Discovering Spatial Relationships Between Approximately Equivalent Patterns in Contact Maps *

Hui Yang, Keith Marsolo, Srinivasan Parthasarathy and Sameep Mehta
Department of Computer Science and Engineering
The Ohio State University
Columbus, Ohio, USA

{yanghu, marsolo, srini, mehtas}@cse.ohio-state.edu

ABSTRACT

We present a method for finding relationships between approximate patterns in contact maps. We examine contact maps generated from protein data in order to discover spatial relationships among the connected patterns contained in those maps. We discuss our criteria for determining whether two patterns are approximately equivalent as well as the motivation behind our work. Finally, we provide results that validate our efforts.

General Terms

Algorithms, Experimentation, Performance

Keywords

Spatial association mining, pattern-set mining, approximation, contact maps

1. INTRODUCTION

Discovering important structures in molecular datasets has been the focus of many recent research efforts in biological and chemical informatics. These efforts have targeted, for example, substructure analysis in small molecules and macromolecules such as proteins and nucleic acids, as well as material defect analysis in molecular dynamics simulations [13; 24; 7; 36; 11; 12; 26; 31]. Most of the work in discovering substructures in molecules has focused on representing the molecule as a 3-D graph and finding frequent subgraphs that are contained within [16; 33; 17; 21; 14; 6; 5; 32]. A problem occurs, however, when trying to determine whether two subgraphs are equal. In general, the problem of subgraph isomorphism is NP-complete, and as such, any efficient solution will require the use of heuristics or similar techniques to keep the running time manageable. Another approach used recently has been to represent a molecule as a contact map. The principle behind a contact map is to only represent the interactions between points, as opposed to an entire three-dimensional structure. Using such a representation reduces the dimensionality of the problem down to a more manageable size. A contact map is essentially an adjacency matrix, where matrix position *A(i,j)* will be set to 1 if residues (or atoms, depending on the resolution used) are "in contact" and 0 otherwise. The definition of "in contact" can change depending on the data being examined, but most applications use the Euclidian distance between two atoms, with a user-specified distance as a cutoff threshold.

We are interested in using contact maps to represent protein molecules. A protein is composed of a series of amino acids. This sequence is commonly referred to as the protein's primary structure. When placed in aqueous solution, a protein will "fold" into a three-dimensional structure, with the structure uniquely determined by the protein's sequence. While the exact steps that a protein undergoes while folding is unknown, it is known that a protein will fold into a series of substructures (α-helices and β-sheets), referred to as secondary structures and these substructures will fold into larger structures, called tertiary structures. Trying to determine the steps, or the pathway that a protein follows while folding remains an open problem in biology. In the protein domain, contact maps are useful in that they provide a visual representation of the secondary structures that make up a protein molecule. For instance, α-helices show up as thick bands on the main diagonal and β-sheets appear as bands either parallel or anti-parallel to the main diagonal, depending on the conformation of the secondary structure. In addition to reducing the dimensionality of the dataset and providing a method of visualization, representing a molecule as a contact map also allows for the efficient use of bit-wise operations during implementation.

In the protein domain, contact maps have been used for a number different applications, including molecular alignment, fold prediction, and the discovery of *non-local structures* (or patterns) [10; 9; 18; 29]. We are also interested in mining contact maps to discover non-local structures, however, we intend to look for *spatial relationships* between the patterns across multiple contact maps, not just within a single map. In a contact map, non-local patterns are indicative of interactions between the tertiary structures of a protein molecule. Thus, if we can find relationships between non-local patterns across several different contact maps, we might be able to shed some insight into the protein folding problem. Finally, we would like to cluster a set of proteins based on the relationships that we generate to determine whether there is any correlation between those relationships and a molecule's function. By incorporating information from a database like the Structural Classification of Proteins (SCOP) database [23], which classifies proteins based on their 3-D structure, we would like to make predictions

*This work was supported by NSF Career Grant IIS-0347662 and NSF Grant CCF-0234273

about a molecule's function based on its contact map. In addition, we would also like to see if the reverse is true: whether we can use the spatial relationships we discover to generate rules that can be used to describe a protein's function.

One problem with protein data is that it is inherently noisy. Therefore, one cannot treat the distances between atoms as absolute. Two different crystallizations of a protein might yield slightly different coordinates for a molecule and lead to different contact maps. Thus, we need to derive a method to discover approximate patterns, and define a notion of approximate equivalence.

In this paper we present an algorithm for generating spatial associations based on approximate patterns within a contact map. We give an overview of the related work in this field in Section 2. Our algorithm is presented in Section 3. We show our experimental results in Section 4 and provide our conclusions and future goals in Section 5.

2. RELATED WORK

A great deal of work has gone into the area of using contact maps in the protein domain. Hu et al. have looked into mining contact maps to generate frequent dense patterns [10]. Additional work has gone into mining non-local patterns in contact maps [9]. A number of researchers have attempted to structurally align two protein molecules by solving the the Maximum Contact Map Overlap problem [18; 4]. Others have shown that it is possible to reconstruct a protein's structure from a contact map even in the presence of a large amount of noise [30]. Zhao and Karypis have developed a technique to predict a molecule's contact map using Support Vector Machines (SVM), which can be beneficial in fold recognition and structure determination [35]. Several groups have looked into using contact maps and the principles of energy minimization to create a system to recognize a protein's folds [28; 19; 3]. Finally, contact maps have been used to create a heuristic solution to the protein fold prediction problem [29].

A number of researchers have been looking into the area of spatial association mining. Koperski et al. [15] have used the technique to find association rules in geographic information system (GIS) databases. The Spatial Mining for Data of Public Interest (SPIN!) project[1] has looked into the mining of GIS databases as well as other areas such as census data. These efforts have primarily looked at defining associations based on a set of spatial predicates. Others have proposed methods to discover metric-based spatial associations [22; 8], though these metrics are defined over points, not over objects.

3. ALGORITHM

In this section, we describe the major steps taken towards generating approximate pattern-associations in contact maps. An overall description of our algorithm is given in Figure 1. We will now describe each step in further detail.

3.1 Contact Map Generation

When generating a contact map, one can examine the distances between individual atoms, between residues, or even secondary structures, depending on the resolution desired.

[1] http://www.ccg.leeds.ac.uk/spin/index.html

1. Generate contact maps for protein molecules.
2. Identify maximally connected patterns for each map and represent each pattern as a *feature vector*.
3. Cluster the *feature vectors* into *approximately* equivalent groups using a *k-means*-based clustering method.
4. Choose the optimal number of clusters based on the clustering entropy.
5. Re-label each pattern in a contact map with its corresponding cluster label.
6. Create an *occurrence vector* for each occurrence of a labeled pattern.
7. Generate *spatial pattern-associations* based on the *occurrence list*

Figure 1: Pattern-Association Mining Algorithm

We chose to look at the distances between the α-carbons (C_α) of each amino acid. Thus, for a protein with N amino acids, we will generate a binary matrix of size $N \times N$. We define each position *C(i,j)* in the contact map in the following manner: Given two amino acids a_i and a_j, if $d(a_i, a_j)$, the Euclidian distance between the C_α atoms of a_i and a_j, is less than a user-specified threshold t, then $C(i,j) = 1$. Otherwise, $C(i,j) = 0$. Since a contact map is symmetric across the diagonal, we only examine half of the matrix when running our experiments. In addition, we ignore the protein backbone (the 1s on the diagonal) in all of our tests.

3.2 Extracting Maximal Connected Patterns

Every non-edge position *(i,j)* in a contact map has eight *neighbor bits* at locations $(i\pm1, j\pm1)$, $(i\pm1, j)$, and $(i, j\pm1)$. For edge positions, we assume the out-of-bound neighbor bits to be 0. We define a *bit pattern* or simply, a *pattern*, to be an arbitrary collection of 1 and 0 bits. A *connected pattern* is a pattern where, for every position that contains a 1, there is a neighboring bit that is also set to 1. The *minimum bounding rectangle (MBR)* is the minimum rectangle that encloses a pattern. We define a *maximally connected pattern* (also referred to as a *feature* in this article) to be a connected pattern p where every neighbor bit not in p is 0. We apply a simple region growth algorithm to identify all *maximally connected patterns* within every protein contact map in a dataset. Connected patterns of size 1 are not considered.

3.3 Generation of Feature Vectors

One of the issues raised when working with contact maps is how to represent a feature. Several different methods have been employed, each with varying success. One simple approach is to represent a feature as a set of positions *(i,j)* where each position in the set corresponds to a 1 in the original pattern [9]. This method works best when the patterns are sparse and spread over a large area. An alternative approach is to represent a pattern as an array of bit strings [10]. Both of these approaches work well when the patterns examined are relatively small. When the number of patterns and the patterns themselves are large, however, both representation methods require an unacceptable amount of storage space.

In this work, we often deal with features that contain thousands of 1s and since we are attempting to identify non-local

features across a large set of contact maps, we must store thousands of unique (and potentially large) patterns. It is clear that representing every 1 in a pattern is not a viable option. Therefore, we must use an approximate representation, one that captures a feature's major characteristics, is storage-efficient and is easily explainable and interpretable. We propose a method using the following fields to represent a pattern:

- *Height*: the number of rows contained in a pattern's MBR.
- *Width*: the number of columns in a pattern's MBR.
- *NumOnes*: the number of 1s in a pattern.
- *Angle*: the general linear distribution trend of all the 1s in the pattern within its MBR.
- *xStdDev*: the standard deviation of all the 1s' x-coordinates (this quantifies how the 1s spread along the x dimension).
- *yStdDev*: the standard deviation of all the 1s' y-coordinates.

Thus, a *feature vector* is a 6-tuple consisting of the above fields. The reason that we require both the height and width of a pattern's MBR instead of simply using the area is that we believe two patterns should be considered "different" when one MBR has a different number of rows or columns than the other, even if both MBRs have the same area. To compute the angle of a connected pattern we use the least-squares method to estimate the slope of a linear regression line. For a pattern containing n 1s, we denote the positions of the 1s as: $(x_1, y_1) \ldots (x_n, y_n)$. The least-squares method then estimates the slope (β_1) as:

$$\beta_1 = \sum_{i=1}^{n}((x_i - \overline{x}) * (y_i - \overline{y}))/\sum_{i=1}^{n}((x_i - \overline{x})^2)$$

Notice that β_1 is a real number in the range $\pm\infty$. This makes the comparison of two patterns' β_1 values difficult. Therefore, we convert the β_1 value of each pattern to its corresponding angle off the x-axis. After this conversion, the values of an angle are in the range of [0, 180). After the feature generation step, we are left with a set of feature vectors. We then normalize those vectors to decrease the impact of attributes with a large range of values.

3.4 Clustering

Our next step is to place the maximally connected patterns into approximately equivalent groups. Two common methods can be used to do this: classification and clustering. Classification is a supervised procedure which requires the user to pre-label a set of connected patterns in order to build up a set of decision rules. Such a requirement is difficult to meet because it requires a domain expert's participation, which is impractical in this case due to the large number and variety of features that are generated. Thus, clustering is used to group the features into *approximately equivalent groups*. Besides being an unsupervised procedure, by using an appropriate similarity metric, a clustering algorithm can place similar elements together while separating dissimilar items. We consider each group generated from the clustering procedure to be an approximately equivalent pattern group. A pattern is assigned to the group to which its feature vector has the highest similarity.

When determining the similarity between two patterns, we believe the most significant parameters of the feature vector to be the dimensions of the MBR. As a result, similar patterns should have similar-sized MBRs. We ensure this property by weighing the *height* and *width* attributes more than the others when clustering the feature vectors.

Once all the vectors have been clustered, we re-label each pattern with its corresponding cluster label. By re-labeling the patterns, we are left with a much smaller set of feature types, as opposed to a large number of individual features. This enables us to study the spatial relationship between patterns in a more effective and efficient manner. We are guaranteed that the information "lost" by our clustering method is minimized by our clustering scheme, discussed next.

Quantitatively Measuring the Clustering Quality

After the completion of any clustering algorithm, one should measure the "goodness" of the clusters. Informally, a "good" cluster is one that has high *intra*-cluster similarity and low *inter*-cluster similarity. If one takes the opposite view and measures the quality of a cluster based on the *dissimilarity* of the features within that cluster, one is left with the quality measure of *entropy*. The higher a cluster's entropy, the greater the degree of dissimilarity among the members of that cluster. Given a set of events e_1,e_2,. . ., e_n, where the probability of an event $e_i's$ occurrence is p_i, then the entropy (H) of the set is defined as:

$$H = -p_i * log_2(p_i)$$

Each feature vector in our dataset is composed of 6 attributes. When computing the entropy of a cluster, we need to compute it in such a way that ensures every attribute contributes to the final entropy value. In addition, once we have computed the entropy for each cluster, we cannot simply sum them to determine the goodness of a clustering run because some clusters are larger than others and thus should not carry the same weight. We propose a goodness measure that weighs each individual cluster's entropy by that cluster's size in relation to the size of the entire dataset. Thus, for a dataset of N records, partitioned into k clusters, c_1,. . ., c_k, where a cluster c_i $(1 \leq i \leq k)$ has an individual entropy H_i and contains N_i elements, then the total entropy of this clustering is given by the following formula:

$$H = \sum_{i=1}^{k} H_i * (N_i/N)$$

Now we look at computing the individual entropy of a cluster. We compute the entropy of a cluster using the non-normalized feature vectors. As stated previously, each feature vector is composed of 6 attributes. The first three attributes, *Height*, *Width* and *NumOnes* are discrete, while the remaining attributes, *Angle*, *xStdDev* and *yStdDev* are continuous. For a discrete attribute, we take every unique value of that attribute in the cluster as a single event. We count the total number of occurrences for that value and then compute the probability of this value by dividing the number of times it occurred by the number of feature vectors in the cluster. For the *Angle* attribute, we assume it has a uniform distribution and compute its entropy as follows:

1. For all the vectors in a cluster, compute the minimum and maximum angle values, denoted $Angle_{min}$

and $Angle_{max}$.
2. Partition the interval [$Angle_{min}$, $Angle_{max}$] into equi-width intervals of length 30.

Each interval is treated as a single event, and we are able to compute the entropy for the *Angle* attribute exactly the same way as we compute it for a discrete attribute. For the other two attributes, *xStdDev* (σ_x) and *yStdDev* (σ_y), we assume they follow a Gaussian distribution and therefore their entropy can be computed by the following formula [27]:

$$H(x) = \log_2(\sqrt{2\pi e} * \sigma_x)$$

Finally, the entropy of a cluster is computed as:

$$H_i = \sum_{i=1}^{6} H(Attribute_i)$$

Choosing the Cluster Size

In order to pick the "optimal" number of clusters for grouping our feature vectors, we run the k-means clustering algorithm [20] on different k values. We then compute the entropy for each run using the method described above and finally, we plot the entropy vs. the number of clusters and choose a value k where the entropy plot begins to show a linear trend.

3.5 Mining Spatial Pattern-Associations

Creation of an Occurrence Dataset

Once the number of clusters has been chosen, we re-label each pattern with its cluster label, i.e. the cluster ID, and for each occurrence of a pattern in a contact map, we create an entry with the following fields:

- p_i: the cluster ID of the pattern.
- m_i: the contact map ID where the pattern is located.
- r_i: the row number of the pattern's MBR's upper left bit within the contact map.
- c_i: the column number of the pattern's MBR's upper left bit within the contact map.
- h_i: the height of the pattern's MBR at location (r_i, c_i) within the contact map.
- w_i: the width of the pattern's MBR at location (r_i, c_i) within the contact map.

The above representation is analogous to the *vertical format* structure used for frequent association mining [34]. The vertical format allows us to efficiently generate spatial pattern associations, as we will see shortly. From this point on, we only deal with the re-labeled patterns.

Computing Pattern Distance

Before we define the problem of spatial pattern-set mining, let us first define how to compute the distance between two connected patterns. The distance between two patterns p_1 and p_2 is defined only if they occur in the same map. Two types of metrics can be used to compute the distance between two patterns, with the first type defined over their feature vectors and the second over their spatial shapes and locations in a map. We do not consider the first type as it does not reflect the spatial distance between two patterns. Several distance metrics are available based on spatial shape and location. They include the Hausdorff distance [1], the distance between the center 1-bits in both patterns, the shortest distance between two 1-bits from each pattern, and the distance between two patterns' MBRs.

In this work, we use the last metric, the distance between two patterns' MBRs. There are several reasons that we use such a distance metric. It gives an approximate spatial distance between two patterns and is easy to explain. Also, it has a better scalability compared to other metrics such as the Hausdorff distance. There are three different cases that can occur when computing the distance between two MBRs:

- *Case 1 (Overlap)*: If two MBRs are overlap, then the distance between them is 0
- *Case 2 (Parallel)*: If two MBRs are parallel to each other, then the distance between them is the Euclidian distance between the two closest edges.
- *Case 3 (Other)*: If two MBRs are neither overlapping nor parallel, the distance between them is the minimum Euclidian distance between any pair of vertices.

Spatial Pattern Creation

Given n spatial patterns $P = \{p_1, p_2, \ldots, p_n\}$, and k 2-D maps $M = \{m_1, m_2, \ldots, m_k\}$, a spatial dataset D can be described as: $D = \{E_i\}$,
where $E_i = <p_i, m_j, r_i, c_i, h_i, w_i>$, and $p_i \in P$ and $m_j \in M$
In the context of contact maps, each E_i corresponds to one occurrence of a maximal connected pattern, with p_i being the pattern's cluster ID. Given a spatial dataset D as described above, we define the problem of spatial association mining as the identification of associations which are not only frequent over the 2-D maps in M, but also meet a user-specified pattern distance criterion.

A *pattern association* or *pattern-set* S of size k is one that consists of k patterns $\{p_0, p_1, \ldots, p_{k-1}\}$, where $p_i \in P$ and $0 \leq i \leq (k-1)$ and $distance(p_0, p_i) \leq maxDist$, where $0 < j \leq (k-1)$ and $maxDist$ is a user-specified distance threshold. Thus, a pattern-set S covers a *circular* area on a 2-D map, with its center located in p_0 and its radius no greater than $maxDist$. p_0 is also called the *center pattern* of S. Unless otherwise noted, the first pattern in S is its center pattern. A pattern-set of size k is denoted as a *k-set*. The *support* of a pattern-set is the percentage of contact maps in the dataset in which it occurs. A *frequent pattern-set* is one whose support is greater than or equal to a user-specified parameter *minSupport*. Note that when we say a pattern association is in a given map, we currently do not consider its specific occurrences in the map, just that it exists. We plan to integrate information regarding in-map occurrences into our future work.

A pattern set S_1 is a *sub-pattern-set* of S_2, if: $\forall p_i \in S_1, p_i \in S_2$ and they have the same center pattern. Accordingly, S_2 is a *super-pattern-set* of S_1. For instance, <A,B,C> is a sub-pattern-set of the set <A,B,C,D>. A *maximal frequent pattern-set* is one that does not have a frequent super-pattern-set.

Pattern-Set Generation: Basic Algorithm

The basic principle of our pattern-set generation algorithm is to generate pattern-sets in an increasing level-wise manner, starting with pattern-sets of size 1. The first step is to identify all the individual patterns that reside in at least *minSupport* contact maps. Given that all pattern occurrences are organized in the vertical format representation,

this step is fairly easy to implement. The second step is to generate all frequent 2-sets, the third step to generate k-sets where $k > 2$, and the last step is to generate only maximal pattern sets, which is optional.

The *anti-monotonicity* property of a frequent spatial pattern-set is used to facilitate the generation of frequent k-sets with $k \geq 2$. The property of anti-monotonicity states that a pattern-set cannot be frequent if one of its sub-pattern-sets is infrequent. Therefore, when generating k-sets, we only need to consider those where all the *(k-1)*-sub-pattern-sets are also frequent. We define a *candidate pattern-set* as one where all its sub-pattern-sets are frequent.

In the basic algorithm, for a candidate 2-set $< p_i, p_j >$, a brute-force method is used to check whether it is frequent by examining all possible location combinations of p_i and p_j in a map. Such a method has very poor performance, given that a pattern can occur at multiple locations within a contact map.

A similar process is used to generate all frequent k-pattern-sets of size $k > 2$ by first generating candidate k-sets based on the frequent $(k-1)$-sets, then computing their support to see if they meet the *minSupport* threshold. Since only circular pattern-sets are considered in this work, we do not need to compute the pattern distance in this step. For example, if we know both $< A, B >$ and $< A, C >$ occur at location (m_i, r, c), where m_i is the ID of a map, and (r,c) is the location of the upper left bit of the MBR, then we know $<A,B,C>$ must also occur at (m_i, r, c), as we are sure that both B and C are within the distance of $maxDist$ from A at (m_i, r, c). By the same token, if both $S_1 = \{s_0, s_1, \ldots, s_{k-1}, p\}$ and $S_2 = \{s_0, s_1, \ldots, s_{k-1}, q\}$ occur at position (m_i, r, c), then $S_3 = \{s_0, s_1, \ldots, s_{k-1}, p, q\}$ must also occur at (m_i, r, c).

Pattern-Set Generation: Optimizations

Two optimization techniques are employed to improve performance when generating all 2-pattern-sets. One quickly eliminates the maps that do not contain a certain candidate pattern-set, the other prunes patterns that are sure not to be within $maxDist$ of a pattern.

To eliminate maps that do not contain a certain candidate 2-set, we assign each map in the dataset a unique ID that remains fixed throughout the entire algorithm. By doing this, we can record the IDs of the first and last map where a pattern or pattern-set appears, denoted as m_{min} and m_{max}. We can then define an interval $[m_{min}, m_{max}]$ which represents a superset of the maps in which a pattern or pattern-set exists. For a 2-set $< p_i, p_j >$, we intersect p_i's $[m_{min}, m_{max}]$ interval with that of p_j's. Then we decide whether a further step is needed to determine this set's support. If the intersected interval spans fewer than $minSupport$ maps, such a set can be immediately discarded; otherwise, a further step is needed to decide whether it is frequent, which can be done much faster than the non-optimized version since we now have a much smaller set of maps to examine.

In order to prune away patterns that are certain to be greater than $maxDist$ from a given pattern, the following method is used. For a given pattern p_i, we order its occurrences by (r_i, c_i) values. Once we have all of a single pattern's locations ordered in a map, the following step can be taken to prune far-away patterns. For a pattern p at location (r, c) in a map, where h is the height of p's MBR at this location and (r,c) is the location of the upper left bit of p's MBR, we first divide the map into the following 3 areas:

- A_1-the area above the row with row number $=\lceil$(r-$maxDist$)$\rceil$
- A_2-the area under the row with row number $=\lceil$(r+h+$maxDist$)$\rceil$
- A_3-the remaining area

We determine into which of these three areas another pattern q is located before computing its distance to p. If q lies in either in A_1 or A_2, we are sure that it cannot be close to p. Since a pattern's occurrences in a map are ordered, we can use a binary search to mark the last occurrence of q in A_1 and the first occurrence of q in A_3. Once this marking is complete, we only need to compute the distances from p to q between the two marked occurrences (i.e. if q lies in A_2).

The other optimization technique introduced to improve the performance is the usage of *equivalence classes*. An equivalence class is a collection of frequent pattern-sets, where all sets have the same *prefix*. A set's prefix is composed of all the patterns in a set except the last one. The size of an equivalence class is defined as the size of its corresponding pattern-sets. Obviously, all sets in an equivalence class have the same center pattern. For instance, suppose $P = \{A, B, C\}$. P would have the following size-2 equivalence classes: $\{< A, A >, < A, B >, < A, C >\}$, $\{< B, A >, < B, B >, < B, C >\}$, and $\{< C, A >, < C, B >, < C, C >\}$ (Note: we assume all the pattern-sets are frequent). One potential size-3 equivalence class for P is $\{< A, A, A >, < A, A, B >, < A, A, C >\}$.

The optimized algorithm to generate frequent pattern sets of size greater than 2 is as follows: We first partition all frequent 2-sets into equivalence classes. As demonstrated in [25], all equivalence classes are independent of one another. Therefore, we can work on one equivalence class a time to generate larger frequent sets. This allows us to efficiently mine frequent spatial associations in a large dataset, as we are dealing with equivalence classes individually instead of the whole dataset.

Evaluating Frequent Pattern-Sets

For a frequent pattern-set, one would like to define a measure of "usefulness." This measurement is often subjective and domain-specific. In the protein context, we propose using a pattern-set's entropy to measure its "usefulness." We do this by integrating the SCOP lineage information of a pattern-set's associated proteins. We realize several other public databases also provide a method of structure-based protein classification, and that their classifications for a given protein may disagree, but for the time being, we use the SCOP classification.

For each frequent pattern-set, we identify the list of proteins contained in that pattern-set. We then classify these proteins into different groups based on a protein's SCOP lineage. A protein's SCOP lineage is organized into 6 levels: $L_1 : class$, $L_2 : fold$, $L_3 : super\text{-}family$, $L_4 : family$, $L_5 : protein$, and $L_6 : species$. In our experiments, we look at the first 4 levels.

Once the N proteins contained in a pattern-set S are classified at a certain SCOP level, we compute the entropy to measure how well these proteins are distributed among different SCOP categories. Take *L1: class* as an example. L_1 is divided into 11 sub-classes, denoted $\{c_1, c_2, \ldots, c_{11}\}$. When computing the entropy for S at this level, we first

count the number of proteins in each sub-class, denoted $\{n_1, n_2, \ldots, n_{11}\}$. The entropy is computed as:

$$H(S) = \sum_{i=1}^{11} -(n_i/N) \times log_2(n_i/N)$$

The pattern-set generation algorithm provides a parameter that allows a user to specify a maximum entropy at a given SCOP level. As with other user-specified parameters, the value of this parameter differs from dataset to dataset and is determined empirically.

As we discussed in Section 2, there has been a great deal of work toward the mining of spatial associations. We feel that our work differentiates itself from existing efforts in a number of areas:

- Vertical format data representation: To the best of our knowledge, there has not been any work in spatial data mining that represents a database using a vertical format. Accordingly, we are the first to use "equivalence classes" to expedite the process of mining spatial data.
- Metric-based spatial associations: Unlike the spatial associations proposed in [15], where a spatial association is defined over a set of spatial predicates (such as *close_to()* and *west_of()*, which are pre-defined and can only approximately describe the relationships between spatial objects), in our work, the relationship between spatial objects in a pattern-set is accurately quantified by Euclidian distances.
- 2-D object-oriented spatial associations: Existing metric-based spatial association mining algorithms [22; 8], have defined their distance metrics over *points* instead of *objects*. In this case, however, using points instead of objects can lead to information loss. The distance metric implemented here functions over actual 2-D objects; in this case, MBRs. The metric quantifies the topological relationship between two MBRs when they overlap or are parallel to each other and takes into account the size of the MBRs otherwise.
- Quantitative measurement of "interestingness": An entropy-based measurement is proposed to indicate whether a particular pattern-set is "interesting."

4. EXPERIMENTAL RESULTS

In this section, we present the experimental results carried out on 3 different datasets.

4.1 Datasets

To generate our contact maps, we used proteins taken from the Protein Data Bank (PDB) [2]. We generated three different sets of contact maps using a cut-off distance between amino acids of 4.5Å, 6Å, and 7.5Å. Table 1 shows the datasets generated. Also given in the table are the number of unique feature vectors, the total number of feature occurrences and the average number of features per protein.

4.2 Clustering Results

To cluster the feature vectors, we used a k-means-based clustering algorithm, where the Euclidian distance between two feature vectors is used as the similarity metric. As mentioned previously, in order to choose an *optimal* number of clusters, we ran the clustering algorithm multiple times for each dataset with different values for k. Once we obtained the clustering results, we computed the entropy for each cluster and plotted the weighted sum of the entropy versus the number of clusters (k). Based on the plot, we chose the value of k where entropy became approximately linear. When this criterion leads to multiple solutions, we chose the one that has smaller $H(Height)$ and $H(Width)$, i.e. the one that gives a tighter clustering solution in terms of a connected pattern's height and width.

Figure 2 shows the entropy plots for the three datasets. Based on the entropy values and $H(Height)$ and $H(Width)$, 24 and 32 clusters are the k values selected for all three datasets. We evaluated both and the results using 24 clusters were always worse in terms of the quality of pattern-sets than those obtained using 32 clusters (see Section 4.3). One possible explanation for this result is that placing the features into 24 approximate equivalence groups eliminates the information that can distinguish one pattern from the other when 32 groups are used.

4.3 Evaluation of Circular Spatial Pattern Sets

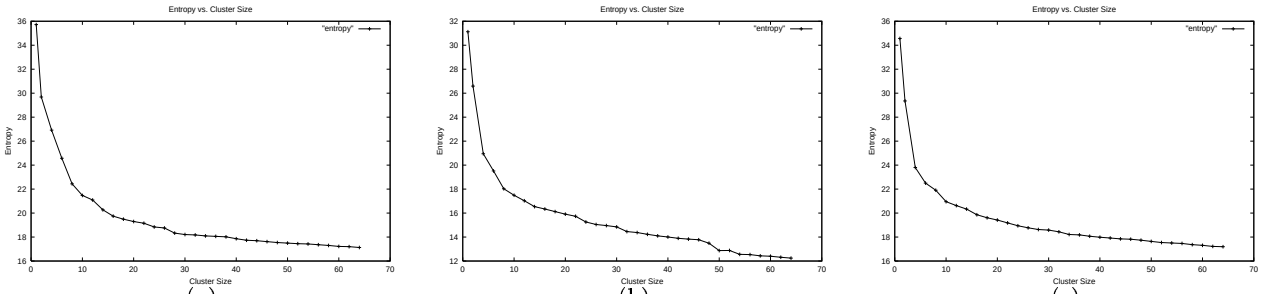
We only consider frequent pattern-sets whose SCOP lineage-based entropy is less than a certain threshold. In addition, when both a pattern-set and its super-pattern-set have an entropy below threshold, only the latter is kept for analysis. We denote the pattern-sets retained after these two steps as *low entropy maximal pattern-sets*, or *quality pattern-sets*. Three parameters are used to generate low entropy maximal pattern-sets, $minSupport$, $maxDist$, and the maximal pattern-set entropy at one or more SCOP levels. For convenience, we refer the last parameter as the *entropy cut-point*. If $maxDist$ is fixed, one can observe that the set of frequent pattern-sets derived at a lower $minSupport$ value must be a superset of the set of frequent pattern-sets derived with a higher $minSupport$. Therefore, in order to get a larger collection of pattern-sets, we set the $minSupport$ to a relatively low value of 0.02. As for the $maxDist$ parameter, various values were applied to each dataset. Based on the experimental results (SCOP-based entropy used as the main leverage), we empirically chose the following values for the 3 datasets: 32 for the 4.5Å dataset, 45 for the 6.0Å dataset and 55 for 7.5Å. Like the other parameters, the maximal SCOP-based pattern-set entropy is selected empirically. In our experiments, we only look at the first 4 SCOP levels. At a given SCOP level, we prefer a pattern-set that has a lower entropy, since a lower entropy usually indicates that a large percentage of the cluster's proteins belong to the same SCOP group. The entropy cut-points we chose corresponding to SCOP levels L_1 and L_2 are 2.0 and 3.2, respectively. For the other two levels, the entropy cut-point is 3.7.

Table 2 presents the number of low entropy maximal pattern-sets generated in each dataset. In order to compare results between the datasets, $minSupport$ is set so that the value of $(Numberofproteins) \times minSupport$ is the same across all datasets. Thus, 0.02 is used for the 6.0Å dataset and 0.01 for the other two.

A closer look at the results shows that the pattern-sets demonstrate different clustering ability. Nearly all the sets from the 4.5Å dataset consist of *Small proteins*, a sub-class at SCOP level L_1. In other words, for most pattern-sets, their associated proteins are classified as small proteins in SCOP. One example is the pattern-set (10 1 22) (Note that each value in a pattern-set corresponds to a clusterID obtained by clustering the individual feature vectors). Such a pattern-set was found in 22 proteins, which had the follow-

Threshold Distance	Number of Proteins	Number of Unique Features	Number of Total Feature Occurrences	Average Number of Features per Protein
4.5Å	2,169	23,148	74,396	34
6.0Å	1,090	36,967	175,525	52
7.5Å	2,122	53,817	410,041	122

Table 1: Contact Map Datasets



(a) (b) (c)

Figure 2: Clustering results: (a)4.5 Å (b)6 Å (c)7.5 Å

dataset	$H(L_1) \leq 2.0$	$H(L_1) \leq 2.0 and H(L_2) \leq 3.2$	#:$H(L_1) \leq 1.5$	#:$H(L_1) \leq 1.5$ *and* $H(L_2) \leq 1.7$)
4.5Å	433	142	19	0
6.0Å	214	93	47	11
7.5Å	1102	468	314	13

Table 2: Low entropy maximal pattern-sets

ing SCOP L_1 distribution: 2 *all-*β, 2 $\alpha + \beta$, 17 small and 1 designed. Unlike the 4.5Å dataset, the pattern-sets from the 7.5Å dataset tend to favor *all-*β proteins.

One other observation to be drawn about the pattern-sets generated for the 4.5Å dataset is that very few of them have $H_{L_1} < 1.5$. Empirically, we found that if a pattern set's entropy at a certain SCOP level was less than 1.5, then nearly all of its associated proteins belonged to the same SCOP sub-group. In addition, we found that the 4.5Å pattern-sets generally occur in very few proteins. One possible explanation for this behavior might be that the 4.5Å contact maps are too sparse to capture most structural information, while the 7.5Å maps are so dense that they introduce too much noise which would confuse the structural distinction for other types of proteins such as *all-*α, $\alpha + \beta$, etc.

The pattern-sets from the 6.0Å dataset have a relatively balanced distribution in terms of the number of protein groups they are able to distinguish. For instance, the pattern set (5 5 10 25 26), was found in 23 proteins with the following SCOP L_1 distribution: 21 *all-*α, 1 *all-*β and 2 α/β. (Please see Figure 3(b) for a visualization of the above pattern-set in the *all-*α protein 1a2f (ID from the PDB)). On the other hand, the pattern set (3 3 7 18) is good at distinguishing *all-*β proteins. Among the 49 proteins where this set occurs, the following SCOP L_1 distribution was found to exist: 1 *all-*α, 39 *all-*β, 2 α/β, 5 $\alpha + \beta$, 1 membrane and cell surface, and 1 designed. (Please see Figure 3(a) for an illustration of the above pattern-set in the *all-*β protein 1a25 (ID from PDB)).

One property illustrated by the pattern-sets from the 6.0Å dataset that does not exist in the the 4.5Å dataset is that there exists a collection of maximal pattern-sets that have low entropy (< 1.5) at the SCOP level L_1 (some are even close to 0) and that there exists a collection of maximal pattern-sets that have low entropy across the first four SCOP levels (see Table 2). An example pattern-set that presents both of these properties is (10 5 10 28). There are 23 proteins that contain this pattern-set, and their distribution at each of the first four SCOP levels is shown in Table 3.

4.4 Performance

In Figure 1, we give an overview of our mining algorithm. In this section, we provide a high-level analysis of the algorithmic running time of each step in that algorithm. Please note that we provide this analysis without proof. Before proceeding, however, we define several variables that will be used to help quantify the running time of each step. Let M be the number of protein molecules in a given dataset, and N_a be the number of amino acid residues contained in the largest protein molecule. N_f is defined to be the maximum number of occurrences of a feature in a contact map, N_u the total number of unique features in the dataset, and N_o the total number of occurrences of all the features in a dataset of M contact maps.

- The generation of contact maps occurs in order $O(M{N_a}^2)$ time.
- The time required to identify all the features in a dataset is $O(M{N_a}^2) + O(N_u)$. Of this, $O(M{N_a}^2)$ is the time required by the region growth algorithm to extract all the maximally-connected patterns in the

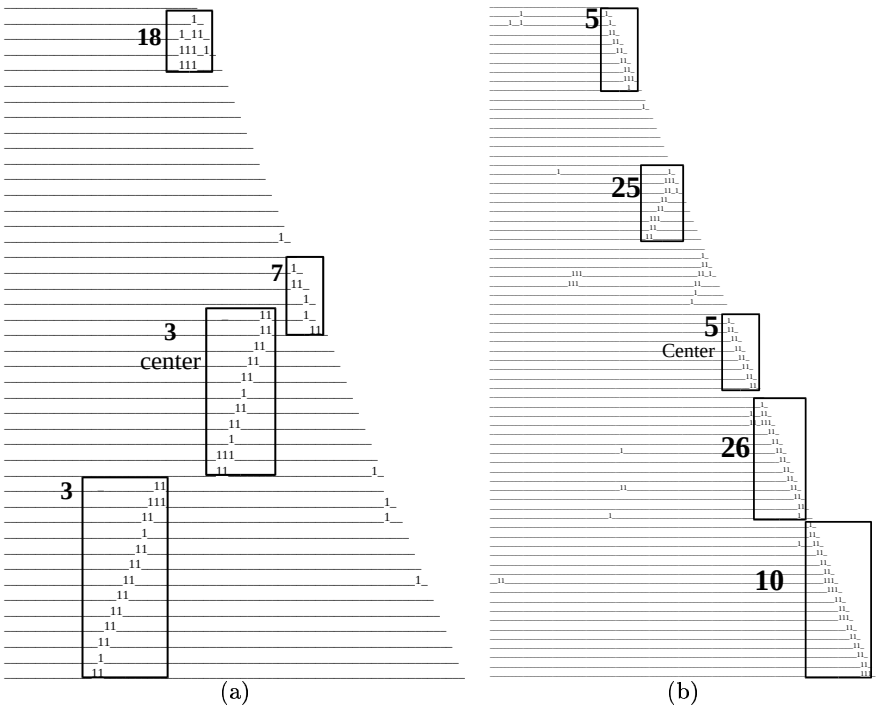


Figure 3: (a) The locations of pattern-set (3 3 7 18) in protein 1a25. (b) The locations of pattern-set (5 5 10 25 26) in protein 1a2f.

L_1:class	L_2:fold	L_3:super family	L_4:family
1 all β	1 SH3-like barrel	1 SH3-like barrel	1 SH3-domain
20 α/β	20 PLP-dependent transferases	20 PLP-dependent transferases	20 AAT-like
1 Peptides	1 Amyloid peptides	1 Amyloid peptides	1 Amyloid peptides
1 designed	1 Alpha-t-alpha	1 Alpha-t-alpha	1 Alpha-t-alpha

Table 3: The distribution of 23 protein containing the maximal pattern-set (10 5 10 28) with a low entropy across the first four SCOP levels.

maps and $O(N_u)$ is the time needed to identify the unique features. For the 7.5Å dataset, which contains over 400,000 feature occurrences, the time to complete this step was less than 20 minutes.

- The time required by one iteration of the *k-means*-based clustering algorithm is $O(kN_u)$ with k being the number of clusters. Of all the steps in the algorithm, this step took the longest, as we must collect a set of cluster solutions before we can decide on an optimal number of clusters. For each value of k, we let the algorithm run for 300 iterations to make sure any solution is approximately optimal. The time for one clustering run ranged from from 1 to 2 hours. The larger the dataset, the longer the clustering would take to complete, but the running time is not affected by the number of clusters that were generated. Once the number of clusters has been selected, it is possible to re-use the clustering solution to label the features for a new set of proteins, provided that the same distance threshold is used when generating the contact maps.
- The fourth step of the algorithm selects an optimal number of clusters based on the clustering entropy of a clustering run. It requires $O(N_u)$ time to compute the entropy for a given run.
- Re-labeling each pattern and creating an occurrence vector requires a running time of $O(N_o)$ for a particular dataset.
- The final step of the algorithm involves the actual generation of frequent spatial pattern-associations. The time required in this step can be decomposed into 3 phases:
  1. Discover all frequent 1-sets. This takes $O(N_o)$ time.
  2. Generate all frequent 2-sets, which can be done in $O(M{N_f}^2)$ time. This is so because in the worst case, one needs to compute the distance between every pair of feature occurrences in a map.
  3. Generate all pattern-sets of size greater than 2. It is hard to quantify the time required to generate a candidate set and all the frequent pattern-sets of a given size, because the time is not only impacted by the two user-specified parameters, *minSupport* and *maxDist*, but is also dataset-specific. As a result, we provide only the time required to confirm whether a candidate pattern-set is frequent, which is $O(MN_f)$. The time is linear to the number of occurrences since there is no need to compute the distance between two feature occurrences in this step.

  Please note that the performance analysis given here assumes the worst-case scenario. In practice, the two threshold parameters, *minSupport* and *maxDist*, can play a significant role in affecting the performance of this step.

5. CONCLUSIONS AND ONGOING WORK

In this paper we present our algorithm for discovering spatial relationships between approximately equivalent patterns in contact maps. While this work is still in the preliminary stages, we were able to find several interesting relationship rules. With further tuning of the algorithm parameters, we hope to find even more biologically-meaningful results.

We are currently extending this work in several aspects: First, we are implementing the other distance metrics described in this paper and intend to run an exhaustive comparison between them. Second, we plan to extend the pattern-set mining algorithm so that it can also generate pattern-sets of other spatial relationships. For example, we are interested in finding pattern-sets that have all pairs of involved patterns within a certain distance, and those that can be spatially arranged as a sequence, with the distance between any two adjacent patterns below a certain threshold. Finally, we plan to take into account a pattern-set's intra-map occurrences, including both the number of occurrences and the locations of those occurrences in a given map.

In addition, we would like to expand this work to other domains. We have access to several datasets containing information about the agricultural yield of farm fields for specific crops over a series of years. By generating contact maps for this dataset and applying our algorithm, it might be possible to determine whether there is any relationship between certain areas and specific crops.

6. REFERENCES

[1] M. J. Atallah. A linear time algorithm for the hausdorff distance between convex polygons. *Information Processing Letters*, 17:207–209, 1983.

[2] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The protein data bank, 2000.

[3] Marco Berrera, Henriette Molinari, and Federico Fogolari. Amino acid empirical contact energy definitions for fold recognition in the space of contact maps. *BMC Bioinformatics*, 4(8), Feb. 2003.

[4] B. Carr, W. Hart, N. Krasnogor, J. Hirst, and E. Burke. Alignment of protein structures with a memetic evolutionary algorithm. In *2002*, 2002.

[5] 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.

[6] 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.

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

[8] Wynne Hsu, Jing Dai, and Mong Li Lee. Mining viewpoint patterns in image databases, 2003.

[9] J. Hu, X. Shen, Y. Shao, C. Bystroff, and M.J. Zaki. Mining non-local structural motifs in proteins. In *BIOKDD 2002*, Edmonton, Canada, 2002.

[10] Jingjing Hu, Xiaolan Shen, Yu Shao, Chris Bystroff, and Mohammed J. Zaki. Mining protein contact maps.

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

[12] J. Kim, E. Moriyama, C. Warr, P. Clyne, and J. Carlson. Identification of novel multi-transmembrane proteins from genomic databases using quasi-periodic structural properties. *Bioinformatics*, 2002.

[13] R. King, A. Karwath, A. Clare, and L. Dehaspe. Genome scale prediction of protein functional class from sequence using data mining. In *The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[14] 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.

[15] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Proc. 4th Int'l Symp. on Large Spatial Databases*, pages 47–66, 1995.

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

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

[18] Giuseppe Lancia, Robert Carr, Brian Walenz, and Sorin Istrail. 101 optimal pdb structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *Proceedings of the fifth annual international conference on Computational biology*, pages 193–202. ACM Press, 2001.

[19] R. Najmanovich M. Vendruscolo and E. Domany. Protein folding in contact map space. *Physical Review Letters*, 82(656), 1999.

[20] 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.

[21] 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.

[22] Yasuhiko Morimoto. Mining frequent neighboring class sets in spatial databases. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 353–358. ACM Press, 2001.

[23] 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.

[24] W. Pan, J. Lin, and C. Le. Model-based cluster analysis of microarray gene-expression data. *Genome Biology*, 2002.

[25] Srinivasan Parthasarathy, Mohammed Javeed Zaki, Mitsunori Ogihara, and Sandhya Dwarkadas. Incremental and interactive sequence mining. In *CIKM*, pages 251–258, 1999.

[26] L. De Raedt and S. Kramer. The level-wise version space algorithm and its application to molecular fragment finding. In *Seventeenth International Joint Conference on Artificial Intelligence*, 2001.

[27] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, 2001.

[28] M. Vendruscolo and E. Domany. Protein folding using contact maps. *Vitamins and Hormones*, 58(171), 2000.

[29] Michele Vendruscolo and Eytan Domany. Efficient dynamics in the space of contact maps. *Folding & Design*, 3(5):329–336, 1998.

[30] Michele Vendruscolo, Edo Kussell, and Eytan Domany. Recovery of protein structure from contact maps. *Folding & Design*, 2(5):295–396, 1997.

[31] J. T. L. Wang, B. A. Shapiro, D. Shasha, K. Zhang, and C.-Y. Chang. Automated discovery of active motifs in multiple rna secondary structures. In *International Conference on Knowledge Discovery and Data Mining*, 1996.

[32] 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.

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

[34] Mohammed J. Zaki and Karam Gouda. Fast vertical mining using diffsets. In *RPI Technical Report 01-1*, 2001.

[35] Ying Zhao and George Karypis. Prediction of contact maps using support vector machines. In *Bioinformatics and Bioengineering*. IEEE, 2003.

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