A GENERAL FRAMEWORK FOR MINING SPATIAL AND
SPATIO-TEMPORAL OBJECT ASSOCIATION PATTERNS
IN SCIENTIFIC DATA

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the
Graduate School of The Ohio State University

By

Hui Yang, B.S., M.S.

* * * * *

The Ohio State University

2006

Dissertation Committee: Approved by
Dr. Srinivasan Parthasarathy, Adviser
Dr. Gagan Agrawal
Dr. Raghu Machiraju

Adviser
Graduate Program in
Computer Science and
Engineering
ABSTRACT

Advances in computational sciences and data collection techniques have resulted in the accumulation of huge amounts of spatial or spatio-temporal data in a wide range of scientific disciplines, such as bioinformatics, astrophysics, meteorology, and computational fluid dynamics. As a result, data mining—the process of discovering hidden and useful information in datasets—has been employed to facilitate the understanding of important phenomena in such disciplines. Many approaches have been proposed to analyze spatial or spatio-temporal data. However, they often suffer from several major limitations. First, they often model spatial entities as points. However, this leads to a loss in information since the geometric properties of such entities (or features) can play an important role in many scientific applications. Second, they lack effective schemes to model the diversity of spatial or spatio-temporal relationships among features. Modeling such relationships are key to understanding the evolutionary behavior of features in many scientific domains. Finally, they are often not cognizant of domain knowledge when modeling these relationships. This can limit the usefulness of the data mining process and inhibit our ability to effectively reason about important scientific phenomena.

In this dissertation, we present a general and modularized framework to address these limitations when mining spatial or spatio-temporal scientific data. We propose different representation schemes to model the geometric properties of spatial entities. We define Spatial Object Association Patterns (SOAPs) to characterize a variety of relationships among
entities. Furthermore, we introduce *SOAP episodes* to capture the evolutionary nature of such relationships. In addition, we propose multiple reasoning strategies to infer important events based on SOAPs or SOAP episodes. We empirically demonstrate the efficacy of this framework on applications originating from the following scientific disciplines: bioinformatics, computational molecular dynamics, and computational fluid dynamics. Our results show that the framework can discover meaningful and important spatial or spatio-temporal patterns. We also demonstrate that the proposed reasoning strategies can make meaningful inferences on important phenomena in such scientific disciplines. Finally, through such applications, we have empirically shown the potential of employing the proposed framework to realize automated or semi-automated data analysis in different scientific disciplines.
ACKNOWLEDGMENTS

First of all, I wish to thank my collaborators during my graduate career. I cannot imagine what a daunting and lonely task it would have become if I were to accomplish this thesis work all on my own. My collaborators include Monika Jankun-Kelly, Douglas Kerr (also my M.S. co-adviser), Tom Lenosky, Raghu Machiraju, Keith Marsolo, Sameep Mehta, Sandeep Reddy, David Thompson, Duygu Ucar, and John Wilkins.

I would also like to thank my colleagues in the Data Mining Research Lab at the Ohio State University. They make the lab a much enjoyable workplace. I will miss them all. They are Sitaram Asur, Greg Buehrer, Amol Ghoting, Matthew Otey, Shirish Tatikonda, Chao Wang, and Huai Wang. Outside of this lab, the following people have offered me great help in different ways. They are Gagan Agrawal, Hakan Ferhatosmanoglu, Yusu Wang, and Xiaodong Zhang.

I wish to thank my friends and my family for their unconditional support and love. Especially my sister and my parents, even the Pacific Ocean cannot stop them there.

Finally, I sincerely thank my adviser Dr. Srinivasan Parthasarathy for his patience, encouragement and guidance. I am also grateful to Dr. Parthasarathy for his tremendous help outside of academic research.
VITA

1993 ........................................ B.S. Computer Science, Huazhong University of Science and Technology, Wuhan, China

2002 ........................................ M.S. Computer Science, The Ohio State University, Columbus, Ohio, USA

PUBLICATIONS


FIELDS OF STUDY

Major Field: Computer Science and Engineering

Studies in Data Mining: Professor Srinivasan Parthasarathy
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapters</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Spatial, Temporal, and Spatio-temporal Data</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1 Spatial Data</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Temporal Data</td>
<td>6</td>
</tr>
<tr>
<td>1.1.3 Spatio-temporal Data</td>
<td>8</td>
</tr>
<tr>
<td>1.2 Limitations of Existing Analytical Techniques</td>
<td>11</td>
</tr>
<tr>
<td>1.3 Our Solution: A General and Modularized Mining Framework</td>
<td>12</td>
</tr>
<tr>
<td>1.4 Target Scientific Applications</td>
<td>12</td>
</tr>
<tr>
<td>1.5 Thesis Statement</td>
<td>13</td>
</tr>
<tr>
<td>1.6 Main Contributions</td>
<td>14</td>
</tr>
<tr>
<td>1.7 Thesis Organization</td>
<td>15</td>
</tr>
<tr>
<td>2. Related Work</td>
<td>16</td>
</tr>
<tr>
<td>2.1 Temporal Data</td>
<td>17</td>
</tr>
<tr>
<td>2.1.1 Representation of Temporal Knowledge</td>
<td>17</td>
</tr>
<tr>
<td>2.1.2 Time Series Analysis in Statistics</td>
<td>20</td>
</tr>
</tbody>
</table>

Abstract                                                                 ii
Acknowledgments                                                           iv
Vita                                                                      v
List of Tables                                                            x
List of Figures                                                           xi

vii
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Description of Protein Contact Map Dataset</td>
<td>80</td>
</tr>
<tr>
<td>3.2 List of $\beta$-protein folds associated with each SOAP type, distType=L-L</td>
<td>81</td>
</tr>
<tr>
<td>3.3 Optimization Results</td>
<td>95</td>
</tr>
<tr>
<td>3.4 SOAPs in major protein groups, dist=C-C</td>
<td>96</td>
</tr>
<tr>
<td>3.5 SOAPs in major protein groups, dist=B-B</td>
<td>96</td>
</tr>
<tr>
<td>4.1 Potential reasons and inferences associated with each SOAP event</td>
<td>101</td>
</tr>
<tr>
<td>5.1 A summary of BBA5 folding analysis.</td>
<td>129</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Shape Representations: (a) Parallelogram (b) Ellipse (c) Irregular</td>
</tr>
<tr>
<td>3.2</td>
<td>SOAP Types: (a) Star (b) Clique (c) Sequence (d) minLink=2</td>
</tr>
<tr>
<td>3.3</td>
<td>Overview of the framework</td>
</tr>
<tr>
<td>3.4</td>
<td>Mining Star SOAPs</td>
</tr>
<tr>
<td>3.5</td>
<td>2-EquiClass Generation</td>
</tr>
<tr>
<td>3.6</td>
<td>Mining Sequence SOAPs</td>
</tr>
<tr>
<td>3.7</td>
<td>Algorithm-Mining $\text{minLink}$ SOAPs</td>
</tr>
<tr>
<td>3.8</td>
<td>Spatio-temporal episode analysis</td>
</tr>
<tr>
<td>3.9</td>
<td>Generating spatial association patterns for protein contact maps</td>
</tr>
<tr>
<td>3.10</td>
<td>Features extracted from proteins 1a0i (ID from PDB)</td>
</tr>
<tr>
<td>3.11</td>
<td>(a) Si surface at $t_0$ (b) Si surface at $t_{3999}$</td>
</tr>
<tr>
<td>3.12</td>
<td>(a) Dimers on Si surface at $t_{2000}$ (b) Dimers on Si surface at $t_{2500}$ (c) Dimer on Si surface at $t_{3999}$</td>
</tr>
<tr>
<td>3.13</td>
<td>Geometry of the serrated wing: it is composed of three serrations, numbered as one through three from left to right, and a large wing at the tail-end.</td>
</tr>
<tr>
<td>3.14</td>
<td>Vortical flows around the serrated wing, where yellow indicates strong vorticity and blue the opposite.</td>
</tr>
</tbody>
</table>
3.15 Main steps to analyze the UAV dataset ........................................... 88
3.16 Vortices on Y-Z planes (slices) at different X locations. ................. 89
3.17 Comparison among different shape-based vortex representations. .... 90
3.18 Illustration of setting equal-arc landmarks (dark dots) on the boundary of a vortex: A landmark is set at the intersection of the boundary and a dividing line. .................................................. 90
3.19 The above figures demonstrate how vortex events were automatically captured by SOAP events. .................................................. 92
3.20 Running Time on 1.5GB Dataset (minDist=5Å) .............................. 94
4.1 Infer critical events based on individual episodes ............................... 102
4.2 Transitions among different SOAP types and their implications w.r.t. inter-object interactions .................................................. 103
4.3 Model interaction history for a set of features and identify transition rules . 104
4.4 SOAP type transition diagram. ’(#)’ beside each transition indicates the number of this transition. .................................................. 105
4.5 (a) Vortices at time 0 (b) SOAP Formation at time 90 (c) Vortex Merging at time 104 .................................................. 108
4.6 Different Vortex SOAPs are discovered at time 991. ....................... 108
4.7 The MBB of SOAP (1 15 61 81) decreases in the episode [39, 49], which eventually dissipated at time 50 due to object amalgamation. (a) Snapshots of the SOAP at time 39, 44, 49, and the vortices in the vicinity of the SOAP right after its dissipation (b) The fitted linear regression line: area(MBB) vs. time .................................................. 109
4.8 The MBB of SOAP (19 32 103 109) increases in the episode [0, 9]. (a) Snapshots of the SOAP at time 1, 5, 9, and vortices in the vicinity of the SOAP right after its dissipation (b) The fitted linear regression line: area(MBB) vs. time .................................................. 110

xii
4.9 Transition: Sequence ⇒ Clique ⇒ Star. The SOAP (23 39 57 58) is a Sequence in the interval [0 18], a Clique in the interval [19 34], and a Star in the interval [35 57]. ..................................................... 110

4.10 Transition: Sequence ⇒ Star ⇒ Clique. The SOAP (19 27 58 36) is a Sequence in the interval [212 243], a Star in the interval [244 287], and a Clique in [288 313]. It eventually dissipated due to the amalgamation of vortices 19 and 58 at time 314. ..................................................... 111

4.11 Transition: Clique⇒ Sequence. The SOAP (7 8 42 99) is a Clique in the interval [0 17], then evolves into a Sequence at time 18 and eventually dissipates at time 49 ..................................................... 112

4.12 Transition: Star⇒ Sequence. The SOAP (205 211 218 230) forms a Star at time 1170, evolves into a Sequence at time 1180, and finally dissipates at time 1202. ..................................................... 112

5.1 Different conformations of BBA5. Points in each plot correspond to the positions of α-carbon atoms. (a)The native NMR structure of BBA5 based on data from the SCOP website. (b)The initial conformation of both folding trajectories. (c)The last conformation in the first trajectory. (d)The last conformation in the second trajectory. ..................................................... 117

5.2 Main steps to analyze protein folding trajectories ..................................... 120

5.3 Mapping between different types of bit-patterns and 3D motifs in conformations of BBAS folding trajectories. The bit-patterns and 3D motifs shown here were randomly selected from their respective group for illustration purpose. ..................................................... 123

5.4 SOAP-based trajectory summarization: a segment in either of the two BBA5 folding trajectories. (a) After superimposing the relative location of each bit-pattern and pruning away redundant SOAPS. (b) After further generalizing each bit-pattern by its corresponding 3D motif. ..................................... 127

5.5 Native-like conformations in both trajectories based on generalized SOAPS. 131

5.6 Four pairs of similar conformations on the consensus folding pathway of BBA5. ..................................................... 132
CHAPTER 1

INTRODUCTION

Data mining is the process of discovering hidden and meaningful knowledge in a data set. As an area combining techniques from database systems, machine learning, and statistical learning, data mining has been successfully applied to many application domains. Web personalization, network intrusion detection, and customized marketing are a few examples of successful applications. Recently, researchers have started to apply data mining techniques to various scientific domains, such as astronomy, bioinformatics, global climate evaluation and forecasting, and computational fluid dynamics, to facilitate the understanding of the underlying scientific phenomena. In this dissertation, we present a general and modularized framework to analyze spatial and spatio-temporal data originating from a variety of scientific disciplines.

Before we move on to discuss the necessity of this framework, it is important for readers to have a general idea of the nature of such data. Readers may notice that we dedicate a section to temporal data, the reason being that it is one of the two inseparable components of spatio-temporal data. Moreover, all the characteristics of temporal data automatically carry on to spatio-temporal data.
1.1 Spatial, Temporal, and Spatio-temporal Data

It is challenging to analyze spatial, temporal, or spatio-temporal data. This can mainly be ascribed to the complex nature of such data themselves. Thus in this section, we discuss the main features of such data.

1.1.1 Spatial Data

Spatial data consists of three components: *map(s)*, *spatial objects or entities*, and *auxiliary information*. The first component, *map(s)*, typically defines a universal reference space for all the involved objects. More than one map can be involved in a spatial dataset. For instance, it is common for data in geographic information systems to consist of maps of multiple cities. Each map contains a set of spatial objects, where each object can be uniquely identified by its location. Furthermore, it is often necessary for a mining task to take into account other properties of spatial objects, including geometric (e.g., shape) and non-geometric properties (e.g., object type). Auxiliary information, the third component of spatial data, usually contains supplementary data, for instance, proprietary information. In this dissertation, we do not consider such information.

To represent spatial data, one generally adopts one of the following two major representation schemes: *raster* and *vector*. The raster scheme is attribute-oriented. It transforms spatial data into a collection of pixel-based “layers”. Each layer is associated with one object attribute of interest (e.g. object size) and is constructed in the following manner. For each object, it first identifies all the pixels that are occupied by the object, and then assigns each identified pixel the attribute value of the said object. Thus a layer represents the spatial distribution of all objects in a given map with respect to the layer’s associated attribute. In
contrast to the raster scheme, the vector representation scheme is object-oriented. It considers each spatial object as a whole, and represents it as a vector of attributes of interest. In this dissertation, we use the vector representation scheme.

We next discuss the main features of spatial data. These features are also the ones that set spatial data apart from non-spatial data.

1. **Spatial Correlation or Autocorrelation**

   Spatial objects in the same vicinity are often related to each other, known as spatial correlation or autocorrelation. Such a phenomenon is also referred to as the first law of geography\(^1\), first proposed by Waldo Tobler [170]. Well-known measures of spatial correlation include Ripley’s K, Moran’s I and Geary’s C [39].

   Although the concept of spatial correlations and autocorrelations is typically applied to geographical data, it has been observed and empirically validated that it can also be applied to spatial data in other domains. For instance, we have shown that spatially correlated (or autocorrelated) non-local structures in protein contact maps can be used to identify structurally similar proteins [186].

   **Implications:** This feature forms the foundation for mining spatial or spatio-temporal association patterns. We believe that such patterns if defined properly, can help uncover the inherent correlation (or autocorrelation) structure among spatial objects.

2. **Complex Spatial Relationships**

   Another notable feature of spatial data is that spatial objects can have multiple types of spatial relationship. They can be either quantitative or qualitative. Note that non-spatial relationships also exist in spatial data, for instance, relationships concerning

\(^{1}\text{Everything is related to everything else, but near things are more related than distant things.}\)
the size of different objects. Spatial relationships are typically categorized into the following types:

- **Topological relations:** These relations are purely qualitative and are preserved under topological transformations such as translation, rotation, and scaling. They characterize the pair-wise intersecting relations among the following three object components: interior, boundary, and complement. A 9-intersection model proposed by Egenhofer [47] is widely used to compute and reason about topological relations among objects.

- **Directional relations:** These relations describe where objects are placed relative to one another. They are purely qualitative and asymmetric. Three basic concepts are required to determine directional relations: the primary object, the reference object, and the frame of reference [36]. An approach proposed by Frank [59] is often adopted to compute and reason about directional relations among objects. Such an approach partitions the underlying space into equal sub-spaces, with the reference object located at the center. It then determines the directional relation between the reference object and any primary object by simply identifying which sub-space the primary object is located in.

- **Metric relations:** These relations are quantitative and usually concerned with distances among spatial objects. For example, the neighboring relationship between two objects is a specific metric relation, as it is concerned with the distance between two objects.

- **Set-oriented relations:** These relations characterize relationships among different sets of objects. For instance, two sets of objects have a *contains* relationship
if one set is a subset of the other. Other typical set-oriented relations include union, intersection, and difference.

**Implications:** Spatial associations capture different relationships among spatial objects. However, due to the existence of multiple spatial relations, it is necessary to have different types of spatial associations in order to capture different types of spatial relationships. For instance, collocation patterns are a type of spatial associations dedicated to capturing neighboring relationships, a type of metric-based relationships, among spatial objects [113, 114, 197]. This is in sharp contrast to non-spatial associations, which only need to consider the co-occurring relationship among items [3].

3. **Large Data Volume**

Typically, spatial datasets, especially scientific spatial datasets, are of very large volume. For instance, a simulation run that simulates the evolution of defects in materials for a few seconds usually produces datasets in the order of GB.

4. **Expensive Spatial Operations**

Given the above features exhibited by spatial data, spatial operations such as range query, nearest neighbor search and spatial join are usually computationally expensive, both in terms of computing resource requirements and complexity. Therefore, there is a strong need for spatial mining algorithms to exploit efficient optimization strategies, including efficient indexing schemes and data structures.
1.1.2 Temporal Data

Temporal data is any data that contains temporal information, where time can be either real or logic. To categorize temporal data, one can adopt different criteria. Below we list several commonly used categorizations for temporal data. Note that such categorizations are not exclusive from each other, as each categorization simply identifies one of many aspects of temporal data.

- **Partially Temporal vs. Fully Temporal:** A temporal dataset is partially temporal if it contains data items whose temporal relationships such as before and after are undecidable. For instance, web log is partially temporal, as it is often impossible to decide the exact access time to the same web page from different web sessions. In contrast, in a fully temporal dataset, the temporal relationship between every pair of data items is decidable.

- **Regularly Timestamped vs. Irregularly Timestamped:** In regularly timestamped data, measurements are recorded at equal-spaced time points. Otherwise, the data is irregularly timestamped.

- **Univariate vs. Multivariate:** Univariate temporal data describes the temporal behavior of one variable. For instance, the dataset that consists of daily temperature in the last 30 days is univariate. Multivariate temporal data on the other hand describes the temporal behavior of more than one variable. For example, trajectories of moving objects in 3D space can be considered as multivariate temporal data, which has three variables: the $x$, $y$, and $z$ locations. Another typical example of multivariate temporal data is a dataset that consists of multiple stock indexes, e.g., Dow Jones Index and NASDAQ Index, during a certain period.
• **Uni-subject vs. Multi-subject:** A uni-subject temporal data involves only one subject, for example, the body temperature of one patient in the last 24 hours. Whereas a multi-subject temporal data involves more than one subject, for instance, the body temperature of five patients in the last 24 hours.

The temporal data studied in this dissertation falls into all the above categories. Please note that *time series*, a data type that has been intensively studied in the statistics community, is a special type of fully timestamped data. In most of the statistical studies, they assume that successive values in a series represent consecutive measures at equally spaced time intervals [22].

We next describe several major characteristics exhibited by temporal data.

1. **Multiple Representation Schemes of Time**
   The following typical questions can be asked when considering representing time:
   - Is time a point or an interval? Does time correspond to a discrete variable or a continuous one? Should it be represented in real time or logic time?

2. **Multiple Temporal Granularity**
   When analyzing temporal data, it is common and often necessary to integrate a concept hierarchy, e.g., year→month→week→day, in the analysis.

3. **Temporal Relationship Reasoning**
   This issue emerges only when time is represented as intervals. As summarized by Allen and others [9, 175, 60, 96], thirteen types of temporal relations can be defined between two intervals. This means that reasoning can become computationally intensive when a dataset contains large amounts of intervals.
**Implications:** While temporal correlation (or autocorrelation) justifies our work on mining temporal associations, the algorithms may need to be able to handle different representation schemes for time. This is because no single representation is suitable for every case. For instance, it is suitable to consider time as points when identifying positions of moving objects. However, when time is associated with evolutionary events such as the merging of two vortices in fluid flows, it is more intuitive to consider time as intervals. Furthermore, the algorithms may also need to take into account the other two aspects i.e., multiple granularity and temporal reasoning.

1.1.3 **Spatio-temporal Data**

Spatio-temporal data records spatial views of objects across time. Data produced from fluid dynamics simulations, or geoinformatics data that tracks the behavior of intrusions are of this type. A fundamental difference between pure spatial data and spatio-temporal data is that objects in spatio-temporal data are under constant change. Regardless of the nature of a change (e.g., location change, shape change), a standard assumption is that change is continuous [36]. This means that while changing, a quantity must pass through all the intermediate values. For example, in the quantity space \{-, 0, +\}, a value cannot change from ’-’ to ’+’ without going through value 0.

It is often impossible to model and represent the continuous properties of changes, especially, when multiple objects are involved. A commonly used approach is to represent a continuously changing system as a sequence of snapshots, where each snapshot records the state of every involved object at a certain time point. We also take such a representation scheme in this dissertation.
Furthermore, the spatial objects of interest are often specific to a given scientific application. For example, one important study in molecular dynamics is to understand different interactions among defect structures. In this case, defect structures are the objects of interest. Whereas, vortices are the objects of interest when studying vortical flows in fluid fields. A closely related issue to scientific spatio-temporal datasets is to detect, extract, and classify objects of interest (e.g., defects in materials and vortices in fluid flows), which is an issue under intensive studies in the past [73, 82, 186, 188]. In this dissertation, we rely on well-known techniques to extract and subsequently classify features in scientific datasets. We next identify the features of spatio-temporal data irrespective of application domains.

1. **Multi-way Correlation or Autocorrelation**

   Objects in spatio-temporal data can be related with each other spatially, temporally, or spatio-temporally. This corresponds to three types of correlation (or autocorrelation), namely, spatial, temporal, or spatio-temporal correlation (or autocorrelation). Thus it is more challenging to analyze and model the correlation structures for spatio-temporal data.

   **Implications:** The problem of discovering spatio-temporal associations is built upon the assumption that spatial objects are related in space or time. However, the correlation strength among objects is dataset dependent. Therefore, it is often necessary to quantize or empirically validate different types of correlations.

2. **Spatio-temporal Relationships**

   Objects in spatio-temporal data not only have spatial and temporal relationships, but also have a new type of relationships, i.e., spatio-temporal relationships. A typical approach to analyze spatio-temporal relationships among objects is to decompose
them into Cartesian product of spatial and temporal relationships [182]. Theoretically, the set of spatio-temporal relations is a Cartesian product of the set of spatial relations and that of temporal relations [182]. In practice, it is often sufficient to study a subset of spatio-temporal relationships. For example, in order to understand the interacting behavior among vortices in fluid flows, it might be enough to study the evolving nature of neighboring relationships among vortices.

Objects in spatio-temporal data also have a new type of relationships, dynamics-based relationships. For instance, relationships concerned with the moving direction and speed of two objects during a certain period.

**Implications:** The diversity of spatio-temporal relationships complicates the design of spatio-temporal association patterns. Unlike spatial or temporal associations, one can normally identify patterns of interest by consulting with domain experts, identifying typical spatio-temporal patterns often turns out to be an iterative and empirical procedure. A potential explanation for this is that it is often not clear what factor(s) dominate(s) a change.

3. **Large Data Volume**

It is typical that many spatio-temporal applications produce large amounts of data. This often leads to poor data accesses and unsatisfying scalability. To alleviate the problems, people often resort to efficient indexing structures.

4. **Spatial or Temporal Mismatch Among Objects**

Another issue in spatio-temporal data is that there might be a mismatch between different objects’ spatial or temporal information. For example, in wetland monitoring studies on rainfall, vegetation index, and soil erosion, measurements on different
subjects might be collected based on different schedules. For instance, rainfall measurements can be collected monthly, while vegetation indexes collected seasonally, and soil erosion collected yearly. Furthermore, these measurements can be associated with very different locations.

**Implications:** The mismatch directly affects the discovered spatio-temporal associations’ quality and strength, as approximation techniques, such as mean averaging or aggregations, will be needed to make sure all objects of interest have similar spatial and temporal distributions.

### 1.2 Limitations of Existing Analytical Techniques

A large number of techniques have been proposed to analyze spatial and spatio-temporal data. Unfortunately, many of them suffer from major limitations. First, most existing techniques model spatial entities as points [113, 114, 197]. However, the geometric properties including shape, size, and orientation, of spatial entities are important in many applications. For instance, they are critical in understanding the interactions among vortices in fluid flows. Correspondingly, when measuring distances between objects, these techniques do not take the shape and extent of an object into account. This can lead to delayed or false detection of interactions among entities. Second, they often consider the case where all spatial objects are located in the same space or map. However, it is also important to analyze how spatial entities relate with each other in different maps (or snapshots — when spatial entities are evolving over time). For instance, spatially proximate structures in protein contact maps have been used to indicate whether proteins are structurally similar [186]. Third, such techniques lack effective methods to model diverse spatial and spatio-temporal relationships among objects. Fourth, to our knowledge, none of these existing techniques
has realized strategies to reason about important phenomena with respect to the underlying domain. Finally, most of these techniques are not cognizant of domain knowledge. This can limit the usefulness of such techniques. In order to understand or explain scientific processes, it is often necessary to integrate domain knowledge in the overall analysis.

1.3 Our Solution: A General and Modularized Mining Framework

In this dissertation, we\textsuperscript{2} present a general and modularized framework to address these limitations in analyzing spatial or spatio-temporal scientific data. The framework supports different representation schemes to capture the geometric properties of spatial entities. It introduces the concept of \textit{Spatial Object Association Patterns} (SOAPs) to model a variety of relationships among entities. It further introduces \textit{SOAP episodes} to characterize the evolutionary nature of such relationships. Furthermore, it supports multiple methods to infer important events with respect to the underlying domain by utilizing the discovered SOAPs or SOAP episodes. We empirically demonstrate the efficacy of this framework on applications in the following scientific disciplines: bioinformatics, computational molecular dynamics, and computational fluid dynamics. Our results show that the framework is both effective and efficient at identifying meaningful and important spatial or spatio-temporal patterns. We also demonstrate that one can utilize these patterns to reason about important phenomena in such scientific domains.

1.4 Target Scientific Applications

To demonstrate the feasibility and generality of this framework, we apply it to four scientific applications, among which two originate in bioinformatics, and the other two

\textsuperscript{2}The use of “we” in this dissertation is a stylistic convention commonly used in the field of Computer Science. The work presented in this dissertation is mainly that of the author unless explicitly identified.
in computational fluid dynamics (CFD) and computational molecular dynamics (CMD) respectively. Below we identify the main goals of each application.

- **Protein structural analysis in bioinformatics**: The main goal is to identify structural “fingerprints” for a given protein class (e.g., α-proteins).

- **Protein folding trajectory analysis in bioinformatics**: There are two main goals in this study. First, we would like to detect different folding events (e.g., the formation of a β-turn) and the occurring order of such events. Second, we intend to identify a consensus partial folding pathway by cross-comparing multiple folding trajectories.

- **Analyzing defect-defect interactions in materials in Computational Molecular Dynamics**: There are two main target issues in this application: (i) to model and capture diverse interactions among defects in materials; and (ii) to infer or predict critical events such as defect amalgamation.

- **Analyzing vortex-vortex interactions in Computational Fluid Dynamics**: The target issues of this application are the same as those of the previous one, except that vortices are the feature of interest.

### 1.5 Thesis Statement

Formally stated, the thesis of this dissertation is as follows: spatially proximate features in many scientific disciplines inherently relate to one another. We believe that such relationships, if properly modeled and captured, can facilitate the understanding of important scientific phenomena. We also believe that it is important and necessary to realize general and extensible solutions that can be applied to applications originating from multiple scientific disciplines.
1.6 Main Contributions

In summary we make the following contributions in this work:

1. We propose a general framework to discover spatial and spatio-temporal association patterns in scientific data produced in multiple domains. This framework is functionally extensible due to its modularized architecture. More importantly, such an architecture allows the effective integration of domain or application specific knowledge in the mining process.

2. We present robust techniques for modeling the shape and extent of features (objects).

3. We implement efficient and scalable algorithms to capture stable and persistent interactions among features through the design of appropriate distance functions and interaction types.

4. We propose automated spatio-temporal inferences based on the discovered spatio-temporal patterns.

5. We empirically evaluate the framework on four applications drawn from different scientific disciplines. Two of these applications analyze scientific simulation data in two disciplines, namely, Computational Fluid Dynamics (CFD) and Computational Molecular Dynamics (CMD). The other two originate in bioinformatics, with one concerning protein structural data, and the other protein folding simulation data.

6. We empirically demonstrate that the proposed framework is promising to automate or semi-automate intelligent data analysis for a variety of scientific applications.
1.7 Thesis Organization

The remainder of this dissertation is organized as follows. Chapter 2 gives a brief overview of existing work that is most germane to this work. Chapter 3 describes the proposed framework and reports empirical results on several case studies. Chapter 4 explains in detail several reasoning approaches that make inferences based on spatio-temporal patterns. Chapter 5 presents the entire procedure of applying this framework to analyze simulated protein folding trajectories. Finally, we conclude and outline several future directions in Chapter 6.
CHAPTER 2

RELATED WORK

Analyzing temporal, spatial, or spatio-temporal data has been the focus of many studies for the past two decades. These studies have largely been carried out by researchers in the following three communities: Artificial Intelligence (AI), Data Base Systems (DBS), and statistics. AI approaches generally focus on developing logic systems to manage uncertainties and support reasoning [9, 175, 60, 96, 47, 59]. Whereas DBS approaches target a variety of issues towards building specialized database management systems. For instance, many algorithms have been proposed in the database community for the following issues: efficient data representations, conceptual data models (e.g., E-R model), indexing structures (e.g., B-tree and R-tree), and algebraic or calculus-based query languages. Finally, statistical approaches [22, 39, 43] typically focus on analyzing the inherent correlation (or autocorrelation) structures in datasets and aim to build appropriate mathematical models. Some commonly applied models include different auto-regression models (e.g., the AR(p) model in time series analysis) and general linear regression models.

The mid 1990s saw the emergence of a new data analysis approach for the aforementioned data, namely, the data mining approach. Such an approach is exploratory in nature and aims to uncover various types of knowledge hidden in the data. For instance, through this approach, one can discover collocation patterns from spatial data [149], event
episodes from temporal data [107], and spatio-temporal episodes from spatio-temporal
data [187]. From the implementation point of view, data mining techniques usually com-
bine different techniques from multiple research areas, including artificial intelligence,
database systems, and statistics.

We next briefly examine the previous work carried out on these three types of data. More
focus will be given to related data mining techniques. Excellent summaries of previ-
ous work on analyzing these three types of data can be found in [159] by Snodgrass, [1] by
Abraham et al., and [135] by Roddick et al..

2.1 Temporal Data

2.1.1 Representation of Temporal Knowledge

Studies on temporal knowledge representation have been focusing on representing or
providing solutions to the following fundamental issues: temporal ontology, temporal logic
or algebra, and temporal data model.

1. Temporal ontology: Ontology provides a representation of concepts in the real world
[165]. Several temporal ontologies have been proposed and studied in the past, each
of which relates to a different perspective of time. Typically, the following perspec-
tives have been well-studied. First, the continuity of time: time can be considered as
either discrete points or intervals. Allen and others discussed the appropriateness of
representing time as intervals as against points and proposed interval-based temporal
logic systems [9, 175]. Second, the dimensionality of time: time can be either single-
dimensional or multi-dimensional. Shueiler is one of the early researchers to notice
that it is necessary to distinguish different kinds of time. For instance, in temporal
database systems, one needs to establish notions such as transaction time and valid
time [145]. Following this pioneering study, many studies have been carried out to address the multi-dimensional characteristic of time [161]. Finally, the linearity of time: time is traditionally viewed as a linear variable, which permanently progresses along a single direction. However, time is sometimes considered as a non-linear variable. For instance, Emerson suggests viewing time as a tree [49]. Alternatively, Wolper suggests that time be viewed as an acyclic graph [181].

2. Temporal logic and algebra: Temporal logic combines abstract, formal semantics with the amenability to efficient implementation in temporal database systems and temporal reasoning [32]. It is worth noting that most of the research activities in this area were conducted in the AI community, emerging as early as in the 1960s. A. Prior’s book on temporal logic (1969) is among the earliest publications on this topic. It is followed by Rescher and Urquhart (1971) and McArther in 1976 [159]. Studies on temporal logic continued in the 1980s and 1990s, however, with a changed focus. The focus in the AI community changed to developing new approaches to managing uncertainty and temporal reasoning. Representative work includes those by Allen [9], Freksa [60], and Krokhin et al. [96]. On the other hand, research in the database community started to focus on the implementation of temporal logic systems. As a result, temporal logic starts to play a role in the design and implementation of query languages and integrity constraints. Nowadays, temporal logic is considered as a relatively mature area. It has been applied successfully to solve many practical problems. Furthermore, mathematical foundations for temporal logic, including formal axiomatic systems and proof systems, are also well-developed and available in literature such as the handbook by Gabbay et al. [61]. Another essential component of temporal logic or algebra is concerned with temporal operators. For
example, the following operators have been identified and implemented in temporal
database systems: time-join, time-equivjoin, event-join, time-equiv outerjoin, contain-
join, contain-semijoin, intersect-join, and temporal natural join [66, 164]. Various
algorithms have been proposed to provide efficient solutions for these operators [80].

3. **Temporal data models**: Data models provide conceptual views of data and do not
depend on a specific application. Ideally, data models should be concise, easy to
explain, and capable of describing various phenomenon in the real world. Other than
abstracting conventional concepts such as entities and non-temporal relationships,
temporal data models also need to characterize a new type of relationships, i.e., tem-
poral relationships [80, 160]. A straightforward approach to construct temporal data
models is to add time as a new dimension to conventional data models, especially,
the well-established E-R model [28]. Actually, about a dozen or so temporally en-
hanced E-R models have been reported in the literature [64]. Other types of models
including event-based models [124] and object-oriented models [176] have also been
studied. However, most of these models have seen limited uses and are mainly em-
ployed by geographical information systems. A fundamental reason of this limited
usage might be attributed to the fact that these models are intrinsically comprehen-
sive, since they are actually spatio-temporal models and aim to provide an integrated
view of spatial and temporal data. As a side note, the data model adopted in the
first temporal database proposed by Wiederhold et al. [180] comprises of a set of
entity-attribute-time-value quadruples.
2.1.2 Time Series Analysis in Statistics

Time series analysis is a well-studied and mature area in the statistic community [22, 37, 43]. Also, many commercial statistical tools are capable of time series analysis, including SAS, SPlus, MATLAB, and BMDP. The main focus of statistical time series analysis is to analyze the dependency or correlation among different observations in time series with the objective of mathematically modeling the data [38]. For instance, the classical decomposition model represents time series as three additive components, namely, trend component, seasonal component, and random (or noise) component.

Time series models can typically be classified into linear and non-linear models. Examples of linear models include MA($p$)–the $p^{th}$ order moving average process, AR($q$)–the $q^{th}$ order autoregressive process, and ARMA($p$, $q$)–the combined model involving both autoregressive and moving average processes. Whereas commonly used non-linear models include bi-linear models, random coefficient AR($p$) models, autoregressive conditional heteroscedacity model of order $p$ (ARCH($p$)), and the generalized autoregressive conditional heteroscedacity models (GARCH models) [171].

Statistical time series analysis has been successfully applied to many domains, including atmospheric sciences, economics, engineering, and medical studies, to target a variety of tasks. Typical tasks include data inference and forecasting in atmospheric sciences, process control in medical studies, and process simulation in engineering. However, statistical studies on temporal data exhibit several major limitations. First, they only deal with numeric values. Second, they do not handle missing values effectively. Third, they often lack the capability to integrate the multi-level (or multi-granularity) characteristics of time. Finally, they are often computationally expensive.

\[^{3}\text{heteroscedacity: non-constant variance}\]
2.1.3 Temporal Databases

Ever since Weiderhold et al. published the first article on temporal database in 1975 [180], a huge amount of studies have been conducted towards realizing temporal database systems. As a specific type of database systems, temporal databases store chronolgical information and provide a variety of tools to facilitate a systematical data management environment. To build such an environment, one not only needs to address every aspect associated with traditional database systems, but also needs to address new aspects arising from the temporal data itself. Below, we summarize several main aspects related to temporal databases. Along with each aspect, we also identify representative studies conducted in the past. For more in-depth surveys on temporal database systems, please refer to the surveys conducted by Jensen [80] and Snodgrass [159]. Note that temporal knowledge representation is also an important issue related to temporal database systems, please refer to Section 2.1.1 for a summary of previous work on this issue.

Temporal query languages allow users to access and manage data in temporal databases. The design and implementation of a temporal query language are mainly determined by two factors: (i) the selected temporal logic system; and (ii) the selected temporal data model. Jones et al. [84] described the first general temporal query language LEGOL 2.0, which is algebra based. Clifford et al. [34], Dadam et al. [40], and Lum et al. [104, 103] were among the pioneers to propose a formal semantics including a data model and a calculus-based query language. After these early efforts, research continued on this topic and a considerable amount of studies were conducted. Some forty temporal query languages were proposed, as summarized by Özsoylu [118]. TSQL2 [44] is one such language proposed in this period. The goal of TSQL2 is to combine the then existing approaches to calculus

21
based query languages, to achieve a consensus extension to SQL-92 [109]. Another essential aspect of query languages concerns the implementation of different algebraic operators, for example, \textit{time-equijoin}.

Indexing structures have long been employed in database systems to facilitate fast data accesses. It is no exception for temporal databases. Dadam \textit{et al.} [40] and Lum \textit{et al.} [104, 103] are among the forerunners to implement temporal indexing structures in temporal databases. Two main strategies have been adopted to develop temporal indexing [143, 172]. The first strategy views time as a set of discrete points. Correspondingly, temporal indexes built under this strategy index on values of a single key, which resembles the classic B$^+$-tree index structure [15]. On the other hand, the second strategy views time as a set of intervals and thus suggests range-based indexing structures. Most of these indexes are based on the classic R-tree [69].

Another two very important issues in temporal database systems are: data storage management and database architecture design. Many related studies had been conducted in the past. However, these issues are generic to every kind of database systems, thus we do not discuss them here. Finally, many temporal data based applications have been identified in the database community. For instance, a variety of algorithms have been proposed to address the following two issues: similarity search and efficient retrieval of time series [6, 85, 56, 112].

We conclude this section by quoting a statement from R. Snodgrass on the status of temporal databases [159]. \textit{By this point in time (mid-1987)$^4$, many of the important aspects of time-oriented databases had been addressed, if only in initial fashion. Both algebraic

$^4$This point is specifically marked by the Conference on Temporal Aspects in Information Systems (TAIS) in 1987.
and calculus-based query languages had been defined, prototype implementations existed, and there were solid results in data modeling and conceptual design.

2.1.4 Temporal Data Mining

A huge amount of work has gone into the area of mining temporal data, aiming to discover various kinds of useful information in the data. Although data mining is often considered as an confluent area of machine learning, database systems, and statistics. Nonetheless, nearly all the studies in temporal mining assume non-database temporal data. This is also observed by Roddick et al. [135]. On the other hand, one does see a trend of utilizing statistical or machine learning techniques in temporal mining. For instance, the framework proposed by Rinsford is capable of accommodating temporal semantics into different mining processes [131].

Many mining issues have been identified and addressed for temporal data. Among them, the following have been intensively studied: (1) temporal association pattern mining; (2) sequence and episode mining; (3) temporal data clustering or classification; and (4) trend and similarity discovery. Two strategic goals can be distinguished for these issues. The first goal is to extract descriptive information about the characteristics of some members in the population or the entire population. And the second goal is to predict the values of a population’s characteristics of interest. These two goals are inter-dependent and often combined to validate different mining processes.

Next, we summarize previous work related to each of the above identified issues. Please refer to [135] for a more comprehensive and in-depth discussion.
1. **Temporal association mining**: The problem of association mining has been typically applied to transactional datasets [5], where each transaction consists of multiple items. The goal is to identify items that frequently appear together in the same transaction. Conventional association mining does not consider temporal information associated with each transaction or the temporal order among items in the same transaction, even if it is available. In contrast, temporal associations accommodate temporal information and identify items that are temporally proximate. Chen et al. [29] and Klosgen [92] have separately explored the idea of using temporal associations to characterize different clients’ purchasing behavior, and subsequently facilitating market planning. Another application of temporal associations is to generate causal rules. A causal rule identifies a possible causal relationship between two sets of items, where the two sets are usually temporally proximate. Causal rules are especially useful in medical studies, where a fundamental problem is to identify factors that may cause or relate to certain medical conditions. Blum [21] and Long et al. [101] proposed solutions to identify causal rules in clinical databases and demonstrated potential uses of these rules in medical investigations.

2. **Sequence and episode mining**: The problem of sequence mining was first defined by Agrawal [8] for sales data, for instance, the purchasing history of customers in a bookstore. Sequence mining can be described as follows. Given a collection of timestamped transactions, where each transaction corresponds to an ordered list of items, the goal is to find sequences with support above a user-specified threshold. A sequence consists of a list of temporally ordered elements, where an element is composed of multiple items appearing together in a transaction. Furthermore, two adjacent elements in a sequence need not be temporally adjacent, as long as the
temporal gap between them is less than a user-specified threshold. The notion of *sliding windows* was later introduced to relax the restriction imposed to elements, which requires all items in an element contained by a single transaction. With sliding windows, a sequence element can be composed of items from multiple transactions provided that they are in the same sliding window. Finally, a semantic taxonomy defined over items can also be integrated into the sequence generation process.

After this pioneering work by Agrawal *et al.*, many other studies have been conducted towards designing more efficient solutions. Zaki proposed the algorithm SPADE (Sequential PAttern Discovery using Equivalence classes) [191, 190] to efficiently discover sequence patterns. SPADE uses a *vertical id-list database* format as against the traditional transactional format. As reported by the author, the vertical format can efficiently expedite the candidate sequence generation step, usually the most expensive step, and cut down resource requirements. Furthermore, SPADE uses a lattice-theoretical approach to decompose the original search (lattice) space into small sub-spaces, which can be processed individually in main memory. Finally, SPADE needs at most three database scans, thus greatly minimizes I/O costs.

PrefixSpan (Prefix-projected Sequential PAtterN mining) [121] is another algorithm that provides fast discovery of sequential patterns. It examines only the prefix subsequences and then projects their corresponding postfix subsequences into projected databases. In each projected database, sequential patterns are grown by exploring local frequent patterns. To further improve mining efficiency, PrefixSpan uses two kinds of database projections: level-by-level and bi-level. It also introduces an optimization technique to make use of pseudo-projection. Other than only focusing on efficiency, Unlike most sequence mining algorithms, which assume the input data be
static, Parthasarathy et al. identified the problem of sequence mining in the presence of database updates (e.g., insertion) and user interaction [120]. Efficient techniques were proposed to address such a problem.

Episode mining is another well-studied problem for temporal datasets, specifically, for datasets that contain collections of temporal sequences (e.g., sequences of alarming events from a security monitoring system). An episode is defined as a conjunction of events associated with certain variables and satisfying a set of unary and binary predicates defined for those variables. Episode mining has been mainly studied by the research group led by Mannila [107]. They implement algorithms to discover the following types of episodes: serial, parallel, simple, and non-simple episodes.

Another temporal mining problem is to discover unusual or rare events, for example, failures in emergency plan simulation systems. Previous studies on this problem include the work by Zaki et al. [192] and that by Weiss et al. [179]. Zaki et al. proposed the PlanMine algorithm to trace events that cause plan failures. The algorithm takes two steps. It first filters out frequent but uninteresting events, in other words, events that are frequent but not preceding plan failures. Frequent events in the filtered dataset, which only includes events that precede a plan failure, are then extracted to analyze different plan failures. The algorithm proposed by Weiss et al. [179] deals with failure prediction by first finding rare events. As this algorithm is more akin to the time series classification problem, we will discuss it in detail later.

3. Trend and similarity discovery: The problem of trend discovery in time series analysis can be informally characterized as the process of identifying time series segments that are of similar shapes, for example, steep peaks and upward or downward moves.
Therefore, a critical issue to extract trends is to identify similarities among multiple series. Although the notion of similarity is domain specific, there is a common challenge in computing similarity among time series, which is how to derive comparable time series. Such a challenge arises from the fact that time series often differ in many ways. For example, measures in one series may be derived on a monthly basis, while the other on a yearly basis. Two solutions are commonly adopted to make time series comparable. One solution makes uses of the dynamic time warping technique [18, 89, 90]. Whereas the other relies on dynamic Fourier transformations [4].

Agrawal et al. take a different view towards discovering trends in time series [6]. They not only identify similar subsequences (i.e., trends, referred to as patterns by the authors) across multiple series, but also attempt to find trends embedded at the pattern level. An important concept associated with the latter problem is temporal proximity. It is a temporal interval (or gap), which dictates under what situations one can assume two temporal events are not related with each other. Such an interval is usually determined by the underlying domain. Agrawal et al. introduce a parameterized time window to address temporal proximity. Events are considered related to each other only if they occurred in the same window.

Studies that focus on discovering specific trends are also available in literature. For instance, Indyk et al. proposed a sketch based method to identify representative trends in massive time series data [78]. Han et al. on the other hand studied the problem of extracting periodic patterns from time series data [71].

Closely related to trend discovery is the effort of implementing general-purpose tools to facilitate interactive retrieval of time series [91]. Towards such an objective, Agrawal et al. defined a Shape Definition Language (SDL) [7]. SDL allows user
to interactively describe the pattern shapes (e.g., steep peaks, upward or downward moves) of interest and consequently retrieves the time series that exhibit the shape(s) of interest.

4. **Temporal clustering and classification:** Another well-studied area in temporal data analysis is the application of clustering or classification techniques. These techniques identify similar time series and automatically group them together. In order to apply classification or clustering techniques, issues related to similarity and temporal proximity must also be addressed.

Weigend *et al.* applied the classification algorithm *AutoClass* [26] to cluster financial time series [178]. A set of features such as market volume and volatility at opening of trade is identified for each series. The similarity between multiple series is then computed according to the identified features. To deal with the problem that measurements of the same feature in different series might not be recorded based on a same time line, they applied smoothing techniques. Oates proposed a clustering technique for multivariate time series [117]. The objective is to discover patterns that are shared with series in the same group, while at the same time group are significantly different from the patterns shared by series in different groups. Dynamic time warping is used to measure similarities among different series.

Das *et al.* applied clustering techniques to time series for a very different purpose: temporal rule induction [41]. More specifically, clustering is applied to discretize time series. The proposed method first segments time series into subsequences based on a user-specified time window. The method then applies the selected clustering technique to group subsequences according a certain similarity measure. Next, it
represents each subsequence in a time series by a symbol that is assigned to the group to which it is clustered. Thus, time series are converted into sequence datasets. Finally, a rule derivation algorithm is applied to the transformed data.

Classification techniques for general temporal sequences are also reported in the literature. Zaki et al. propose FeatureMine to extract features from sequence data [193]. Such features are then used to facilitate the classification of temporal sequences. A feature in this context is a sequence of events of items. Weiss et al. implemented a supervised learning technique, timeweaver, to predict rare events [179]. Timeweaver uses a genetic algorithm to generate "prediction patterns", which are used for rare event prediction. The selection, crossover, and mutation operators in the genetic algorithm are designed to reach two goals: to prevent premature convergence and to ensure that most events in the population are covered. The reported results are generally satisfactory.

2.2 Spatial Data

Similar to temporal data, research on spatial data has been conducted by researchers from the AI, database, statistics, and data mining fields. We next identify major research directions in each field and briefly go over previous work along these directions.

2.2.1 Representation of Spatial Knowledge

Spatial knowledge representation, often referred to as spatial ontology and spatial algebra in the AI and database communities respectively [165, 67], is concerned with fundamental problems in spatial data analysis. These problems and their associated studies are listed as follows.
1. **Representation of spatial objects:** A traditional approach, especially in mathematical theories, is to represent spatial entities as points (or points and lines) [36]. However, a minority tradition regards this point-oriented representation as a philosophical error, as pointed out by Simons in [157]. A well-accepted approach in both AI and database fields is to divide objects into simplex objects and complex objects. Simplex objects include point, line, and region in 2D space and basic 3D objects such as cuboid and sphere [68]. Whereas complex objects are represented by combining multiple simplex objects [153].

2. **Representation of physical space:** In the database community, the following two schemes are often considered to represent the physical space: partition-based and network-based [68]. The former scheme represents the underlying space as a set of disjoint partitions. For example, a common approach is to uniformly partition the space into disjoint grids. Spatial objects are then associated with one or more grid where an object is located or occupies. On the other hand, the network-based representation only considers the objects of interest and thus views the space as one embedded with various networks. For instance, in geographic information systems, one can construct networks that include only rivers or interstate highways. Note that these two schemes are very similar to the two typical representation schemes—raster and vector—adopted in geographic information systems or land information systems. Please refer to Section 1.1.1 for an detailed description on these two schemes.

---

5Simons says: “No one has ever perceived a point, or ever will do so, whereas people have perceived individuals of finite extent”.  

30
3. **Spatial relationships:** Spatial relationships can be either qualitative or quantitative. As discussed in Section 1.1.1, topological, directional, and set-based relationships are qualitative, and metric-based relationships are quantitative [47, 59, 152].

4. **Spatial operations:** Güting [67] identifies four types of spatial operations from an algebraic point of view: *spatial selection*, *spatial join*, *spatial function application*, and *other set operations*. Spatial selection is an operation that returns from a set of objects that satisfy a set of predicates. Spatial join compares two sets of objects and returns objects that fulfill predicate(s) defined on spatial attributes. Spatial function application operations are typically applied to a set of objects to derive new spatial data types. For example, the *extend* operator appends a new attribute to objects. Other set operations manipulate whole sets of spatial objects in a special way and are often application specific. For instance, the *overlay* operation computes the elementary regions resulting from overlaying two partitions.

5. **Data models:** Two different taxonomies are used in the AI and database communities. In the AI community, spatial models are typically classified into (i) *quantitative models*, which are well known and commonly adopted in fields such as robotics and vision; (ii) *qualitative models*, which represent space in a pure qualitative manner; and (iii) *hybrid models*, which are a mixture of the quantitative and qualitative models [74]. Hybrid models agree with the *poverty conjecture*, proposed by Forbes [58], which says: “There is no problem independent, purely qualitative representations of space or shape.” Common data models in the database area include *relational*, *object-oriented*, and *object-relational* models [153].
6. **Spatial reasoning**: The last aspect in spatial knowledge representation is to reason about spatial characteristics based on known knowledge. Reasoning can be applied to individual spatial attributes and spatial entities. A lot of research in spatial reasoning is concerned with a specific type of spatial relations. For instance, Egenhofer discussed how to reason about binary topological relations [46]. Work reported in [158] and [158] focus on reasoning directional relations. Spatial reasoning thus far has been dealing with qualitative characteristics, referred to as *qualitative spatial reasoning*. Please refer to [36] for an excellent survey conducted by Cohn and Hazarika on this topic.

### 2.2.2 Spatial Data Analysis in Statistics

Spatial statistics is a well-studied area and a variety of techniques are available to process spatial data from various applications, such as soil science applications [39, 72]. Studies in spatial statistics generally deal with spatially distributed numerical variables. From the knowledge representation point of view, spatial statistics represents spatial entities as points, and only considers objects’ numeric attributes. Many problems have been identified and intensively studied in the statistics community, among which the following are most relevant to spatial mining.

- Measuring and testing for non-zero spatial correlation or autocorrelation: Commonly used spatial correlation measures include *Moran’s I* (Moran, 1950), *Geary’s C* (Geary, 1954), and *Ripley’s K*. A design matrix (or contiguity matrix) is required to compute these measures. Such a matrix essentially quantifies the first law of geography—that nearby things are more related than distant things—by identifying
neighbor locations of a given location and weighing different neighbors differently. A larger weight indicates a stronger spatial correlation.

- **Spatial autoregressive analysis**: The goal of such an analysis is to mathematically model the data. Spatial Auto-regression Model (SAR) and Markov Random Fields Gaussian Mixture Model (MRFGMM) are among the many commonly applied mathematical models [39].

- **Location based prediction for a response variable**: This is usually accomplished through realizing a suitable mathematical data model, for instance, the SAR model.

- **Spatial data sampling**: Sampling is usually used as a pre-processing step for analyzing large amount of data.

As mentioned earlier, statistical spatial analysis can only deal with numerical variables. It also exhibits two other major limitations. One limitation is that spatial autocorrelation analysis often requires expertise, generally only available from domain experts. It is especially so for the contiguity matrix involved in the computation of different autocorrelation measures such as Moran’s I. The other limitation is that it generally does not scale well and is computationally expensive.

### 2.2.3 Spatial Databases

A Spatial Database Management System (SDBMS) is a database system that manages spatial and non-spatial knowledge. The key difference between SDBMS and conventional DBMS is that the former supports spatial data representations and provides efficient spatial data manipulation [67]. Studies conducted on spatial databases have been focusing on several aspects as follows.
1. Spatial data representation or spatial algebra: This aspect identifies potential schemes to represent different types of spatial knowledge, such as spatial data types, spatial operations, and spatial relationships [63, 68].

2. Modeling spatial data: Data models provide conceptual views of the real world. Three data models, namely, relational, object-oriented, and object-relational, are commonly adopted for spatial data [148, 183]. Relational and object-relational models facilitate the integration of spatial data management with conventional relational databases. Object-oriented models on the other hand aim to build database systems that specialize in spatial data administration.

3. Spatial algebra based query languages: For instance, Egenhofer [45] proposed a spatial SQL to support different spatial data types and operations.

4. Spatial indexing: This is probably the most intensively studied area. The main goal is to provide efficient spatial data accesses by accounting for spatial data’s unique characteristics (e.g., large size, location-oriented). Most of the indexing structures reported in literature are tree-based and can be further categorized into two classes, value-oriented and region-oriented. Value-oriented indexes resemble the B-tree index [15] and are based on the values of a chosen key attribute such as location. Region-oriented indexes partition the underlying space and organize spatial entities according to their association with the partitioned space. Region-based indexes support operations on points, for example, quad-tree [169] and k-d tree [129]. They also support operations on objects with spatial extent (e.g., rectangles), for example, R-tree and many of its variations $R^+$-tree and $R^*$-tree [69, 146, 16].
2.2.4 Spatial Data Mining

Spatial data mining is the process of discovering interesting, previously unknown, and potential useful patterns from spatial data. Several mining methods had been intensively studied in the context of spatial data in the past decade. These methods include: spatial clustering, spatial association mining, spatial classification, and spatial outlier detection. Below we identify representative studies reported in the literature for each method. Many in-depth surveys on these methods are also available in the literature [93, 55, 50, 152].

1. **Spatial clustering:** Cluster analysis as a branch in statistics has been under intensive study for a long time. The goal is to identify interesting structures or clusters from the dataset in an unsupervised fashion, such that members in a cluster are as similar as possible whereas the members from different clusters differ as much as possible. PAM (Partitioning Around Medoids) and CLARA (Clustering LARge Applications) are two well-known clustering methods used in statistics [86]. Both methods take a heuristic strategy to find $k$ clusters, where $k$ is usually a user-specified parameter. PAM starts with randomly selecting a representative point (or seed) for each of the $k$ cluster. Such a seed is referred to as the cluster medoid, as it is the most centrally located point in the cluster. PAM then repeatedly tries to find a better medoid for each cluster by comparing all pair-wise distances between non-medoid points and medoids. Such a process continues until certain criteria are fulfilled. PAM performs poorly on large datasets as it considers every data point in a given dataset. To overcome this limitation, Kaufman and Rousseeuw, also the authors of PAM, proposed the sampling based clustering method CLARA. For large datasets, CLARA first randomly samples data points. It then proceeds in a similar fashion as PAM to identify
the $k$ clusters. However, studies show that neither PAM nor CLARA delivers satisfactory performance. An improved approach CLARANS (Clustering large Applications based upon RANdomized Search) was later proposed by Ng and Han [116]. CLARA and CLARANS differ in their sampling strategy. While CLARA has a fixed sample at every stage, CLARANS introduces some randomness in each sampling step. CLARANS can be further improved by taking advantage of efficient spatial indexing techniques such as R-tree and R*-tree, as reported by Ester et al. [54]. One major limitation of CLARA or CLARANS is that the resulting clusters can be anti-intuitive, or simply incorrect. To address this issue, Ester et al. proposed a density-based clustering algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) [52]. DBSCAN defines a cluster as one that consists of at least $\text{minPts}$ points, and for every point in the cluster there exist another point in the same cluster within distance less than a distance threshold. The minPts criterion effectively avoids small clusters, which are considered as noise. DBSCAN also makes use of the R*-tree index structure to locate points in the vicinity. DBSCAN outperforms CLARANS by a factor of between 250 and 1900. Furthermore, DBSCAN does not require the number of clusters ($k$ in CLARANS) a priori. A generalized version of DBSCAN, GDBSCAN, was proposed by Jörg et al. later [144] to deal with objects of extent (e.g., polygons). One main idea of GDBSCAN is to generalize the notion of neighborhood, which is defined as a binary predicate. Moreover, instead of simply counting the objects in a neighborhood of an object as in DBSCAN, GDBSCAN allows the use of other measures to define an equivalent of the cardinality of a neighborhood. Zhang et al. presented another algorithm, BIRCH (Balanced Iterative Reducing and
Clustering), for clustering [196]. BIRCH is capable of constructing clusters incrementally and provides mechanisms to adjust memory requirements to the available memory size. This is realized by introducing two main concepts: Clustering Feature and CF tree, where a CF tree is a balanced tree with two tunable factors.

2. **Spatial classification:** The task of classification is to assign an object to a class based on the attribute values of the object. In the context of spatial data, due to spatial correlation or autocorrelation, objects in the vicinity can also influence the membership of an object of interest. Therefore, a classification scheme needs to take neighboring objects into account as well. Ester *et al.* [53] proposed a spatial classification algorithm based on the well-known ID3 algorithm [130], which was originally designed for relational databases. They extended ID3 to spatial attributes by considering the attribute of objects on a *neighborhood path* starting from the current object. Another classification method proposed by Koperski *et al.* takes a different route [95]. It first transforms the data into predicates and then applies conventional classification algorithms to construct a decision tree. Such a transformation is usually costly and is limited to a few spatial relationships. Chelghoum *et al.* implemented a two-step classifier [195]. The classifier first computes the *spatial-join* between the collection of target objects and other themes. It then builds a conventional decision tree on the join result. As pointed by the authors, such a method may classify a target object into multiple classes. They later proposed another decision-tree based classifier aiming to overcome this limitation and handle multi-layered spatial data [27].

3. **Spatial association mining:** The main goal of spatial association mining is to identify collections of spatial entities that satisfy a set of pre-defined spatial relationships.
These collections are often represented in a form analogous to *itemsets* or *association rules*, as used in conventional association mining [5]. In the context of spatial mining, they are referred to as *spatial association patterns*. Spatial association mining has been applied to many applications, including geographic information applications [94], image analysis [75], census analysis [106], location-aware E-commerce applications [113, 197], and ecological studies [184].

Koperski *et al.* proposed an algorithm that extracts spatial association rules at multiple levels. However, the algorithm only generates “task relevant” rules. A task specifies the kind of spatial knowledge that a user is interested in and must be explicitly expressed by an end-user. An example task would be “Find all the coffee shops that are less than one kilometer from a park.” For a given task, the algorithm first collects all the task-relevant data from the underlying database. This step also converts the relevant data into generalized spatial predicates like g:\texttt{close}.to() (generalized close.to), which only considers the minimum bounding rectangles of involved objects when identifying close objects. The algorithm then proceeds to extract frequent association at the level of generalized spatial predicates by applying traditional association mining approaches such as *Apriori* [3]. Finally, the algorithm identifies associations that involved more detailed and finer, but more expensive, spatial computations, for example, adjacent.to, which requires the exact distance between two objects instead of the corresponding minimum bounding boxes. One main limitation of such an approach is that it is task relevant. Another limitation is that it separates spatial computations from the process of generating spatial associations.

Recently, researchers start to look at another problem, the discovery of *collocation patterns* [149, 113, 114, 197]. Early literature defines a collocation pattern as a set
of spatial entities that are frequently located near to each other, where *near to* is a parameterized spatial predicate. However, collocation patterns can be generalized to reflect other types of spatial relations as well, such as directional relations and topological relations [187]. New measurements are also defined to quantify the spatial characteristics of collocation patterns. They include *participation ratio*, *prevalence*, *confidence*, and *coverage ratio* [77, 149, 184]. Both Morimoto [113] and Shekhar *et al.* [149] implemented algorithms to discover *clique* collocation patterns in a single map. A clique collocation pattern requires all involved parties are near to each other. Built on the work proposed by Shekhar *et al.* [149, 77], Zhang *et al.* implemented a partition-based optimization strategy aiming at finding collocation patterns more efficiently [197]. However, such a strategy is limited as it requires all objects to be points. Consequently, it can not be applied to objects of different shapes and sizes. Recently, Xiong *et al.* proposed an approach to discover collocation patterns for extended spatial objects including line-strings and polygons [184]. For each object, they identify its neighborhood and then consider objects that have overlapped neighborhood as candidate collocation patterns. We notice that previous work on collocation mining exhibits several limitations. First, spatial objects are represented as points. Thus, they do not take objects’ geometric properties such as shape and size into account. Second, all objects are considered to be located in the same map. Third, nearly all the previous work only considers the neighboring relations among objects. To address these limitations, we propose a general framework. Please refer to Chapter 3 for detailed information on this framework.

4. Other spatial mining methods: Other spatial mining methods include spatial outlier detection, location-based prediction, and database-oriented mining approaches. 
Shekhar et al. proposed statistical approaches to detect spatial outliers [154, 151]. They also implemented a framework for location-based prediction through applying autoregression analysis [87, 150]. Ester et al. discussed the advantages of integrating database techniques into spatial mining tasks [51]. They identify a set of database primitives and algorithms that can facilitate diverse data mining tasks. They also proposed a R-tree based indexing scheme, *neighborhood indexing*, which they show is shown able to effectively improve the performance of several typical mining tasks, such as clustering, classification, association mining, and trend detection.

### 2.3 Spatio-temporal Data

Space and time have long been recognized as two indispensable components in various applications, for example, detecting intrusions in geographic information systems. However, due to the complexities of spatio-temporal data, (See Section 1.1.3 for detailed discussion), earlier work in this area often decomposes spatio-temporal data into its two components–spatial and temporal data–and regards spatio-temporal analysis as an extension to either spatial data analysis or temporal data analysis. As a result, many previous spatio-temporal studies are either spatial data or temporal data centric. Recently, there has been an increasing trend of finding efficient solutions to integrate analysis on these two data components. Since spatio-temporal data analysis is a relatively new research area, there are considerably fewer studies reported in the literature. Therefore, instead of reviewing previous studies according to research fields, we divide them into the following topics: spatio-temporal ontology, spatio-temporal reasoning, spatio-temporal indexing, and spatio-temporal data mining.
1. **Spatio-temporal ontology:** The main goal of researches on spatio-temporal ontology is to provide a representation as close to the real world as possible, in the mean time, also as representative as possible. A spatio-temporal ontology needs to support representations of both spatial and temporal knowledge. However, the focus here is how to represent time-varying spatial knowledge. A typical approach is to abstract the time-vary knowledge as *changes*, which can be associated with objects, events, attributes, or relationships among spatial objects [36, 177, 1]. Changes are sometimes categorized into *continuous* and *discrete* in applications such as environmental studies [125]. However, such a categorization does not always hold its place in the field of spatial reasoning, where a change is always considered as a continuous phenomenon [36]. As summarized by Roshannejad and Kainz [136], an object can experience eight possible changes, by changing one or more of the following components associated with spatial objects: spatial geometry, spatial attributes, and spatial topology. Changes associated with spatial objects often lead to various changes in spatial relationships among objects. Most previous work focuses on studying how topological relationships evolve in different spatio-temporal scenarios. A conceptual neighboring relationship graph is often constructed to characterize the evolving paths among different topological relations [48, 132]. For example, the two relations “overlaps” and “meets” are considered as neighbors in such a graph.

2. **Spatio-temporal data models:** Spatio-temporal data models lay at the core of Spatio-temporal Information Systems (STIS). A data model typically consists of the following main components: data types, relationships, operations, and rules to maintain

---

6Strictly speaking, there exists no such thing in the real world which changes in a discrete manner.
data integrity. A substantial number of spatio-temporal data models had been developed or proposed in the last two decades. Although traces of conventional data models such as Entity-Relationship and Object-Oriented models can be easily identified, the main objective is to provide seamless integration between spatial and temporal data. Pelekis et al. provide an excellent review on major spatio-temporal models proposed in the last decade [123]. In this review, spatio-temporal data models are classified into the following ten classes:

- The snapshot model: where spatio-temporal data is modeled as a sequence of pure spatial data recorded at different time moments;
- The space-time composite model: which is based on the principal that every line in space and time can be projected to the spatial plane and be intersected with each other to create a polygon mesh;
- Data models based on simple time-stamping, where each object is simply tagged with two timestamps associated with the object’s creation and cessation;
- The event-oriented model, where events are of major concern and spatial objects are only participants of events;
- The three-domain model: which identifies semantics, spatial domain, and, and temporal domain for spatio-temporal data. The links between space and time are described through different semantics;
- The history graph model: the main purpose of this type of models is to identify all types of changes. Thus it manage both events and objects;
- The spatio-temporal entity-relationship model: which extends the well-known Entity-Relationship (E-R) data model to spatio-temporal data;
• The object-relationship model, where processes that cause different changes are abstracted as a type of relationship among spatio-temporal objects;

• The spatio-temporal object-oriented model, where a spatio-temporal object is defined as a unified object with both spatial and temporal extents;

• The moving object data model, where spatio-temporal data is abstracted as a collection of moving objects including points and regions. It models time as an integral part of spatial entities and captures both changes and movements.

The review also gives a comprehensive comparison of the ten classes from the following aspects: temporal semantics, spatial semantics, spatio-temporal semantics, and query capabilities.

3. *Spatio-temporal reasoning:* As mentioned earlier, spatio-temporal reasoning has been focusing on qualitative spatial relationships, including topological and directional relationships [36]. The underlying assumption for spatio-temporal reasoning is that changes are continuous in space and time. A main strategy to reason about time-varying spatial relations among objects is to construct the conceptual neighboring relationship graph [132, 48]. Coenen *et al.* proposed a constraint-guided approach for quantitative spatio-temporal reasoning in multi-dimensional space [35]. The underlying N-dimensional space is decomposed into hierarchical sub-spaces until an application dependent resolution is reached. This representation is referred to as tesseral representation by the authors. Such an approach is not general as it requires pre-defined constraints. Spatio-temporal reasoning is also studied in a database environment. Ryu *et al.* proposed a spatio-temporal reasoning method [139]. Several
reasoning mechanism were studied to make inferences on moving objects’ future behaviors, where objects are assumed to move along regular and known directions. For example, objects can only move along designated routes.

4. Spatio-temporal indexing: A main goal of indexing is to facilitate spatio-temporal data retrieval, which includes historical data retrieval and predicative data retrieval. Correspondingly, two types of spatio-temporal databases (STDB) can be identified: historical STDBs and predicative STDBs. A historical STDB stores the status of each involved object at past timestamps. On the other hand, predicative STDB stores the most up-to-date status of each involved object and the motion function that describes its current movement [30]. A straightforward approach to spatio-temporal indexing is to create spatial index at each timestamp. The RT-tree [185], proposed by Xu et al. is among the early R-tree [69] based indexing structures for spatio-temporal data. The first partially persistent structure⁷ HR-tree [115], which creates indexes for historical data, is also based on R-tree. Tao et al. later proposed another R-tree based indexing structure, MV3R-Tree [168], to overcome redundancy in HR-tree. Vazirgiannis et al. took a different route to use R-tree for spatio-temporal data. They proposed the 3D R-tree [173], where time is considered as an extra dimension in addition to the two spatial dimensions. Other spatial indexes such as Quadtree and B+-tree were also adapted for the purpose of indexing spatio-temporal data. For instance, Tayeb et al. adapted Quadtree to index the movement of 1D objects [169]. Whereas Jensen et al. proposed a B+-tree based indexing scheme for moving objects [81].

⁷Compared to the straightforward approach, which creates a R-tree at each timestamp, partially persistent structures reduce space consumption by allowing R-trees associated with consecutive timestamps sharing common nodes, if the objects in these nodes do not incur extent changes.
Recently, researchers start to look at the problem of indexing the past, present, and future positions of moving objects. Agarwal et al. [2] use the ideas of so-called kinetic data structures to address the problem. Chakka proposed a two-level indexing structure SETI (Scalable and Efficient Trajectory Index) [25], where spatial and temporal dimensions are treated separately. Time-parameterized R-tree (TPR-tree) [142] and its variations provide another approach to index moving objects. As a TPR-tree variation, the STAR-tree proposed by Procopiuc et al. [128] seems to be best suited for workloads with infrequent updates. With the objective of enabling efficient deletion of data that is no longer valid, Saltenis and Jensen [141] propose the R$^{EXP}$-tree, which extends the TPR-tree to accommodate data with so-called expiration times. Tao et al. [167] adopt assumptions about the query workload and propose another variation of TPR-tree: TRP*-tree. As pointed out by Saltenis et al., these above mentioned structures can only index positions up to the most recent sample point. To overcome this limitation, they propose yet another TPR-tree based indexing scheme [122]. The index is claimed to be the first one that is capable of indexing past, present and future positions of moving objects with partial persistence. The index supports objects moving in one, two, and three dimensions, and it is applicable to continuous variables other than geographical position. While most indexing schemes assume known motion function of moving objects, Tao et al. proposed an indexing scheme that can also index moving objects with unknown motion patterns [166]. We refer readers to [111] for a comprehensive survey on different spatio-temporal indexing structures.
5. *Spatio-temporal data mining*: Similar to research in other fields such as database systems, a straightforward approach to explore spatio-temporal data takes the following steps. It first separates a spatio-temporal dataset into spatial and temporal component. It then applies existing mining techniques to either component. Finally, it derives spatio-temporal knowledge by integrating the discovered spatial and temporal knowledge. The approach proposed by Mennis and Liu for discovering spatio-temporal association from geographic datasets follows such a path [110]. The approach needs a pre-processing step that transforms each variable of interest into a pre-defined hierarchical structure. Such a structure includes both spatial and temporal dimensions. It then applies a conventional association mining algorithm CBA (Classification Based on Associations) [100] to derive spatio-temporal association rules from the transformed datasets. We propose algorithms (see Chapter 3) to discover different types of spatio-temporal patterns from scientific datasets (e.g., datasets drawn from the Fluid Flow Dynamics domain) [187]. To derive spatio-temporal patterns, the algorithms first discover spatial patterns while keeping record of temporal information associated with each pattern, where temporal information is simply identified by the IDs of the snapshots that contain the pattern. Temporal information is then integrated with the discovered spatial patterns to derive spatio-temporal patterns.

We also propose approaches to reason about a variety of spatial-temporal phenomena based on the derived spatio-temporal associations. Such approaches share some of the objectives with approaches for spatial reasoning. Bailey-kellogg and Zhao [11] propose a methodology for reasoning about such problems, referred to as qualitative spatial reasoning (QSR). Their work is methodology driven and mainly focuses conceptual topics such as data representations and manipulations. They also discuss the
use of different spatial primitives to model objects of different shapes and spatial relationships among objects [11]. Fernyhough et al. implemented techniques to detect events by identifying frequently occurring spatial relationships [57, 36]. However, their proposed technique only considers pair-wise relationships. Thus interactions involving more than two features will be missed.

Another research direction in spatio-temporal data mining is to make use of well-developed statistical methods. For instance, Pokrajac and Obradovic apply spatio-temporal autoregression models for location-based predictions [126, 127]. Basak et al. apply Independent Component Analysis (ICA) techniques to model weather data and further derive stable patterns of the identified independent components [14].
CHAPTER 3

MINING SPATIAL AND SPATIO-TEMPORAL OBJECT ASSOCIATION PATTERNS IN SCIENTIFIC DATA

3.1 Introduction

Analyzing spatial data is an important problem in many application domains, including geographical information systems, bioinformatics, scientific and engineering informatics, and computer aided design. The main difference between analyzing such data and data in a transactional form is that an object can influence the properties of objects located in the same spatial neighborhood [170] and therefore must be modeled in the analysis. Analyzing and reasoning about relationships among spatial objects is further complicated if the data is time varying in nature. Data produced from protein folding or fluid dynamics simulations, or geoinformatics datasets that track the behavior of intrusions are examples that have this additional constraint. Mining spatial relationships in these datasets is an important and interesting problem, since specific relationships may be indicators or predictors of upcoming events (e.g., vortex (hurricane) dissipation, crack propagation and amalgamation in materials).
Unfortunately, mining relationships among spatial objects is an extremely challenging task. First, almost all the related work done to date on this problem models each feature as a point in a multi-dimensional space [113, 114, 197]. However, especially in physical sciences, the extent and shape of a feature can play an important role in determining its influence on neighboring objects. Second, there is a strong need to develop techniques to capture and reason about the interactions among the features. These interactions if captured properly can help domain experts to understand the underlying processes in very effective manner. Third, one needs to develop effective techniques to incorporate temporal information in the overall analysis and reason about them. Finally, recent technological advances in computational sciences have resulted in huge amounts of data. Traditional statistical approaches to model such interactions do not scale very well to large datasets. In this chapter we propose a general-purpose framework to address these challenges.

We have noted that a feature’s geometric properties such as shape and extent can play an important role in determining neighborhood relationships. For example as noted by Silver and Wang [156], two (or more) features can combine to form one new large feature. To capture and represent this event in a correct fashion, it is imperative that the change in shape and size is accounted for. Commonly used point based feature representation will not suffice. A straightforward solution to this problem would be to represent the feature by its Minimum Bounding Box (MBB). However MBBs are not ideal for all situations. For example vortices in fluid flows are very well represented by ellipsoids [140] but not by minimum bounding boxes. Alternatively, defect structures in materials may require irregular shape descriptors [108] such as landmarks[133]. Our framework thus supports

---

8We define a feature to be an object or entity of interest. A feature can have multiple attributes (including non-spatial ones) associated with it. For example, a vortex is a feature in a fluid flow simulation. Associated with this feature are spatial attributes such as location and non-spatial attributes such as velocity, swirl etc.
multiple shape types. Figure 3.1 shows different representation schemes supported by the proposed framework.

The interaction among spatially proximate features are important. An appropriate distance metric must be defined to detect when features are close enough to be considered interacting. Simply measuring the distance between centroids of two features is often inadequate for discovering such interactions. For example, in fluid flow, if two vortices are close to each other, they will possibly merge after a few time steps. However, the distance between their centroids can be very large. Therefore, using a centroid-based distance metric will miss this interaction and subsequently the merging event. The distance metric should thus be capable of taking into account the shape and extent of the features. We evaluate the use of four distance metrics and demonstrate that for different applications different metrics are appropriate.

In addition to distance metrics, we need a method to capture the nature of interactions. We define Spatial Object Association Patterns (SOAP) to characterize the interaction among different types of features (objects) at a given moment. We have identified four SOAP types, namely, *clique*, *star*, *sequence*, and *minLink* (Figure 3.2). Different SOAP types can characterize different types of interaction. For example, *clique* SOAPs in Molecular Dynamics data indicate the presence of compact defects. Compact defects are usually more stable and often govern the properties of materials. *Sequence* SOAPs on the other hand can indicate the formation of long extended defects from smaller point defects [134]. In this chapter we propose fast and efficient algorithms for detecting frequently occurring SOAPs. Our algorithms are scalable and can handle large out-of-core datasets.

We have also developed a very simple approach to model evolving SOAPs or sets of features in data produced by scientific simulations. We define and identify three types of
events to describe SOAPs’ evolutionary behavior: formation, dissipation, and continuation. Formation and dissipation of a SOAP indicate the start and end of an interaction among involved features. Whereas continuation characterizes the stability of an interaction. The continuation of a SOAP from its formation to its next dissipation is abstracted as a spatio-temporal episode. Critical events such as merging can then be inferred by analyzing these episodes. Furthermore, by combining episodes associated with multiple SOAP types, we can also model how the interactions among features change over time.

In summary we make the following contributions in this work:

1. We present robust techniques for modeling the shape and extent of features (objects).

2. We have developed fast algorithms for extracting frequent spatial object interactions through the design of appropriate distance functions and interaction types.

3. We have developed a simple yet effective approach for mining spatio-temporal episodes of SOAP patterns. We further demonstrate that an approach that combines information from multiple SOAP models is capable of reasoning about critical events.

4. We have empirically evaluated our approaches on real case study applications and show that the algorithms scale well and are capable of processing large datasets.

We validate our framework on three case study applications drawn from the scientific and engineering community. Two of the applications analyze scientific simulation data, namely, Computational Fluid Dynamics (CFD) and Computational Molecular Dynamics (CMD) and the other case study application is drawn from protein contact map analysis. The main challenges for these applications include feature detection, classification, and then extracting and modeling spatio-temporal or spatial interactions. Many techniques have
been proposed to detect, extract and classify features from such data in the past [73, 82, 186, 188]. In this work we limit our discussion to the last aspect, namely, discovery of spatial or spatio-temporal patterns.

### 3.2 Related Work

Spatial object association patterns proposed in this work share some commonalities with collocation patterns, which identify features that are frequently located in the same neighborhood. However, most work on collocation pattern mining is limited to applications where subjects of interest can be represented as points [149, 113, 114, 197]. Furthermore, all these points are located in a single map, which correspond to a snapshot in our work. Recently, Xiong et al. proposed an approach to discover collocation patterns for extended spatial objects including line-strings and polygons [184]. For each object, they identify its neighborhood and then consider objects that have overlapped neighborhood as candidate collocation patterns. This is similar to the Clique SOAPs defined in this article. However, in their approach they only consider objects located in the same map.

Our research on spatio-temporal association patterns shares some of the objectives with approaches for spatial reasoning. Bailey-kellogg and Zhao [11] propose a methodology for reasoning about such problems called qualitative spatial reasoning (QSR). Their work is methodology driven and mainly focuses conceptual topics such as data representations and manipulations. They also discuss the use of different spatial primitives to model objects of different shapes and spatial relationships among objects [11]. Our framework is an efficient realization of their conceptual methodology for scientific data. In addition we also support the discovery of spatio-temporal episode patterns that is not explicitly considered in their work. Fernyhough et al. implemented techniques to detect events by identifying frequently
occurring spatial relationships [57, 36]. However, their proposed technique only considers pair-wise relationships. Thus interactions involving more than two features will be missed by only considering pair-wise relationships.

3.3 Basic Concepts

3.3.1 Spatial Feature Representation

We propose three different representation schemes: parallelepiped (or parallelogram in 2D), ellipsoid (or ellipse in 2D), and landmarks based representation, where landmarks are sampled boundary points [133]. These schemes can be used to model features from a variety of scientific domains. For instance, parallelograms are suitable to model non-local structures in protein contact maps. Whereas ellipsoids or ellipses are well-suited for vortices. Finally, landmarks are very effective to model highly irregular-shaped features such as defect structures in materials. The number of landmarks needed to represent a feature is domain dependent. The framework also supports elemental shapes such as lines and splines.

As shown in Fig. 3.1a-b, the shape descriptor of a parallelogram or an ellipse can be described as a vector $A_{\text{basic}} = < l_1, l_2, \theta >$. If landmarks are used (Fig. 3.1c), the shape descriptor is $A_{\text{landmark}} = < (x_i, y_i) : 1 \leq i \leq \nu >$, where $(x_i, y_i)$ is the position of the $i^{th}$ landmark. These shape descriptors can be easily extended to 3-D cases.

3.3.2 Data Representation

The dataset $\mathcal{D}$ consists of $n$ features extracted from $r>1$ maps, denoted as $M = \{m_1, m_2, \ldots, m_r\}$. In time-varying data, the $r$ maps correspond to $r$ time frames or snapshots, which are taken at time $t_1, t_2, \ldots, t_r$ ($t_1 < t_2, \ldots, t_r$). For spatial data that involves multiple maps, one can arbitrarily assign a unique ID to each map. For example, the
set of contact maps from different proteins can be treated in the same manner as that for
time-varying data, where the arbitrarily assigned map ID is used in place of a snapshot’s
associated time. The \( n \) features in \( \mathbb{D} \) are categorized into \( l \) types, which are governed by
the underlying domain. The \( l \) feature types are identified by \( l \) unique labels, denoted as
\( \Sigma = \{c_1, c_2, \ldots, c_l\} \). A feature’s geometric properties such as shape and size are captured
based on one of the three representation schemes described in Section 3.3.1. Thus a feature
\( f \) appearing at time \( t_i \) can be described as a vector \( f = < t_i, \text{location}, A_{\text{geo}}, \text{type} > \), where
\( \text{type} \in \Sigma, \text{location} \) identifies \( f \)’s position at \( t_i \), and \( A_{\text{geo}} \in \{A_{\text{basic}}, A_{\text{landmark}}\} \) models the
geometric properties of \( f \) at \( t_i \).

Note that in the rest of the chapter, we refer to a feature corresponding to the above
vector as a \textbf{spatial object}. We also assume the existence of an ordering among the \( l \) feature
types: \( c_1 < c_2 \ldots < c_l \). Furthermore, we refer to a snapshot’s associated time \( t_i \) as its ID.

### 3.3.3 Object-based Distance Metrics

The framework uses the following metrics to measure the distance between two objects
\( o_i \) and \( o_j \) existing in the same snapshot.

- **Point-Point distance**: This is 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 upper and lower 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 $o_j$. If $o_i$ and $o_j$ are ellipsoids (or ellipses), the line-line distance is the shortest distance between the two major axes.

- **Boundary-Boundary distance**: This is the shortest pairwise distance between the sampled boundary points (landmarks) of $o_i$ and $o_j$.

Notice that the last two metrics take objects’ geometric properties into account. The framework also supports Hausdorff distance [10]. 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 $\varepsilon$ 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)$^9$, 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: $(z_i > z_j) \lor [(z_i = z_j) \land [(y_i > y_j) \lor ((y_i = y_j) \land (x_i < x_j))]]$ in 3D, or $(y_i > y_j) \lor [(y_i = y_j) \land (x_i < x_j)]$ in 2D.

### 3.3.4 Spatial Object Association Pattern (SOAP)

A **Spatial Object Association Pattern** (SOAP) of size $k$, denoted as $k$-SOAP, characterizes the **closeTo** or **isAbove** relationships among $k$ object types. The framework supports the discovery of four SOAP types: **star**, **clique**, **sequence**, and **minLink** (Figure 3.2). They can be abstracted as undirected graphs, where a node corresponds to an object-type $c_i \in \Sigma$.

---

$^9$For a 3D MBB, it is defined as the upper-left corner of its top surface.
and an edge \( (c_i, c_j) \) indicates a closeTo or isAbove relationship between \( c_i \) and \( c_j \). These SOAP types can also be represented as lists with different constraints for different SOAP types. We describe each SOAP type and its corresponding list representation as follows.

- **Star SOAPs** (Fig. 3.2a) have a center object-type, which is required to have a closeTo relationship with all the other object-types in the same SOAP. Let \( c_{cntr} \) be the center of a star \( k \)-SOAP \( p \), and \( \{c_{[i]}; i \in [1,k-1]\} \) be the other \( k-1 \) object-types in \( p \), where \( c_{[1]} \leq \ldots \leq c_{[k-1]} \). The SOAP \( p \) can then be represented as the list \( p=(c_{cntr}, c_{[1]}, \ldots, c_{[k-1]}) \), where closeTo\( (c_{cntr}, c_{[i]})=true \) (\( i \in [1,k-1] \)).

- **Clique SOAPs** (Fig. 3.2b) require a closeTo relationship hold between every pair of involved object-types in the same SOAP. Let \( \{c_{[i]}; i \in [1,k]\} \) be the \( k \) object-types in a clique SOAP, it can then be described by the following list \( (c_{[i]}; i \in [1,k]): \forall i,j \in [1,k] closeTo(c_{[i]}, c_{[j]}). \)

- **Sequence SOAPs** (Fig. 3.2c) of size \( k \), \( p=(c_{[i]}; i \in [1,k]) \), satisfy two constraints: (1) closeTo\( (c_{[i]}, c_{[i+1]})=true \) and (2) isAbove \( (c_{[i]}, c_{[i+1]})=true \), where \( 1 \leq i \leq k-1 \).

- **minLink SOAPs** (Fig. 3.2d) are a parameterized SOAP type, where the value of minLink is user-specified. Let minLink=\( l \), minLink SOAPs include all the SOAPs that have linkage \( \geq l \). The linkage of a \( k \)-SOAP \( p_k \) is defined as follows. Let \( n_{i} \) (\( 1 \leq i \leq k \)) be the number of neighbors that the \( i^{th} \) object-type has in \( p_k \), the linkage of \( p_k \) is then defined as \( \min\{n_{i}; 1 \leq i \leq k\} \). Thus, we can represent a \( (\text{minLink}=l) \) \( k \)-SOAP by the following list: \( (c_{[i]}; i \in [1,k]): \forall i \in [1,k](\#\text{Neighbors Of } c_{[i]} \geq l), \) where \( (c_{[1]} \leq \ldots \leq c_{[k]} \). Note that when minLink=\( l \), all star, clique, and sequence SOAPs will be generated.

A SOAP is said to be **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 spatial objects that meet all the requirements specified by $p$. For instance, $(o_i, o_j)$ is an instance of the clique 2-SOAP $(c_1, c_2)$, where $o_i$.type$=c_1$, $o_j$.type$=c_2$, and closeTo$(o_i, o_j)=$true,

![Figure 3.2: SOAP Types](image)

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

It is straightforward to show that frequent star, clique, sequence, and (minLink=1) SOAPs have the anti-monotone property [3]. This means that a $k$-SOAP cannot be frequent if one of its sub-SOAPs is not frequent. A sub-SOAP of a SOAP $p$ corresponds to a sub-list of $p$’s corresponding list. For a star SOAP, its sub-SOAPs also need to have the same center. However, when minLink$>1$, minLink SOAPs will not be anti-monotonic. We illustrate this by an example. Let minLink$=2$, the 2-SOAP $(c_1, c_2)$ has linkage of 1, thus is not a valid minLink SOAP. However, the clique SOAP $(c_1, c_2, c_3)$, which has linkage of 2, might be a valid minLink=2 SOAP. Extra processing is needed to handle this case. For the
first three SOAP types and \( \text{minLink}=1 \) SOAPs, we leverage the anti-monotone property to efficiently prune the search space.

### 3.3.5 Spatio-temporal Episodes

Spatial relationships among objects (or features) evolve over time. As a result, SOAPs of different types also evolve over time. We identify the following three evolutionary events for a SOAP \( p \):

- **Formation**: when the number of \( p \)'s instances changes from zero to non-zero.

- **Dissipation**: when all \( p \)'s instances become invalid. The dissipation of a SOAP can occur due to many reasons. For example feature(s) involved in a SOAP may cease to exist or merge into a new one. Another possible reason is end of interactions among the involved features.

- **Continuation** from time \( t_i \) to time \( t_{i+1} \): if there exists at least one instance of \( p \) in each snapshot taken between \( t_i \) and \( t_{i+1} \), including \( t_i \) and \( t_{i+1} \).

These events characterize the stability of interactions among different features. They are useful to model and subsequently predict features’ future spatio-temporal behaviors.

Formation and dissipation events can occur to a SOAP many times. Thus a SOAP can exist in multiple disjoint temporal intervals, where each interval starts at a formation event and ends at a dissipation event. We refer to a SOAP’s continuation in each of the above temporal intervals as a **spatio-temporal episode**. Let \( I_p^t \) be the set of SOAP \( p \)'s instances at time \( t \), an episode of \( p \) in the discrete interval \([t_s, t_e]\) can then be described as: \( E_p[t_s, t_e] = \{ I_p^t; t_s \leq t \leq t_e \} \). The framework identifies all the episodes associated with each frequent
SOAP. It also extracts information to abstract how an interaction changes over time within an episode or across multiple episodes.

3.4 Framework and Algorithms

An overview of the framework is given in Figure 3.3. In this article, we focus on four tasks enclosed by the dashed rectangle in the figure. We will also present solutions to facilitate spatio-temporal analysis and inferences.

We organize the dataset $\mathcal{D}$ 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 first ordered by the time they appeared, then by their locations at a certain time. This data organization is analogous to the vertical format used for association rule mining [194].
3.4.1 Equivalence Classes

Based on the list-based SOAP representation (Section 3.3.4), we organize SOAPS as equivalence classes. A \(k\)-equivalence class, denoted by \(k\text{-EquiClass}\), is defined as the set of \(k\)-SOAPS that (1) are of the same SOAP type; and (2) have the same prefix, where the prefix of a \(k\)-SOAP consists of its first \(k-1\) elements. For instance, star SOAPS \(<(c_1, c_1, c_1), <(c_1, c_1, c_2), \text{ and } <(c_1, c_2, c_4)\) will be organized into the same 3-EquiClass, with \(<(c_1, c_1)\) being the prefix.

By using equivalence classes, our mining algorithms only need to compute the closeTo or isAbove relationships among objects once (when generating 2-SOAPs). The equivalence classes also help to improve locality while generating SOAPS [194]. As a result, our algorithms can efficiently discover different types of SOAPS from large amounts of data.

3.4.2 Mining Star SOAPS

Figure 3.4 outlines the algorithm that discovers star SOAPS. The first step, \(\text{gen1SOAP}\), generates frequent 1-SOAPs. For each object-type \(c_i\), the procedure counts the number of snapshots that contain at least one object of type \(c_i\). If the count \(\geq \text{minSupp}\), 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, \(\text{gen2EquiClass}\), discovers 2-SOAPs and organizes them into 2-EquiClasses (line 2). The pseudo-code of \(\text{gen2EquiClass}\) is described in Fig.3.5. A 2-EquiClass is generated for each frequent 1-SOAP (Fig.3.5: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 \((c_j)\in P_1\) (Fig.3.5:line 3). For each candidate 2-SOAP \(p_2=(c_i, c_j)\), the procedure identifies all the snapshots where \(p_2\) occurs (Fig.3.5:lines 7-12). An instance of \(p_2=(c_i, c_j)\) is an object pair \((o_i, o_j)\), where
Algorithm mine_starSOAP( \( D \) )

Parameters: minSupp, minRealization, distType, \( \varepsilon \)

1. \( P_1 \leftarrow \text{gen1SOAP}(\); //freq. and prev. 1-SOAPs
2. \( EC_2 \leftarrow \text{gen2EquiClass}(\; D, \; P_1, \; \text{star}, \; \text{parList}); \) //parList: parameters
3. \( k \leftarrow 2; \)
4. while (1){
5. \( EC_{k+1} \leftarrow \emptyset; \) //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. \( E_{k+1} \leftarrow \emptyset; \) //the prefix of \( E_{k+1} \) is \( p_{k,i} \)
9. foreach k-SOAP \( p_{k,j} \in E_k \) \& \( i \leq j \)
10. \( p_{k+1} \leftarrow \text{append}(p_{k,i}, \text{lastElement}(p_{k,j})); \)
11. \( S_{p_{k+1}} \leftarrow \{ t_i : t_i \text{ contains } p_{k,i} \text{ and } p_{k,j} \}; \)
12. foreach \( t_i \in S_{p_{k+1}} \)
13. if ( cntStarInstances( \( t_i; p_{k,i}; p_{k,j} \)) \geq 1) occCnt++;
14. if ( occCnt \geq \text{minSupp} ) \( E_{k+1} \leftarrow E_{k+1} \cup \{ p_{k+1} \}; \)
15. if ( \( E_{k+1} \neq \emptyset \)) \( EC_{k+1} \leftarrow EC_{k+1} \cup E_{k+1}; \)
16. if ( \( \text{minRealization} > 1 \)) markPrevFreqSOAPs( \( E_{k+1}; \)
17. if ( \( EC_{k+1} = \emptyset \)) break; //terminate
18. \( k++; \) //increase SOAP size } //while(1)

Figure 3.4: Mining Star SOAPS

The algorithm next discovers SOAPS of size \( \geq 2 \) (Fig.3.4: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 \( p_{k,i} \) (Fig.3.4: line 10), the algorithm identifies all the snapshots where \( p_{k+1} \) occurs (Fig.3.4:lines 11-14). The \( i^{th} \) \( k \)-SOAP in a \( k \)-EquiClass is denoted as \( p_{k,i} \). The procedure \text{cntStarInstances}( \( t_i; p_{k,i}; p_{k,j} \)) (Fig.3.4:line 13) computes the instances of \( p_{k+1} \) in snapshot \( t_i \) 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

\((o_i.type = c_i) \land (o_j.type = c_j) \land (\text{distance}(o_i, o_j, \text{distType}) \leq \varepsilon)\). If \( p_2 \) occurs in \( \geq \text{minSupp} \) snapshots, then it is frequent and added to \( E_{2,c_i} \) (Fig.3.5:line 13).
Algorithm gen2EquiClass(D, P₁, soapType)

Parameters: minSupp, minRealization, distType, ε

1. \( EC₂ \leftarrow \emptyset \); //the set of 2-EquiClasses
2. foreach \((c_i) \in P₁ \) //\( P₁ \):ordered freq. 1-SOAPs
3. \( E₂,c_i \leftarrow \emptyset \) //2-EquiClass with prefix of \( c_i \);
4. foreach \((c_j) \in P₁ \)
5. if ( \((soapType \notin \{ star \ or \ sequence \}) \land (c_i > c_j)\) ) continue;
6. \( p₂ \leftarrow (c_i, c_j) \); //a candidate 2-SOAP
7. \( S_p₂ \leftarrow \{t_i : (t_i \ contains \ c_i \ and \ c_j)\} \);
8. foreach \( t_i \in S_p₂ \)
9. foreach \((o_i, o_j) \in t_i : (o_i.label = c_i) \land (o_j.label = c_j)\)
10. if ( \((soapType=sequence) \land (\not\exists \ isAbove(o_i, o_j))\) ) continue;
11. if \( distance(o_i, o_j, distType) \leq ε \) addInstance(\( p₂, (o_i, o_j)\));
12. if ( cntInst(\( p₂, t_i \) \( \geq 1 \) ) occCnt++;
13. if ( \( occCnt \geq \text{minSupp} \) ) \( E₂,c_i \leftarrow E₂,c_i \cup \{p₂\} \);
14. if ( \( E₂,c_i \neq \emptyset \) ) \( EC₂ \leftarrow EC₂ \cup E₂,c_i \);

Figure 3.5: 2-EquiClass Generation

they have the same first \( k \)-1 objects and different last object. Frequent SOAPs of the same prefix are organized into one equivalence class as they are being generated (Fig3.4:lines 5,8,14,15). The mining process stops when all the frequent SOAPs have been discovered (Fig. 3.4:line 17).

To discover frequent SOAPs, the algorithm only needs to consider a SOAP’s presence in a snapshot (Fig.3.4: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 (Fig.3.4:line 16). It is necessary to keep SOAPs that are frequent but not prevalent. We explain this by one simple example.

Let \( \{a_1, b_1, b_2\} \) be the only objects in a snapshot and they are neighbors. Furthermore, let \( minRealization = 2 \). In order to derive the two instances \((a_1, b_1)\) and \((a_1, b_2)\) of the 2-SOAP
Algorithm mine_sequenceSOAP( \( D \) )

**Parameters:** \( \text{minSupp}, \text{minRealization}, \text{distType}, \varepsilon \)

1. \( P_1 \leftarrow \text{genSOAP}_1() \); //freq. and prev. 1-SOAPs
2. \( EC_2(\text{or}P_2) \leftarrow \text{gen2EquiClass}(D, P_1, \text{sequence}, \text{parList}) \); //parList:parameters
3. \( k \leftarrow 2 \); //max. size of SOAPs discovered
4. while (1)
5. \( E_{k+1} \leftarrow \varnothing \); //initialize the set of \((k+1)\)-SOAPs;
6. foreach SOAP \( p_{k,i} = \langle c_0, \ldots, c_{k-1} \rangle \in P_k \)
7. \( E_{2,c[i]} \leftarrow E_{2,c[i]} \in EC_2 \) \( \land \left( E_{2,c[i]} . \text{prefix} = c_k \right) \);
8. foreach 2-SOAP \( p_{2,j} \in E_{2,c[i]} \)
9. \( p_{k+1} \leftarrow \text{append}(p_{k,i}, \text{lastElement}(p_{2,j})) \);
10. \( S_{p_{k+1}} \leftarrow \{ t_i : t_i \text{ contains both } p_{k,i} \text{ and } p_{2,j} \} \);
11. foreach \( t_i \in S_{p_{k+1}} \)
12. if (countSeqInstances\((t_i, p_{k,i}, p_{2,j})\) \( \geq 1 \)) occCnt++;
13. if (occCnt \( \geq \text{minSupp} \)) \( P_{k+1} \leftarrow P_{k+1} \cup \{ p_{k+1} \} \);
14. if (\( P_{k+1} = \varnothing \)) return; //terminate the process;
15. if (\( \text{minRealization}>1 \)) markPrevFreqSOAPs\((P_{k+1})\);
16. \( k++ \); //increase SOAP size

//while(1)

Figure 3.6: Mining Sequence SOAPs

(a, b), the 1-SOAP (a) needs to be maintained even if its realization is 1 (<\text{minRealization})
in the snapshot.

### 3.4.3 Mining Clique SOAPs

Clique SOAPs can be discovered by applying two changes to the algorithm for mining star SOAPs (Fig.3.4). The first change is applied to gen2EquiClass (Fig.3.4:line 2) to eliminates 2-SOAPs in the form of \((c_i, c_j): c_i > c_j\), as they are not in lexicographic order (Fig. 3.5: line 5). The second change takes place at line 13 in Fig.3.4, which replaces the procedure cntStarInstance\((t_i, p_{k,i}, p_{k,j})\) with countCliqueInstance\((t_i, p_{k,i}, p_{k,j})\). The procedure identifies the instances of the candidate \((k+1)\)-SOAP \( p_{k+1} \) in snapshot \( t_i \), where \( p_{k+1} \) is obtained from combining two k-SOAPs \( p_{k,i} \) and \( p_{k,j} \) in the same equivalence class. Two
instances from \(p_{k,i}\) and \(p_{k,j}\) are joined to produce an instance of \(p_{k+1}\) if they have the same first \((k-1)\) objects and there is a closeTo relationship between the last object of the two instances.

### 3.4.4 Mining Sequence SOAPs

The pseudo-code is described in Fig.3.6. Unlike mining star or clique 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 frequent 1-SOAPs and 2-SOAPs. The isAbove relationship is checked by the procedure \(isAbove(o_i, o_j)\) (Fig. 3.5:line 10). 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\((c_{[k]}, c_{[j]})\)\& 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 or clique SOAPs, a candidate \((k+1)\)-SOAP \(p_{k+1}\) is frequent if it appears in \( \geq minSupp \) snapshots (lines 10-13). The procedure \(countSeqInstances(t_i, p_{k,i}, p_{2,j})\) (line 12) identifies instances of \(p_{k+1}\) in snapshot \(t_i\), where \(p_{k+1}\) is a candidate SOAP based on \(p_{k,i}\) and \(p_{2,j}\). Instances of \(p_{k+1}\) are computed by combining the instances of \(p_{k,i}\) and \(p_{2,j}\). The two instances \(I_{k,i}\) and \(I_{2,j}\), from \(p_{k,i}\) and \(p_{2,j}\) respectively, can be combined to produce a \(p_{k+1}\) instance 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 14). The algorithm stops when no more SOAPs can be discovered (line 15).
Algorithm mine_minLinkSOAP( $D$ )

Parameters: $minLink$, $minSupp$, $minRealization$, $distType$, $e$

1. $P_1 \leftarrow \text{genSOAP}_1();$ //freq. and prev. size 1 SOAPs
2. $EC_2(P_2) \leftarrow \text{gen2EquiClass}(D, P_1, minLink, parList);$ //parList: parameters
3. $k \leftarrow 2;$ //maximum SOAP size discovered
4. while (1)
5. \hspace{1em} $P_{k+1} \leftarrow \emptyset;$ //initialize the set of ($k+1$)-SOAPs;
6. \hspace{1em} $blackList \leftarrow \emptyset;$ //infreq. SOAPs;
7. \hspace{1em} foreach SOAP $p_k \in P_k$
8. \hspace{2em} $C_{\text{cand}} \leftarrow \text{genCand}(p_k);$ //class-labels that can potentially grow $p_k$
9. \hspace{2em} foreach $c \in C_{\text{cand}}$
10. \hspace{3em} $p_{k+1} \leftarrow \text{append}(p_k, c);$;
11. \hspace{2em} if ( $(p_{k+1} \in P_{k+1}) \lor p_{k+1} \in blackList$ ) continue;
12. \hspace{2em} $M_{p_{k+1}} \leftarrow \{t_i : t_i \text{ contains } p_k \text{ and } c\};$
13. \hspace{2em} foreach $t_i \in M$
14. \hspace{3em} if ( countInstances$(t_i, p_k, c) \geq minRealization$) occCnt++;
15. \hspace{3em} if ( $minLink > 1$) { //check if $p_{k+1}$ meets minLink
16. \hspace{4em} foreach instance $e$ of $p_{k+1}$
17. \hspace{5em} if ( getLinkage$(e) \geq minLink$) cnt++;
18. \hspace{5em} if ( cnt $\geq minRealization$) occCntMinLink++;
19. \hspace{3em} }
20. \hspace{2em} }
21. \hspace{2em} if ( ($minLink>1) \land (occCntMinLink \geq minSupp)$ ) $p_{k+1}.flag=true;$
22. \hspace{2em} if ( occCnt$ \geq minSupp$) $P_{k+1} \leftarrow P_{k+1} \cup \{p_{k+1}\};$
23. \hspace{2em} else $blackList \leftarrow blackList \cup \{p_{k+1}\};$
24. \hspace{2em} if ( $P_{k+1} = \emptyset$ ) return; //terminate the process;
25. \hspace{2em} empty(blackList); //terminate the process;
26. \hspace{2em} if ( $minRealization>1$) markPrevFreqSOAPs($P_{k+1}$);
27. \hspace{2em} k++; //increase SOAP size }

Figure 3.7: Algorithm-Mining $minLink$ SOAPs
3.4.5 Mining minLink SOAPs

The pseudo-code for mining minLink SOAPs is described in Figure 3.7. The algorithm starts with identifying all frequent and prevalent 1-SOAPs and 2-SOAPs (lines 1-2). All the features in a minLink SOAP are required to be in lexicographic order (see Section 3.4.1). Therefore, the procedure gen2EquiClass(), which generates 2-EquiClasses, prunes away 2-SOAPs that are not in lexicographic order (line 4 in Fig. 3.5).

To generate a \((k+1)\)-SOAP, the algorithm uses one \(k\)-SOAP and one 1-SOAP. For each \(k\)-SOAP \(p_k\), the procedure genCand\(p_k\) (line 8) is called to collect the set of features (1-SOAPs) that have a closeTo relationship with at least one feature in \(p_k\). These features can be easily identified from 2-EquiClasses\(EC_2\). Let \(C\) be the set of all such 1-SOAPs. A candidate \((k+1)\)-SOAP \(p_{k+1}\) is then generated by appending a feature \(c_i \in C\) to \(p_k\) (line 10). However, the same \(p_{k+1}\) can be generated multiple times. For example, consider the following 2-SOAPs \(p_1 : \langle c_1, c_2 \rangle\) and \(p_2 : \langle c_1, c_3 \rangle\), \(p_1\) can be joined with \(c_3\) to form a 3-SOAP \(\langle c_1, c_2, c_3 \rangle\). Similarly \(c_2\) can be joined with \(p_2\) to obtain the same 3-SOAP. Therefore, the SOAP will be considered twice. To avoid redundant computation, the algorithm checks if a candidate SOAP has been discovered before processing it. The candidate \((k + 1)\)-SOAP can be either frequent or infrequent. If it is frequent, it can be found in \(P_{k+1}\), the set of \((k+1)\)-SOAPs discovered so far. The algorithm also maintains a blacklist to keep all the infrequent candidate \((k+1)\)-SOAPs that have been processed so far (lines 6, 24, and 26). In Line 11, the algorithm searches both \(P_{k+1}\) and the blacklist to make sure that a candidate SOAP is not re-processed. It proceeds to process \(p_{k+1}\) if it is not in \(P_{k+1}\) or the blacklist (lines 12-16). The procedure countInstances\(t_i; p_k, c\) (line 14) identifies \(p_{k+1}\)'s valid instance in snapshot \(t_i\) by combining instances from \(p_k\) and \(c_i\). An instance \(I_k\) of \(p_k\) can be combined with an object of type \(c_i\) to produce an instance of \(p_{k+1}\).
if they meet the following conditions: (1) \( o_i \) is not in \( I_k \); (2) \( o_i \) has a closeTo relationship with at least one object in \( I_k \); and (3) the linkage of \( I_{k+1} \) is \( \geq \text{minLink} \).

Note that when \( \text{minLink} > 1 \), in order to discover all frequent \((k+1)\)-SOAPs, the algorithm needs to identify all the frequent \( k \)-SOAPs with \( \text{linkage} \geq 1 \). We explain this by an example. Let \( \text{minLink}=2 \), to find the clique 3-SOAPs \((c_1,c_2,c_3)\) (linkage=2), the algorithm needs to join the 2-SOAP \((c_1,c_2)\) and \( c_3 \). Thus, \((c_1,c_2)\) cannot be pruned away even though its linkage is 1 (\(< \text{minLink}\)). As a result, some of the discovered SOAPs can fail to meet the \( \text{minLink} \) criteria. To address this issue, the algorithm computes a pattern’s linkage value (line 18) and flags those patterns that have \( \text{linkage} \geq \text{minLink} \) (line 22). For the same reason stated before, when \( \text{minRealization}>1 \), the algorithm calls the procedure \text{markPrevFreq SOAPs} to label SOAPs being both prevalent and frequent (line 14). The algorithm stops when no more SOAPs can be discovered.

### 3.4.6 Analyzing Spatio-temporal Episodes

For each discovered SOAP, its spatio-temporal episodes can be constructed by identifying its associated formation and dissipation events. Such events can be easily derived by checking the SOAP’s presence at a given time. Let \( \lambda \) be the number of episodes associated with SOAP \( p \), then its episodes correspond to \( \lambda \) disjoint time intervals \( \{[t_{si}, t_{ei}]: 1 \leq i \leq \lambda \} \), where \( t_{si} \) and \( t_{ei} \) mark \( p \)’s \( i \)th formation and dissipation.

We use episodes to address two important issues. First, to make inferences on critical events such as the merging of multiple features. Second, to model how the interactions among a certain set of features evolve over time.

To address the first issue, we analyze episodes individually. The pseudo-code is described in Figure 3.8. Let \( E_p[t_{si}^i, t_{ei}^i] \) be an episode associated with SOAP \( p \). For each of \( p \)’s
Algorithm: Spatio-temporal Episode Analysis

**Input:** $E$: spatio-temporal episodes of different SOAP types

**Analysis 1: Infer critical events**

1.1 Foreach episode $E_p[t^i_s, t^i_e] \in E$
1.2 Foreach instance $I^i_p: t \in [t^i_s, t^i_e]$
1.3 $A^i_p \leftarrow$ size of the MBB of $I^i_p$
1.4 Fit a simple linear regression model over $(A^i_p, t): t \in [t^i_s, t^i_e]$

**Analysis 2: Model interacting history of** $F=\{c_1, \ldots, c_l\}$

2.1 $E_F \leftarrow$ all episodes in $E$ that are associated with $F$
2.2 Sort $E_F$ in increasing order of $e.t_s: e \in E_F$

---

Figure 3.8: Spatio-temporal episode analysis

instance appearing during $[t^i_s, t^i_e]$, we compute the size of the instance’s minimum bounding box (MBB) (line 1.3), where the MBB of a SOAP instance encompasses every object in the instance. We then apply a simple linear regression model to model the trend that an instance’s MBB varies over time (line 1.4). In the model, time is considered to be an explanatory variable, and the size of an instance’s MBB as a dependent variable. Inferences can then be drawn based on this trend. For instance, if the MBB of an instance decreases from $t^i_s$ to $t^i_e$, it is very possible that the involved objects will merge at its corresponding SOAP’s dissipation.

To model the interacting behavior among a set of features of interest, denoted as $F=\{c_1, \ldots, c_l\}$, we take the following two step (See Figure 3.8). We first find all the episodes that are associated with $F$ (line 2.1). These episodes can be associated with different SOAP types. We then order all these episodes by their formation time (line 2.2). The resulting ordered episode sequence is then used to model the interactions among features in $F$. 

68
3.4.7 Optimizations

1. **Neighborhood-based pruning:** This strategy facilitates fast identification of an object’s neighbors. To find all the neighbors of an object o in snapshot ti, the algorithm first identifies o’s *neighborhood* in ti, which is a spherical (or circular) area around o with radius=\(\varepsilon\) (the distance threshold). The algorithm then only considers the objects in o’s neighborhood to locate its neighbors, as all objects outside the neighborhood cannot be o’s neighbors. Objects in o’s neighborhood can be efficiently located due to the data organization scheme described earlier.

2. **Short-circuiting:** This strategy targets at eliminating infrequent candidate SOAPs at an early stage. For each candidate SOAP \(p_{\text{cand}}\) derived from combining two SOAPs of smaller size, \(p_i\) and \(p_j\), the strategy works as follows. It first intersects the two intervals \([p_i.m_{\text{min}}, p_i.m_{\text{max}}]\) and \([p_j.m_{\text{min}}, p_j.m_{\text{max}}]\), which correspond to the range of snapshot IDs that \(p_i\) and \(p_j\) are associated with. If the intersected interval has less than \(\text{minSupp}\) snapshots, \(p_{\text{cand}}\) is infrequent. Otherwise, it proceeds to collect the set of snapshots, denoted as \(M\), that contain both \(p_i\) and \(p_j\). If \(\|M\| < \text{minSupp}\), then \(p_{\text{cand}}\) is infrequent. If \(p_{\text{cand}}\) is still a candidate at this point, the algorithm proceeds to compute its instances in each snapshot \(t_i\in M\). However, snapshots in \(M\) are processed in increasing order of \((\#\text{instances of } p_i)+ (\#\text{instances of } p_j)\). (The more instances a snapshot has, the more time will be required to generate the instances of \(p_{\text{cand}}\).) A counter \(\text{occCnt}\) is maintained to indicate the number of snapshots which contain \(p_{\text{cand}}\) so far. After some point (e.g., when 25% of the snapshots have been processed), the algorithm starts to regularly compare \(\text{minSupp}\) with the sum of \(\text{occCnt}\) and the number of snapshots in \(M\) to be processed. If the sum is less than \(\text{minSupp}\), \(p_{\text{cand}}\) is infrequent and can be discarded.
3. Efficient Data Structures: Efficient data structures are also used to make the algorithms more efficient, both in performance and storage requirement. Specifically, we use bit-sets and fast bit-set operations [24] to realize fast spatial-join, which is implemented by the \textit{countInstances} procedure in each algorithm.

In addition, the SOAP mining algorithms identify frequent $k$-SOAPs ($k > 2$) by processing one snapshot (or map) a time. Thus the algorithms can efficiently handle large out-of-core datasets.

3.5 Experimental Evaluation

In this section we evaluate our framework on datasets from three different scientific domains, namely, Bioinformatics, Computational Molecular Dynamics and Computational Fluid Dynamics. We also evaluate the performance and scalability of this framework.

3.5.1 Case Study 1: Protein Datasets

Discovering important structures in molecular datasets has been the focus of many recent research efforts in biological and chemical informatics. One approach to address this problem is 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. Particularly, we are interested in discovering spatial relationships among non-local patterns across multiple contact maps. 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 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.
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 and create an occurrence vector for each occurrence of a labeled pattern.
6. Generate spatial object association patterns based on the occurrence list.

Figure 3.9: Generating spatial association patterns for protein contact maps

Thus, we need to derive a method to discover approximate patterns, and define a notion of approximate equivalence.

Next, we describe the major steps taken towards generating approximately equivalent non-local patterns in contact maps. Figure 3.9 lists the major steps. We will now describe all the steps except the last one in detail.

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. We chose to look at the distances between the \( \alpha \)-carbons (\( C_\alpha \)) 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 \( C_\alpha \) 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.
2. *Generation of Feature Vectors* Every non-edge position \((i,j)\) in a contact map has eight *neighbor bits* at locations \((i \pm 1, j \pm 1)\), \((i \pm 1, j)\), and \((i, j \pm 1)\). 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 parallelogram (or rectangle)* (*MBP or MBR*) is the minimum parallelogram (or 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.

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 [76]. 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 [76]. 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 MBP (or MBR).
- **Width**: the number of columns in a pattern’s MBP (or 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. It corresponds to \( \theta \) when a pattern is represented as a parallelogram (Figure 3.1(a)).
- **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.
- **\( \phi \)**: this field is required when a pattern is represented as a parallelogram as illustrated in Figure 3.1(a). In the case of using the MBR representation, \( \phi = 0 \).

Thus, a feature vector is an 8-tuple consisting of the above fields. The reason that we require both the height and width of a pattern’s MBP (or MBR) instead of simply using the area is that we believe two patterns should be considered “different” when one MPB (or MBR) has a different number of rows or columns than the other, even if both MBPs (or 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 (\( \beta_1 \)) as:

\[
\beta_1 = \frac{\sum_{i=1}^{n} ((x_i) \cdot (y_i - \bar{y}))}{\sum_{i=1}^{n} ((x_i - \bar{x})^2)}
\]
Notice that $\beta_1$ is a real number in the range $\pm\infty$. This makes the comparison of two patterns’ $\beta_1$ values difficult. Therefore, we convert the $\beta_1$ 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. **Clustering Feature Vectors** 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 MBP (or MBR). As a result, similar patterns should have similar-sized MBPs (or 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.

4. **Choosing the Cluster Size**  
   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 \( e_1, e_2, \ldots, e_n \), 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 8 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, \( c_1, \ldots, c_k \), 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:

(a) For all the vectors in a cluster, compute the minimum and maximum angle values, denoted $Angle_{\text{min}}$ and $Angle_{\text{max}}$.

(b) Partition the interval $[Angle_{\text{min}}, Angle_{\text{max}}]$ into equi-width intervals. We set the interval width to 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, $xStdDev$ ($\sigma_x$) and $yStdDev$ ($\sigma_y$), we assume they follow a Gaussian distribution. Therefore, their entropy can be computed by the following formula [147]:

$$H(x) = \log_2(\sqrt{2\pi e*x*\sigma_x})$$
Finally, the entropy of a cluster is computed as:

\[ H_i = \sum_{i=1}^{6} H(\text{Attribute}_i) \]

In order to pick the “optimal” number of clusters for grouping our feature vectors, we run the \( k \)-means clustering algorithm [105] 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.

5. **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 MBP’s (or MBR’s) upper left bit within the contact map.
- \( c_i \): the column number of the pattern’s MBP’s (or MBR’s) upper left bit within the contact map.
- \( h_i \): the height of the pattern’s MPB (or MBR) at location \((r_i, c_i)\) within the contact map.
- \( w_i \): the width of the pattern’s MBP (or MBR) at location \((r_i, c_i)\) within the contact map.
- \( \theta, \phi \): the two angles formed by a pattern when its geometry being modeled by its minimum bounding parallelogram.

6. **Discovering and Evaluating Frequent SOAPs** We apply the mining algorithms to the dataset to discover different types of frequent SOAPs. To evaluate the “usefulness”
of a SOAP, in the protein context, we propose using a SOAP’s entropy. We do this by integrating the SCOP lineage information of a SOAP’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 SOAP, we identify the list of proteins contained in that SOAP. 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-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 SOAP $S$ are classified at a certain SCOP level, we compute the entropy to measure how well these proteins are distributed among different SCOP categories. Take $L_1$: class as an example. $L_1$ is divided into 11 subclasses, 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 SOAP 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 is determined empirically.

A total of 1,009,755 features are extracted from the 8,732 contact maps. These features are clustered into 28 classes. Many of these features correspond to well-known protein secondary structures and have been validated by domain experts. We represent each feature by its minimum bounding parallelogram and label the parallelogram by the feature’s class ID. For instance, the three features extracted from protein 1a0i (ID from PDB) are classified
into three classes, referred to by their identifiers 5, 21, and 22. Figure 3.10 shows three features (from one part of a large contact map) represented as parallelograms. Table 3.1 describes the characteristics of the input dataset.

<table>
<thead>
<tr>
<th>#maps</th>
<th>#objType</th>
<th>#objects</th>
<th>Avg. #obj/map</th>
</tr>
</thead>
<tbody>
<tr>
<td>8732</td>
<td>28</td>
<td>1,009,755</td>
<td>115</td>
</tr>
</tbody>
</table>

Table 3.1: Description of Protein Contact Map Dataset

We used the protein structural hierarchy\(^{10}\) described in the database of Structural Classification of Proteins (SCOP)\(^{11}\) to evaluate the quality of our results. We find that different types of SOAPs can actually distinguish different protein folding structures. Table 3.2 lists the $\beta$-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

---

10. We use the first two levels: $l_1 : class$ and $l_2 : fold$ in our experiments

11. http://scop.mrc-lmb.cam.ac.uk/scop/
types. Folds in other protein classes such as \(\alpha\)-protein show a similar trend. For example, the \(\alpha\)-protein fold “Cyclin-like” is primarily associated with sequence SOAPS.

<table>
<thead>
<tr>
<th>Star</th>
<th>Immunoglobulin-like beta-sandwich</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Concanavalin A-like lectins/glucanases</td>
</tr>
<tr>
<td></td>
<td>Trypsin-like serine proteases</td>
</tr>
<tr>
<td></td>
<td><strong>Cupredoxin-like</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Acid proteases</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Cysteine proteinases</strong></td>
</tr>
<tr>
<td>clique</td>
<td>Immunoglobulin-like beta-sandwich</td>
</tr>
<tr>
<td></td>
<td>Concanavalin A-like lectins/glucanases</td>
</tr>
<tr>
<td>Sequence</td>
<td>Immunoglobulin-like beta-sandwich</td>
</tr>
<tr>
<td></td>
<td>Concanavalin A-like lectins/glucanases</td>
</tr>
<tr>
<td></td>
<td>Trypsin-like serine proteases</td>
</tr>
<tr>
<td></td>
<td><strong>Lipocalins</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Nucleoplasmin-like/VP</strong></td>
</tr>
</tbody>
</table>

Table 3.2: List of \(\beta\)-protein folds associated with each SOAP type, distType=L-L

### 3.5.2 Case Study 2: Molecular Dynamics Simulation Datasets

The Molecular Dynamics (MD) dataset is generated using the Object-oriented High-performance Multi-scale Multi-resolution Simulator for material science (OHMMS) [134]. A slice of Silicon (Si) lattice is simulated to understand the creation of stable structures and dimer rows. The dataset consists of 4000 Silicon slices, corresponding to 4000 snapshots. There are 94 atoms in each slice.

**Creation of stable structures**

Figures 3.11(a) and (b) show the slice of Si lattice at \(t=0\) and at \(t=3999\) respectively. At \(t=0\), there are no stable structures in the lattice. However, stable structures are formed at \(t=3999\) and can be easily spotted from Figure 3.11(b). These structures corresponds to
the pentagon-like rings formed by five Si atoms. Our framework was able to discover these structures automatically by mining \textit{minLink}=1 SOAPs from the dataset. Two atoms are considered as neighbors if the distance between them \( \leq 2.8\,\text{Å} \), which is a value suggested by domain experts. Our framework was able of detect the formation to these structures. These structures were discovered by using \textit{minLink} SOAP type with \textit{minlink}=1.

Figure 3.11: (a)Si surface at \( t_0 \) (b)Si surface at \( t_{3999} \)

\textbf{Creation of dimer rows}

A dimer is defined as a pair of connected atoms. Please note that not all pairs of bonded atoms qualify as a dimer. \textit{The atoms should be oriented at an angle (around 45 degrees) to be considered a dimer.} Orientation of features is thus very important here and should be considered for mining spatial patterns. A dimer row is created when individual dimers align themselves in a particular fashion. For example, Figure 3.12(c) shows a detected dimer row (enclosed in the rectangle). The figure also shows other dimer rows, which are at different stages of evolution. Dimer rows were discovered by mining sequence SOAPs. As suggested by domain experts, 2 dimers are considered as neighbors if their distance is \( \leq 5.0\,\text{Å} \). This is our distance threshold.
By discovering different types of spatial association patterns, *minLink* and *sequence* in this case, we were able to find two very different types of meaningful structures in the same dataset. This validates our belief in characterizing different types of interactions by different types of association patterns.

SOAP evolution in this dataset is illustrated in Figure 3.12(b) and Figure 3.12(c). Figure 3.12(b) shows two 2-SOAPs. These SOAPs have no interaction between them, however after few time steps a new SOAP is created between them. This new SOAP acts as a link between the other SOAPs and the whole structure - consisting of 5 dimers- is discovered as a sequence SOAP. This episode points to amalgamation of two features and creation of a new larger one. Such events are automatically discovered by our toolkit.

![Figure 3.12: (a)Dimers on Si surface at $t_{2000}$ (b)Dimers on Si surface at $t_{2500}$ (c) Dimer on Si surface at $t_{3999}$](image)

**3.5.3 Case Study 3: Vortical Flows in Aerodynamics**

Advances in Computational Fluid Dynamics (CFD), coupled with today’s high computing power, have led to successful large-scale simulations of complex configurations. For
instance, CFD techniques have been utilized to simulate air flows around spinning missiles [20] and serrated wings of Un-manned Aerial Vehicles (UAV) [70]. Correspondingly, such simulations can be employed to design or to analyze the afore-mentioned devices.

CFD simulations have also been widely used to study and understand the evolving nature of vortices in fluid flows. The reason for this is twofold. First, vortices are commonly considered to be the most important feature that control the dynamics of flow fields [13]. Second, vortices can usually produce many undesirable effects. For example, vortices surrounding an aircraft can lead to reduced lift, control buffeting, audible noise and airframe vibration [88].

Traditional studies on vortices mainly focus on designing algorithms to address the following issues: vortex detection and identification, representation, reconstruction, and visualization. To date, a great deal of techniques have been proposed and developed. We refer readers to [83, 137] for comprehensive surveys on such techniques.

As vortex detection and identification techniques continue to mature, it becomes increasingly important to develop techniques that can address the following issues:

- To automatically capture interactions among vortices and reason about potential consequences of such interactions;

- To provide insights to understand or extrapolate important physical phenomena in fluid flows, such as vortex amalgamation and vortex dissipation.

Following Newton’s law, two vortices are said to be interacting if they are sufficiently close to each other in space. This is quantitatively defined in Section 3.5.3.

12Several definitions are available to define a vortex. The following is one from Lugt [102]: A vortex is the rotating motion of a multitude of material particles around a common center.
In our case study, we are given a CFD simulation dataset of vortical flows around the serrated wing\textsuperscript{13} of an Unmanned Aerial Vehicle (UAV). We refer to this data as the UAV dataset in this chapter. Figure 3.13 illustrates the geometry of this serrated wing. Our colleagues at Mississippi State University have developed algorithms to detect and identify such vortical flows [79]. (See Figure 3.14 for the detection result.)

To address the above-listed issues for the UAV dataset, we utilize the spatio-temporal data mining framework described in Section 3.4. We adopt a landmark-based representation to capture the spatial extent of vortices. We utilize SOAPs and their episodes to model and capture different interactions among vortices around the wing. Furthermore, we show that such patterns can be employed to automatically detect and reason about critical events, such as vortex creation and amalgamation.

**The UAV Vortical Flow Dataset**

The dataset simulates the vortical flows at a given instant around the serrated wing of a UAV. Figure 3.13 shows the geometry of this wing, with the nose pointing to the left side and the tail to the right. This wing consists of three serrations–namely, the first, second and third serration in order from left to right– and a large wing at the tail end. Figure 3.14 visualizes the detected core regions of the vortical flows around this serrated wing. We refer readers to [79] for details of the detection and visualization algorithms. One can observe that several vortical events occurred along the wing, including a vortex creation event above the third serration, and two amalgamation events at the left and right ends of the large wing, respectively. Similar to what Silver et al. proposed in [156], we identify five types of vortical events, namely, vortex creation, dissipation, amalgamation, bifurcation, and continuation.

\textsuperscript{13}This type of wing platforms has been suggested of rendering high maneuverability [70].
Main Objectives and Issues in Analyzing the UAV Dataset

For this UAV dataset, we intend to reach two main objectives. First, we would like to automatically detect the five types of vortical events. Second, we would like to reason about these events, in other words, to identify potential causes for such events.

Our solution relies on two main observations on vortical flows in fluid fields. First, all vortices in the same field interact with each other. However, vortices that are spatially proximate interact more strongly than those that are spatially distant. This is analogous to the First Law of Geography [170]. Second, many vortical events can be attributed to strong interactions among vortices. For instance, considering the two vortical flows above the third serration, they interacted with each other and this interaction became stronger as
they moved closer along the wing from left to right. Eventually, this strong interaction led to their amalgamation at the left end of the large wing, as a result, the creation of a new vortical flow.

Therefore, if we can effectively capture the interactions among vortices and the evolutionary nature of such interactions, we will be able to detect and reason about different events in vortical flows. We utilize SOAPs to model the interaction among vortices, and SOAP episodes to characterize the evolutionary process of the captured interaction. We then reason about the occurrence of different vortical events based on different SOAP events (e.g., SOAP formation and dissipation).

In addition, we need to address the following issues that are specific to the UAV dataset. First, this dataset records a snapshot of the vortical flows in 3D space. In order to apply the proposed spatio-temporal framework, it is necessary to convert this dataset to a virtual spatio-temporal domain. Second, we need to identify a representation scheme to capture the geometry of vortices, since we cannot simplify them to points. Finally, it is imperative to model interacting vortices in a quantitative fashion. In other words, we need to answer the question: Under what conditions are vortices considered interacting? We will address such issues when we next describe the main analysis steps.

Main Steps to Analyze the UAV Dataset

Figure 3.15 lists the main steps taken to analyze the UAV dataset. We next explain each step in further detail.

1. Detecting Regions of The Vortical Flows Our colleagues at Mississippi State University carried out this task by calculating the swirl parameter [17]. This parameter defines a scalar quantity called swirl. Its value indicates the tendency for the fluid to
1. Detect the core regions for each vortical flow
2. Convert the vortical flows from 3D space to a virtual 2D spatio-temporal domain
3. Build correspondence between vortices over “time”
4. Represent vortices by a fixed number of landmarks
5. Discover frequent Star SOAPs of vortices and their episodes
6. Detect and reason about vortical events

Figure 3.15: Main steps to analyze the UAV dataset

swirl at a given point. Contiguous regions of non-zero swirl values are considered
to correspond to distinct vortices. See Figure 3.14 for an illustration of the vortical
flows identified along the serrated wing.

2. Converting The Vortical Flows to a Virtual 2D Spatio-temporal Domain As men-
tioned earlier, to apply the proposed spatio-temporal mining framework to this UAV
dataset, we need to convert it to a virtual spatio-temporal domain. We do this by
converting the 3D space centering around the wing into a sequence of Y-Z planes (or
slices) along the X-axis. We then project the 3D vortical flows to these Y-Z planes.
Each projected vortex on these slices can be considered as a 2D vortex. We next
study how these 2D vortices interact and evolve on these slices along the X-axis.
This converted space can be viewed as a virtual spatio-temporal domain since one
can treat the X-axis as the temporal dimension. For the UAV dataset, we convert it
to a series of 32 slices. They are distributed as follows: 1 slice in front of (i.e., to the
left of) the nose, 4 slices for each of the three serrations, 18 slices for the large wing,
and 1 slice behind (i.e., to the right of) the tail. Figure 3.16 shows several of these
Y-Z planes with the projected vortical areas.
Figure 3.16: Vortices on Y-Z planes (slices) at different X locations.

3. **Building Correspondence Between 2D Vortices** Given two adjacent slices at time $t$ and $t+1$, this task identifies whether two vortices appearing at $t$ and $t+1$, respectively, correspond to two snapshots of a same vortex. We address this by computing the overlapped area occupied by two vortices in different slices. A correspondence is established if two vortices overlap, and no correspondence otherwise.

4. **Representing 2D Vortices by Boundary Landmarks** As described in Section 3.3.1, to model the geometry of features, the proposed framework supports three representations schemes: parallelepiped (or parallelogram in 2D), ellipsoid (or ellipse in 2D), and landmark-based representation. Recall a landmark is a sampled boundary point. We have observed that the projected 2D vortices in the UAV dataset demonstrate a great diversity with respect to their geometric properties. Therefore, we choose the landmark-based representation to capture the geometry of vortices. Accompanying this choice, we need to decide the number of landmarks for a vortex and where to set these landmarks. A simple and more accurate solution would be using all the available boundary points as shown in Figure 3.17(b). However, this is often unnecessary in practice. As shown in Figure 3.17(c), one can still capture the geometry of a
vortex reasonably well by strategically selecting a small number of landmarks on the boundary. We empirically adopt an 8-landmark representation. These 8 landmarks divide the boundary of a vortex into 8 equal arcs. Figure 3.18 illustrates this scheme.

![Figure 3.17: Comparison among different shape-based vortex representations.](image)

![Figure 3.18: Illustration of setting equal-arc landmarks (dark dots) on the boundary of a vortex: A landmark is set at the intersection of the boundary and a dividing line.](image)

5. Discovering Star SOAPs and Episodes The proposed framework supports four types of SOAPs, namely, Star, Clique, Sequence, and minLink=1, where each SOAP type captures a special type of interacting behavior among features, vortices in this case. We choose to discover minLink=1 SOAPs in the UAV dataset, due to their capability to encompass all the other types of SOAPs.
However, before we proceed, we must address the question raised at the end of last section: Under what conditions are vortices considered interacting? Following our observation that many vortical events are usually a consequence of strong interactions, we consider vortices involved in “strong interactions”. We quantify “strong interactions” by following another observation, which states spatially proximate vortices interact more strongly than distant ones. Henceforth, two vortices are said to have strong interaction if their boundary-boundary distance is \( \leq \delta \), where \( \delta \) is a user-specified threshold. In our analysis, we set \( \delta \) to 200.

We applied the mining algorithm to identify all the minLink=1 SOAPs and their corresponding SOAP episodes. This resulted in a total number of 14 SOAPs. Each SOAP is associated with one episode.

6. Detecting and Reasoning About Vortical Events In this step, we detect and reason about different types of vortical events along the serrated wing based on the SOAP episodes discovered previously. Two main techniques are employed to automate this task. First, at a SOAP formation (or dissipation) event, we compare the vortices involved in the SOAP with neighboring vortices in the slice right before the creation event (or right after the dissipation event). This comparison provides clues that can be used to weigh the possibility of different events. For instance, at a SOAP formation event, if we identify some of the vortices in the newly created SOAP also appear in the previous slice, there is a high possibility that the remaining vortices in the SOAP are newly created. In other words, this SOAP formation event is due to one or more vortex creation events. Second, we further improve the certainty of our reasoning results by identifying evolving trends during a SOAP continuation event. For example, we employ simple yet effective approaches to deduce whether vortices
involved in a SOAP are moving towards each other or vice versa. Such approaches include characterizing the changing trend of pair-wise distances between involved vortices or the minimum area covered by such vortices during a SOAP’s continuation.

![Diagram of SOAP and vortex events](image)

**Figure 3.19:** The above figures demonstrate how vortex events were automatically captured by SOAP events.

To illustrate these reasoning techniques, let us look at several SOAP episodes discovered in the UAV dataset. Based on the formation event of a 3-SOAP at slice 21 (Figure 3.19(b)), we first identify all the three involved vortices. We then identify their neighboring vortices at slice 20. As shown in Figure 3.19(a), two neighboring vortices are identified at slice 20. We next compare these two sets of vortices and establish correspondences between them if applicable. This establishes the correspondence for the two vortices located in the upper-left of slice 21 (Figure 3.19(b)). Thus, we can infer that the vortex at the lower-right corner was newly created. And it is this creation that causes the formation of the 3-SOAP at slice 21.
This SOAP continues to exist until it dissipates at slice 26. Upon this dissipation event, similar to what we do at a formation event, we identify and compare two sets of vortices, where the second set of vortices are identified at the slice right after the dissipation event, i.e., slice 27 (Figure 3.19(c)). After this comparison, we were only able to establish the correspondence for the vortex located at the lower-right corner at slice 27 (Figures 3.19(c)). So, we proceed to examine the general changing trend of the distance between the two vortices located in the upper-left quarter from slice 21 through 26. Since this distance keeps decreasing during the SOAP’s continuation, we infer that the two vortices in the upper-left corner merged into one vortex at slice 27. By the same token, based on the SOAP formation event at slice 28 (Figure 3.19(d)), we reason that the two vortices appearing at slice 27 merged into a larger vortex at slice 28. Note that all the above inferences are automatically produced by the reasoning module in our framework.

3.5.4 Performance Evaluation

In this section we present the results on scalability of our framework. We also present result highlighting the importance of our optimization schemes. All the experiments were carried out on a Pentium 1.7GHz machine of 1GB main memory. We applied the SOAP mining algorithms to a large Molecular Dynamics dataset produced by OHMMS. The dataset consists of 387,999 slices of Silicon lattice and is of size 1.5GB. We first identify all the dimers in each slice and then apply the framework to find different types of interactions among dimers.
Scalability Results

The fact that we can process the data incrementally allows the approach to scale well and handle out of core datasets. Figure 3.20 shows the time taken to discover all types of SOAPs at different support thresholds for the 1.5GB dataset. First of all, even for a large dataset we are able to mine all SOAPs in reasonable amount of time. Time increases as support threshold decreases, because more SOAPs will be generated at a lower support value. However, this increase in time is linear. For example, at 20\% support we take 180 seconds to discover all the star SOAPs, the time just increases by 50 seconds when it is at a very low support threshold 0.5\%. Figure 3.20 shows the performance results on the 1.5GB dataset. Please note that minLink SOAP subsumes all other SOAP types. As a result the algorithm can compute all the other SOAP types within roughly the same time (negligible overhears) it takes to compute min-link patterns.

![Figure 3.20: Running Time on 1.5GB Dataset (minDist=5Å)](image)

Figure 3.20: Running Time on 1.5GB Dataset (minDist=5Å)
**Optimization Evaluation**

We next present experimental results to show the significance of using different optimizations. The most time-demanding component in mining frequent SOAPs is to identify valid SOAP instances for a SOAP candidate in each map (or snapshot). This is especially true when the dataset involves large number of maps. Thus, the main target of our optimization strategies is to prune away infrequent SOAPs as early as possible. Table 3.3 summarizes numbers of SOAP candidates that are pruned away at an early stage, referred to as *Early-Prunes* in the table. All results are based on the 1.5GB MD dataset with $\text{minSupp}=5\%$. The last column in Table 3.3 lists the ratio between the number of early-pruned candidates and total number of candidates. The average ratio is around 40% which is roughly in line with the savings in computational time.

<table>
<thead>
<tr>
<th>SOAP Type</th>
<th># Total</th>
<th># Early-Prunes</th>
<th>Gain%</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAR</td>
<td>646</td>
<td>221</td>
<td>34.2</td>
</tr>
<tr>
<td>CLIQUE</td>
<td>367</td>
<td>165</td>
<td>44.9</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>327</td>
<td>132</td>
<td>40.3</td>
</tr>
</tbody>
</table>

Table 3.3: Optimization Results

### 3.5.5 Impact of Distance Metrics

In order to capture different interactions among evolving objects (or spatial relationships in spatial data), it is critical to use an appropriate distance metric. In this section we present results demonstrating the importance of using distance metric that is capable of taking objects’ shape and extent into account. We compare the number of SOAPs generated from using an object-based distance metric and that from the commonly used point-based
metric. Due to space constraint, we only present results generated from the most informative shape representation scheme, which is parallelogram in the protein case, and ellipse in the vortex case.

Tables 3.4-3.5 show the number of SOAPs discovered based on two different distance metrics on protein datasets, centroid-centroid (C-C) distance in Table 3.4 and boundary-boundary (B-B) distance in Table 3.5. SOAPs are further grouped according to their associated protein classes. Using B-B distance, we are able to find more SOAPs, 406 in total as compared to only 235 SOAPs when C-C distance is used. The difference is even more radical in small proteins. With point-point distance only 42 SOAPs were discovered in small proteins. However, it increases to 142 if boundary-boundary distance is used. We would like to point out that most of these identified SOAPs are also verified by domain experts

<table>
<thead>
<tr>
<th>Type</th>
<th>#(α)</th>
<th>#(β)</th>
<th>#(small)</th>
<th>#(peptide)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star</td>
<td>2</td>
<td>103</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>Clique</td>
<td>-</td>
<td>37</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Sequence</td>
<td>4</td>
<td>30</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>6</td>
<td>170</td>
<td>42</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 3.4: SOAPs in major protein groups, dist=C-C

<table>
<thead>
<tr>
<th>Type</th>
<th>#(α)</th>
<th>#(β)</th>
<th>#(small)</th>
<th>#(peptide)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star</td>
<td>8</td>
<td>145</td>
<td>91</td>
<td>13</td>
</tr>
<tr>
<td>Clique</td>
<td>2</td>
<td>39</td>
<td>26</td>
<td>5</td>
</tr>
<tr>
<td>Seq.</td>
<td>6</td>
<td>40</td>
<td>25</td>
<td>6</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>16</td>
<td>224</td>
<td>142</td>
<td>24</td>
</tr>
</tbody>
</table>

Table 3.5: SOAPs in major protein groups, dist=B-B

96
3.6 Conclusions

In this chapter, we present a general framework for mining spatial associations and spatio-temporal episodes for scientific datasets. Features are modeled as geometric objects rather than points. We define multiple distance metrics that take into account objects’ shape and extent and thus are more robust in capturing the influence of an object on other objects in its spatial neighborhood. We have developed algorithms to discover four different types of spatial object association patterns across multiple maps. We also extend our approach to mine for spatial temporal episodes and thereby present a methodology for reasoning about critical events.

Empirical results on three real case study applications, drawn from the scientific and engineering disciplines serve to validate the framework. We show that the discovered interactions are meaningful and can be used to uncover important spatial and spatio-temporal patterns in the underlying scientific domain. We further demonstrate that the different association pattern types our framework when used in conjunction can provide an effective mechanism to support qualitative spatio-temporal reasoning. Performance studies carried out on large out-of-core datasets indicate that the framework is both efficient and scalable.
CHAPTER 4

OBJECT ASSOCIATION PATTERN BASED SPATIO-TEMPORAL REASONING

4.1 Introduction

In Chapter 3, we describe a generalized framework to discover spatial and spatio-temporal association patterns in scientific data drawn from a variety of scientific domains. We also demonstrate the efficacy of this framework by applying it to several scientific applications, including protein structural analysis in bioinformatics, vortical events detection and reasoning in Computational Fluid Dynamics (CFD), and analysis of defect evolution in materials in Computational Molecular Dynamics (CMD).

In summary, the proposed framework implements multiple representation schemes to capture the geometric properties of features. It also employs object-based distance metrics that take into account such geometric properties. Furthermore, the framework supports the discovery of different types of Spatial Object Association Patterns (SOAP), namely, Star, Clique, Sequence and minLink=1. These SOAPs are used to characterize the interaction among different features (objects) at a given moment. Finally, the framework utilizes SOAP episodes to effectively integrate the temporal dimension into the overall analysis. As
discussed in Chapter 3, SOAP episodes can be used to characterize the evolving nature of feature-feature interactions.

In this chapter, we focus on the last functional module in this framework: spatio-temporal reasoning. We present several SOAP-based reasoning methods to make inferences on the following five types of events: creation, dissipation, amalgamation, bifurcation, and continuation [156]. We also report preliminary results on scientific simulation data drawn from Computational Fluid Dynamics. Other main implementation issues for this application include detecting, extracting, and classifying vortices, and then modeling different interactions among vortices. Many techniques have been proposed to detect, extract and classify vortices from such data in the past [73, 82, 188]. In this chapter, we limit our discussion to the problem of spatio-temporal association based reasoning.

4.2 Object Association Based Spatio-temporal Reasoning

Please refer to Figure 3.3 in Chapter 3 for an overview of the proposed framework. In this article, we focus on the two tasks listed in the shaded area, namely, spatio-temporal episodes generation and spatio-temporal reasoning.

In Chapter 3, we present algorithms to efficiently discover frequent and prevalent Star, Clique and Sequence SOAPs from spatio-temporal datasets. These algorithms provide the following information for each discovered SOAP $p$:

- SOAP type, which can be one of the three types, namely, Star, Clique, and Sequence.
- The list of snapshots that contain valid instance(s) of $p$.
- Properties such as location, shape and size of all the objects in an instance of $p$. 

99
We next discuss how this information can be used to reason about the time-varying interactions among objects.

To do this, we first generate spatio-temporal episodes for each discovered SOAP. We then utilize such episodes, to address the following issues:

- Infer critical events based on individual episodes.
- Model the interacting history for a set of features by combining multiple temporally overlapped episodes.
- Reason about future interactions among features based on their interacting history.
- Model the interacting behavior among multiple sets of features.

### 4.2.1 Generation of Spatio-temporal Episodes

Informally, a spatio-temporal episode of the SOAP $p$ corresponds to a time interval during which $p$ stays valid. Relationships among objects can change over time in time-varying data. As a result, SOAPS also change over time as they characterize such relationships. We identify the following three evolutionary events for a SOAP $p$:

- **Formation**: This event occurs when the number of $p$’s instances changes from zero to non-zero.
- **Dissipation**: This event occurs when all the instances of $p$ become invalid. The dissipation of a SOAP can occur due to many reasons. For example, feature(s) involved in a SOAP may cease to exist or merge to form a new feature. The objects can also move away from each other, implying no interaction or very weak interaction.
- **Continuation**: if there exists at least one instance of $p$ in each snapshot taken between $t_s$ and $t_e$, including $t_s$ and $t_e$. 

Formation and dissipation events can occur to a SOAP many times. Therefore, a SOAP can exist in multiple disjoint temporal intervals, where each interval starts at a formation event and ends at a dissipation event. We refer to a SOAP’s continuation in each of the above temporal intervals as a **spatio-temporal episode**. Let \( I_p^t \) be the set of instances of SOAP \( p \) at time \( t \), an episode of \( p \) in the *discrete* interval \([t_s, t_e]\) can then be described as:

\[
E_p[t_s, t_e] = \{ I_p^t : t_s \leq t \leq t_e \}.
\]

To generate spatio-temporal episodes for a SOAP, we need to identify its associated formation and dissipation events. Such events can be easily derived by checking the the SOAP’s presence at a given time. Let \( \lambda \) be the number of episodes associated with SOAP \( p \), then its episodes correspond to \( \lambda \) disjoint time intervals \([t_s^i, t_e^i] : 1 \leq i \leq \lambda \), where \( t_s^i \) and \( t_e^i \) mark the \( i^{th} \) formation and dissipation of \( p \).

### 4.2.2 Inferences Based on SOAP Events

The afore-mentioned events characterize the stability of interactions among different features. In Table 4.1, we list some of the potential reasons for these events and information that can be inferred from these events.

<table>
<thead>
<tr>
<th>Event</th>
<th>Reason</th>
<th>Inference(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formation</td>
<td>Features start interacting</td>
<td>Features moving closer&lt;br&gt;Features may merge</td>
</tr>
<tr>
<td>Continuation</td>
<td>Equilibrium is reached</td>
<td>Creation of stable SOAPs</td>
</tr>
<tr>
<td>Dissipation</td>
<td>Features stop interacting</td>
<td>Dissipation of one or more features&lt;br&gt;Features merged and formed new feature</td>
</tr>
</tbody>
</table>

Table 4.1: Potential reasons and inferences associated with each SOAP event
Most of the inferences listed are easy to understand. Here we explain only the continuation event and associated inference in greater details. For expository purpose, let us consider a SOAP $p$ composed of two objects $A$ and $B$. If SOAP $p$ continues for a long period of time, it can point to some equilibrium. The exact nature of equilibrium is domain specific. For vortices, it is well-known that two vortices rotating in different directions (one in clockwise and the other in anti-clockwise) tend to destroy each other. Therefore a SOAP with long continuation time can imply that $A$ and $B$ are rotating in the same direction (clockwise or anti-clockwise). If however, $p$ dissipates very soon after it is formed, it can imply that the two vortices are rotating in opposite directions.

### 4.2.3 Inferences Based on Individual Episodes

**Algorithm: Infer critical events**  
**Input**: $\mathbb{E}$: spatio-temporal episodes of different SOAP types

1. foreach episode $E_p[t^i_s, t^i_e] \in \mathbb{E}$
2. Foreach instance $I^i_p$: $t \in [t^i_s, t^i_e]$
3. $A^i_p \leftarrow$ size of the MBB of $I^i_p$
4. Fit a simple linear regression model $A_p = \alpha + \beta t$
   over $(A^i_p, t): t \in [t^i_s, t^i_e]$
5. Inferences:
   - if $(\beta > 0) \implies$ (Decreasing interaction)
   - if $(\beta < 0) \implies$ (Increasing interaction$\implies$Object amalgamation)

Figure 4.1: Infer critical events based on individual episodes

By analyzing episodes individually, we can make inferences on critical events such as feature amalgamation and the involved features’ future interacting behavior. The main analysis steps are described in Figure 4.1. Let $E_p[t^i_s, t^i_e]$ be the $i_{th}$ episode associated with
SOAP $p$. For each instance of $p$ in $[t^i_s, t^i_e]$, we compute the size of the instance’s Minimum Bounding Box (MBB) (line 3), where the MBB of a SOAP instance encompasses every object in the instance. We then apply a simple linear regression model to quantify the trend that an instance’s MBB varies over time: $A_p=\alpha+\beta t$ (line 4), where $A_p$ is the area of a SOAP instance’s MBB, and $t$ is a time point in the interval $[t^i_s, t^i_e]$. Inferences can then be drawn based on this trend (line 5). For instance, if the MBB of an instance decreases from $t_s$ to $t_e$, i.e. $(\beta < 0)$, we can infer that the involved objects merge at its corresponding SOAP’s dissipation. Otherwise, we can infer that the interaction among the involved objects becomes weaker over time.

### 4.2.4 Modeling Features’ Interaction History and Generating Transition Rules

![Transition Rules](chart.png)

Figure 4.2: Transitions among different SOAP types and their implications w.r.t. inter-object interactions

As mentioned earlier, different SOAP types characterize different types of interactions among objects. Therefore, for a set of features, the evolutionary nature of their spatial
relationships can be captured by a series of SOAP episodes associated with different SOAP types. We would also like to point out that the three SOAP types are not mutually exclusive in a given interval. For instance, a Clique SOAP corresponds to multiple Star SOAPS centered at each of the involved object. To address this issue, we impose the following order among the three SOAP types: Clique ≪ Sequence ≪ Star, where Clique is the most spatially constrained and Star the least constrained. In the case where two or more overlapped episodes are identified for the same set of features, we choose the episode that is associated with the most constrained SOAP type. Figure 4.2 summarizes the transitions between the three SOAP types. Also given in the figure are the implications and inferences one may make based on each transition.

**Algorithm: Model interacting history and Identify Transition rules**

**Input:** $E$: spatio-temporal episodes of different SOAP types

$F = \{c_1, \ldots, c_i\}$: the set of features of interest

1. $E_F \leftarrow$ all episodes in $E$ that are associated with $F$
2. Sort $E_F$ in increasing order of $e.t:s: e \in E_F$
3. if (episodes have overlapped intervals)
4. Keep the episode that is associated with the most constrained SOAP type
5. Construct $F$’s relationship transition diagram
6. Make inferences using the Transition Table in Figure 4.2
7. Identify relationship transition rules for $F$

Figure 4.3: Model interaction history for a set of features and identify transition rules

The interaction history of a set of features of interest, denoted as $F=\{c_1, \ldots, c_i\}$, is modeled by combining its overlapped episodes. The resulting model is referred to as the Relationship Transition Diagram of $F$ as shown in Figure 4.4. The transition diagram of
$F$ consists of the complete path from $F$’s first SOAP formation event till its last SOAP dissipation event. Note that this path is continuous in space and time.

Figure 4.3 describes the main steps to derive the relationship transition diagram for $F$. We consider all of its associated SOAP episodes as input. We first find all the episodes that are associated with $F$ (line 1). These episodes can be associated with different SOAP types. We then order all these episodes by their formation time (line 2). In the case where multiple overlapped episodes of different SOAP types are identified for $F$, we only keep the most constrained SOAP type (lines 3-4). We next construct the transition diagram of $F$ (line 5). Based on this diagram, we can make important inferences about the spatial relationship among objects in $F$. Note that it is possible that $F$ is not associated with any episode. In such a case, no transition diagram will be constructed.

Based on relationship transition diagrams, we can further characterize the evolutionary nature of spatial relationships through constructing relationship transitional rules. A relationship transitional rule for a set of features $F$ is in the form $F : T_{pre} \Rightarrow T_{post}(n/N)$, where $T_{pre}$ and $T_{post}$ are the relationship types before and after the transition, $n$ is the
number of this transition that has occurred to $F$, and $N$ is the total number of different transitions that have occurred to $F$ when the $T_{pre}$ relationship holds for $F$. These rules can facilitate predictions on features’ future spatio-temporal behaviors. For instance, based on the rule $\{A, B, C, D\}:Star \Rightarrow Clique(9/10)$, if we observe that $A$, $B$, $C$, and $D$ form a Star SOAP in a certain period, we can predict that they are very likely to form a Clique after some time.

4.2.5 Inferences Based on Multiple Sets of Features

So far, we have only considered inferences that involve one set of features. We are currently investigating potential approaches that involve multiple sets of features. For example, after the dissipation of SOAP $(A \ B \ C)$ at time $t_i$, we observe that at time $(t_i + 1)$, objects $A$ and $E$ appear in the same neighborhood where objects $A, B$, and $C$ are located at $t_i$, we can then infer the following: objects $B$ and $C$ merged into $E$. We plan to extend this idea by constructing spatio-temporal traces, where a trace is obtained by splicing a series of temporally and spatially overlapped episodes. Since the splicing point of two intervals in a trace usually indicates a significant interaction change, traces can be used to reason about the causal relationships between different evolutionary SOAP events. However, we note that for a given situation, one can often have multiple inferences. For instance, we may also have the following inferences for the above example: (1) $B$ and $C$ dissipated at the SOAP’s dissipation; and (2) $B$ evolved into $E$ and $C$ dissipated. We are currently investigating other approaches to address this multi-inference issue. One potential solution is to take into account the interaction among different SOAPS. To do this, we need to consider multiple SOAPS in the same vicinity.
4.3 Evaluation

In this section, we present preliminary results on vortex simulation data to evaluate the proposed approaches. The vortex dataset is generated by implementing a simplistic version of the algorithm proposed by Christian [33]. The dataset consists of 1970 snapshots, with around 200 vortices in each snapshot. The number of vortices in each frame changes over time, as an existing vortex can dissipate, new vortex can be created, or two vortices can merge to form a new vortex. We apply a locally implemented algorithm [82], to detect and extract vortices from the simulation dataset. We then use an entropy-based clustering algorithm to cluster all the detected vortices into groups (or classes) [31], where each vortex is represented by an ellipse. This clustering procedure ensures that similar vortices are grouped into the same group. Finally, we label each detected vortex by the ID of the group to which it is clustered. We next extract different types of SOAPs by applying our SOAP mining algorithms. The boundary-boundary distance is used to measure the distance between two vortices, and two vortices are considered to be of interaction if they are within the distance of 10 (\(\epsilon = 10\)). A SOAP is frequent if it appears in at least 10 snapshots, i.e., \(minSupport = 10\). Finally, the parameter \(minRealization\) is set to 1.

Figure 4.5 shows a fraction of three simulation snapshots at different times. Figure 4.5(a) shows the initial configuration of the vortices with no formed SOAPs. Figure 4.5(b) shows the creation of a SOAP. This SOAP is created because the two vortices are moving towards each other and eventually their distance becomes less than the distance threshold. Figure 4.5(c) shows a very interesting result. The two vortices involved in the SOAP came closer and eventually merged into a new vortex. This event is captured by the SOAP’s dissipation (resulting from an amalgamation of two features within the SOAP). Note that SOAPs’ dissipation does not necessarily imply dissipation of features. We are
Figure 4.5: (a) Vortices at time 0 (b) SOAP Formation at time 90 (c) Vortex Merging at time 104

able to make inference on such events by identifying the changing trend of the correspond-
ing instance’s MBB.

Figure 4.6: Different Vortex SOAPs are discovered at time 991.

Figure 4.6 shows a Clique and a Sequence SOAP identified in the same snapshot taken at time 991. One can observe that different SOAP types can capture very different spatial relationships (or interactions) among vortices.
4.3.1 Inferences Based on Individual Episodes

Figures 4.7-4.8 demonstrate two opposite cases that lead to a SOAP dissipation event. The SOAP in Figure 4.7 dissipates due to a vortex amalgamation event. Whereas the SOAP in Figure 4.8 dissipates because the interaction among the involved vortices becomes weaker. We are able to make inferences on these two types of behaviors by identifying the changing trend of the SOAP’s MBB during an episode. Figure 4.7(a) shows three snapshots of a Clique SOAP instance during the episode from time 39 to 49. It also shows a snapshot of the SOAP’s neighborhood area immediately after the dissipation. One can observe that the MBB of this instance decreases as the SOAP instance evolves. Such a changing trend is successfully identified by the linear regression model (Figure 4.7(b)). Therefore, we can infer that a vortex amalgamation event may have occurred around the SOAP’s dissipation. This is exactly what happened at the SOAP’s dissipation, where three vortices merged into one new vortex. Figure 4.8(a) illustrates the case where all the involved objects are moving away from each other. The linear regression line captures this trend (Figure 4.8(b)).

Figure 4.7: The MBB of SOAP (1 15 61 81) decreases in the episode [39,49], which eventually dissipated at time 50 due to object amalgamation. (a) Snapshots of the SOAP at time 39, 44, 49, and the vortices in the vicinity of the SOAP right after its dissipation (b) The fitted linear regression line: area(MBB) vs. time
Figure 4.8: The MBB of SOAP (19 32 103 109) increases in the episode [0, 9]. (a) Snapshots of the SOAP at time 1, 5, 9, and vortices in the vicinity of the SOAP right after its dissipation (b) The fitted linear regression line: area(MBB) vs. time

4.3.2 Capturing Features’ Interaction History

As mentioned earlier, combining episodes associated with different SOAP types can be used to capture the evolving nature of interactions among different features.

Figure 4.9: Transition: Sequence ⇒ Clique ⇒ Star. The SOAP (23 39 57 58) is a Sequence in the interval [0 18], a Clique in the interval [19 34], and a Star in the interval [35 57].
Figure 4.9 demonstrates how the interaction among four vortices evolves over time and is captured by combining multiple episodes formed at different times. The interaction history of the four vortices is modeled by three SOAP types, corresponding to three episodes. In the first episode, which starts at time 0 and ends at time 18, the four vortices form a Sequence SOAP (Figure 4.9(a)). As the interaction evolves, the four vortices start to form a Clique, which ends at time 34. Following the dissipation of this interaction, the four vortices start to move away from each other. This evolving behavior is captured by the transition of Clique→Star. Through identifying the changing trend of the Star SOAP’s MBB, which increases over time, we can infer that the four vortices will keep moving apart and may cease to interact after some time. This inference is verified by observing the simulation data.

Figure 4.10: Transition: Sequence ⇒ Star ⇒ Clique. The SOAP (19 27 58 36) is a Sequence in the interval [212 243], a Star in the interval [244 287], and a Clique in [288 313]. It eventually dissipated due to the amalgamation of vortices 19 and 58 at time 314.

Figure 4.10 models the other type of evolving relationship among features. The four vortices begin their interaction as a Sequence and changes into a Star after some time. After staying as a Star SOAP for some time, they then evolve into a Clique. The transition
Figure 4.11: Transition: Clique $\Rightarrow$ Sequence. The SOAP (7 8 42 99) is a Clique in the interval [0 17], then evolves into a Sequence at time 18 and eventually dissipates at time 49.

*Star$\Rightarrow$Clique* implies that the interaction among the four vortices increases, which may lead to vortex amalgamation. Such an implication is further strengthened since we see a decreasing trend of the SOAP’s MBB during the time interval [288 313].

Figure 4.12: Transition: Star $\Rightarrow$ Sequence. The SOAP (205 211 218 230) forms a Star at time 1170, evolves into a Sequence at time 1180, and finally dissipates at time 1202.

Figure 4.11 and 4.12 illustrate the other two transitions that can occur to features: *Clique $\Rightarrow$Sequence* and *Star $\Rightarrow$Sequence*. We observe that the transition from *Clique
to Sequence often indicates that the involved features will very likely cease to interact after some time. However, such an observation does not hold for the Star $\Rightarrow$ Sequence transition.

4.4 Related Work

Our research on spatio-temporal association patterns shares some of the objectives with approaches for spatial reasoning. Bailey-kellogg and Zhao [11] propose a methodology for reasoning about such problems called qualitative spatial reasoning (QSR). Their work is methodology driven and mainly focuses on conceptual topics such as data representations and manipulations. They also discuss the use of different spatial primitives to model objects of different shapes and spatial relationships among objects. Our framework is an efficient realization of their conceptual methodology for scientific data. In addition we also support the discovery of spatio-temporal episode patterns that is not explicitly considered in their work. Fernyhough et al. implemented techniques to detect events by identifying frequently occurring spatial relationships [36, 57]. However, they only consider pair-wise relationships, thereby missing interactions among more than two features.

4.5 Conclusions

In this Chapter, we propose spatio-temporal association based approaches to make inferences on important events and reason about features’ interacting behaviors. We present preliminary results on vortex simulation data to validate the efficacy of these approaches.
CHAPTER 5

CASE STUDY: PROTEIN FOLDING TRAJECTORY ANALYSIS

5.1 Introduction

The three dimensional (3D) native structures of proteins have important implications in proteomics. Understanding the structure of a protein enables us to explore the function of the protein, explain substrate and ligand binding, perform realistic drug design and potentially cure diseases caused by misfolding. The protein folding problem is therefore one of the most fundamental yet unsolved problems in computational molecular biology. One major challenge in simulating the protein folding process is its complexity. Snow et al. [162] state that performing a Molecular Dynamics (MD) simulation on a mini-protein for just 10 μs would require decades of computation time on a typical CPU. The Folding@home distributed computing project [155] recently proposed using worldwide distributed computing to tackle protein folding simulations.

With the increasing number of trajectories produced by distributed computing, there is a need to analyze, understand, and understand the available data. Previously, researchers have examined several summary statistics (e.g. radius of gyration, root mean square deviation (RMSD)) for this purpose. Although summary statistics are commonly used for
comparison, they can only capture a biased and limited global property of the conformation. Recently, Russel et al. [138] suggested using geometric spanners for mapping a simulation to a more discrete combinatorial representation. They consider using geometric spanners to discover the proximity between different segments of a protein across a range of scales, and track the changes of such proximity over time.

To overcome the difficulties in managing and analyzing the large amount of simulation data, Berrar et al. [19] proposed designing a data warehouse system. They embed their warehouse in a grid environment to enable the sharing of the actual simulation data. They also propose implementing a set of data mining algorithms to facilitate commonly needed data analysis tasks.

In this chapter, we propose a method to analyze folding trajectories of the mini protein BBA5 produced by the Folding@home project. We utilize the spatio-temporal data mining framework that we have developed and described earlier in Chapter 3. In Section 3.5.1, we have applied this framework to 8732 proteins taken from the Protein Data Bank to identify structural fingerprints for different classes of proteins. Each protein is associated with a set of objects that are extracted from their contact maps. To effectively capture spatial relationships among objects, we define Spatial Object Association Patterns (SOAP). Furthermore, by associating SOAPs with proteins in different protein classes, we establish the connections between different types of SOAPs and protein classes.

It is apparent that protein folding trajectory data have both spatial and temporal components. Each protein in a MD simulation consists of a number of residues spatially located in the 3D space that move over time. Each frame of the trajectory can be represented as a contact map in 2D, capturing the pair-wise 3D distance of residues. Similar to the case study described in 3.5.1, we extract non-local bit-patterns from these contact maps. We then
use an entropy-based clustering algorithm to cluster such bit-patterns into groups. These bit-patterns are further associated to form spatial object association patterns (SOAPs). By the use of SOAPs, we effectively represent and analyze folding trajectories produced by MD simulations. A major advantage of this representation is its appropriateness for cross-comparison across different simulations. Key benefits of our framework include:

- **Effective, informative and scalable representation of folding simulations:** We represent each frame by a set of SOAPs, where each SOAP in turn characterizes the spatial relationship (or interactions in the folding case) among multiple bit-patterns. SOAPs are not only easily obtainable but also, as we will show, are able to capture landmarks along a folding trajectory.

- **Cross-analysis of trajectories to reveal a consensus partial pathway:** By representing each frame as a set of SOAPs, one easily carry out analysis across different trajectories. Such analysis includes detecting critical events and identifying consensus partial folding pathways across trajectories.

### 5.2 Analysis of Protein Folding Trajectories

#### 5.2.1 Protein Folding Trajectories

Advances in high-performance computing technologies and molecular dynamics have led to successful simulations of folding dynamics for (small) proteins at atomistic level [119]. Such simulations result in a large number of *folding trajectories*, each of which consists of a series of 3D conformations of the protein under simulation. These conformations are usually sampled regularly (e.g., every 200fs) during a simulation. In this chapter, we also refer to each conformation as a *folding frame* or simply a *frame*. Furthermore, to represent
a protein conformation, we adopt one of the commonly adopted representation schemes, where a conformation is represented as a sequence of α-carbons ($C_\alpha$) located in 3D space.

![Figure 5.1](image)

Figure 5.1: Different conformations of BBA5. Points in each plot correspond to the positions of α-carbon atoms. (a) The native NMR structure of BBA5 based on data from the SCOP website. (b) The initial conformation of both folding trajectories. (c) The last conformation in the first trajectory. (d) The last conformation in the second trajectory.

As mentioned earlier, we obtained two folding trajectories of the designed mini-protein BBA5 (Protein Data Bank ID) from the Folding@home research group at Stanford University\(^{14}\). BBA5 is a 23-residue protein that folds at microsecond timescale. The native structure (or fold) of BBA5 shows a β-hairpin involving residues 1-10 and centering about residues 4-5. It also includes an α-helix involving the remaining residues 11-23. By convention, residues are numbered increasingly from the N-terminal to C-terminal of a protein. Figure 5.1(a) illustrates the native conformation of BBA5. The two folding trajectories, referred to as $T_{23}$ and $T_{24}$ respectively, are of different length. $T_{23}$ consists of a series of 192 conformations (or frames), while $T_{24}$ 150 frames\(^{15}\). Each conformation is described at atomistic level in PDB (Protein Data Bank) format.

\(^{14}\)http://folding.stanford.edu/

\(^{15}\)Please refer to [119, 163] for details on the simulation model employed to produce such trajectories.
5.2.2 Comparing Conformations of BBA5 Across Trajectories

Although both trajectories start from the same extended conformation as shown in Figure 5.1(b), when we examine the visualized frames, they seem to identify two very different folding processes. Figures 5.1(c) and (d) illustrate the last frame in $T_{23}$ and $T_{24}$ respectively. This seeming difference might be attributed to the stochastic nature of the folding simulation process [119, 163]. However, it is also desirable to characterize the similarities (or dissimilarities) across multiple trajectories.

To compare two trajectories, a key issue that must be addressed is: how can we compare two protein conformations? Several measures have been commonly used to do such comparison, including RMSD (root mean squared distance) [189], contact order [97], and native contacts [42]. However, all these measures are designed to quantify the global topology of a conformation. Furthermore, based on our empirical analysis of these measures, we notice that they are generally too coarse and thus can often be misleading. Even more important, such measures fail to identify similar local structures (or motifs) between conformations. This is especially crucial for small proteins like BBA5. As demonstrated in both experimental and theoretical studies, small proteins often fold hierarchically and begin locally [12]. For instance, it has been shown that BBA5 tends to first form secondary structures such as $\beta$-turns and $\alpha$-helix, then conform to its global topology [163]. Finally, as suggested by Pande [119], both sterics (local motifs) and global topology might play an important role in protein folding. Therefore, to compare conformations of (small) proteins, a more reasonable comparison should consider both local and global structures. Moreover, it should also take the native topology of the protein under study into account.

To meet these requirements, we propose the following approach to compare conformations of BBA5. First, we loosely partition the 23 residues of BBA5 into four fragments:
(i) \( F_1 \): N-terminal 1-10 \( \beta \)-hairpin; (ii) \( F_2 \): C-terminal 11-23 \( \alpha \)-helix fragment; (iii) \( F_3 \): the first half of \( F_1 \) and the second half of \( F_2 \); and (iv) \( F_4 \): the second half of \( F_1 \) and the first half of \( F_2 \), i.e., the middle section in the primary sequence. Second, we recognize the secondary structure propensity in each fragment. Two conformations are said to be similar if they demonstrate the same secondary structure propensity in the same fragment. For instance, the conformation pair in Figure 5.6(a) are similar as residues in \( F_1, F_2 \) and \( F_4 \) from both conformations indicate a \( \beta \)-turn like local motif. Please note that the orientation of local motifs does not affect the comparison. For instance, in Figure 5.6(d), we say the two conformations have a similar structure in \( F_1 \) fragment, even though the \( \beta \)-turn motifs have different orientations.

To realize the comparison of conformations, two more issues must still be addressed. First, how can we effectively capture and represent local motifs? Second, how can we represent the global topology of a conformation in terms of local motifs? To address the first issue, we leverage the non-local patterns in protein contact maps. For the second, we characterize the spatial arrangement among non-local patterns. We will explain such ideas in detail in Section 5.3.

### 5.2.3 Folding Trajectory Analysis: Objectives

There are two goals we would like to achieve in analyzing the folding trajectories. First, we would like to address the following folding issues for a given trajectory: (1) to detect (or even predict) significant folding events, including the formation of \( \beta \)-turns, \( \alpha \)-helices, and native-like conformations; and (2) to recognize the temporal ordering of important folding events in the trajectory. For instance, between the two secondary structures \( \alpha \)-helix and
I: Data preprocessing
1.1 Generate contact maps for every conformation in the two folding trajectories
1.2 Identify maximally connected bit-patterns in all contact maps
1.3 Cluster bit-patterns into approximately equivalent groups based on geometric properties
1.4 Re-label each bit-pattern with its corresponding cluster label

II: Discovering frequent spatio-temporal object association patterns (SOAPs)
2.1 Discover frequent (minLink=1)-SOAPs of bit-patterns in either folding trajectory

III: Folding trajectory analysis
3.1 Summarize each folding trajectory based on frequent SOAPs
3.2 Detect folding events and recognize the ordering of folding events in a trajectory
3.3 Identify the consensus partial folding pathway across trajectories

Figure 5.2: Main steps to analyze protein folding trajectories

β-hairpin in BBA5, which forms earlier? What is ordering of the two events preceding a β-hairpin formation: formation of two extended strands or formation of the turn?

If the first goal concerns individual trajectories, the second goal concerns multiple trajectories. Specifically, we would like to identify a sub-sequence of similar conformations in both trajectories. This sub-sequence of conformations is referred to as the consensus partial folding pathway. This is analogous to the Longest Common Sub-sequence (LCS) problem [62], but much more challenging in two important ways. First, we are dealing with time series of 3D protein structures. Second, we are not looking for an exact match between conformations across different trajectories. Instead, we are looking for similar conformations across trajectories.
5.3 Algorithm

In this section, we describe in detail our approach for analyzing protein folding trajectories. As shown in Figure 5.2, this analysis consists of three main phases: (I) Data preprocessing, (II) Spatio-temporal object association pattern mining, and (III) Trajectory analysis. We next discuss each phase in further details.

5.3.1 Data Preprocessing

Contact Map Generation

As in our previous studies on protein structural analysis (Section 3.5.1, we represent 3D protein conformations by contact maps. To generate contact maps, we consider the Euclidean distances between $\alpha$-carbons ($C_\alpha$) of each amino acid. Two $\alpha$-carbons are considered to be in contact if their distance is within 8.5 Å. Since contact maps are symmetric across the diagonal, we only consider the bits below the diagonal. Furthermore, we also ignore the pairs of $C_\alpha$ atoms whose distance in the primary sequence is $\leq 2$, as they are sure to be in contact. This step essentially transforms the two BBA5 trajectories into two series of contact maps, with each map of size $23 \times 23$.

Identifying Maximally Connected Bit-patterns

We apply a simple region growth algorithm to identify all maximally-connected patterns\(^{16}\) in every contact map in the two trajectories. A total of 352 maximally-connected bit-patterns were extracted from the two folding trajectories of BBA5.

As discussed earlier and also in the literature [76, 98, 174], such bit-patterns can effectively capture the secondary structures of proteins. In the context of protein folding, we have observed that they are powerful enough to represent a wide range of local structural

\(^{16}\)Please see Section 3.5.1 for definition.
motifs. We can even measure approximately the strength of secondary structure propensity in a conformation based on the bit-patterns. For instance, we have identified bit-patterns that correspond to “premature” $\alpha$-helices and native-like $\alpha$-helices respectively. Henceforth, we refer to the 3D structure formed by all the participating residues of a bit-pattern as the 3D motif of the bit-pattern.

Clustering Bit-patterns into Approximately Equivalent Groups

We apply an entropy-based clustering algorithm to group the bit-patterns into $l$ clusters, where the bit-patterns in a cluster show similar geometric properties (e.g., shape and size). The value of $l$ is determined using an entropy-based measure that quantitatively indicates the quality of a clustering result. (Please refer to Section 3.5.1 for details.) For the BBA5 folding data, the clustering step groups the 352 bit-patterns into 10 clusters (or types).

Intuitively, the 3D motifs of the bit-patterns in a cluster will also have similar 3D geometric properties. This is verified based on our analysis on the BBA5 trajectories. Figure 5.3 illustrates the representative 3D motifs for 9 of the 10 types of bit-patterns. We omit type 0, as bit-patterns of this type correspond to a wide variety of 3D motifs.

This demonstrates, to a certain extent, the advantage of using 2D contact maps to analyze 3D protein conformations. Undoubtedly, using contact maps greatly reduces the computational complexity of our algorithm. More importantly, by exploiting different features in contact maps (bit-patterns in this work), we are able to connect 2D features with features in 3D space. In our case, by identifying 10 types of bit-patterns in contact maps, we indirectly recognize 10 different 3D structural motifs in the folding conformations.
Figure 5.3: Mapping between different types of bit-patterns and 3D motifs in conformations of BBA5 folding trajectories. The bit-patterns and 3D motifs shown here were randomly selected from their respective group for illustration purpose.
Re-labeling Bit-patterns with The Corresponding Cluster Label

In this step, we re-label all the previously identified bit-patterns with their corresponding cluster label. Let $p$ be a labeled bit-pattern. It can be represented as follows: $p = (\text{trajID}, \text{frameID}, \text{listC}_\alpha, \text{label})$. Here, $\text{trajID}$ identifies a folding trajectory, and $\text{frameID}$ indicates the frame where $p$ occurs, $\text{listC}_\alpha$ consists of all participating $\alpha$-carbons of $p$, identified by their position in the primary sequence. Finally, $\text{label}$ is the cluster label of $p$. For BBA5, $\text{label} \in \{g_0, g_1, \cdots, g_9\}$, corresponding to the 10 approximately equivalent groups (or types).

5.3.2 Mining Spatio-temporal Object Association Patterns

The pre-processing steps transform a 3D protein conformation into a set of labeled 2D bit-patterns, that indirectly capture the local 3D structural characteristics of the conformation. For the two BBA5 trajectories, each conformation contains an average of 6 bit-patterns.

As BBA5 folds, the dynamics among its 23 residues will be constantly changing until it reaches an equilibrium\(^\text{17}\). This means that two residues previously in contact may become out of contact later. As a result, bit-patterns present in one conformation may be absent in the next. The evolving nature of contacting residues and in turn bit-patterns, is essentially the consequence of a variety of weak interactions among amino acids at different levels. Such weak interactions include hydrogen bonds, electrostatic interactions, van der Waal’s packing and hydrophobic interactions [65]. To capture these (potential) interactions, a simple yet effective method is to consider how close two amino acids are located from each other in 3D. We also adopt this method here. Specifically, we consider interactions between

\(^{17}\)According to the “folding funnel” theory [99, 23], an equilibrium (or the native folded conformation) has the global minimum energy. However, this might not be the case for simulated folding trajectories.
local 3D motifs captured by labeled bit-patterns. We denote such interactions as “interactions among bit-patterns”. Let $p_i$ and $p_j$ be two bit-patterns in a protein conformation, and $p_i.listC_\alpha$ and $p_j.listC_\alpha$ be the list of $\alpha$-carbons involved in $p_i$ and $p_j$, respectively. We define $p_i$ and $p_j$ as interacting bit-patterns if at least one pair of $\alpha$-carbons from $p_i.listC_\alpha$ and $p_j.listC_\alpha$, respectively, are located within a short distance $\delta$. (The value of $\delta$ should be greater than the distance that is being used to identify contacting $\alpha$-carbon atoms when generating contact maps.) In our analysis, we set $\delta = 10\,\text{Å}$. 

So far, we have discussed our approach of using bit-patterns in contact maps to characterize local 3D motifs and further represent a protein conformation during folding. We also define the notion of interacting bit-patterns in the folding context. We are ready to present our method of summarizing folding trajectories to fulfill the two objectives described in Section 5.2.3. The main idea is that we can summarize a folding trajectory by characterizing the evolutionary behavior of interactions among different types of bit-patterns and in turn, the interactions among local 3D motifs.

As discussed in Chapter 3, such interactions can be modeled and captured by discovering different types of spatial object association patterns (SOAPs). Essentially, SOAPs characterize the specific way that objects, bit-patterns in this case, are interacting with each other at a given time. Among the proposed SOAP types, for folding analysis, we empirically select ($\text{minLink} = 1$) SOAPs to model the interacting bit-patterns in the folding process. We refer readers to Section 3.3.4 for the definition of this SOAP type and its mining process.

We further restrict ourselves to focus only on SOAPs that occur frequently during the folding process ($\text{frequent SOAPs}$). Recall that a SOAP is frequent if it appears in $\geq \text{minSupp}$ frames in a trajectory. In our studies, we set $\text{minSupp} = 5$. 

125
SOAP Episodes

As mentioned earlier, small proteins like BBA5 often fold hierarchically and begin with local folded structures. As BBA5 folds, new SOAPs can be created and existing one can dissipate. To capture such evolutions, we leverage SOAP episodes, which are defined in Chapter 3.

In summary, this mining phase produces the following results: (i) A list of \( \text{minLink} = 1 \) SOAPs of bit-patterns that appeared in at least 5 conformations in each folding trajectories; and (ii) A list of episodes, ordered by beginning frame \( F_{\text{beg}} \), associated with each of these SOAPs.

5.3.3 Folding Trajectory Analysis

In this section, we describe our strategy on utilizing SOAPs to summarize a folding trajectory and address the two folding analysis issues described in Section 5.2.3.

SOAP-based Trajectory Summarization

The previous mining phase discovers a collection of frequent \( \text{minLink} = 1 \) SOAPs and the associated episodes in each trajectory. Therefore, it identifies all the conformations in the trajectories that contain at least one frequent \( \text{minLink} = 1 \) SOAPs. For instance, the last conformation in trajectory \( T_{23} \) (Figure 5.1(c)) has two SOAPs of size 2: \( (5\ 8) \) (i.e., association of a type 5 and a type 8 bit-pattern) and \( (7\ 8) \), and three SOAPs of size 1: \( (5) \), \( (7) \), and \( (8) \), while the last conformation in trajectory \( T_{24} \) has three SOAPs: \( (7\ 8) \), \( (7) \) and \( (8) \). This leads to our SOAP-based approach for folding trajectory summarization.

To summarize a folding trajectory, we perform the following three steps. First, for each conformation, we identify all the frequent SOAPs that appear in it and use these SOAPs to represent this conformation. Note that not every conformation contains frequent SOAPs,
especially when $\minSupp$ is set high. Second, for each SOAP-representable conformation, we carry out the following two tasks on its associated SOAPs. First, for each SOAP, we mark the relative location of each involved bit-pattern in the primary sequence of BBA5. This is done by identifying the segment of BBA5 where the majority of a bit-pattern’s $\alpha$-carbons are located. The segment can be one of the following as described in Section 5.2.2: $F_1$, residues 1–10; $F_2$, residues 11–23; $F_3$, residues 6-17; and $F_4$: residues 1-5 and 18-23.

Let us again take the last conformation in $T_{24}$ as an example. It can be summarized by three SOAPs: (7 8), (7) and (8). When we look at the list of $\alpha$-carbons involved in these bit-patterns, we find out that 7 is mainly located in $F_2$ and 8 in $F_1$. Therefore, we mark the three SOAPs as follows: (8.1 7.2), (7.2) and (8.1). (We re-arrange the bit-patterns in a SOAP by relative location in BBA5.) This super-imposes BBA5-specific spatial information to a SOAP. The next step is to prune away redundant SOAPs after marking each bit-pattern with its relative location in BBA5. A SOAP is redundant if it is embedded in another SOAP. For instance, in the previous example, we can prune away (8.1) and (7.2) as both are embedded in (7.2 8.1). After pruning, most conformations in such a small protein can
often be represented by a single SOAP. We can even take this summarization a step further, where we replace a bit-pattern with its corresponding 3D motif, as illustrated in Figure 5.3. For instance, SOAP (7.2 8.1) will be transformed into (β.1 α.2). We refer to such SOAPs as generalized SOAPs, and the corresponding trajectory as a generalized trajectory. Note that in generalized trajectories, multiple types of bit-patterns can be mapped into a single type of 3D motif. For instance, the α-motif corresponds to three types of bit-patterns 4, 7, and 9 (Figure 5.3). Figure 5.4 shows a segment in each summarized BBA5 folding trajectory before and after being generalized with 3D motifs.

**Detecting Folding Events and Recognizing Ordering Among Events**

Once each folding trajectory is summarized into generalized SOAPs, it is fairly straightforward to detect folding events such as the formation of α-helix or β-turn like local structures. This can be done by simply locating the frames that contain the local motif(s) of interest. We can also easily identify native-like conformations, by finding those that contain the generalized SOAP (β.1 α.2). Finally, based on the summarization, one can quickly identify the ordering of folding events in a trajectory. For instance, to check which secondary structure forms more rapidly, α-helix or β-hairpin, one can simply compare the first occurrence of these structures in the summarized trajectory (Figure 5.4(b)).

**Identifying the Consensus Partial Folding Pathway Across Trajectories**

To do this, we simply compute the longest common sub-sequence (LCS) in the two summarized trajectories. One can utilize the summarization either before the 3D motif generalization (Figure 5.4(a)) or after(Figure 5.4(b)). We use the latter in our analysis. Based on the LCS of generalized SOAPs, we construct the consensus folding pathway by
<table>
<thead>
<tr>
<th>Protein</th>
<th>PDB Identifier: BBA5; Primary sequence: 23 residues; Designed protein; Native fold: N-terminal 1-10 $\beta$ hairpin, C-terminal 11-23 $\alpha$-helix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory</td>
<td>Two trajectories: $T_{23}$ and $T_{24}$; $T_{23}$: 192 conformations; $T_{24}$: 150 conformations</td>
</tr>
<tr>
<td>Contact map</td>
<td>Based on contacts between $\alpha$-carbons. Two $\alpha$-carbons are in contact if their Euclidian distance is $\leq 8.5\text{Å}$</td>
</tr>
<tr>
<td>Bit-patterns</td>
<td>A total of 352 unique maximally connected bit-patterns were identified from all conformations; Average number of bit-patterns per conformation is 6; Bit-patterns are further classified into 10 approximately equivalent types</td>
</tr>
<tr>
<td>Interacting bit-patterns</td>
<td>If at least one pair of $\alpha$-carbons, one from each bit-pattern, is of Euclidian distance $\leq 10\text{Å}$</td>
</tr>
<tr>
<td>Frequent SOAPs</td>
<td>A SOAP is frequent if it appears in $\geq 5$ conformations; A total of 444 frequent SOAPs identified in trajectory $T_{23}$, and 258 in $T_{24}$</td>
</tr>
<tr>
<td>Consensus partial folding pathway</td>
<td>We identified a consensus partial folding pathway across the two trajectories. It is composed of 71 pairs of similar conformations, one from each trajectory</td>
</tr>
</tbody>
</table>

Table 5.1: A summary of BBA5 folding analysis.

identifying pairs of conformations, one from each trajectory, associated with those generalized SOAPs in the LCS\textsuperscript{18}. The resulting consensus pathway is a sequence of conformation pairs of similar 3D structures.

Notice here that the comparison between 3D protein conformations (as described in Section 5.2.2) is done by using bit-patterns to model local structural motifs, and associations of bit-patterns (SOAPs) to characterize the global structure. It is a hierarchical comparison, which matches the hierarchical folding process of BBA5.

\textsuperscript{18} Ambiguity might arise in this process as different conformations can be represented by the same generalized SOAPs. However, this can be easily resolved by taking temporal dimension into account.
5.4 Results

In this section, we report the results from analyzing two trajectories of the protein BBA5. We have described this protein, its trajectories, and the concepts employed in the analysis process in great details earlier. Such information is summarized in Table 5.1.

5.4.1 Detecting and Ordering Folding Events

We summarize both folding trajectories with a sequence of SOAPs as illustrated in Figure 5.4. Coincidentally, both summarized trajectories consist of 64 conformations.

Based on these summarized trajectories, we can quickly identify all the conformations where the first \( \alpha \)-helix-like or \( \beta \)-turn-like local motifs were formed. For trajectory \( T_{23} \), the first \( \alpha \)-helix-like motif was identified in frame 26, and the first \( \beta \)-turn-like local motif was formed in frame 63. For the other trajectory \( T_{24} \), the frames were 29 and 38, respectively. This agrees with the experimental result, which says that \( \alpha \)-helices generally fold more rapidly than \( \beta \)-turns.

For the two events related to \( \beta \)-turn formation, formation of two extended strands and formation of the turn, we found that for both trajectories, the formation of extended strands preceded the formation of the turn.

Also, we identify two conformations in each trajectory that show native-like structure. We do this by locating the conformations associated with the generalized SOAP \( (\beta.1 \alpha.2) \). Figure 5.5 presents the 3D structure of these native-like conformations along with the native conformation of BBA5. One can see that our SOAP-based comparison does well in identifying similar 3D conformations.
5.4.2 Consensus Partial Folding Pathway Across Trajectory

Based on the generalized trajectory summarization, we identify a consensus partial folding pathway of length 71. In other words, 71 pairs of conformations, one from each trajectory, are considered similar to each other. Furthermore, they evolve from one to the next in the same order. Figure 5.6 displays four such pairs along this consensus folding pathway. Note that by using bit-patterns, we naturally realize a rotation-invariant comparison. For instance, in Figure 5.6(d), even though the two β-turns at the N-terminal are of very different orientation, the two conformations are still identified as similar.

Currently, we rely on visual tools to justify this pathway. We did attempt to use several measurements that have been used previously to quantify the similarity between 3D protein conformations, but to no avail. These measurements include RMSD, contact order, and native contacts. If we identify the pathway based on the best match given by any of the above measurements, where two conformations are a best match if they have the lowest RMSD or have the smallest difference in contact order or native contacts, for instance, we often ended up with a very short consensus pathway (as short as 10 frames). Sometimes, the two conformations along this consensus pathway were visually dissimilar. Moreover,
different best-matched measurements rendered very different consensus pathways. We are investigating alternative methods for quantitative validation.

5.5 Conclusions and Future Work

In this chapter, we present a novel approach to analyze protein folding trajectories and a case study using the small protein BBA5. We capture a variety of local motifs in the 3D protein conformations by non-local bit-patterns identified in their contact maps. Furthermore, by modeling the interactions or spatial relationships among bit-patterns as SOAPs, we effectively characterize the evolutionary nature of the folding process. We also describe two methods to summarize folding trajectories by super-imposing BBA5-specific information and 3D local structures onto SOAPs. Utilizing the summarized trajectories, we
demonstrate that we can detect folding events and the ordering between events, and also identify a consensus folding pathway across trajectories.

We realize that protein folding is a very hard problem. Based on the results of our analysis, we are not in the position to make any general comments on the protein folding problem. However, the approach presented here is meant to be general. It is applicable to any folding trajectories.

There are several issues we plan to address in the future. First, we would like to realize an automatic mapping between bit-patterns and 3D motifs. Second, we will further analyze the consensus folding pathway and validate it through other means. Third, it is well-known that the side chains of a protein play a very crucial role in the folding process. So, we would like to investigate ways to involve side chains in our analysis. We also plan to apply such approach to trajectories of another protein obtained from the Folding@home group. Finally, we plan to study whether bit-patterns can be used to index protein folding simulation data.

**Acknowledgment:** We thank Dr. Yusu Wang for providing the folding simulation data and sharing many constructive and insightful thoughts with us during this study.
CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The thesis of this dissertation is that: spatially proximate features in many scientific disciplines inherently relate to one another. Such relationships, if properly modeled and captured, can facilitate the understanding of important scientific phenomena. Furthermore, it is important to realize general and extensible solutions that can be leveraged to address applications originating from multiple scientific disciplines.

In this dissertation, we describe a general and modularized framework to explore this thesis. We define multiple types of spatial object association patterns (SOAPs) to model diverse relationships or interactions among features (or objects) of interest. We construct SOAP episodes to capture the evolving nature of such interactions in spatio-temporal applications. We also propose multiple reasoning approaches to infer critical events (e.g., vortex amalgamation). Additionally, due to the framework’s modularized design, domain-specific knowledge can be readily orchestrated into the framework. To demonstrate the efficacy of this framework, we have applied it to applications in multiple scientific domains, namely, bioinformatics, computational molecular dynamics and computational fluid dynamics. The empirical results from such applications clearly support the thesis of this dissertation.
To summarize, we have made the following contributions:

- **Realizing a general framework:** The framework is modularized and consists of the following modules:
  
  - *Feature construction:* This module is required if there does not exist well-defined features (e.g., vortices in fluid flows) in the underlying domain. For instance, we identify a new type of features, *maximally connected bit-patterns* in protein contact maps, to identify structural fingerprints for different protein classes. In order for the constructed features to be meaningful, one often needs to take domain-specific knowledge into account. Thus, it is critical to involve domain experts in this step.
  
  - *Data preprocessing:* This module consists of the following tasks: (i) detecting and extracting features of interest in a dataset. For well-defined features, we rely on existing feature detection techniques to carry out this task; (ii) identifying important properties of a feature such as geometry and location and determining an appropriate scheme to represent such properties. This is another task that often requires domain expertise; and (iii) clustering or classifying features based on their similarity. This task can usually lead to better data manageability. It can also effectively reduce the negative impact caused by noisy data, which is common to many scientific applications.
  
  - *Spatial and spatio-temporal association pattern mining:* This module consists of algorithms that discover four types of SOAPs (Spatial Object Association Patterns), and their corresponding SOAP episodes. Namely, the four SOAP
types are: *Star*, *Clique*, *Sequence*, and $\text{minLink}=l$. Each SOAP type is employed to model and capture an important type of relationships (or interactions) observed or hypothesized among spatially proximate features. SOAP episodes on the other hand provide a simple yet effective means to characterize the evolutionary behavior of such relationships.

- *Association pattern-based reasoning*: This module contains multiple reasoning approaches that infer critical events based on SOAP or SOAP episodes. For instance, one approach examines individual episodes to reason about the occurrence of critical events such as vortex amalgamation in air flows. Another approach combines multiple episodes to infer or predict the evolving trend of the captured feature-feature interactions.

This framework distinguishes itself from related work in the following aspects:

- *Shape-based feature representation*: In contrast to most existing work in this area, features are modeled as geometric objects rather than points. In addition, domain specific properties can be readily incorporated in the proposed shape-based feature representation schemes.

- *Object-based distance measurements*: We define multiple distance measurements to take into account the shape and extent of features. This allows one to effectively capture the interactions among features.

- *Spatial object association patterns (SOAPs)*: We introduce multiple types of SOAPs to model and capture different spatial relationships or interactions amongst features. Furthermore, we have implemented scalable algorithms to discover such patterns from large datasets.
– **SOAP episodes**: We define SOAP episodes, a simple yet effective concept, to characterize the evolving nature of feature-feature interactions.

– **Association-based spatio-temporal reasoning**: We propose multiple reasoning strategies to infer or predict critical events (e.g., vortex amalgamation) based on SOAPS and SOAP episodes.

- **Applying the framework to discover knowledge in a variety of scientific applications**: We have utilized the proposed framework to address important applications arising in several scientific disciplines, namely, bioinformatics, computational molecular dynamics, and computational fluid dynamics. Below we briefly summarize each application.

  – **Protein structural analysis**: The main goal is to identify structural “fingerprints” for a given protein class (e.g., α-proteins). We employ protein contact maps to address this issue. We first identify a new type of features, *maximally connected bit-patterns*, in contact maps. We then discover different types of SOAPS of such bit-patterns. Finally, we examine these SOAPS to derive structural “fingerprints” for a protein class. Domain knowledge is effectively incorporated through the course of analysis.

  – **Protein folding trajectory analysis in bioinformatics**: There are two main goals in this study: (i) to detect different folding events (e.g., the formation of a β-turn) and identify the occurring order of such events; and (ii) to identify a consensus partial folding pathway by cross-comparing multiple folding trajectories. Similar to the previous application on protein structural analysis, we
resort to protein contact maps and bit-patterns in such maps. To reach the two goals, we utilize SOAPS of bit-patterns and their corresponding episodes.

- **Analyzing defect-defect interactions in materials in Computational Molecular Dynamics:** There are two main target issues in this application: (i) to model and capture diverse interactions among defects in materials; and (ii) to infer or predict critical events such as defect amalgamation. To address the first issue, we discover different types of SOAPS of defects and their SOAP episodes. For the second issue, we apply different reasoning strategies based on the discovered SOAPS and SOAP episodes.

- **Analyzing vortex-vortex interactions in Computational Fluid Dynamics:** The target issues of this application are similar to those of the previous one, except that vortices in fluid flows are the features of interest. The analysis procedure is in general similar to that of analyzing defect-defect interactions. However, to more effectively model the geometric properties of vortices, we use a different representation scheme, i.e., ellipse-based, as against the line-based scheme for defects. Furthermore, we employ different SOAP types to model the interactions among vortices. Specifically, we use *Star* SOAPS for vortices, whereas *Sequence* SOAPS for defects.

### 6.1 Current Limitations and Future Research Directions

In this section, we first identify the current limitations of this framework. We then highlight several closely related directions that one can pursue in the future.
6.1.1 Current Limitations of The Proposed Framework

The current framework is limited in the following aspects. First, the proposed spatial
or spatio-temporal patterns mainly capture the distance-based relationship among features.
One can overcome this by designing patterns to also capture other types of spatial relationships, for instance, topological and directional relationships. We expect that such patterns will be more powerful in helping domain experts to understand or explain important scientific phenomena. Second, the current framework only supports the discovery of frequent association patterns. In the future, to address this limitation, one can design algorithms to discover rare but important patterns. This can be very challenging when the dataset consists of a large number of maps or snapshots. Finally, we notice in our case studies that simply based on SOAPs or SOAP episodes, it is often impossible to distinguish different events. For instance, a SOAP formation can be caused by two possible events: the creation of a new feature or the bifurcation of an existing feature. Therefore, it is necessary to improve upon and extend our reasoning approaches so that one can confidently identify such events based on probability (or plausibility).

6.1.2 Future Work: Mining Spatio-temporal Data in Life Sciences

As a result of advanced data collection techniques and high storage capacity, one witnesses an increasing accumulation of huge amounts of spatial and spatio-temporal data in life sciences. Among many applications arising in this domain, we are particularly interested in the following:

- **Protein structural analysis:** Proteins are the basic building blocks and functional modules of life. For a protein to function properly, it must assume a certain 3D
structure. To date, biologists have learned the 3D structure of around 26,000 proteins. And this number is still growing.

In Chapter 3 we have demonstrated that one can exploit contact maps and bit-patterns in such maps to identify structural “fingerprints” for a class of proteins. In the future, one can study the potential of using bit-patterns to index protein structures. This will allow one to efficiently retrieve structural information given a large number of protein structures. Furthermore, we conjecture that such bit-patterns can be further explored to address much more challenging issues such as: (1) building a structure-based classification scheme that can effectively identify proteins of similar structure; (2) using such a classification scheme to potentially unify extant classification schemes such as SCOP and CATH; and (3) realizing a structure-function mapping for proteins based on this classification scheme.

- **Protein folding analysis**: In Chapter 5, we present a method to analyze protein folding trajectories of a small protein. As mentioned in the last section of that chapter, one still needs to address several important issues to make this method more objective and verifiable. Such issues include setting up a mapping between bit-patterns and 3D structural motifs, considering the side-chains in the analysis, and quantitatively validating the identified consensus folding pathway. One also needs to evaluate this methodology on more trajectories and for more proteins, once such data is available.

- **Gene regulatory network analysis**: A gene regulatory network (GRN) describes a dynamic and collaborative molecular process, in which a collection of gene segments in a cell interact with each other and other substances in the cell, thereby governing
the rates at which genes are transcribed into RNA. Each RNA transcript then functions as the template to synthesize a specific protein by the process of translation. Studies have shown that GRNs can behave differently under different cellular environments. One would like to model and predict the evolutionary behaviors of GRNs in response to different conditions. We are interested in designing spatio-temporal mining methods to address this issue.

- **Transmissible disease control:** Transmissible disease control has captured much attention recently due to a potential pandemic outbreak of bird flu. The goal is to minimize the number of infected subjects as early as possible, and in as limited a geographic region as possible. There are many factors that can contribute to this goal. One critical factor is to understand how people in the infected area move around and relate with each other over time. In other words, one needs to characterize and capture the interacting patterns among people and the evolution of such patterns in the control process. We believe that spatio-temporal data mining approaches can play an important role here.

One can also apply spatio-temporal mining approaches to address applications in other domains. For instance, spatio-temporal approaches might be useful to detect wormhole attacks in sensor networks. In a wormhole attack, the attacker builds a tunnel-like channel involving generally no more than three sensors. By doing this, the attacker can redirect packets only through these sensors, thereby severely disrupting the sensor network. Wormhole attacks essentially change the topological relationships among sensors in the network. If such changes can be captured properly and timely, one will be able to detect potential wormhole attacks.
6.2 Concluding Remarks

In this dissertation, we describe a general framework to discover spatial and spatio-temporal association patterns in scientific data. We represent features as geometric objects rather than single points. To effectively model the spatial relationships or interactions among features, we employ distance measurements that can take into account a feature’s geometric properties. We define multiple types of spatial object association patterns (SOAPs) to model different spatial relationships among features. Furthermore, we define SOAP episodes to characterize the evolving nature of feature-feature interactions. Finally, we implement multiple reasoning strategies to infer critical events that are a result of strong feature-feature interactions. To demonstrate the efficacy of this framework, we have carried out a variety of case studies in several scientific disciplines, namely, bioinformatics, computational fluid dynamics, and computational molecular dynamics. Our empirical results show that the framework can efficiently and effectively discover meaningful knowledge with respect to the underlying domain, thereby facilitating the understanding of important scientific processes. Through such applications, we also show that the proposed framework is promising towards automating or semi-automating the analysis process for scientific data.
BIBLIOGRAPHY


[38] P. F. Craigmile. Stat635 lecture notes, ohio state university.


156


