3: Top six protein clusters based on the alignment of 312 non-redundant proteins and an alignment threshold of at least 56 atoms

ing two calcium-binding proteins, 1AHR and 5CPV, with the inclusion of physio-chemical properties (amino acid constraints) and without. Our alignment was comparable to one obtained through DALI and it should be noted that we able to verify the existence of the calcium-binding site in our results.

4.4 Alignment-Based Clustering

As an experimental test of our alignment algorithm, we ran an all-against-all alignment of 312 non-redundant (sharing less than 20 amino acids in sequence) proteins and then clustered the results based on the number of atoms that can be aligned between the molecules. We set an alignment threshold of 56 (meaning at least 56 atoms can be aligned between the molecules) and were left with 218 clusters. Most of the clusters contained a single protein molecule, however there were several clusters that contained multiple proteins, and even more striking, the cluster proteins showed functional similarity in addition to their structural similarity. The clusters containing more than five proteins are shown in Table 3. As mentioned above, the dataset uses only non-redundant proteins. Thus, most of the closely related proteins are removed from the dataset. However, antibodies are generated through gene shuffling, which leads to large sequence diversity. The clustering results shows that these antibodies, though different in sequence, still share structural similarity.

4.5 Comparison with DALI

In this work we present a new method for the alignment of protein molecules based on local substructures as well as sequence information. There are a number of other publicly-available protein alignment methods that work based on structure, DALI being one of the most popular. We reran several of our experiments using DALI and DaliLite (a stand-alone version of the DALI Server) [22] to see how our results compared to the results returned by those programs. We found that our results were comparable to what was returned by DaliLite, differing by only a few amino acid residues at most. Aligning proteins pdb1a2y_B and pdb1a4j_L (discussed in Section 4.2) with DaliLite yielded the same amino acid sequence that our program found (Figure 11, bottom). In addition, the running time of our algorithm was equivalent to the running time of DaliLite (or faster). Our algorithm runs in a completely different fashion than DALI (and DaliLite), so it is difficult to compare running times. DALI works by computing a pair-wise distance matrix for each protein and then uses a Monte Carlo optimization procedure to try and minimize the distance between the matrices. The resulting alignment is returned as the best alignment between proteins. Our algorithm works by mining local substructures and then using the underlying sequence information to determine conflicts between structures. Structures that do not conflict are aligned. Those that do are not. It is interesting to note that although both methods are orthogonal in nature, they produce consistent results, provided that we do not include physio-chemical properties. When we do include such properties, we achieve results that, while more concise, still retain their biological relevance.

5. CONCLUSIONS

In this work we present extensions to MotifMiner that allow for the efficient detection of substructures in protein molecules using both biological and structural information. These extensions enable us to detect substructures that vary due to the noise inherent in protein data and to approximate a molecule's amino acid sequence based on varying physio-chemical properties. We have tested our algorithm against a well-established structural alignment tool, DALI, and found that our work performs favorably, even providing some benefits not available in DALI. One benefit that our algorithm has over DALI is that we are able to handle the chirality inherent in some protein molecules. Chirality refers to the "handedness" of a protein. By only dealing with pair-wise distances, DALI is not able to distinguish between chiral proteins. Our algorithm can make such a distinction. In addition, we have the ability vary the physio-chemical properties in our cost analysis. With further testing, we hope to provide more examples as to the usefulness of our algorithm as well as a statistical metric that can be used to determine the quality of the results returned by our program.

6. ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers of this paper for their comments and suggestions.

7. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *20th VLDB Conf.*, September 1994.

[2] Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., 1974.

[3] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: A computer-based archival file for macromolecular structure. *J. Mol. Biol.*, 112:535–542, 1977.

[4] C. Borgelt and M. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *IEEE International Conference on Data Mining*, Dec 2002.

[5] L. P. Chew, D. Huttenlocher, K. Kedem, and J. Kleinberg. Fast detection of common geometric substructures in proteins. *RECOMB*, 1999.

[6] M. Coatney, S. Mehta, A. Choy, S. Barr, S. Parthasarathy, R. Machiraju, and J. Wilkins. Defect detection in silicon and alloys. In *IEEE Workshop on Visualization in Bioinformatics and Cheminformatics*, 2002.

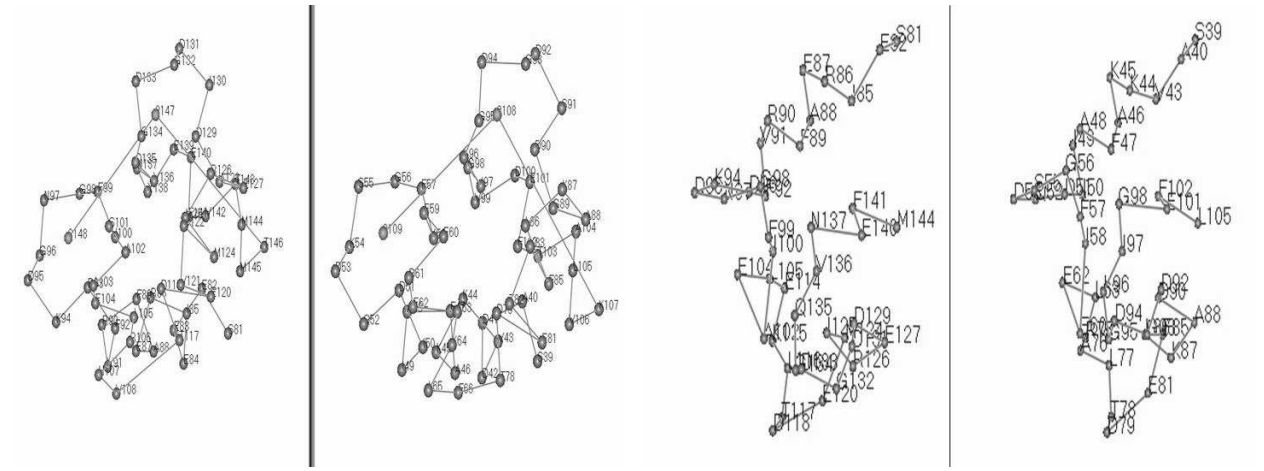


Figure 12: The result of structurally aligning two calcium-binding proteins. The two proteins (1AHR,5CPV) on the left were aligned using structure alignment. The same two proteins were then aligned using approximate matching based on physio-chemical properties. It is apparent that the alignment on the right is contained within the structure-based alignment, however, the structure on the right is, as a whole, more biologically active than the structure on the left. This gives credence to the idea that the incorporation of physio-chemical properties into the alignment algorithm produces more biologically relevant results without losing the important structural motifs.

[7] M. Coatney and S. Parthasarathy. Motifminer: A general toolkit for efficiently identifying common substructures in molecules. In *IEEE International Conference on Bioinformatics and Bioengineering (to appear)*, 2003.

[8] M. Coatney and S. Parthasarathy. Motifminer: Efficient discovery of common substructures in biochemical molecules. In *Knowledge and Information Systems, to appear*, 2003.

[9] D.J. Cook, L.B. Holder, S. Su, R. Maglothin, and I. Jonyer. Structural mining of molecular biology data. *IEEE Engineering in Medicine and Biology*, 20(4):67–74, 2001.

[10] M. O. Dayhoff and R. M. Schwartz. Matrices for detecting distant relationships. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5 of *3*, pages 353–358. Natl. Biomed. Res. Found., Washington, DC, 1978.

[11] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5 of *3*, pages 345–352. Natl. Biomed. Res. Found., Washington, DC, 1978.

[12] L. Dehaspe, H. Toivonen, and R. King. Finding frequent substructures in chemical compounds. In *The Fourth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1998.

[13] J. Dougherty, R. Kohavi, and M. Shami. Supervised and unsupervised discretization of continuous features. In *ICML*, 1995.

[14] D. Durocher, I. A. Taylor, D. Sarbassova, L. F. Haire, S. L. Westcott, S. P. Jackson, S. J. Smerdon, and M. B. Yaffe. The molecular basis of fha domain:phosphopeptide binding specificity and implications for phospho-dependent signaling mechanisms. *Mol Cell*, 6(5):1169–82, Nov 2000.

[15] S.F. Altschul et al. Basic local alignment search tool. *J. of Mo. Biol.*, 1990.

[16] S.F. Altschul et al. Gapped blast and psi-blast: A new generation of protein database search programs. *Nucleic Acids Research*, 1997.

[17] M. Gerstein and M. Levitt. Using iterative dynamic programming to obtain accurate pair-wise and multiple alignments of protein structures. In *Proc. Fourth Int. Conf. on Intell. Sys. for Mol. Biol*, pages 59–67, Menlo Park, CA, 1997. AAAI Press.

[18] H.M. Grindley, P.J. Artymiuk, D.W. Rice, and P. Willett. Identification of tertiary resemblence in proteins using a maximal common subgraph isomorphism algorithm. *J. of Mol. Biol.*, 229(3):707–721, 1993.

[19] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. In *Proc. Natl. Acad. Sci*, volume 89, pages 10915–10919, USA, 1992.

[20] K. Hofmann and P. Bucher. The fha domain: A putative nuclear signalling domain found in protein kinases and transcription factors. *Trends in Biochemical Science*, 20:347–349, 1995.

[21] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233:123–138, 1993.

[22] Liisa Holm and Jong Park. Dalilite workbench of protein structure comparison. *Bioinformatics*, 16(6):566–567, 2000.

[23] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, NY, 1986.

[24] I. Jonassen, I. Eidhammer, D. Conklin, and W. Taylor. Structure motif discovery and mining the pdb. In *German Conference on Bioinformatics*, 2000.

[25] I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *J. of Comp. Biol.*, 3(2):289–306, 1996.

[26] M Kuramochi and G. Karypis. Frequent subgraph discovery. In *IEEE International Conference on Data Mining*, Nov 2001.

[27] M Kuramochi and G. Karypis. Discovering frequent geometric subgraphs. In *IEEE International Conference on Data Mining*, Dec 2002.

[28] Y. Lamdan and H. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. In *ICCV*, 1988.

[29] H. Li and S. Parthasarathy. Automatically deriving multilevel protein structures through data mining. In *HiPC Conference Workshop on Bioinformatics and Computational Biology*, Hyderabad, India, 2001.

[30] R. Machiraju, S. Parthasarathy, D. S. Thompson, J. Wikins, B. Gatlin, T. S. Choy, D. Richie, M. Jiang, S. Mehta, M. Coatney, S. Barr, and K. Hazzard. Mining Complex Evolutionary Phenomena. In H. Kargupta *et al.*, editor, *Data Mining for Scientific and Engineering Applications*. MIT Press, 2003.

[31] J MacQueen. Some methods for classification and analysis of multivariate observation. In L.M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[32] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[33] E.M. Mitchell, P.J. Artymiuk, D.W. Rice, and P. Willett. Use of techniques derived from graph theory to compare secondary structure motifs in proteins. *J. Mol. Biol.*, 212:151–166, 1990.

[34] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol*, 247:536–540, 1995.

[35] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *SIGMOD*, pages 13–24, 1998.

[36] U.S. Department of Energy. Human genome program. *Genomics and Its Impact on Science and Society: A 2003 Primer*, 2003.

[37] S. Parthasarathy and M. Coatney. Efficient discovery of common substructures in macromolecules. In *IEEE International Conference on Data Mining*, 2002.

[38] J. S. Richardson and D. C. Richardson. Principles and patterns of protein conformation. In G. D. Fasman, editor, *Prediction of protein structure and the principles of protein conformation*, pages 43–75. Plenum, 1989.

[39] A. P. Singh and D. L. Brutlag. Hierarchical protein structure superposition using both secondary structure and atomic representations. In *Proc. Fifth Int. Conf. on Intell. Sys. for Mol. Biol*, pages 284–293, Menlo Park, CA, 1997. AAAI Press.

[40] R. Srikant and R. Agrawal. Mining generalized association rules. In *21st VLDB Conf.*, 1995.

[41] Alexander S. Szalay, Peter Z. Kunszt, Ani Thakar, Jim Gray, Don Slutz, and Robert J. Brunner. Designing and mining multi-terabyte astronomy archives: the Sloan Digital Sky Survey. In *Proc. ACM SIGMOD*, pages 451–462. ACM Press, 2000.

[42] M. S. Venkatarajan and W. Braun. New quantitative descriptors of ami no acids based on multidimensional scaling of large number of physical-chemical properties. *Journal of Molecular Modeling*, 7:445–453, 2001.

[43] X. Wang, J.T.L. Wang, D. Shasha, B.A. Shapiro, I. Rigoutsos, and K. Zhang. Finding patterns in three-dimensional graphs: Algorithms and applications to scientific data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):731–749, jul/aug 2002.

[44] H. Wolfson and I. Rigoutsos. Geometric hashing: An overview. In *In IEEE Computational Science and Engineering*, Oct 1997.

[45] X. Yan and J. Han. gspan: Graph based substructure pattern mining. In *IEEE International Conference on Data Mining*, Dec 2002.

[46] X. Zheng and T. Chan. Chemical genomics: A systematic approach in biological research and drug discovery. *Current Issues in Molecular Biology*, 2002.