Mining Spatial Object Associations for Scientific Data*

Hui Yang, Srinivasan Parthasarathy and Sameep Mehta
Department of Computer Science and Engineering
Ohio State University
2015 Neil Avenue, Columbus, OH 43210

Abstract

In this paper, we present efficient algorithms to discover spatial associations among features extracted from scientific datasets. In contrast to previous work in this area, features are modeled as geometric objects rather than points. We define multiple distance metrics that take into account objects' extent. We have developed algorithms to discover two types of spatial association patterns in scientific data. We present experimental results to demonstrate the efficacy of our approach on real datasets drawn from the bioinformatic domain. We also highlight the importance of the discovered patterns by integrating the underlying domain knowledge.

1 Introduction

Advances in simulation and data collection techniques in different scientific domains such as bioinformatics, computational fluid dynamics, and computational molecular dynamics, have resulted in huge amounts of data. It is necessary to develop computational techniques to extract features from scientific datasets, for instance, vortices in fluid flow fields and substructures from molecular datasets. It is also very important to subsequently uncover meaningful relationships among extracted features. Such relationships can provide valuable information towards understanding or explaining the underlying scientific phenomenon. Many algorithms [Kramer et al., 2001; Jiang et al., 2003; Yang et al., 2004] have been proposed to extract features from these scientific datasets. In this paper we focus on the latter problem of finding relationships among features.

Recently, researchers have started to explore the relationships [Morimoto, 2001; Munro et al., 2003; Zhang et al., 2004] among features. Most of the previous techniques represent features by single points. However, this representation leads to loss of important information. The shape, size, and orientation of a feature in scientific datasets are very important. A simple solution to this problem is to use Minimum Bounding Boxes (MBB). However, MBBs are not well-suited for every domain. For example, ellipsoids are more suitable for capturing the shape of vortices in fluid flow datasets [Sadarjoen et al., 1998]. Alternatively, defect structures in materials may require irregular shape descriptors [Mehta et al., 2004]. In this work, we propose to use different shape descriptors for features from different domains.

Frequently recurring relationships among features across different datasets can guide the domain expert to find useful knowledge. Such relationships are especially useful in bioinformatics. One important issue in bioinformatics is to identify structurally similar proteins. To address this issue, one can first discover non-local patterns that frequently occur in proteins known to be structurally similar [Zaki, 2002; Yang et al., 2004]. Such patterns can then be used to indicate whether a new protein is potentially similar to known proteins. We propose algorithms that discover relationships across multiple maps. This is different from previous work on spatial association mining [Koperski and Han, 1995; Shekhar and Huang, 2001; Morimoto, 2001; Zhang et al., 2004], where features are located in the same dataset and represented as points.

We define Spatial Object Association Patterns (SOAP) to characterize spatial relationships among object types. Our algorithm finds two different types of SOAPs: *Star* and *Sequence* (Figure 2). They capture different aspects of neighborhood relationships among features. For example, in protein contact maps, the formation of *Star* SOAPs among non-local patterns indicates such patterns have a compact spatial relationship. The formation of *Sequence* SOAPs on the other hand indicates an extended spatial relationship among the involved patterns. Furthermore, by extracting SOAPs from contact maps that are associated with proteins in different protein classes, we can establish the associations between different types of SOAPs and protein classes. Such associations can help us to identify structural characteristics of different protein classes.

The rest of this document is organized as follows. In Section 2, we present the key ideas underpinning our work: (i) the notion of spatial feature representation using extents and shapes, (ii) the different object-oriented distance metrics, and (iii) the different association pattern types used to characterize spatial relationships. In Section 3, we detail the algorithms and efficient realizations of the key ideas. Section 4 describes the efficacy of our approach on datasets drawn from bioinformatics. Finally, we present our conclusions and outline directions of ongoing and future research in Section 5.

*This work is funded by NSF grants ITR-NGS ACI-0326386, CAREER IIS-0347662, and SOFTWARE ACI-0234273. All correspondence should be addressed to Srinivasan Parthasarathy at srini@cse.ohio-state.edu.

2 Background and Definitions

2.1 Spatial Feature Representation

We propose two basic shape representation schemes: parallelepiped (or parallelogram in 2-D) and ellipsoid (or ellipse in 2-D). As demonstrated in Section 4, parallelograms are more appropriate to capture the shape and extent of non-local structures in protein contact maps as opposed to MBBs. Whereas ellipsoids (or ellipses) are suitable for vortices in fluid flow data as mentioned earlier. Note that the first scheme subsumes the MBB representation, and the second scheme subsumes circles. For highly irregular-shaped features such as defect structures in materials, we plan to use sampled boundary points, known as landmarks [Rao and Suryawanshi, 1996].

As shown in Figures 1(a) and 1(b), the shape descriptors of a parallelogram and an ellipse can be described as $A_{parallel} =< (x, y), l_1, l_2, \theta, \phi >$ and $A_{ellipse} =< (x, y), l_1, l_2, \theta >$, respectively. These descriptors can also be extended to 3D.

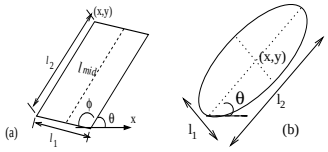


Figure 1: Object type: (a)Parallelogram (b)Ellipse

2.2 Dataset Representation

The dataset $\mathbb{D}$ consists of n features located in r maps ($r >1$), denoted as M=$\{m_1, m_2, \ldots, m_r\}$. The n features are categorized into l types, corresponding to l unique labels Σ=$\{c_1, c_2, \ldots, c_l\}$. The categorization of features is governed by the underlying domain. A feature's geometric properties such as shape and size are captured by adopting one of the supported representation schemes. A feature f thus can be described as a vector <*mapID, location*, A_{geo}, *type*>, where *type*$\in \Sigma$, *mapID*$\in M$ indicates the map where f occurs, *location* identifies f's position within the map, and $A_{geo} \in \{A_{parallel}, A_{ellipse}\}$ captures the shape of f.

Note that in the rest of the paper, we refer to a feature corresponding to the above vector as a *spatial object*. We assume the following order among the r maps: $m_1 < m_2 < \ldots < m_r$. If the r maps correspond to r snapshots, they are ordered temporally. If they are from different datasets (e.g., different protein contact maps), the order among them is imposed by arbitrarily assigning each map a unique ID. Furthermore, the lexicographic order among the l feature types is imposed as follows: $c_1 < c_2 < \ldots < c_l$.

2.3 Object-oriented Distance Metrics

The proposed algorithms use the following metrics to measure the distance between two objects o_i and o_j located in the same map.

- **Point-Point distance**: This is simply the Euclidian distance between object centroids.
- **Line-Line distance**: If o_i and o_j are parallelepipeds (or parallelograms), we first identify the line segment between the midpoints of the top and bottom surfaces (or sides) in each object, then compute the shortest distance between these two line segments as the line-line distance between o_i and o_j. We identify the top and bottom surfaces (or sides) by selecting a reference axis in the underlying Cartesian coordinate space, specifically, the z-coordinate in 3D and y-coordinate in 2D. If o_i and o_j are ellipsoids (or ellipses), the line-line distance is between the two major axes.
- **Boundary-Boundary distance**: This is the shortest distance between the boundaries of o_i and o_j. When o_i and o_j are represented as ellipses, the boundary-boundary distance is the shortest pair-wise distance between points sampled on the boundaries (or surfaces in 3D). The number of sampled points is user-specified.

Notice that the line-line and boundary-boundary metrics are able to take objects' geometric properties into account. The algorithms also support Hausdorff distance [Atallah, 1983]. Since this distance is not applicable to the applications described in this article, we do not discuss it here.

Two objects o_i and o_j have a *closeTo* relationship if the distance between them is $\leq \varepsilon$, where ε is a user-specified parameter. Two objects are *neighbors* if they have a closeTo relationship. We also define the isAbove relationship between o_i and o_j. In a coordinate system, o_i is said to have a *isAbove* relationship with o_j, if the upper-left corner of o_i's Minimum Bounding Box (MBB)[1], denoted as (x_i, y_i, z_i), and the upper left corner of o_j's MBB, denoted as (x_j, y_j, z_j), meets the following condition: $(z_i > z_j) \vee [(z_i = z_j) \wedge [(y_i > y_j) \vee ((y_i = y_j) \wedge (x_i < x_j))]]$ in a 3-D map, or $(y_i > y_j) \vee [(y_i = y_j) \wedge (x_i < x_j)]$ in a 2-D map.

2.4 Spatial Object Association Pattern (SOAP)

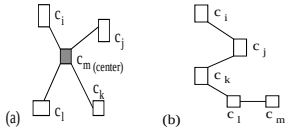


Figure 2: SOAP Types: (a)Star (b)Sequence

A *Spatial Object Association Pattern (SOAP)* of size k, denoted as k-SOAP, characterizes the *closeTo* relationships among k object types. In this article, we focus on two types of SOAPs: *Star* and *Sequence* (Figure 2). As discussed earlier, these two SOAP types can characterize different spatial relationships among objects. These two SOAP types can be abstracted as undirected graphs, where a node corresponds to an object-type $c_i \in \Sigma$, and an edge (c_i, c_j) indicates that c_i and c_j are required to have a closeTo relationship.

- **Star SOAPs** (Figure 2a) have a *center* object-type, which is required to have a *closeTo* relationship with all the other object-types in the same SOAP.
- **Sequence SOAPs** (Figure 2b) of size k p=($c_{[i]}$:$i\in$[1,k]) satisfy two constraints, where $c_{[i]}$ is the i^{th} element in p: (1) *closeTo*($c_{[i]}, c_{[i+1]}$)=*true* and (2) *isAbove*($c_{[i]}, c_{[i+1]}$)=*true*, where $1 \leq i \leq k$-*1*. Sequence SOAPs are mainly motivated by our observation on protein contact maps, where non-local structures, i.e, features, tend to line up in a Sequence like manner.

[1] For a 3D MBB, it is the upper-left corner of its top surface.

These two SOAP types can also be represented as lists. Let c_{cntr} be the center of a star k-SOAP p, and $\{c_{[i]}: i\in[1,k\text{-}1]\}$ be the other *k-1* object-types in p, where $c_{[1]} \leq \ldots \leq c_{[k-1]}$. SOAP p can then be described by the list p=(c_{cntr}, $c_{[1]}, \ldots, c_{[k-1]}$), where *closeTo*($c_{cntr}, c_{[i]}$)=*true* ($i\in[1,k\text{-}1]$). Whereas elements in a sequence SOAP correspond to a list by definition and cannot be forced into lexicographical order. For instance, the two sequence SOAPs $(A\ B\ C)$ and $(A\ C\ B)$ are different.

A SOAP is *autocorrelated* if an object-type occurs multiple times. For example, (c_1, c_1, c_2) is an autocorrelated 3-SOAP, where c_1 occurs twice. An *instance* of a SOAP p is the set of objects that meet all the requirements specified by p, including those on object-types and *closeTo* (or *isAbove*) relationships.

We define two measures-*support* and *realization*- to characterize the importance of a SOAP. The support of a SOAP p is the number of maps in the dataset where p occurs. Assume support*(p)*=*s*, let n_i be the number of p's instances in the i^{th} map where p appears, realization(p)=*min*$\{n_i\}$. A pattern p is *frequent* if support(p)$\geq$*minSupp*, and *prevalent* if realization(p)$\geq$*minRealization*, where minSupp and minRealization are user-specified parameters.

3 Algorithms

3.1 Data Organization

We organize $\mathbb{D}$ in the following manner. The n objects are first grouped into l partitions, where each partition is composed of objects of the same type. Within each partition, objects are ordered by their map IDs and locations in a map. This data organization is analogous to the format used for association rule mining [Zaki et al., 1997]. Note that each object can be uniquely identified by combining the following information: *mapID*, *label*, and *locationID* in the map.

3.2 Equivalence Classes

Based on the list-based SOAP representation (Section 2.4), we organize SOAPs into equivalence classes. A *k-equivalence class*, denoted as *k-EquiClass*, is defined as the set of k-SOAPs that (i) are of the same SOAP type, and (ii) have the same prefix, where the *prefix* of a k-SOAP is its first *k-1* elements. By using equivalence classes, our mining algorithms only need to compute the *closeTo* or *isAbove* relationships between objects once. The equivalence classes also help to improve the memory locality of the algorithms, which result in significant performance gains. Moreover, equivalence classes also enable the algorithms to smoothly scale to large datasets [Zaki et al., 1997].

The next two sections describe the algorithms that discover *Star* and *Sequence* SOAPs. For each frequent SOAP, the algorithms store information about all of its instances. Such information allows the algorithms to locate every object involved in an instance.

3.3 Mining *Star* SOAPs

Figure 3 outlines the algorithm that discovers *star* SOAPs. The first step generates frequent 1-SOAPs (line 1). For each object-type c_i, the procedure *gen1SOAP* counts the number of maps that contain at least one object of type c_i. If the count $\geq$ *minSupp*, then p_1=(c_i) is a frequent 1-SOAP. The set of all frequent 1-SOAPs is denoted by P_1.

The next step, *gen2EquiClass*, discovers 2-SOAPs and organizes them into 2-EquiClasses (line 2). The pseudo-code of *gen2EquiClass* is described in Figure 4. A 2-EquiClass is generated for each frequent 1-SOAP (Figure 4:line 2). The 2-EquiClass of 1-SOAP (c_i)$\in P_1$, denoted by E_{2,c_i}, contains frequent 2-SOAPs in the form of $(c_i, *)$. To generate E_{2,c_i}, the procedure considers the following 2-SOAPs as candidates: (c_i, c_j), where (c_j)$\in P_1$ (Figure 4:line 3). For each candidate 2-SOAP p_2=(c_i, c_j), the procedure identifies all the maps where p_2 occurs (Figure 4:lines 6-11). An instance of p_2=(c_i, c_j) is an object pair (o_i, o_j), where (o_i.*type*=c_i)$\wedge$(o_j.*type*=c_j) $\wedge$ (*distance*(o_i, o_j,*distType*)$\leq \varepsilon$). If p_2 occurs in $\geq$*minSupp* maps, then it is frequent and is added to E_{2,c_i} (Figure 4:line 12).

Algorithm mine_starSOAP($\mathbb{D}$,*minSupp, minRealization, distType*, ε)
1. $P_1 \leftarrow$ gen1SOAP(); // 1-SOAPs
2. $EC_2 \leftarrow$ gen2EquiClass($\mathbb{D}$, P_1, *star, parList*); //parList: parameters
3. $k \leftarrow 2$;
4. while (1){
5. $EC_{k+1} \leftarrow \varnothing$; //the set of (k+1)-EquiClasses
6. foreach k-EquiClass $E_k \in EC_k$
7. foreach k-SOAP $p_{k,i} \in E_k$
8. $E_{k+1} \leftarrow \varnothing$; // the prefix of E_{k+1} is $p_{k,i}$
9. foreach k-SOAP ($p_{k,j} \in E_k$) $\wedge$ ($i \leq j$)
10. $p_{k+1} \leftarrow$ append($p_{k,i}$, lastElement($p_{k,j}$));
11. $M_{p_{k+1}} \leftarrow \{m_i : m_i$ contains $p_{k,i}$ and $p_{k,j}\}$;
12. foreach $m_i \in M_{p_{k+1}}$
13. if (countStarInstances(m_i , p_i , p_j)$\geq$ 1) mapCnt++;
14. if (mapCnt $\geq minSupp$) $E_{k+1} \leftarrow E_{k+1} \cup \{p_{k+1}\}$;
15. if ($E_{k+1} \neq \varnothing$) $EC_{k+1} \leftarrow EC_{k+1} \cup E_{k+1}$;
16. if (*minRealization*>1) markPrevFreqSOAPs(E_{k+1});
17. if ($EC_{k+1} = \varnothing$) return; //terminate
18. k++; //increase SOAP size }//while(1)

Figure 3: Mining *Star* SOAPs

The algorithm next discovers SOAPs of size>2 (Figure 3:lines 4-17). Two k-SOAPs in the same k-EquiClass are combined to construct a candidate (k+1)-SOAP. For each candidate (k+1)-SOAP p_{k+1} derived by appending the last element of $p_{k,j}$ to $p_{k,i}$ (Figure 3: line 10), the algorithm identifies all the maps where p_{k+1} occurs (Figure 3:lines 11-14). (The i^{th} k-SOAP in a k-EquiClass is denoted as $p_{k,i}$.) The procedure *countStarInstances*($m_i, p_{k,i}, p_{k,j}$) (Figure 3:line 13) computes the instances of p_{k+1} in map m_i by combining instances of $p_{k,i}$ and $p_{k,j}$. Two instances from $p_{k,i}$ and $p_{k,j}$ are joined to produce a p_{k+1} instance if they have the same first k-1 objects and different last object. SOAPs with the same prefix are organized into one equivalence class as they are being generated (see Figure 3:lines 5, 8, 14, 15). The mining process stops when all the frequent SOAPs have been discovered (Figure 3:line 17).

To discover frequent SOAPs, the algorithm only needs to consider a SOAP's presence in a map (Figure 3:line 13). Hence, some of the discovered frequent SOAPs may not be prevalent if *minRealization*>1. In this case, the procedure *markPrevFreqSOAPs* is called to identify the SOAPs that are both prevalent and frequent (Figure 3:line 16). It is necessary to keep SOAPs that are frequent but not prevalent. We explain this by a simple example. Let $\{a_1, b_1, b_2\}$ be the three neighboring objects in a map and *minRealization=2*. To derive the two instances of SOAP (a, b), (a_1, b_1) and (a_1, b_2),

Algorithm gen2EquiClass ($\mathbb{D}$, P_1, $soapType$,*parList*) //P_1 :freq. 1-SOAPs
1.$EC_2 \leftarrow \varnothing$; //the set of 2-EquiClasses
2.foreach (c_i)$\in P_1$
3. $E_{2,c_i} \leftarrow \varnothing$ //2-EquiClass with prefix of c_i;
4. foreach (c_j)$\in P_1$
5. $p_2 \leftarrow (c_i, c_j)$; //a candidate
6. $M_{p_2} \leftarrow \{m_i : (m_i$ contains c_i and $c_j)\}$;
7. foreach $m_i \in M_{p_2}$
8. foreach $(o_i, o_j) \in m_i : (o_i.label = c_i) \wedge (o_j.label = c_j)$
9. if ((*soapType*=*Sequence*)$\wedge(\neg$ isAbove$(o_i, o_j))$) continue;
10. if distance$(o_i, o_j$, *distType*)$\leq \varepsilon$) addInstance(p_2, (o_i, o_j));
11. if (cntInst(p_2, m_i) $\geq$ 1) mapCnt++;
12. if (mapCnt $\geq$ *minSupp*) $E_{2,c_i} \leftarrow E_{2,c_i} \cup \{p_2\}$;
13.if ($E_{2,c_i} \neq \varnothing$) $EC_2 \leftarrow EC_2 \cup E_{2,c_i}$;

Figure 4: 2-EquiClass Generation

the 1-SOAP (a) must be maintained even though its realization is 1 (< 2) in the map.

Correctness: It is straightforward to show that the algorithm discovers all the frequent 1-SOAPs and 2-SOAPs. Thus to prove the algorithm is correct, we only need to show that every frequent k-SOAP (k>2) will be considered as a candidate. Assume p_k=($c_{[1]}, \ldots, c_{[k-2]}, c_{[k-1]}, c_{[k]}$) is frequent, then the two (k-1)-SOAPs $p_{k-1,i}$= ($c_{[1]}, \ldots, c_{[k-2]}, c_{[k-1]}$) and $p_{k-1,j}$= ($c_{[1]}, \ldots, c_{[k-2]}, c_{[k]}$) must also be frequent and in the same equivalence class. Thus the algorithm will consider p_k as a candidate. It is trivial to show that the procedure *countStarInstances* identifies all the instances of a candidate SOAP. □

3.4 Mining *Sequence* SOAPs

The pseudo-code is given in Figure 5. Unlike mining Star SOAPs, which uses two k-SOAPs in the same equivalence class to generate a candidate (k+1)-SOAP ($k\geq$2), the algorithm joins one k-SOAP and one 2-SOAP.

The first two steps (lines 1-2) discover all the frequent 1-SOAPs and 2-SOAPs. The *isAbove* relationship is checked by the procedure *isAbove*(o_i, o_j) (Figure 4:line 9). For each k-SOAP p_k=($c_{[1]}, \ldots, c_{[k]}$), the algorithm first locates the 2-EquiClass $E_{2,c_{[k]}}$, in which every 2-SOAP is in the form ($c_{[k]}, c_{[j]}$):*closeTo*($c_{[k]}, c_{[j]}$)$\wedge$isAbove($c_{[k]}, c_{[j]}$) (line 7). A set of candidate (k+1)-SOAPs are then generated by combining p_k with each 2-SOAP in $E_{2,c_{[k]}}$ (lines 8-9). Same as mining Star SOAPs, a candidate (k+1)-SOAP p_{k+1} is frequent if it appears in $\geq$*minSupp* maps (lines 10-13). The procedure *countSeqInstances*($m_i, p_{k,i}, p_{2,j}$) (line 12) identifies instances of p_{k+1} in map m_i, where p_{k+1} is a candidate SOAP based on $p_{k,i}$ and $p_{2,j}$. Two instances $I_{k,i}$ and $I_{2,j}$, from $p_{k,i}$ and $p_{2,j}$ respectively, are combined to produce an instance of p_{k+1} if the last object in $I_{k,i}$ is the same as the first object in $I_{2,j}$. For the same reason explained before, the algorithm calls the procedure *markPrevFreqSOAPs* to label SOAPs being both prevalent and frequent if *minRealization*>1 (line 15). The algorithm stops when no more SOAPs can be discovered (line 14).

Correctness: For each frequent sequence k-SOAP p_k=($c_{[1]}, \ldots, c_{[k-1]}, c_{[k]}$), the following two SOAPs must also be frequent: p_{k-1}=($c_{[1]}, \ldots, c_{[k-1]}$) and p_2=($c_{[k-1]}, c_{[k]}$). Thus, p_k will be considered as a candidate and be discovered. It is trivial to show that the procedure *countSeqInstances* identifies all the instances of a candidate SOAP.□

Algorithm mine_sequenceSOAP($\mathbb{D}$,*minSupp, minRealization, distType,* ε)
1. $P_1 \leftarrow$ gen1SOAP();
2. EC_2 ($or\ P_2$) $\leftarrow$ gen2EquiClass($\mathbb{D}$, P_1, *Sequence, parList*);//parList:parameters
3. $k \leftarrow 2$;
4. while (1){
5. $P_{k+1} \leftarrow \varnothing$; //initialize the set of $(k+1)$-SOAPs;
6. foreach SOAP $p_{k,i} =< c_{[0]}, \ldots, c_{[k-1]} > \in P_k$
7. $E_{2,c_{[k]}} \leftarrow E_{2,c_{[k]}} \in EC_2) \wedge (E_{2,c_{[k]}}.prefix = c_{[k]})$;
8. foreach 2-SOAP $p_{2,j} \in E_{2,c_{[k]}}$
9. $p_{k+1} \leftarrow$ append($p_{k,i}$, lastElement($p_{2,j}$));
10. $M_{p_{k+1}} \leftarrow \{m_i : m_i$ contains both $p_{k,i}$ and $p_{2,j}\}$;
11. foreach $m_i \in M_{p_{k+1}}$
12. if (countSeqInstances($m_i, p_{k,i}, p_{2,j}$) $\geq$ 1) mapCnt++;
13. if (mapCnt $\geq$ *minSupp*) $P_{k+1} \leftarrow P_{k+1} \cup \{p_{k+1}\}$;
14. if ($P_{k+1} = \varnothing$) return; //terminate the process;
15. if (*minRealization*>1) markPrevFreqSOAPs(P_{k+1});
16. k++; //increase SOAP size}//while(1)

Figure 5: Mining *Sequence* SOAPs

Other SOAP types can also be defined. For instance, we can define *Clique* SOAPs, which require every pair of objects in the same SOAP have a *closeTo* relationship. We can also define SOAPs that involve other types of spatial relations such as topological relations.

4 Experimental Evaluation

In this section, we evaluate the algorithms on a protein contact map dataset. We start with a brief description on the dataset generation process and the domain-specific usefulness measurement adopted in this work. We then present results to analyze the impact from different distance metrics and SOAP types. Finally, we present performance characterization of the algorithms.

4.1 Data Preprocessing and Representation

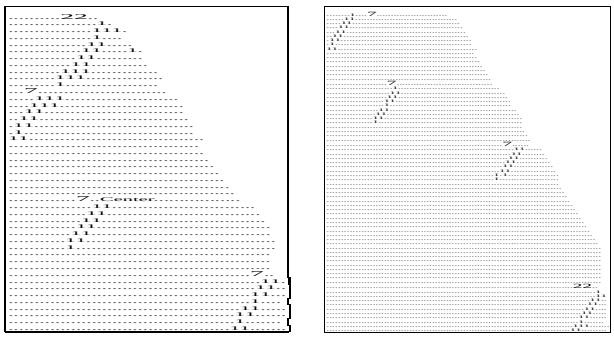
We first generate contact maps for $8,732$ proteins taken from the Protein Data Bank [Berman et al., 2000]. For a protein with *N* amino acids, its contact map C is a $N \times N$ binary matrix. The position $C(i,j)$ is set to 1 if the distance between the i^{th} and j^{th} residues is less than a threshold and 0 otherwise [Vendruscolo and Domany, 1997]. We use 6Å as the threshold as suggested in the literature [Zaki, 2002]. We then extract features in contact maps. A *contact map feature* is composed of a set of positions, where each position and at least one of its eight neighbors contain a '1' (see Figure 6 for examples). We apply a simple region growing approach to extract features in contact maps [Yang et al., 2004]. We then use an entropy-based clustering algorithm to cluster features into l groups (or classes) [Cheng et al., 1999]. Features in a class have similar geometric properties such as shape and extent.

A total of $1,009,755$ features are extracted from the $8,732$ contact maps. These features are clustered into 28 classes. The average number of features in each map is about 115. Many of these features correspond to well-known protein secondary structures and are also validated by domain experts. We represent each feature by its minimum bounding parallelogram and label the parallelogram by the feature's class ID.

4.2 Domain Specific Usefulness Measurement

The structure of a protein often provides information about its functionality. Thus, one can predict a protein's function based on the function of other structurally similar proteins.

Frequent SOAPs in contact maps characterize proteins' structure. They can be used to generate signatures for different proteins. For instance, the SOAP (7 7 7 22), which is automatically discovered by our algorithms as both *Star* and *Sequence*, can be used to identify the following *structurally and functionally similar* β-proteins: 1h3t, 1h3u and 1h3v (IDs from PDB). (See Figures 6(a) and 6(b) for an illustration of these SOAPs.)



(a) (b)

Figure 6: SOAP (7 7 7 22) in proteins 1h3u, 1h3t &1h3v as: (a) Star and (b) Sequence

However, a SOAP is unlikely to add any value to the signature, if it is randomly associated with proteins of different classes. To prune away such SOAPs, we use an entropy-based approach. For a given SOAP, we first identify its associated proteins. We then compute the SOAP's entropy by integrating the proteins' lineage information from the database of Structural Classification of Proteins (SCOP)[2]. A protein's SCOP lineage is organized into 6 hierarchical levels according to its structure. We look at the first two levels: L_1:$class$, and L_2:$fold$. L_1 consists of 11 classes such as *α-protein*, *β-protein*, and *small protein*. L_2 further divides proteins into sub-classes based on proteins' folding structures. Let N be the number of proteins whose contact maps contain the SOAP p, and n_i be the number of proteins among these N proteins in the i^{th} class, the entropy of p at the first SCOP level is then computed as: $H(p) = \sum_{i=1}^{11} -(n_i/N) \times log_2(n_i/N)$ [Shannon, 2001]. $H(p)$ essentially measures how well p's associated proteins distribute among different proteins classes. In our experiments, we observe that SOAPs with entropy $\leq$ 2.0 show good quality and can be used to generate signatures. This observation is also validated by domain experts.

For each SOAP with entropy $\leq$ 2.0, we next identify the protein class that dominates its associated proteins. For example, the dominating protein class of the clique SOAP (5 7 7 22) is *β-protein*, as 191 out of its 215 associated proteins are β-proteins. As suggested by domain experts, we would like to have many SOAPs for a certain protein class. A large number of SOAPs not only cover a wide range of proteins, but can also be used to identify structurally similar proteins from different aspects. This criterion is used to evaluate the discovered SOAPs. Note that even if two SOAPs have the same dominating protein class, their associated proteins can be very different.

[2]http://scop.mrc-lmb.cam.ac.uk/scop/

4.3 Impact of Distance Metrics

Due to lack of space, we only report the results produced from the following parameter setting: *minSupp*=1%, *minRealization*=1, and ε=5$\mathring{A}$.

Figure 7 shows the impact of the three distance metrics on different SOAP types. One can observe that the Line-Line (L-L) distance generates the most number of SOAPs for both SOAP types, whereas the Point-Point (P-P) distance the least. This is expected as the P-P distance does not consider the shape and size of spatial objects, in contrast with the other two. One may argue that more SOAPs can be identified by increasing the distance threshold in the case of P-P distance. However, it is very difficult to find an appropriate threshold value, as the size of a feature, i.e., the number of bit-1 positions, varies drastically, from 2 up to several thousand.

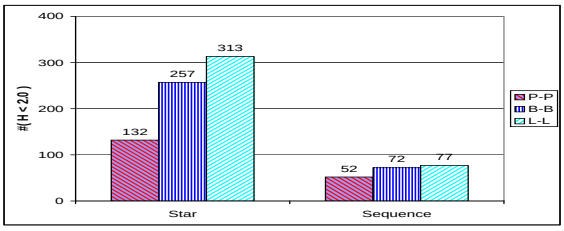


Figure 7: Distance metric vs #SOAPS discovered

The advantage of using L-L and B-B distance metrics becomes more significant when looking at the SOAPs' dominating protein classes. Tables 1-3 summarize the number of SOAPs in major protein classes. One can see that, if the P-P distance is used (Table 1), only 31 SOAPs are discovered in small proteins as against 133 SOAPs in β-proteins. On the other hand, SOAPs based on L-L or B-B distance show a relatively more balanced behavior(Tables 2-3). Compared with P-P distance, much more SOAPs in β-proteins or small proteins are discovered. For instance, in the L-L case (Table 2), there are 180 and 203 SOAPs are identified in β-proteins and small proteins respectively. Also the B-B distance produces the most number (14) of SOAPs for α-proteins (Table 3).

The differences exhibited by SOAPs from different distance metrics show that it is important for our algorithms to support multiple distance metrics, especially metrics which consider objects' extent.

Type	#(α)	#(β)	#(small)	#(peptide)
Star	2	103	19	7
Sequence	4	30	12	4
Total	6	133	31	11

Table 1: #SOAPs in major protein groups, distType=P-P

Type	#(α)	#(β)	#(small)	#(peptide)
Star	1	133	167	12
Sequence	4	37	36	6
Total	5	170	203	18

Table 2: #SOAPs in major protein groups, distType=L-L

4.4 Impact of SOAP Types

As we move down to the second SCOP lineage level, L_2:$fold$, we discover that different types of SOAPs can actually distinguish different protein folding structures. Proteins in the same class (e.g., β-protein) are further classified

Type	#(α)	#(β)	#(small)	#(peptide)
Star	8	145	91	13
Seq.	6	40	25	6
Total	14	185	116	19

Table 3: #SOAPs in major protein groups, distType=B-B

into structurally similar sub-groups according to their folding structures. Table 4 lists the β-protein folds that are distinguished by each SOAP type, where SOAPs are generated based on the L-L distance. The folds in bold are those that are associated with only one SOAP type. Whereas other folds in the table are distinguished by two or more SOAP types. Folds in other protein classes show a similar trend. For example, the α-protein fold "Cyclin-like" is only associated with sequence SOAPs.

The above results can potentially help domain experts address some important biological issues, for instance, predicting a protein's function based on the SOAPs contained in its contact map.

Star	Immunoglobulin-like beta-sandwich Concanavalin A-like lectins/glucanases Trypsin-like serine proteases **Cupredoxin-like** **Acid proteases** **Cysteine proteinases**
Sequence	Immunoglobulin-like beta-sandwich Concanavalin A-like lectins/glucanases Trypsin-like serine proteases **Lipocalins** **Nucleoplasmin-like/VP**

Table 4: List of β-protein folds associated with each SOAP type, distType=L-L

Type	minSupp=2%	minSupp=1%
Star	15.54	21.53
Sequence	4.83	7.79

Table 5: Running Time (in seconds) (distType=L-L)

4.5 Running Time

Table 5 shows the time taken to discover the two types of SOAPs at two *minSupp* values. All the experiments were carried out on a Pentium 1.7GHz computer of 256MB main memory. To discover SOAPs in $8,732$ protein contact maps containing $1,009,755$ objects, the algorithms take about 20 seconds when *minSupp*=2% and about 30 seconds when *minSupp*=1% . Although running time increases as *minSupp* decreases, the algorithm scales very well. We attribute this good performance to the use of equivalence classes.

5 Conclusion and Ongoing Work

In this paper, we present a general framework to uncover two types of spatial association patterns among features in scientific data. The framework represents features as spatial objects instead of points. It also supports multiple distance metrics. Empirical results on protein contact maps show that the framework is both efficient and scalable. Furthermore, the discovered SOAPs are meaningful and can potentially be used to address important biological issues.

The SOAP mining problem in this article shares some similarity with frequent subgraph mining [Yan and Han, 2002]. In order to apply the conventional graph mining algorithms for SOAP mining, the notion of nodes needs to be extended to integrate spatial properties such as location and shape. Also, the notion of edge needs to be modified to reflect different spatial relationships such as *closeTo*.

We are currently extending the framework in several directions. First, we are examining other shape representation schemes such as the landmark-based approach for highly irregularly shaped features. Second, we are evaluating other types of object association patterns, for example, clique SOAPs, where each object has a *closeTo* relationship with every other object in the same SOAP. Third, we are interested in evaluating the algorithms on datasets from other scientific domains, including molecular dynamics and fluid flow dynamics. Some of our ongoing work is reported in a technical report [Yang *et al.*, 2005]. Finally, we are investigating potential approaches towards association based spatio-temporal reasoning.

Acknowledgments: We thank D. Polshakov and K. Marsolo for helping us validate the experimental results, and thank Dr. J. Wilkins and Dr. R. Machiraju for valuable comments on the initial ideas.

References

[Atallah, 1983] M. J. Atallah. A linear time algorithm for the hausdorff distance between convex polygons. *Information Processing Letter, 17(207-209)*, 1983.

[Berman et al., 2000] H. Berman et al. The protein data bank. *Nucleic Acids Research, 28(235-242)*, 2000.

[Burdick et al., 2001] C. Burdick et al. Mafia: A maximal frequent itemset algorithm for transactional databases. In *ICDM*,01.

[Cheng et al., 1999] C. Cheng et al. Entropy-based subspace clustering for mining numerical data. In *SIGKDD*, 1999.

[Jiang et al., 2003] M. Jiang et al. Feature mining paradigms for scientific data. In *SIAM*, 2003.

[Koperski and Han, 1995] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *SSD*, 1995.

[Kramer et al., 2001] Stefan Kramer et al. Molecular feature mining in HIV data. In *SIGKDD*, 2001.

[Mehta et al., 2004] S. Mehta et al. Detection and visualization of anomalous structures in molecular dynamics simulation data. In *IEEE VIS*, 2004.

[Morimoto, 2001] Y. Morimoto. Mining frequent neighboring class sets in spatial databases. In *SIGKDD*, 2001.

[Munro et al., 2003] R. Munro et al. Complex spatial relationships. In *ICDM*, 2003.

[Rao and Suryawanshi, 1996] C. R. Rao and S. Suryawanshi. Statistical analysis of shape of objects based on landmark data. *Proc National Academy of Science, USA*, 1996.

[Richie et al., 2001] D.A. Richie et al. Real-time multiresolution analysis for accelerated molecular dynamics simulations. In *American Physics Society March Meeting*, 2001.

[Sadarjoen et al., 1998] A. Sadarjoen et al. Selective visualization of vortices in hydrodynamic flows. In *IEEE VIS* , 1998.

[Shannon, 2001] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mobile Computing Community Review*, 5(1):3–55, 2001.

[Shekhar and Huang, 2001] S. Shekhar and Y. Huang. Discovering spatial co-location patterns: A summary of results. *Lecture Notes in Computer Science*, 2001.

[Vendruscolo and Domany, 1997] M. Vendruscolo and E. Domany. Recovery of protein folding from contact maps. *Folding and Design*, 2(5):295–306, 1997.

[Yan and Han, 2002] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM* , 2002.

[Yang et al., 2004] H. Yang et al. Discovering spatial relationships between approximately equivalent patterns. In *BIOKDD*, 2004.

[Yang *et al.*, 2005] H. Yang, S. Mehta, and S. Parthasarathy. A generalized framework for mining spatio-temporal patterns in scientific data. In *Technical Report OSU-CISRC-5/05-TR14, Ohio State University*, 2005.

[Zaki et al., 1997] M.J. Zaki et al. New algorithms for fast discovery of association rules. In *Technical Report TR651, Rensselaer Polytechnic Institute*, 1997.

[Zaki, 2002] M.J. Zaki. Mining protein contact maps. In *BIOKDD*, 2002.

[Zhang et al., 2004] X. Zhang et al. Fast mining of spatial collocations. In *SIGKDD*, 2004.