

To the Graduate Council:

I am submitting herewith a dissertation written by Evan C. Ezell entitled “Temporal Data Techniques for Analyst Sensemaking.” I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Data Science and Engineering.

Seung-Hwan Lim, Major Professor

We have read this dissertation
and recommend its acceptance:

Seung-Hwan Lim

Robert Stewart

Russell Zaretzki

Audris Mockus

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

To the Graduate Council:

I am submitting herewith a dissertation written by Evan C. Ezell entitled “Temporal Data Techniques for Analyst Sensemaking.” I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Data Science and Engineering.

Seung-Hwan Lim, Major Professor

We have read this dissertation
and recommend its acceptance:

Seung-Hwan Lim

Robert Stewart

Russell Zaretski

Audris Mockus

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Temporal Data Techniques for Analyst Sensemaking

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Evan C. Ezell

August 2025

© by Evan C. Ezell, 2025
All Rights Reserved.

To my Wife, Mom, and Dad.

Acknowledgements

I would like to thank my doctoral committee for their contributions and mentorship in preparing this work. Foremost, I would like to thank my advisor, Seung-Hwan Lim, for his tremendous help in driving the direction of my research throughout the past several years. Secondly, I would like to thank Robert Stewart, who took a chance by allowing me to work with him eight years ago. He has provided me with funding through the WSTAMP project and invaluable advice in navigating the vicissitudes and toils of graduate student life.

Throughout my life, I have been fortunate to learn from remarkable institutions, each enriched by outstanding teachers and professors. Their wisdom, guidance, and dedication have shaped me in countless ways, leaving a lasting impact on both my education and personal growth. The following teachers and professors deserve special mention for their indelible mark on my journey.

Sequoyah High School

Mr. Jeff Beinke - Computer Science

Mr. Andy Kohler - Mathematics

Maryville College

Dr. Jeff Bay - Statistics

Dr. Robert Lowe - Computer Science

Dr. Maria Siopsis - Mathematics

Dr. Jesse Smith - Mathematics

Mr. Lee Wittenberg - Computer Science

University of Tennessee

Dr. Michael Langston - Computer Science

Dr. James Plank - Computer Science

“Never had it so good.” – Pa

Abstract

Many of the systems we care about—like the economy, public health, transportation, and socioeconomics—change over time. The data that captures these changes is referred to as temporal data. The overarching goal of this work is to develop methods that make temporal data more interpretable, explorable, and actionable to analysts across many domains. Each method presented in this dissertation addresses a different modality or aspect of temporal data analysis. Together, they form a methodological toolkit that enables analysts to explore temporal patterns and extract meaningful insights from dynamic systems. Specifically, we focus on algorithmic and visualization techniques for time series and dynamic networks. We present a novel time series representation method called Shape Profiling that automatically extracts and describes time series based on intuitive shapes. The versatility of our technique allows for a wide range of query-by-content searches, such as identifying co-occurring patterns, concomitant patterns, and sequential patterns executed across various time specifications and filterings. These queries provide efficient and accurate results for practitioners that are consistent with human intuition, enabling an in-depth exploration of how shapes and trends are arranged across complex time series datasets. Our next contribution is a technique for uncovering temporal patterns in dynamic network sequences. We apply the Multidimensional Matrix Profile to snapshot graph embeddings of dynamic networks. This data structure yields a new way to study dynamic networks based on the all-pairs similarity-join problem that has shown much success in time series data mining. Finally, we present Community

Fabric, a novel visualization technique for simultaneously displaying communities and structure within dynamic networks. By hybridizing two popular network and community visualizations, Community Fabric improves upon existing state-of-the-art in several key areas. Our approach allows viewers to interpret and understand the interplay of structural change and community evolution in dynamic networks more easily. For each contribution, we apply our approaches to use cases of interest and present the results.

Table of Contents

1	Introduction	1
1.1	What is Temporal Data and Why Does it Matter?	2
1.2	Scope and Purpose	3
1.3	Practical Challenges	5
2	Shape Profiling: A Shape-Based Representation for Time Series Analysis	8
2.1	Abstract	9
2.2	Introduction	9
2.3	Background	14
2.3.1	Representation Methods	15
2.3.2	Distance Measures	23
2.3.3	Indexing	27
2.3.4	Segmentation	30
2.3.5	Classification	33
2.4	Methods	34
2.4.1	Semantic Segmentation	34
2.4.2	Taxonomic Classification of Time Series Segments	47
2.5	Data	52
2.6	Results	53
2.7	Limitations	58

2.8	Conclusion	60
2.9	Future Work	61
2.10	Author Contributions	67
3	Dynamic Network Matrix Profile: Uncovering Temporal Patterns in Dynamic Graphs	68
3.1	Abstract	69
3.2	Introduction	69
3.3	Background	71
3.4	Methods	74
3.5	Capital Bikeshare Results	80
3.5.1	Data	80
3.5.2	Model Training	81
3.5.3	Embeddings	85
3.5.4	Multidimensional Matrix Profile	87
3.6	Atlas Coffee Trade Results	98
3.6.1	Data	98
3.6.2	Model and Embeddings	98
3.6.3	Multidimensional Matrix Profile	99
3.7	Limitations	102
3.8	Conclusion	103
3.9	Future Work	104
3.10	Author Contributions	105
4	Community Fabric: Visualizing Communities and Structure in Dynamic Networks	106
4.1	Abstract	107
4.2	Introduction	107
4.3	Methods	112
4.3.1	Crossing Minimization	115

4.4	Results	122
4.4.1	Hospital Ward Interaction Network	123
4.4.2	Synthetic Dynamic Network	128
4.4.3	Atlas Economic Trade Data	130
4.5	Conclusion	131
4.6	Future Work	132
4.7	Author Contributions	133
4.8	Acknowledgements	133
	Bibliography	135
	Appendices	164
A	Dynamic Time Warping (DTW)	165
A.1	Background	165
A.2	Method	166
A.3	Variants and Optimization	169
	Vita	172

List of Tables

2.1	Summary of Shape Mining Methods	20
2.2	Archetype Shapes	50
2.3	Japan’s Gross Domestic Product (GDP) Segment Behavior Class Dynamic Time Warping (DTW) Distances	52
2.4	Symbolic Index Alphabet	66
3.1	Nearest Neighbor Weekday Confusion Matrix	95
3.2	Yearly Snapshot Clusters from Trade Network Embeddings	99
4.1	Lancichinetti-Fortunato-Radicchi Graph Configuration Parameters . .	128

List of Figures

2.1	Japan's Gross Domestic Product (GDP) Matrix Profile	39
2.2	Japan's Gross Domestic Product (GDP) Matrix Profile Arc Plot . . .	41
2.3	Japan's Gross Domestic Product (GDP) Arc Curve	41
2.4	Idealized Arc Curve (IAC)	43
2.5	Japan's Gross Domestic Product (GDP) Corrected Arc Curve (CAC)	44
2.6	Japan's Gross Domestic Product (GDP) Segmentation using Unsuper- vised Fast Low-cost Unipotent Semantic Segmentation (UFLUSS) . .	48
2.7	Template Trends	51
2.8	Japan's Gross Domestic Product (GDP) Shape Profile	51
2.9	Japan's Population Growth Shape Profile	54
2.10	Japan's Population Growth (Ages 65 and Older) Shape Profile	56
2.11	Japan's Personal Remittances Shape Profile	56
2.12	Japan's Livestock Production Shape Profile	57
2.13	Japan's Crop Population Shape Profile	57
2.14	Japan's Health Expenditure Shape Profile	59
2.15	Japan's Inflation of Consumer Prices Shape Profile	61
3.1	Dynamic Graph as a Sequence of Snapshots	74
3.2	Dynamic Graph Matrix Profile Vector	80
3.3	District of Columbia Capital Bikeshare Station Locations	82
3.4	Capital Bikeshare Station Daily Trip Volume	83
3.5	Vocabulary Size of DynNode2Vec Model on Capital Bikeshare	85

3.6	Graph Embedding Heatmap	86
3.7	3D t-SNE Plot of Graph Embedddings	88
3.8	Dynamic Network Matrix Profile	89
3.9	Distance Profile for Query	91
3.10	Snapshot Graph Metrics of Capital Bikeshare Motif	93
3.11	Query’s Median and Worst Match Snapshot Graph Metrics	94
3.12	Corrected Arc Curve of Capital Bikeshare DNMP	97
3.13	Vocabulary Size of DynNode2Vec Model on Coffee Trade	100
3.14	3D PCA Plot of Graph Embedddings on Coffee Trade	101
3.15	Corrected Arc Curve of Coffee Trade DNMP	102
4.1	Visualizing Communities in Dynamic Networks State-of-the-Art	111
4.2	Node-link diagram versus Biofabric diagram	113
4.3	Alluvial flow community diagram.	113
4.4	Community Fabric Visualization of Vehicle Trade	116
4.5	Transition Network	118
4.6	Transition Network Adjacency Matrix	121
4.7	Hospital Interaction Network Adjoined Node-Link Diagram	125
4.8	Community Fabric Visualization of Hospital Interaction Network	125
4.9	Unoptimized Community Fabric Diagram	126
4.10	Node-Link Diagram of Synthetic Dynamic Network	129
4.11	Community Fabric Visualization of Synthetic Dynamic Network	130
4.12	Community Fabric Visualization of Cotton Trade	131
4.13	Interactively Focused Community Fabric Diagram	132
A.1	Warping Path Matrix and Corresponding Temporal Alignment	170
A.2	Warping Path Constraints	171

Chapter 1

Introduction

1.1 What is Temporal Data and Why Does it Matter?

Many of the systems we care about—like the economy, public health, transportation, and socioeconomics—change over time. The data that captures these changes is referred to as *temporal data*. For example, by analyzing time series data describing a country’s socioeconomic indicators, we can gain insights into its development, stability, or vulnerability. Similarly, by studying daily network snapshots from the D.C. Capital Bikeshare system, we can uncover patterns such as recurring movement motifs, identify unusual events like system-wide disruptions, and detect structural changes in how stations are used over time. In the context of global trade, we can examine temporal data on coffee exports to identify sudden shifts in trading behavior that might signal economic instability or geopolitical events. Likewise, in healthcare settings, analyzing how individuals interact in a hospital ward over time can reveal evolving community structures that may influence the spread of infections or the effectiveness of care. These examples illustrate that temporal data can vary in form, ranging from individual scalar measurements to dynamic networks, but all share the challenge of capturing and interpreting change over time.

As sensors, systems, and digital records increasingly generate large volumes of temporal data, manually reviewing and interpreting these data points is no longer practical. Consequently, we rely on computational methods and algorithms to assist in analyzing, summarizing, and understanding complex temporal patterns. These tools enable us to detect trends, identify anomalies, and highlight significant changes—tasks that would otherwise be infeasible for human analysts.

The ability to accurately interpret temporal data carries varying levels of importance depending on the context. In global trade analysis, such as monitoring coffee exports, delays in identifying shifts or anomalies may lead to incorrect economic forecasts or missed market opportunities. In urban mobility systems like the D.C. Capital Bikeshare network, failing to detect patterns in daily station usage could result

in inefficient resource allocation or disruptions to service. In contrast, in a hospital setting, overlooking patterns of interaction between individuals can have far more serious consequences, potentially compromising patient safety or the effectiveness of infection control measures. These examples highlight that accurate and timely interpretation of temporal data is not merely a technical challenge, it is often essential for informed decision-making and responsible action.

This dissertation focuses on developing methods to support the analysis and interpretation of time-evolving data, including both time series and dynamic networks. The techniques introduced in the following chapters are designed to improve the interpretability of temporal data, uncover patterns and behaviors that would otherwise remain hidden, and facilitate decision-making in contexts where understanding how systems change over time is essential.

1.2 Scope and Purpose

Analyst sensemaking refers to the process of constructing meaningful interpretations and actionable insights from complex, uncertain, or ambiguous data. In the context of temporal analysis, sensemaking involves not only identifying patterns, changes, and anomalies in data over time, but also interpreting their significance, drawing connections across variables and contexts, and generating insights that inform real-world decision-making. Effective sensemaking is crucial in domains such as public policy, scientific research, and system monitoring, where decision-making depends on a deep understanding of dynamic behaviors.

Temporal data arises across nearly every domain where systems evolve over time. From sensor readings and economic indicators to mobility patterns and social interactions, analysts increasingly rely on temporal datasets to answer questions about behavior, change, and structure. Yet as temporal datasets grow in size and complexity, so too do the problems associated with understanding them. Analysts are often faced with the task of interpreting noisy, high-dimensional signals, identifying

patterns across time, and relating the findings to the real-world situations they are investigating, guided by their domain expertise and contextual knowledge. This dissertation addresses these challenges by introducing three distinct exploratory temporal data analysis methods, each designed to support the analyst sensemaking process.

The overarching goal of this work is to develop methods that make temporal data more interpretable, explorable, and actionable. Each method presented in this dissertation addresses a different modality or aspect of temporal data analysis. Together, they form a methodological toolkit that enables analysts to explore temporal patterns and extract meaningful insights from dynamic systems. These methods are: (1) Shape Profiling, for semantically segmenting and labeling shapes in time series; (2) Dynamic Network Matrix Profile (DNMP), for motif and anomaly discovery in dynamic networks; and (3) Community Fabric, a visualization technique for exploring the evolution of community structure in time-evolving graphs.

The first contribution, Shape Profiling (Chapter 2), introduces a hands-off technique for transforming raw time series into interpretable sequences of labeled regimes. By using a semantic segmentation algorithm and matching segments to a curated set of archetypal shapes, Shape Profiling creates a representation that is both expressive and intuitive. This approach allows analysts to recognize behavioral trends, search for patterns, and compare time series not only by numerical similarity but also by their underlying shape composition.

The second contribution, DNMP (Chapter 3), extends the Matrix Profile concept to dynamic network data. It does so by representing each graph snapshot as an embedding vector and stacking these vectors over time, effectively transforming a dynamic network into a multivariate time series. This formulation supports the application of the all-pairs similarity join to graph snapshot subsequences, enabling a wide range of data mining tasks across the temporal evolution of a network. DNMP bridges the gap between time series analysis and dynamic graph mining, unlocking new capabilities for pattern discovery in networked systems.

The third contribution, Community Fabric (Chapter 4), addresses the challenge of visualizing both community dynamics and network structure in evolving graphs. Existing techniques tend to emphasize one aspect at the expense of the other, leading to fragmented or visually overwhelming displays. Community Fabric integrates Biofabric-style structural visualizations with community alluvial diagrams, providing a cohesive view of how communities evolve over time in relation to the underlying network topology. This technique supports exploration, validation, and explanation of community behaviors and structural shifts in dynamic networks.

The three methods introduced above share a common purpose: to enhance how analysts engage with, interact with, and explore temporal data in support of deeper understanding and sensemaking. Shape Profiling, designed for time series, helps answer what types of behaviors are occurring over time. DNMP and Community Fabric, both focused on dynamic graphs, address different aspects of temporal structure: DNMP helps identify when and how often structural behaviors recur or deviate, while Community Fabric helps contextualize why those behaviors may be emerging by revealing changes in network topology and community structure. Together, these methods form a versatile toolkit for analyzing and interpreting complex time-evolving data. Each method is presented with a focus on its individual challenges, goals, and role in advancing temporal data sensemaking, collectively representing an overall advancement to the methodological landscape of exploratory temporal data analysis, with the shared goal of supporting interpretable and insightful analyst reasoning.

1.3 Practical Challenges

While the methods presented in this dissertation are designed to be broadly applicable across domains, many of the practical challenges they address were first surfaced and shaped through engagement with the World SpatioTemporal Analytics and Mapping Project (WSTAMP) [140, 112, 142], which served as a key source of inspiration.

WSTAMP provides a compelling illustration of the kinds of real-world problems analysts face when working with large-scale, heterogeneous temporal datasets.

WSTAMP is a long-running, practitioner-focused research and analytics initiative that curates, integrates, and analyzes open-source spatiotemporal data across a wide range of socioeconomic and environmental domains. Its user base—primarily applied geographers, public policy researchers, and government analysts—often confronts complex analytical questions using diverse datasets that span decades, regions, and data quality levels. These users are rarely specialists in statistics or computer science, yet they are tasked with interpreting temporal patterns, comparing structural dynamics, and making decisions based on evidence found in noisy, high-dimensional data across a broad range of socioeconomic and cultural indicators.

To support its users, WSTAMP provides a cohesive, browser-based analytical environment for discovering, analyzing, and visualizing socioeconomic spatiotemporal data with the goal of enabling users to interpret the past, operate in the present, and anticipate future developments [113]. WSTAMP strives to provide analytical and visualization capabilities that allow detailed and comprehensive data exploration while remaining accessible to analysts of varying skill levels. Its analytical capabilities encompass both standard time series analysis and advanced data mining approaches*. Altogether, WSTAMP provides a source of methods, tools, and software for working with complex, non-traditional spatiotemporal data.

The WSTAMP platform aggregates over 25,000 attributes collected from major organizations and open-source datasets such as World Bank (WB), World Health Organization (WHO), United Nations (UN), Armed Conflict Location and Event Data (ACLED), Harvard Growth Lab (ATLAS), National Aeronautics and Space Administration (NASA), National Oceanic and Atmospheric Administration (NOAA), Bureau of the Census, and over 30 other sources. Its data spans indicators of violence,

*WSTAMP’s advanced analytical techniques include Spatial Principal Component Analysis Trend Maps [141], Attribute Portfolio Distance [114, 141, 112], Find Signature Trends [113], Attribute Stability Index [115, 113], Categorize Trends [143], and Dynamic Time Warping based trend comparisons.

health, education, climate, commerce, agriculture, and more, resulting in a database with approximately 55 million time series and thousands of product trade networks.

While WSTAMP offers a vast number of time series, many of these are relatively short in duration—often consisting of just a single annual observation. At the same time, the dataset is exceptionally wide, containing tens of thousands of attributes distributed across countries, subnational regions, and data vendors. This combination of short temporal length and broad attribute coverage poses unique challenges for analysis and influenced our methods.

Through engagement with WSTAMP and its users, three analyst needs emerged:

- The need to represent and compare behavioral trends in time series data using interpretable semantically meaningful shapes.
- The need to identify motifs and anomalies in dynamic networks over specific time ranges, rather than at individual time points.
- The need to visualize evolving communities in dynamic networks while understanding the structural relationships that led to those communities.

These challenges, though highlighted by WSTAMP, are by no means unique to it. They reflect a broader class of problems encountered by analysts across many domains such as climate science, economics, health, and national security, where making sense of time-evolving data is critical for insight and action. The methods developed in this dissertation—Shape Profiling, Dynamic Network Matrix Profile (DNMP), and Community Fabric—directly address these practical needs.

Chapter 2

Shape Profiling: A Shape-Based Representation for Time Series Analysis

2.1 Abstract

We introduce *Shape Profiling*, a novel time series representation that automatically extracts and characterizes sequences using intuitive shape-based segments. Our method belongs to the class of approaches that first segment time series and then extract patterns from the resulting segments. Existing techniques in this space often rely on segmentation strategies designed for dimensionality reduction, which may obscure meaningful structure, and can require significant analyst intervention and technical expertise. To address these gaps, Shape Profiling uses semantic segmentation to preserve structural regimes within segments and automatically extracts patterns with minimal analyst involvement. We achieve this by developing Unsupervised Fast Low-cost Unipotent Semantic Segmentation (UFLUSS), a Matrix Profile-based algorithm that identifies regime changes without requiring labeled data. Each resulting segment is then classified by comparing it to a set of prototypical shapes using Dynamic Time Warping (DTW), producing a labeled Shape Profile for each time series. The result is an interpretable, low-effort representation that supports scalable analysis and pattern discovery in large-scale time series datasets.

2.2 Introduction

Time series data mining is critical across various fields, offering valuable insights into patterns and trends within complex systems. Time series data can encompass anything from collected physical measurements to logical indicators representing different systems. Researchers and analysts often focus on identifying specific trends or anomalies within particular periods of a time series. For example, in the context of a nuclear reactor, engineers might be interested in identifying one-hour periods where sensor readings differ significantly from others. In socioeconomics, a geospatial analyst may examine which decades saw a consistent upward trend in a country’s gross domestic product and whether these trends exhibit any spatial relationships across

neighboring regions. Furthermore, practitioners may investigate whether certain countries exhibited unusual trends during these periods, suggesting economic outliers in contrast to regional baselines.

Inspired by such diverse use cases and analyst needs, we introduce a novel technique called Shape Profiling, designed to extract, index, and analyze the shapes and trends of time series data. There are several existing works that aim to extract meaningful shapes or patterns from time series data [80, 162, 50, 43, 135, 3, 47, 143]. These techniques are thoroughly described in Section 2.3.1 and an overview is presented in Table 2.1. Some of these approaches do not apply segmentation [3, 47, 143] and instead require the analyst to specify the length of the pattern or query window in advance. This requirement can be difficult to tune across large, heterogeneous datasets, where appropriate pattern lengths may vary substantially. Other methods incorporate segmentation to divide the time series into more interpretable parts [80, 162, 50, 43, 135], but the segmentation strategies themselves were typically designed for dimensionality reduction rather than for identifying meaningful behavioral changes. For instance, equal-length slicing imposes fixed breakpoints at regular intervals regardless of the underlying signal [135], while Piecewise Linear Approximation (PLA) [133] incrementally fits linear segments and places a breakpoint when the residual error exceeds a threshold [43, 80]. As Gharghabi et al. [42] note, a more appropriate strategy is *semantic segmentation*, in which “we are interested in regimes that are defined by changes in the shapes of the time series subsequences, which can change without any obvious effect on the statistical properties.” Thus, semantic changes can be difficult for segmentation methods based on dimensionality reduction to capture effectively.

Shape Profiling fills these gaps by offering a fully automated framework that leverages semantic segmentation to preserve coherent temporal regimes and performs shape classification with minimal analyst intervention. Central to our method is Unsupervised Fast Low-cost Unipotent Semantic Segmentation (UFLUSS), a novel extension of the Matrix Profile-based FLUSS algorithm [42]. Unlike the original

FLUSS method, UFLUSS does not require the number of regimes to be specified, enabling fully unsupervised semantic segmentation. Thus, UFLUSS allows our method to be applied to unlabeled datasets. Once segmented, each regime is classified by comparing it to a set of synthetically generated shape exemplars using Dynamic Time Warping (DTW) [126, 152], resulting in a labeled Shape Profile that compactly captures the essential structure of each time series. These profiles support querying, retrieval, and interpretation tasks across time series datasets.

Our approach begins by taking a micro-level view of each time series through the computation of the Matrix Profile [166]. The Matrix Profile is a widely used data structure that efficiently solves the all-pairs similarity-join problem: finding, for every subsequence in a time series, its nearest neighbor according to a given distance metric (e.g. Euclidean or z-normalized Euclidean distance) over a fixed subsequence length. This powerful abstraction enables a wide range of downstream data mining tasks, including motif discovery, anomaly detection, and segmentation.

In Shape Profiling, we use the Matrix Profile solution to the all-pairs similarity-join problem for semantic segmentation by employing UFLUSS. UFLUSS builds upon FLUSS [42], a segmentation method that detects changes in subsequence similarity structure using the arc curve, which captures the recurrence of similar patterns over time. While FLUSS requires the number of segments to be specified in advance, a barrier to usability with our data, UFLUSS eliminates this requirement by greedily finding all local minima in the arc curve. This adaptation allows UFLUSS to operate in a fully unsupervised manner, detecting regime boundaries without prior knowledge of how many behaviors exist in the time series.

The output of UFLUSS is a series of non-overlapping segments, each corresponding to a coherent semantic regime. These regimes capture distinct behaviors or system states, where local subsequence behavior is similar. Semantic segmentation is the natural fit for this type of problem, since we ultimately want to extract *where* behaviors are occurring. By isolating individual regimes, analyst do not need to worry

about manually tuning specific time ranges or tuning query parameters as is the case with non-segmentation-based approaches [3, 47, 143].

While semantic segmentation helps determine *where* behaviors are occurring in the time series, the pattern extraction step identifies *what* those behaviors are. Each segmented regime is treated as a coherent unit whose shape reflects an underlying trend or system state. By classifying these segments into archetypal behavior classes, we obtain a symbolic and interpretable representation of the time series. This two-stage process, semantic segmentation followed by shape classification, transforms raw time series data into a higher-level summary that is both compact and aligned with human intuition.

To perform this classification, we use a vocabulary of ten predefined archetypal shapes, originally developed in collaboration with professional analysts to capture patterns commonly observed in temporal datasets [143]. These include simple trends such as *up*, *down*, and *constant*; cyclic patterns such as *wave*; localized anomalies such as *spike-up* and *spike-down*; and structural shifts such as *shift-up*, *shift-down*, *smile*, and *frown*. These shapes align closely with shape vocabularies used in prior shape-based time series methods [135, 43, 162, 80, 47], reflecting common temporal behaviors that recur across application domains. Additionally, our method allows for straightforward extension, enabling domain-specific applications to define and incorporate custom shape definitions as needed.

To assign shape labels, we synthetically generate a representative exemplar for each archetype—a clean, idealized time series encoding the canonical version of the behavior. These exemplars are not fixed templates; instead, they are generated dynamically by a function that adjusts each shape to match the length of the segment being classified. This strategy was originally introduced in Stewart et al. [143], where it was applied to full-length time series. In our work, we use this strategy at the segment level, allowing each regime identified by UFLUSS to be classified independently based on its local behavior.

Each regime is compared to all shape exemplars using the Dynamic Time Warping (DTW)* distance [126, 152]. DTW is well-suited for this task due to its ability to non-linearly align sequences; it can align shapes even when they are phase-shifted, stretched, or compressed in time. This makes DTW more robust than pointwise measures like Euclidean distance. The regime with the lowest DTW distance to an exemplar is labeled with the corresponding behavior class. The result is a *Shape Profile* (see Figure 2.8), a simplified yet semantically rich representation of the time series that captures its structural evolution through a sequence of labeled regimes. This compact and interpretable summary supports a range of search, retrieval, and indexing tasks in time series datasets.

In summary, Shape Profiling offers a novel approach to pattern mining in large time series datasets. We improve upon existing methods by using semantic segmentation to identify meaningful regime boundaries and by requiring minimal analyst intervention to produce results. Furthermore, Shape Profiles can be precomputed and stored, enabling rapid search and retrieval across datasets. The result is an interpretable, low-effort representation that supports scalable analysis and pattern discovery in large-scale time series datasets.

The remainder of the chapter is organized as follows. Section 2.3 reviews key time series background material, including Dynamic Time Warping (DTW), Matrix Profile, time series representations, shape-based pattern mining, and information retrieval. Section 2.4 formally details our proposed method and outlines the algorithmic steps involved in Shape Profiling. Section 2.6 presents experimental results using select indicators from an open-source geospatial time series dataset. Section 2.7 provides the limitations of our approach. Section 2.8 summarizes the main findings, and Section 2.9 outlines directions for future research.

*A full discussion of DTW is provided in Appendix A.

2.3 Background

The gathering of measurements over time is ubiquitous across almost every field of the sciences. The purpose of collecting these measurements is to use them to extract, understand, and reason with patterns, shapes, and correlations across the data. Time series analysis, originating in the early 20th century, has been fundamental in this endeavor. British statistician George Udny Yule laid the groundwork with his seminal 1927 paper [171], introducing methods to analyze temporal data. Around the same period, Russian economist Eugen Slutsky applied time series analysis to economic data [136], demonstrating how random events could produce cyclical patterns observed in economies.

Humans have a natural propensity to see complex patterns within time series; however, their scope of vision is limited to a relatively small set of data compared to that of an advanced computer. Even though computers solve the problem of combing through massive amounts of data, it is still a complex and challenging task to develop algorithms that can suss out meaningful patterns and descriptions of the data. Beginning in the mid-20th century, organizations like the National Bureau of Economic Research (NBER) started leveraging computers for time series analysis of economic indicators [94], marking a significant advancement in the field. For the last several decades, the time series community has continued to utilize computers to enhance human interaction with temporal data. These advancements have enabled researchers to analyze and dissect larger datasets, providing insights that were previously unattainable.

[37] provide an excellent survey covering some of the most important topics of modern time series data mining including representation techniques, distance measures, and indexing methods. This paper serves as a strong background for our work, as each of these three major topics is highly relevant to Shape Profiling. While the extensive field of time series analysis includes many other important topics such

as forecasting and prediction, they are less pertinent to our Shape Profiling technique and are not covered here.

2.3.1 Representation Methods

Representation methods aim to represent the fundamental shape characteristics of a time series [37]. It is typically desired that the representation maintain critical information, while reducing or keeping the same dimensionality from the original data size. This is because efficient retrieval is needed only data subsequences that fulfill domain-specific criteria, rather than for arbitrary points. Consequently, it is neither necessary nor feasible to store the entire dataset in a way that allows immediate access to every part at all times. Instead, it is sufficient to preserve key characteristics of the data and maintain those representations, rather than the complete sequences [133]. This leads to an important question: what information should be kept? The answer can be dependent on the domain and data task at hand. Some of the most basic time series representations include sampling points or strategically choosing certain points, reducing storage while trying to maximize the ability to reconstruct the original sequence or desired sequence information. There are various strategies for retaining points, including *Landmark* in [111], *Important Point* in [119], and *Perceptually Important Point* in [41]. If one wants to find discords within their data they would likely keep different information than if they want to find motifs. However, more recently, domain agnostic representations like Matrix Profile [166] have proved the possibility of approaches that are useful across many tasks and domains. Despite this progress, these newer domain agnostic representations are still the minority of techniques currently published in the literature.

Traditionally, time series representation strategies have been broken down into three categories, namely non-data adaptive, data adaptive, and model based [37, 82]. Non-data adaptive approaches do not consider specific characteristics of the data, applying the same transformation to all datasets. Discrete Fourier Transform (DFT)

[38] and Discrete Wavelet Transform (DWT) [24] were some of the earliest approaches in this category. These spectral techniques decompose the time series into components (e.g. sine and cosine components in the case of DFT) without adjusting for specific features of the data they are being performed on. Non-data adaptive approaches can often struggle to retain specific local information due to their global focus causing localized events or variations in frequency or amplitude to get smoothed out.

Data adaptive approaches adjust their method or parameters based on the structure and characteristics of the specific time series being analyzed. Approaches like Singular Value Decomposition (SVD) [161] inherently account for local variations within the data falling into this category. Additionally, non-data adaptive approaches can often be trivially transformed into data adaptive approaches by adding a selection step. This can be seen in [144] where they use a data-sensitive approach to prune maxima from the wavelet transform representation.

In contrast, model based representations assume that the time series data was a result of some underlying generative model. By fitting models like Markov Chains (MCs) [130], Hidden Markov Models (HMMs) [105], or Auto-Regressive Moving Average (ARMA) models [61], these approaches aim to represent the time series by the parameters of the fitted model. Model based representations can be particularly useful for capturing the stochastic nature of time series data and provide a robust framework for prediction tasks when forecasting is the goal.

Moving on from the above three-class ontology, there is also a class of representations that first segment the time series and then represent those segments by some feature or aspect of the segment. These representations could fit into any of the above three categories depending on the chosen segmentation algorithm and representation of each segment. These representations are most similar to the Shape Profile approach presented here.

Examples of this class of representations include the Piecewise Aggregate Approximation (PAA) [65, 168]. In PAA, the time series is divided into equal-length segments, and the mean of each segment is calculated to represent each segment.

The segment length is chosen by the user based on their specific data and application domain, thus putting this approach into the non-data adaptive category since the segment length remains the same across all time series in a dataset. An update to this approach called the Adaptive Piecewise Constant Approximation (APCA) [23] allows for varying length segments and puts it into the data-adaptive category of techniques. Both PAA and APCA tend to smooth high-variability time series but can also be misleading when there are significant outliers causing acute average segment values. Other approaches such as the Piecewise Linear Approximation (PLA) [133] retain more information by approximating segments as real-valued functions. Most commonly, this approach uses first-degree polynomials, or lines, calculated using linear regression or linear interpolation to represent each segment[†]. PLA approaches can be considered model-based approaches since they fit a model to each extracted segment within the time series.

In addition, there are approaches that use symbols as a representation for each segment. Symbolic representations have the advantage that they are discrete, whereas real-valued representations suffer from the inability to use certain algorithms and data structures coming from the text search domain (e.g. hashing, Markov models, suffix trees, etc.). Perhaps the most popular symbolic approach found in the literature is the Symbolic Aggregate approXimation (SAX) family of algorithms [82, 134, 15, 16]. SAX is based on the ideas behind PAA, first using PAA as its initial dimensionality reduction strategy. From here, SAX uses the property that normalized time-series values come from a Gaussian distribution by breaking the average values into a user-selected number of bins. Each bin is mapped to a letter, and each segment’s average value is represented by its corresponding bin’s letter. Thus, the time series can be represented by a word consisting of letters in the alphabet whose cardinality is the number of bins chosen. These word representations can be indexed and used to answer time series queries with both exact and approximate solutions. Moreover, additional

[†]A comprehensive study of PLA segmentation strategies can be found in [66].

enhancements have been introduced into SAX to optimize indexability in iSAX [134], iSAX2 [15], and iSAX2+ [16], which are discussed further in Section 2.3.3.

We mention symbolic approaches because our Shape Profiling representation can be trivially transformed into a symbolic representation opening the door for us to utilize text indexing strategies. We further discuss the possibility of a Shape Profile symbolic representation in Section 2.9 and various indexing strategies in Section 2.3.3. A comprehensive overview of traditional time series symbolic representations is provided in [31] and an evolution of modern symbolic data series indexes is provided in [103].

More similar to our work are the class of time series representations that take a shape-based view of the time series. For example, Agrawal et al. [3] presented a query language called Shape Description Language (SDL) that allows for the retrieval of patterns in time series histories. SDL employs a user-defined alphabet to describe point-to-point changes using lower and upper bounds and value constraints. This alphabet can be used to build more complex patterns and shapes, similar to the exemplar trends generated by Stewart et al. [143]. However, constructing robust patterns with SDL can be challenging due to the fine granularity required in the alphabet description. Manually creating specific patterns that strike the right balance of information in large datasets—avoiding too few or too many results—can be cumbersome and difficult to achieve. To make the alphabet description a useful query tool for complex datasets, extensive hyperparameter optimization might be necessary to select appropriate thresholds and constraint values.

Furthermore, Gregory [46] studied the creation of user-defined shapes for identifying, comparing, and ranking time series. As a part of this work, the authors examined eight primary user-defined shapes: *lines*, *spikes*, *sinks*, *rises*, *drops*, *plateaus*, *valleys*, and *gaps* [47]. These are similar to the general shapes in [143]; however, each shape and its ranking are defined using different techniques, making the interpretation of results more complex since one must understand each shape’s specific criteria separately.

There is also a more specific class of representations that first segment each time series and then take a shape view of the individual segments. Our Shape Profiling approach belongs to this more specialized class of methods, as discussed below. Table [2.1](#) provides a summary of existing shape-based methods, distinguishing between those that rely on segmentation and those that operate directly on the full time series or a selected range of the time series.

Method	Segmentation Strategy	Pattern Extraction Strategy
<i>Segmentation-Based Methods</i>		
Shape Profiling	Matrix Profile-based semantic segmentation using Unsupervised Fast Low-cost Unipotent Semantic Segmentation (UFLUSS).	DTW used to compare segments to synthetic shape templates for classification.
Multiple Abstraction Level Mining (MALM) [80]	PLA with linear regression on each segment.	Regression coefficients, residual statistics, and MSE are extracted and clustered into rise, fall, flat with subcategories.
Xie and Yan [162]	Stable local extrema selected using local neighborhood.	Convex/concave patterns defined between extrema to support shape comparison.
Derivative Segment Approximation (DSA) [50]	Slope-based segmentation using first derivative and sliding window.	Segments mapped to angular values from their mean slopes.
Time Series Shape (TSS) [43]	PLA-based; segments form overlapping intervals.	Angles classify intervals into 25 primitive shapes; 5 additional features are extracted.
Shirato et al. [135]	Equal-length segmentation.	Triangle area determines shape: Up, Down, Constancy, Peak, or Trough.
<i>Non-Segmentation-Based Shape Methods</i>		
Shape Description Language (SDL) [3]	N/A	Symbolic alphabet defines shape primitives via constraints; pattern matching is sensitive to alphabet granularity.
Gregory and Shneiderman [47]	N/A	Predefined shapes (e.g., lines, spikes, valleys) identified and ranked based on various rules.
Categorize Trends [143]	N/A	DTW used to compare raw time series to synthetic shape templates for classification.

Table 2.1: Summary of segmentation-based and non-segmentation-based shape mining methods for time series analysis.

One of the earliest techniques in this category is Multiple Abstraction Level Mining (MALM) [80]. MALM segments a time series using Piecewise Linear Approximation (PLA) [133] mentioned previously. After segmentation, each segment is approximated by a linear regression model. Features are then extracted from the regression, such as the regression coefficients, the mean squared error between the original subsequence and the regression curve, and various statistics derived from the residuals. These features are used as input to standard clustering algorithms to group and label time series segments. MALM introduces the idea of a vocabulary of patterns, including labels like *rise*, *flat*, and *fall*, with further subcategories such as *steep* and *slow*. Although the authors do not explicitly describe the classification process, it is implied that segments are classified based on the extracted regression features. Importantly, MALM develops the concept of querying across multiple abstraction levels: raw time series (time series level), extracted regression features (feature level), and classified pattern labels (semantic level) providing a flexible framework for domain experts looking for dynamic control of their similarity search.

Xie and Yan [162] segment time series based on stable extremes, that is, minima and maxima. Stable extremes are determined by finding local minima and maxima that have adjacent neighbors that are not local maxima or minima themselves, respectively. The stable extremes are then used as candidate points upon which primitive convex and concave patterns are defined. These patterns ultimately determine where a time series is segmented. A convex pattern can be defined as a sequence of stable extremes in which the start and end points are a stable minimum, there is at least one stable maximum between the start and end points, and there is no stable minimum below the start or end points on either side of every stable maximum. The inverse of these conditions are used to define a concave pattern. Time series segments with their corresponding patterns can then be used for further analysis and comparison of time series within a dataset.

In Gullo et al. [50], the authors present an approach called Derivative Segment Approximation (DSA) that follows segmentation by shape-based feature extraction

to represent time series. DSA first transforms the original time series values using the first derivative estimate of each value. The transformed time series is segmented with a sliding window approach capturing contiguous points with close slope. The series is broken on a point when the slope of the preceding segment exceeds a threshold. This threshold can be defined using a *dataset oriented* standard deviation of all time series, a *series oriented* standard deviation of individual time series, or a *segment level* standard deviation. The authors used the latter threshold, which segmented the values based on more localized patterns in the data compared to the other thresholds. Following segmentation, DSA uses the arctangent of the mean segment values to map each segment to an angular value. The segment mapping of angular values is used as the final representation of the time series.

In Gmati et al. [43], the authors introduce a time series representation and abstraction framework called Time Series Shape (TSS). Like MALM, TSS segments time series using Piecewise Linear Approximation (PLA) [133]. Every two adjacent segments are then combined into an interval, with overlapping intervals that share a segment. Each time series is thus represented by the collection of primitive shape classifications and features extracted from its intervals. Primitive shapes are determined for each interval based on the angles of its two segments relative to the horizontal axis. These angles define the classification of the interval into one of five primary shape categories: *Increase*, *Decrease*, *Pick Up*, *Pick Down*, and *Stable*. Each primary category can be further subdivided based on specific angle values, resulting in a total of 25 possible primitive shapes. In addition to shape classification, five shape features are extracted for comparison. Two features, specific to *Pick Up* and *Pick Down*, measure the symmetry of amplitude and length between the two composing segments. The remaining three features, applicable to all shapes, capture the overall amplitude, length, and mean of the interval. Finally, the authors define a similarity measure over these shape features to enable retrieval of time series with similar structural patterns.

In Shirato et al. [135], the authors introduce the concept of *episodes*, which are essentially multivariate time series where multiple attributes or features vary over time. In WSTAMP [142], a similar idea is referred to as a *basket* of attributes. These multivariate series can be decomposed into collections of univariate patterns extracted from individual features. Although Shirato et al. [135] does not perform semantic segmentation, its method is closely related to others in this category and is thus included here. In this work, segments are defined simply as equal-length intervals chosen by the user. Segments are then classified into one of five shapes: *Up Trend*, *Peak*, *Constancy*, *Trough*, or *Down Trend*. Segment classification is achieved through a downsampling technique that selects the point maximizing the triangle area between an interval’s start and end points [139]. Features of this triangle, such as the area, range, and whether the middle point is a maximum or minimum point are used to determine classification based on a decision tree with feature thresholds. Beyond detecting patterns within episodes, the study emphasizes post-analysis by investigating transitions between patterns and the co-occurrence of patterns within an episode. While this style of post-processing is not entirely novel—having been previously explored in WSTAMP through both the *Find Signature Trends* tool [112] and *Categorize Trends* tool [143] in the context of socioeconomic data—most prior work has primarily focused on pattern detection and representation rather than post-analysis.

2.3.2 Distance Measures

While representations allow us to succinctly capture important characteristics of a time series, for most data mining tasks we also need a way to compare individual time series. Distance measures address this need by quantifying the differences between time series data, enabling computers to directly assess the similarity between sequences. As mentioned previously, humans are exceptionally skilled at observing patterns and individual characteristics of time series, simultaneously accounting for

amplitude, scaling, temporal warping, noise, and outliers. Likewise, a good distance measure should also abstract these complexities and recognize perceptually similar time series in a manner consistent with human intuition [37].

Distance measures are a fundamental concept in the mining of time series data, and research in this domain has proliferated over the past few decades. Distance measures have been extensively evaluated to understand which measures provide superior performance and accuracy [157, 132, 45, 52]. Most recently, [52] compared 56 distance measures from highly cited time series research papers against the well-known University of California at Riverside (UCR) time series data suite [30]. Distance measures are crucial for time series classification, clustering, anomaly detection, and feature engineering. Significantly, distance similarity plays an important role in our Shape Profiling technique allowing us to compare time series segments to synthetically generated sequences, resulting in a classification for each segment.

Distance measures can be broken down into four primary categories: shape-based, edit-based, feature-based, and structure-based distances [37]. Other typologies have been suggested [157, 93]; however, Esling’s classification appears to be the most comprehensive when augmented with an additional combined distance measure category proposed by [52] to accommodate modern hybrid techniques.

Shape-based distance measures compare the overall form of time series by evaluating differences between raw data points. This evaluation can involve directly matching time series points in lock-step fashion or allowing an elastic mapping between points, such as one-to-many or one-to-none correspondences. The most prominent examples of shape-based distance measures are Euclidean distance and Dynamic Time Warping, a lock-step approach and an elastic approach, respectively. Euclidean distance and other Lp-norms [168] are widely used because of their simplicity and low computational cost. The issue with these distances is the lack of robustness to noise, outliers, time shifting, value amplification, and other properties that foolproof a distance measure against common time series variances. Elastic approaches such as Dynamic Time Warping (DTW) handle local distortions by

allowing the time axis to *warp*. In this context, *warping* is a sophisticated term describing how certain segments or points of one time series can be aligned to different segments of another time series during comparison. This gives DTW more robustness to the common aforementioned time series variances.

In this paper, we use the DTW distance measure in classifying time series segments. This classification strategy was first developed by us and others in [143] for classifying whole time series based on general trends. DTW, a dynamic programming algorithm, was first used by [126] and [152] for speech recognition and was only later known as DTW in the data mining community. DTW for time series analysis was popularized in the seminal work [14] and research on the topic has since exploded. It has been used in many domains, and variants and optimizations of the original algorithm are profuse. For a comprehensive overview of the many refinements to the original DTW framework, see [1].

While DTW can be computed traditionally in $O(n^2)$ time complexity, most commonly used optimizations have come in the form of temporal constraints that restrict the warping path yielding an $O(n)$ time complexity without sacrificing result quality. These temporal constraints include the Sakoe-Chiba band [125], Itakura parallelogram [56], Keogh’s upper and lower envelopes [64], and learned windows using machine learning [17, 169, 173]. Another approach to restrict warping path is by controlling the length of the overall path [174]. While these warping path constraints inherently reduce the computational complexity, another benefit is that they conciliate pathological warping paths. A pathological warping path presents itself when a relatively small number of features of one time series is mapped onto a large number of features of the other time series, producing an unintended correspondence [63]. Other DTW optimizations use implementation strategies like FastDTW, which leverages a multilevel approach by starting with a coarse resolution warping path that is recursively refined to finer resolutions [127]. In addition to optimizations, DTW variants have been developed for domain specific purposes or to strive for general improvement over the original DTW algorithm. Although some

of these variants have been shown to outperform the original DTW algorithm on a case-by-case basis, the original DTW algorithm with a Sakoe-Chiba warping band constraint does not perform statistically worse than other techniques on a wide variety of datasets [45].

Edit-based distance measures try to transform one time series into another using operations such as insertion, deletion, and substitution. These operations are counted producing a distance between two series. This approach is analogous to text-based edit distances, such as the Levenshtein distance commonly used for spell-checking and other text applications. The advantage of using edit-based distances is that they can compensate for outliers and noisy data due to the ability to delete and substitute for certain points. The deletion operation also provides time invariance when sequences are out of phase. [29] defines a real-valued variant of the longest common subsequence (LCSS) algorithm to match common points between time series. Likewise, the Edit Distance on Real sequence (EDR) [26] is an edit distance defined for real values. EDR penalizes large gaps in the edit, and further iterations attempt to combine shape-based Lp-norms with edit distance [25]. Several other edit-based distance measures exist that attempt to constrain the edit or penalize edit operations in different ways [88, 96, 27, 138].

Feature-based distances involve extracting a set of descriptive characteristics or statistics from each time series, such as mean, variance, autocorrelation, or frequency components [106, 154, 145]. These measures are a natural extension of representing time series using the same feature by computing a distance between two time series features. For example, a distance measure is defined using the Discrete Fourier Transform (DFT) [38] and the Discrete Wavelet Transform (DWT) [24]. By encapsulating essential aspects of the data into feature vectors, standard vector distances can be computed between time series in a reduced dimensional space.

Structure-based similarity compares higher-level structures within time series focusing more on global features. This can include both model-based and compression-based approaches. Model-based approaches fit a model to each time series and

compare the model parameters. The underlying model can be assumed to be an autoregressive moving average (ARMA) [61], autoregressive integrated moving average (ARIMA) process [116], a Markov Chain [130], or a Hidden Markov Model [137], for example. Structure-based similarity also includes distance measures using the idea of shared knowledge such as Kolmogorov complexity. Compression-based approaches join time series and compute the overall compression ratio to be used as the distance measure. Examples of this approach are the compression-based dissimilarity measure (CDM) [67] and its predecessor, the normalized compression distance [81].

Combined distance measures are a convex combination of multiple distance measures [52]. By leveraging more than one approach, these hybrid measures capture different aspects of the data and can outperform single measures by balancing their respective strengths and weaknesses. However, they do introduce additional complexity by necessitating hyperparameter tuning, typically through cross-validation. [52] classify 14 out of 56 covered distance measures into the combined distance measure category. Many of these methods incorporate some form of Dynamic Time Warping (DTW) in conjunction with other techniques. Combining DTW with derivative information has been shown to significantly enhance performance compared to using derivative or DTW measures alone [52]. Notable examples include the combined derivative with DTW measure [44] and the more recent combined derivative DTW with Longest Common Subsequence (LCSS) measure [51].

2.3.3 Indexing

The overarching goal of time series indexing is to build structures that minimize computational costs and storage overhead while enabling rapid retrieval of relevant data through queries. By creating an efficient index, users can locate specific time series or subsequences quickly, rather than resorting to expensive sequential scans

over massive datasets. This can dramatically improve the ability to analyze larger amounts of time series data leading to better research outcomes.

Over the past two decades, numerous indexing methods for data series—and time series in particular—have been developed to provide fast and effective similarity searches [104, 102, 58]. These methods are typically optimized for two fundamental query types: k -nearest neighbor (k -NN) searches, which retrieve the k most similar time series to a given query; and range searches (also known as radius queries), which find all time series within a specified distance of the query. Such similarity searches are nontrivial for traditional database management systems (DBMSs) because their architectures are not built to handle sequence-based search operations natively [104].

There is a large overlap between time series indexing and general multidimensional indexing, as raw time series and most time series representations can be treated as multidimensional vectors. Traditionally, these multidimensional vectors have been indexed using a spatial access method (SAM) data structure. SAMs are used extensively for geographic information systems, graphics processing, and more general multimedia indexing. [128] provides a comprehensive overview of these types of SAM structures.

The challenge of using SAMs with raw time series is the explosion of dimensionality [22], whereas a graphics application typically has three or fewer dimensions. We can reduce the burden of using SAMs with time series by first applying dimensionality reduction to the data. Many of these dimensionality reduction techniques were initially developed as representation strategies, such as the ones discussed above (see Section 2.3.1). This common strategy of first using a reduced dimension representation and then applying an off-the-shelf spatial access method (SAMs) to index the representation was first developed in [2] and later generalized in [39] as the General Multimedia Indexing (GEMINI) framework. GEMINI uses a two-stage “filter-and-refine” approach in which (1) a SAM-based index on reduced-dimension vectors generates candidate matches quickly, and (2) a linear scan on the raw time series data confirms true matches among those candidates. According to [68], over

two-thirds of surveyed time series indexing methods up until 2002 had utilized the GEMINI framework with varying ways of reducing dimensions.

Although GEMINI has proved effective in many scenarios, multidimensional vectors with only 16 to 20 dimensions can still lead to excessive sequential scans when using traditional SAMs [54]. As a result, researchers have continued to develop more specialized representations and indexing schemes tailored to the specific nuances of time-series data. Among the proposed techniques, the SAX lineage of solutions stands out as the current state-of-the-art [104]. The Symbolic Aggregate approXimation (SAX) [82] introduced a way to discretize time series after applying Piecewise Aggregate Approximation (PAA) [65]. SAX maps segments of a time series to symbolic bins using their average values, creating binary represented vector approximations that facilitate bounded distance computations. The bounded distance computations can be used to optimize queries using a file-based vector approximation scheme [160] resulting in much faster queries since block-level chunking and precomputed lookup tables can be used.

Building on SAX, indexable SAX (iSAX) [134] enhances the indexing process by taking a multidimensional approach, similar to a wavelet representation. iSAX allows varying cardinalities among the symbolic bins and defines a comparison mechanism for time series with different bin resolutions. This ultimately enables a group of time series with the same approximate representation to be split at any point by using a higher dimensionality. This flexibility encourages a more balanced data distribution and avoids the performance pitfalls that occur when too many similar series are lumped together. Consequently, this approach has emerged as a popular choice for indexing large-scale time series data and has given rise to numerous derivative methods that provide additional capabilities.

Because the index can take a while to build on a large dataset, a second iteration of iSAX, called iSAX2, was created to provide a bulk loading strategy for the iSAX index [15, 16]. The concept of bulk loading draws on well-researched methods developed for standard database structures (e.g., B-Trees). The Adaptive Data Series (ADS) index

allows querying before the index has been fully built [176, 178, 177]. Furthermore, several methods have exploited different types of parallelism to improve performance [108, 110, 109, 164, 78, 165]. The ULtra compact Index for variable-length Similarity SEarch (ULISSE) allows variable-length queries with a SAX-based index [83, 84]. The Compact and Contiguous Sequence Infrastructure (Coconut) index improves iSAX by making vector approximations sortable, allowing bottom-up index creation, leading to a more balanced structure [72, 73, 74]. See [104] for a complete discussion of these indexes.

SAX has paved the way for the crossover between time series specific indexing to be applied in unexplored domains that require search across high-dimensional vectors [104]. As a result, time-series indexing and more general high-dimensional indexing techniques are starting to be seen as converging research areas. [158] provide an excellent survey of both graph-based and tree-based indexes for high-dimensional vector searches which includes the SAX based tree indexes discussed above.

2.3.4 Segmentation

In the literature, the term *time series segmentation* is used in varying contexts due to its broad utility in addressing different analytical tasks in time series data. This versatility has led to segmentation being applied in a range of scenarios, from event detection, anomaly identification, activity recognition, and system monitoring. Time series segmentation can be broken down into three categories: change point detection, boundary detection, and state detection [156]. Change point detection emphasizes the identification of real statistical shifts such as changes in mean, variance, or spectral density [42]. Boundary detection focuses on identifying regime shifts without labeling the internal states of those regimes, whereas state detection assigns a state label to each time point within a segment and is concerned with creating internally consistent regimes [42], which has also been called *semantic segmentation* by others [172]. These distinctions, however, are not always rigid—the lines between them can often blur, as

elements of each method tend to overlap. For example, even though the perspective of boundary detection and state detection is different, both approaches can be used to achieve *semantic segmentation*. In our approach, we emphasize *semantic segmentation*, aiming to segment the time series into meaningful internally consistent regimes followed by separate classification into a particular state.

Recent advancements in time series segmentation have led to the development of more adaptive and efficient algorithms, with particular focus on methods that reduce parameter input. The majority of these algorithms are suited only for batch use with few being able to handle streaming data. [156] classify current unsupervised time-series semantic segmentation methods into five categories: entropy-based, statistical, representation-learning, compression, and profile-based techniques. The statistical class is the oldest segmentation approach and uses Markov models and semi-Markov models to infer latent states in time series [36, 170, 59]. Compression-based methods such as AutoPlait [89] and StreamScope [62] also use HMM, but pair it with minimum description length (MDL) to summarize and compress data. Both IGTS [124] and ESPRESSO [32] use an entropy-based information gain technique. ESPRESSO is a hybrid approach that additionally integrates profile-based pruning of the boundary search space. Profile-based segmentation uses a special profile data structure (e.g. Matrix Profile, Neighbor Profile, and Classification Score Profile) that captures both the local and global characteristics of a time series [166, 53, 129]. Representation learning approaches such as Time2State [155] and E2USD [75] are among some of the newest techniques.

Among these classes, profile-based techniques like FLUSS and ClaSP stand out. Their computational efficiency, domain agnosticism, and ability to preserve shape information make them particularly suitable for our needs. Profile-based segmentation is a method used to divide time series data into distinct segments by utilizing a profile curve, such as the Matrix Profile [166], neighbor profile [53], or classification score profile [129]. These profiles are generated by computing similarity measures between neighbor subsequences of the time series and identifying areas

where these subsequences exhibit distinct differences, which are indicative of segment boundaries. FLUSS (Fast Low-cost Unipotent Semantic Segmentation) uses the Matrix Profile to detect regime changes by searching for regions where subsequences have few similarity connections with other segments, which typically indicates a boundary [42]. This approach is efficient because Matrix Profile leverages the Fast Fourier Transform (FFT) algorithm to quickly compute the similarity between all subsequences, also known as the all-pairs similarity-join problem. We further discuss Matrix Profile in Section 2.4.1. ClaSP (Classification Score Profile) generates a classification score profile by evaluating potential splits in the time series, where each split is treated as a classification task [129]. The profile curve is created from the classification scores, and the local maxima on this curve are used as boundary points to identify significant regime changes.

We present a modification of FLUSS used in Shape Profiling, designed to eliminate the need for specifying the number of regimes’ parameter, requiring only the subsequence length hyperparameter. As noted by Esling and Agon [37], the need to fine-tune parameters in time series systems remains a significant challenge. While ClaSP does avoid the need to specify the number of regimes, its classification-based approach introduces higher computational complexity due to the recursive process of splitting and refining segmentation boundaries. By modifying FLUSS, we retain its computational efficiency while removing the need to specify the number of regimes, enabling profile-based segmentation to be applied directly to unlabeled data. For a complete overview of current unsupervised time series segmentation techniques see the excellent review by Wang et al. [156].

It is also worth noting that other segmentation strategies arose from the necessity to segment time series for a specific representation [133, 65, 82]. Even though these segmentation techniques are not generally signaling semantics [156], there is a large overlap in implementation with boundary and state detection techniques. In representation approaches, poorly chosen boundaries risk obscuring essential features of the original data; whereas in semantic approaches, inadequate boundaries can yield

inconsistent regimes. Common algorithms for both types of approaches include *sliding window*, *top-down*, and *bottom-up*. *Sliding window* approaches extend a segment until it violates a specified error threshold [133]; *top-down* approaches recursively partition the data until meeting a termination condition [80, 124]; and *bottom-up* techniques start with small segments and iteratively merge them while desired properties hold [69]. Keogh et al. [66] provides a comprehensive study of representation segmentation implementation strategies.

2.3.5 Classification

After semantic segmentation, our Shape Profiling approach classifies individual time series segments into their best-fit shape. This method builds on the approach introduced by us and others in [143], who were among the first to apply this type of analysis to geospatial time series data. This paper defines 10 behavior classes, or shapes, that capture common patterns observed in real-world economic data by geospatial analysts. Because our dataset and use case are similar, we adopt the same 10 behavior classes. As mentioned previously, it is possible to define new behavior classes for specific domains and problems. Each behavior class is underlaid by a function that generates a representative shape for any specified length. These synthetically generated trends are then used to classify individual segments of time series data using Dynamic Time Warping (DTW) (see Section 2.3.2). Notably, while our approach follows that of [143], we focus on classifying segments rather than entire series, thereby capturing more nuanced and localized behaviors within the data. Segmentation enhances the classification performance because full time series are less likely to align exactly with synthetic patterns, whereas segments more effectively reflect the internal semantics of time series.

Similar techniques of comparing real time series to template or synthetic trends using DTW have been applied to other application domains. For instance, Kim et al. [70] and Tsinaslanidis and Guijarro [148] utilized DTW to analyze stock price

trends, while Guan et al. [49] and Mondal and Jeganathan [92] employed it to classify agricultural land use.

2.4 Methods

In this section we discuss our approach to extracting symbolic indexes from time series using semantic segmentation and shape classification of individual time series segments called Shape Profiling (SP). We describe all necessary related work and mathematical definitions that our SP method is based upon. We first discuss our semantic segmentation algorithm which is an unparameterized version of the Matrix Profile based time series segmentation algorithm called Fast Low-cost Unipotent Semantic Segmentation (FLUSS) presented in the Matrix Profile VIII paper [42]. Next, we discuss the process of categorizing individual segments into archetype shapes, produced by comparing time series using the dynamic time warping (DTW) similarity measure to synthetically generated patterns of interest (Section 2.4.2). This approach to classifying geospatial time series was originally presented by ourselves and others [143].

2.4.1 Semantic Segmentation

This section describes the semantic segmentation algorithm we use to segment time series for shape profiling. Our segmentation approach uses a version of the FLUSS algorithm that was adapted by us to be unsupervised. This was necessary since our dataset did not consist of any labels or intuitive values to use for FLUSS’s *number of regimes* parameter. The FLUSS algorithm is a profile-based segmentation technique (see Section 2.3.4) using the Matrix Profile. Therefore, we will first describe Matrix Profile and move on to describing our modification of the FLUSS algorithm called Unsupervised Fast Low-cost Unipotent Semantic Segmentation (UFLUSS).

Matrix Profile

Matrix profile is a revolutionary time series technique first published in 2016 [166] that made great progress in solving the *all-pairs similarity-join* problem. The all-pairs similarity-join problem for time series is that given a series of data points, find the nearest neighbor for all subsequences of some size. The solution to this time series problem can lead to other important data mining findings such as discovering regimes, motifs, discords, segmentations, and other impactful attributes.

Until the Matrix Profile approach was published, there was no efficient way of computing a solution to the similarity join problem. This made the problem intractable for large time series. Matrix profile utilizes the Fast Fourier Transform (FFT) algorithm and Euclidean distance to efficiently calculate similarity joins in $O(n \log n)$ time by exploiting the overlap of subsequences for each dot product calculation.

We begin by denoting the formal Matrix Profile definitions that will be used later.

Definition 2.1. A time series T is a sequence of real-valued numbers t_i :

$$T = t_1, t_2, \dots, t_i, \dots, t_n$$

where $1 \leq i \leq n$ and n is the length of T .

Definition 2.2. A subsequence $T_{i,m}$ of T is a continuous subset of the values from T of length m starting from position i . $T_{i,m} = t_i, t_{i+1}, \dots, t_{i+m-1}$, where $1 \leq i \leq n - m + 1$. We can also use range notation to define a subsequence. That is, $T_{i:j} = t_i, t_{i+1}, \dots, t_j$ where, $i \leq j \leq n$.

To compute the Matrix Profile, we need to compute the distance between each subsequence to all other subsequences within the time series. A subsequence's distance to all other subsequences can be stored as an ordered vector. This ordered vector of distances is known as a distance profile.

Definition 2.3. Given a subsequence query, a distance profile D is a vector of the Euclidean distances between the query and each z-normalized subsequence in an all-subsequences set (see Definition 2.4).

Definition 2.4. An all-subsequences set A of a time series T is an ordered set of all possible subsequences of T obtained by sliding a window of length m across T :

$$A = \{T_{1,m}, T_{2,m}, \dots, T_{n-m+1,m}\}$$

where m is a user-defined subsequence length. We use $A[i]$ to denote $T_{i,m}$.

The subsequence length m , also known as the window size, is the only parameter that must be defined for the Matrix Profile algorithm. According to the authors, the choice of m can be determined by the desired length of pattern to search for. For example, data collected from weather station sensors with a resolution of one minute, $m = 60$ would be a reasonable value since it would represent one hour. In our use case, our choice of m was informed based on the visual inspection of the quality of our downstream segmentation and shape classification results.

A traditional similarity join problem would join two time series together, hence the name *similarity join*, to find the nearest neighbors between two different time series. In this case, the similarity join function can be defined as follows:

Definition 2.5. 1NN-join function: given two all-subsequences sets A and B and two subsequences $A[i]$ and $B[j]$, a 1NN-join function $\theta_{1NN}(A[i], B[j])$ is a Boolean function which returns "true" only if $B[j]$ is the nearest neighbor of $A[i]$ in the set B .

Likewise, the similarity join set is defined below.

Definition 2.6. Similarity join set: given all-subsequences sets A and B , a similarity join set J_{AB} of A and B is a set containing pairs of each subsequence in A with its nearest neighbor in B : $J_{AB} = \{\langle A[i], B[j] \rangle \mid \theta_{1NN}(A[i], B[j])\}$. We denote this formally as $J_{AB} = A \bowtie_{\theta_{1NN}} B$.

Matrix profile is named as such because if we were to keep the distance profile vector D of all subsequence queries, we would have a matrix of distances. However, we actually only keep the nearest neighbor distances which gives us sufficient information with much less storage.

Definition 2.7. A Matrix Profile P_{AB} is a vector of the Euclidean distances between each pair in J_{AB} .

In the case of SP, we are only concerned with the *self-similarity join*. A self-similarity join joins its own subsequence set to itself.

Definition 2.8. A self-similarity join set J_{AA} is a result of a similarity join of the set A with itself. We denote this formally as $J_{AA} = A \bowtie_{\theta_{1NN}} A$. To prevent *trivial matches*, where a subsequence's nearest neighbor contains itself, an exclusion region is enforced such that the distance between $T_{i,m}$ and any subsequence beginning in the range $[i - m/2 : i + m/2]$ is assumed to be *infinity*. The Matrix Profile of a self-similarity join set is just P_{AA} since it is a vector of the Euclidean distances between each pair in J_{AA} .

It should be noted that the exclusion region excludes more indexes from matching than just the same subsequence match. This is because it is also likely that the distance measure will be near zero for nearby subsequences also resulting in a trivial match. To update the 1NN-join function, $\theta_{1NN}(A[i], A[j])$ is “false” when $i - m/2 \leq j \leq i + m/2$ ($A[j]$ is a trivially matched pair).

While the Matrix Profile P_{AB} tells us the Euclidean distances of the nearest neighbor pair, we need another data structure to keep track of where the nearest neighbor is. This is called the Matrix Profile index and is defined as follows:

Definition 2.9. A Matrix Profile index I_{AB} of a similarity join set J_{AB} is a vector of integers where $I_{AB}[i] = j$ if $\langle A[i], B[j] \rangle \in J_{AB}$. A Matrix Profile index I_{AA} of a self-similarity join set J_{AA} is a vector of integers where $I_{AA}[i] = j$ if $\langle A[i], A[j] \rangle \in J_{AA}$.

The nearest neighbor does not have to be symmetric (i.e. i 's nearest neighbor may be j , but j 's nearest neighbor may be k). An example of a time series and its corresponding Matrix Profile is shown in Figure 2.1.

Unsupervised Fast Low-cost Unipotent Semantic Segmentation (UFLUSS)

In this section, we move on to describing our time series segmentation technique. The definitions in this section will build upon the the previous section's definitions (Section 2.4.1).

Time series semantic segmentation aims to partition a times series into individual regimes each containing specific behaviors or conditions. Our technique is built on the Matrix Profile based algorithm called Fast Low-cost Unipotent Semantic Segmentation (FLUSS). Using the Matrix Profile for semantic segmentation was first mentioned in the original Matrix Profile paper [166] and FLUSS was formally defined later in the Matrix Profile VIII paper [42]. An advantage to using a Matrix Profile based semantic segmentation algorithm like FLUSS is its domain agnosticism, whereas various other segmentation approaches are heavily domain dependent. This will allow our shape profile technique to be more applicable to other domains beyond socioeconomic data.

Ultimately the purpose of a time series record T is to measure the phenomenon of a system through some kind of physical or indirect measurement. A system is a physical or logical process containing one or more discrete states. The goal of the segmentation task is to find the boundaries separating these discrete states in a system. For example, one such task might be trying to determine whether someone is running or walking based on physical measurements from a wearable accelerometer. The boundary would be the transition from a running to walking state.

Definition 2.10. Segmentation task: Given a time series T that monitors some aspect of a system, find the set of boundaries B between changes of state.

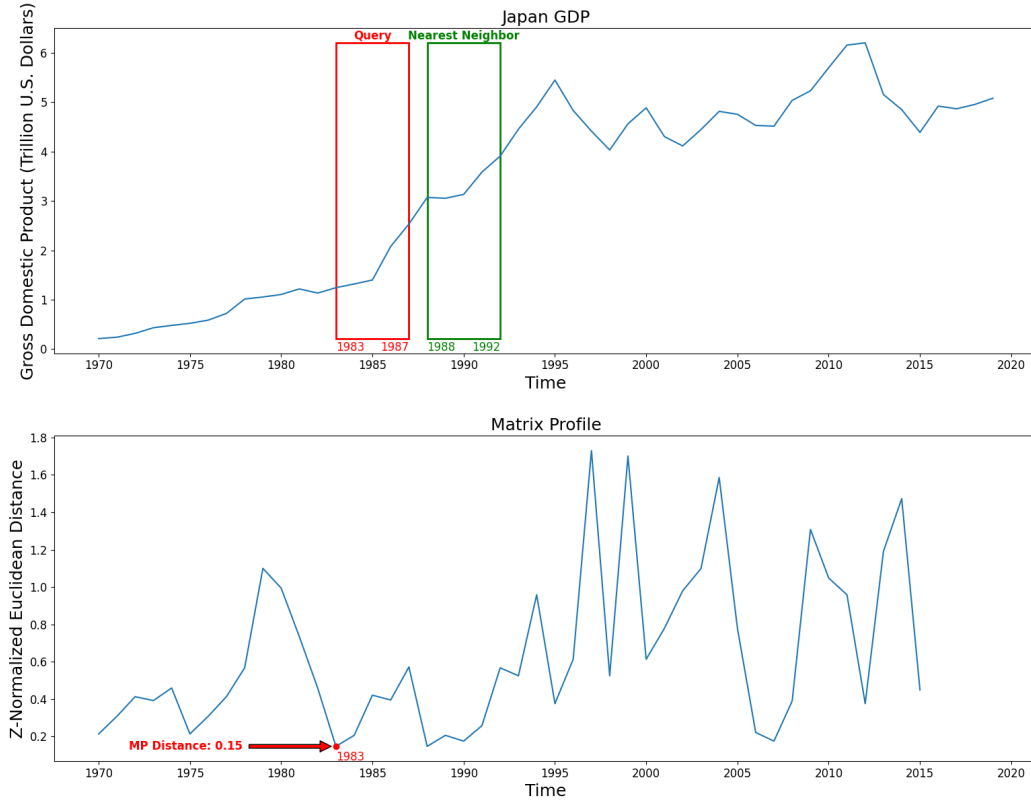


Figure 2.1: Japan's Gross Domestic Product (GDP) Matrix Profile: An example trend and its corresponding Matrix Profile. The top plot shows Japan's gross domestic product time series while the bottom plot shows the corresponding Matrix Profile for the self-similarity join. In the top plot, the red box highlights the query subsequence while the green box highlights the nearest neighbor match to the query. The corresponding Matrix Profile distance between this query and its nearest neighbor match is shown in the bottom plot. Intuitively, both the query and the nearest neighbor match have a similar trend. Each are trending slightly up in the beginning and then trend more aggressively up in the latter part of their subsequence.

Definition 2.11. A boundaries set B is an ordered set of zero or more indexes existing in T . That is, $B = \{b_1, b_2, \dots, b_k\}$, where k is the number of boundaries and $1 \leq b_i \leq n - m + 1$.

FLUSS tries to solve the segmentation task by using the aforementioned Matrix Profile as the foundation of its algorithm. The crucial insight of FLUSS is that we expect a state (i.e. a *regime* or a *segment*) to have many nearest neighbors within the same state, while we expect different states to have few nearest neighbors between them. This intuitive idea allows us to use the nearest neighbor information stored in the Matrix Profile index I_{AA} to find segments within the time series. Visually, we can imagine each subsequence’s nearest neighbor as an arc from the starting index of the query to the starting index of the nearest neighbor subsequence. Figure 2.2 shows an arc plot of all the arcs for the Japan GDP example used in the previous figures. A Matrix Profile Arc Plot is a good way to inspect local structure of time series and see where clusters of nearest neighbor matches are occurring.

While the MP Arc Plot provides a visual of the MP index, under the hood FLUSS builds an Arc Curve (AC) to find continuous subsequences of the time series that have spatially arranged nearest neighbors. The AC is just a count of the number of arcs that cross a given index (as shown by the Arc Plot in Figure 2.2). An example AC is presented in Figure 2.3 and is the corresponding AC of the Japan GDP example in previous figures.

Definition 2.12. The Arc Curve (AC) for a time series T is a vector of natural number with the same length as its Matrix Profile index I_{AA} . The i^{th} index in the AC specifies how many nearest neighbor arcs from the Matrix Profile index I_{AA} spatially cross over location i .

As mentioned previously, a regime should have many local nearest neighbors. This property will yield a higher AC value across the regime. Consequently, the local minima of the AC represent points where adjacent indexes have few nearest neighbors spanning that location. These local minima provide an ideal segmentation boundary,

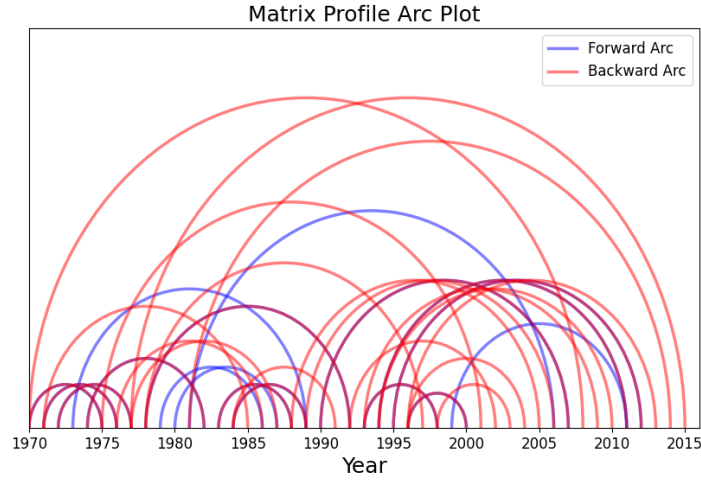


Figure 2.2: Japan's Gross Domestic Product (GDP) Matrix Profile Arc Plot: This Matrix Profile Arc Plot visualizes an arc from the starting index of the query to the starting index of the nearest neighbor for all queries. Blue arcs show forward arcs, meaning that the nearest neighbor occurred after the query. Red arcs show backward arcs, meaning that the nearest neighbor occurred before the query. The dark red arcs are when both a forward and backward arc was present on the same indices, signifying a symmetric match between subsequences. Taller arcs indicate nearest neighbors that occurred further away from the query, whereas shorter arcs indicate a nearest neighbor that occurred closer to the query.

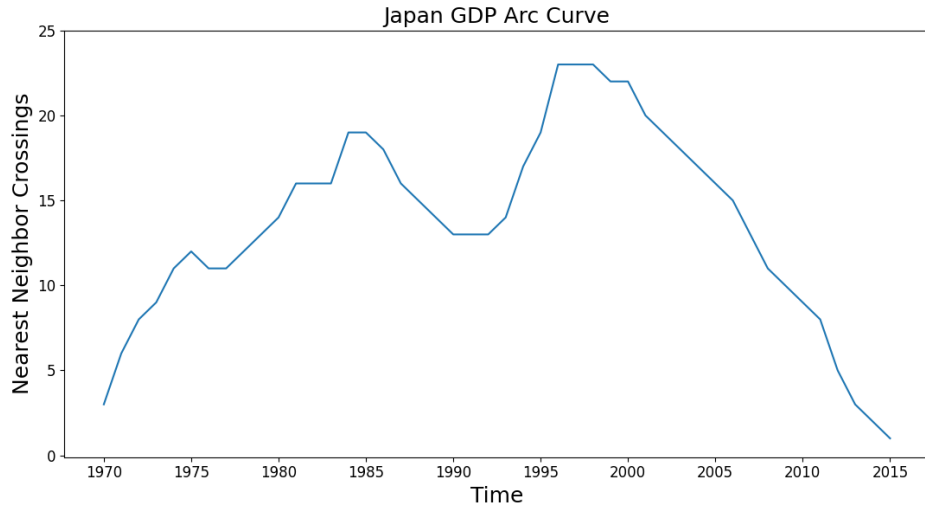


Figure 2.3: Japan's Gross Domestic Product (GDP) Arc Curve: This is the corresponding arc curve for the Matrix Profile presented in Figure 2.1.

separating regimes throughout the time series. The AC can be efficiently calculated in linear time using a cumulative sum [42].

The AC can be used to directly find the boundaries that will segment the time series. However, there is one issue with this approach. Looking at the AC in Figure 2.3, we notice that it appears to be similar to an inverted parabolic curve. From a probabilistic standpoint, this is because there is more likelihood of crossings in the middle of the AC than toward the periphery. Due to this property, computing local minima on the raw AC could produce biased boundaries. FLUSS adjusts for this bias by computing a Corrected Arc Curve (CAC) that normalizes the AC by an idealized arc curve. The Idealized Arc Curve (IAC) is the theoretical curve given that all possible subsequences in the self-similarity join set J_{AA} have a random probability of being selected as the nearest neighbor to any query. The theoretical IAC happens to follow a beta distribution with $\alpha = 2$ and $\beta = 2$ [42]. This can be seen empirically in Figure 2.4 which shows the theoretical IAC along with an AC generated from random simulation. FLUSS defines the CAC as follows:

Definition 2.13. Corrected Arc Curve:

$$CAC_i = \begin{cases} \min\left(\frac{AC_i}{IAC_i}, 1\right) & \text{if } m < i < n - m + 1 \\ 1 & \text{otherwise} \end{cases}$$

Notice that the CAC also sets the first and last m elements to one. This is done to prevent boundary points from being too close to the extremities of the time series. Following along with the Japan GDP example presented in this section, Figure 2.5 shows the CAC for this time series.

Furthermore, the paper provides the algorithm used to extract the regimes using the CAC local minima called the Regime Extracting Algorithm (REA). REA contains a parameter defining the number of regimes to find. However, in our case we need the extraction of segments to be unsupervised. We do not know how many regimes will be present in a given time series or have any prior training data to learn a value

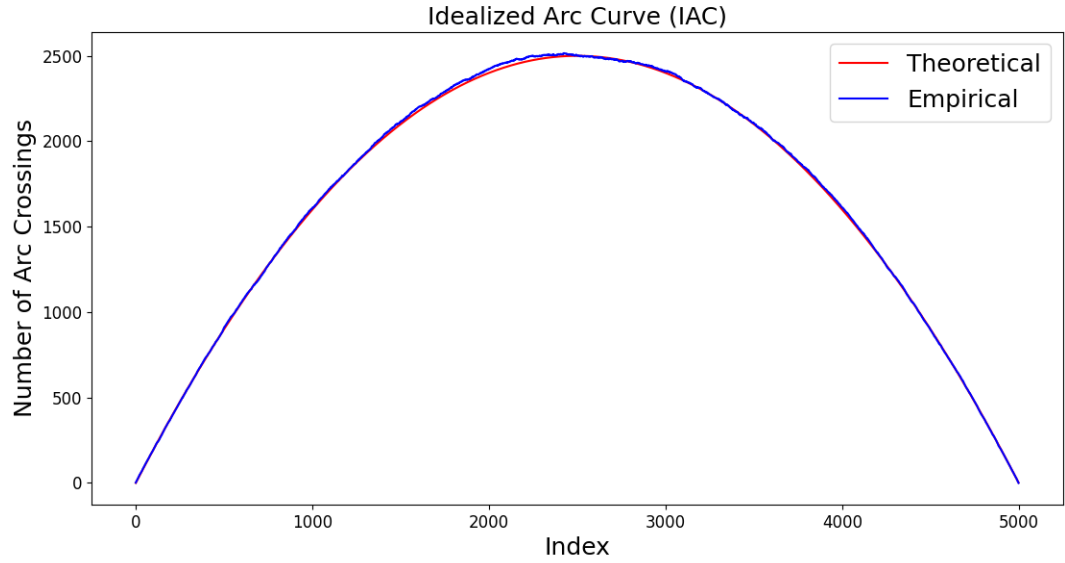


Figure 2.4: Idealized Arc Curve (IAC): The IAC is the theoretical arc curve given a time series that has no localized structure. The IAC (red) follows a beta distribution with $\alpha = 2$ and $\beta = 2$. The empirical arc curve (blue) was generated through simulation where all possible subsequences have a random probability of being selected as the nearest neighbor to any query. The empirical curve is the median curve of ten simulations.

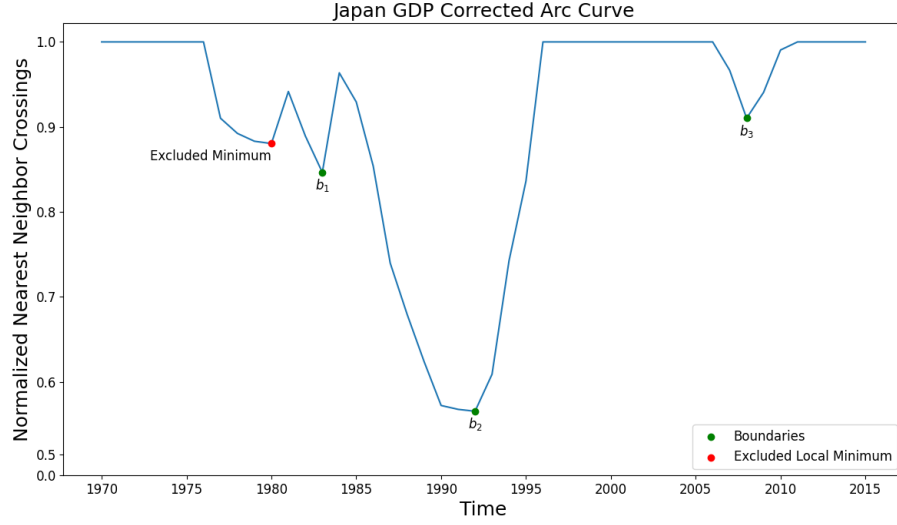


Figure 2.5: Japan’s Gross Domestic Product (GDP) Corrected Arc Curve (CAC): The CAC is the normalized arc curve that takes into account edge-effect bias. The boundaries extracted using UFLUSS are shown in green. An excluded local minimum is shown in red. This minimum was excluded because it is inside the exclusion zone of the boundary b_1 .

for the number of regimes. Therefore, instead of finding a specified number of local minima, we find local minima until no more local minima exist. We call our approach Unsupervised Fast Low-cost Unipotent Semantic Segmentation, or UFLUSS. Like FLUSS, we use an exclusion zone to prevent boundaries that are too close together. In the original FLUSS paper [42] an exclusion zone of five times the window size is used. The paper states that the exclusion zone *surrounds* the starting index of the query. However, the provided REA pseudocode is too ambiguous to determine exactly how the exclusion zone surrounds the query. Even though no explicit REA source code is provided with the paper, an open source library called STUMPY that aims to accurately and efficiently implement MP algorithms implements REA [77]. STUMPY’s implementation of REA uses an exclusion zone that is the window size multiplied by an exclusion factor (set to five by default) and sets this exclusion zone on both the right and left of the query subsequence’s starting index. In UFLUSS, we

use this interpretation of the exclusion zone. It is stated in the MP paper that an exclusion factor of five was chosen based on the assumption that regimes are likely to have many repetitions. While this worked well for their data, we have found an exclusion factor of five to exclude too many local minima in our dataset. This is partly due to the fact that our time series are not necessarily expected to contain repetition. Another reason for over exclusion is that our dataset contains series with significantly shorter length than the authors' dataset. After empirically evaluating UFLUSS results on our data, we chose to use an exclusion zone of twice the window size. It should be noted that for other domains the exclusion zone might need to be updated given the characteristics of the data. We present the UFLUSS algorithm below. The first listing, Algorithm 1, shows how to extract the local minima given a CAC. The second listing, Algorithm 2, shows the full UFLUSS algorithm using the local minima helper function from Algorithm 1 to find segments from the CAC.

Algorithm 1 Extract Local Minima from Corrected Arc Curve

Require: CAC - list of normalized arc crossing values

Ensure: $minima$ - list of indices of local minima

```

1:  $minima \leftarrow$  empty list
2: for  $i = 1$  to  $\text{len}(CAC) - 2$  do
3:   if  $CAC[i - 1] == CAC[i]$  then
4:     continue
5:   end if
6:   if  $CAC[i - 1] > CAC[i] < CAC[i + 1]$  then
7:     append  $i$  to  $minima$ 
8:   end if
9: end for
10: return  $minima$ 

```

Algorithm 1 goes through the CAC and finds indexes that have adjacent indexes with a higher arc count considering each of these a local minimum. This algorithm returns a list of all the local minima found.

Algorithm 2 presents the entire UFLUSS algorithm. UFLUSS takes in a CAC, computes the minima using the local minimum extraction routine presented in Algorithm 1, and computes the time series boundaries from the minima. UFLUSS

Algorithm 2 Unsupervised Fast Low-cost Unipotent Semantic Segmentation (UFLUSS)

Require: CAC - list of normalized arc crossing values

Require: $window_length$ - window size

Require: $exclusion_factor$ - exclusion zone multiplier

Ensure: $segments$ - list of (start, end) tuples of non-overlapping segments

```
1:  $minima \leftarrow local\_minima(CAC)$ 
2:  $minima\_queue \leftarrow$  create queue from minima
3:  $exclusion\_zone \leftarrow window\_length \times exclusion\_factor$ 
4:  $boundaries \leftarrow$  empty list
5: while  $minima\_queue$  is not empty do
6:    $index \leftarrow pop\ minima\_queue$ 
7:   while  $minima\_queue$  is not empty do
8:     if  $minima\_queue\ top \leq index + exclusion\_zone$  then
9:        $pop\ minima\_queue$ 
10:    else
11:      break
12:    end if
13:  end while
14:  append  $index$  to  $boundaries$ 
15: end while
16:  $segments \leftarrow$  build list of  $(start, end)$  tuples from boundaries
17: return  $segments$ 
```

puts all of the candidate boundaries, the local minima, into a queue. From here, we pop the candidates out of the queue one-by-one to see if they are a valid boundary. A candidate minima can only be a valid boundary if it falls outside the exclusion zone. Figure 2.5 shows the boundaries extracted by UFLUSS from the CAC. Notice the local minimum that was excluded because it falls outside of the exclusion zone of boundary b_1 . The UFLUSS algorithm returns the time series segmentation as a list of tuples containing the intervals of each non-overlapping segment. The segments or regime intervals can be directly inferred from the boundary indices.

Definition 2.14. A time series segmentation S_T is a set of $k + 1$ subsequences that split T on boundaries B . Given $B = \{b_1, b_2, \dots, b_k\}$, a segmentation S_T can be defined as follows:

$$S_T = \{T_{1:b_1}, T_{b_1+1:b_2}, \dots, T_{b_k+1:n}\}$$

Continuing with our Japan GDP example, we present the UFLUSS time series segmentation in Figure 2.6. If we look closely at Japan’s GDP time series, we notice four extracted segments. There is a period of slight positive growth from 1970 - 1983 (the blue segment), strong positive growth from 1984 - 1992 (the orange segment), a period of oscillation from 1993 to 2008 (the green segment), and a different pattern from 2009 to 2019 (the red segment).

Given this time series segmentation, we can move on to describing the segmentation classification portion of the Shape Profile technique.

2.4.2 Taxonomic Classification of Time Series Segments

After extracting the segments using semantic segmentation (see Section 2.4.1), we then classify each time series segment into a *shape* or *archetype* taxonomic class describing the pattern of that segment. This is the final step in computing the Shape Profile for a time series. A Shape Profile (SP) is a time series segmentation accompanied by the taxonomic class for each segment in the time series.

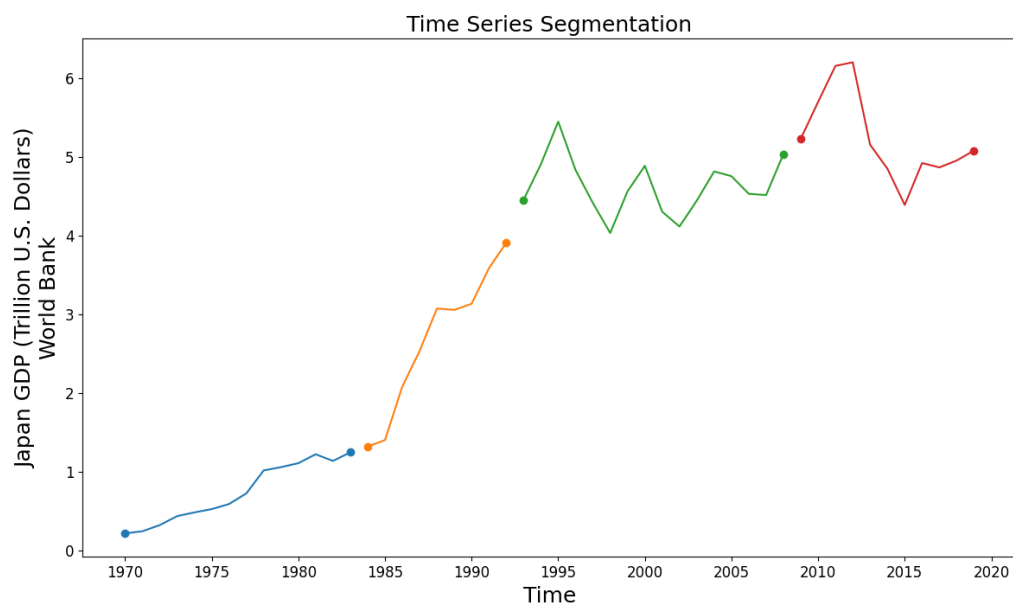


Figure 2.6: Japan's Gross Domestic Product (GDP) Segmentation using Unsupervised Fast Low-cost Unipotent Semantic Segmentation (UFLUSS): There are four extracted segments: a period of slight positive growth from 1970 - 1983 (the blue segment), strong positive growth from 1984 - 1992 (the orange segment), a period of oscillation from 1993 to 2008 (the green segment), and a different pattern from 2009 to 2019 (the red segment).

The work in this section uses the same classification techniques used in the paper, *A Taxonomic Classification Approach for Global Spatio-temporal Data* [143], developed by us and other authors. The Categorize Trends tool within that work classifies the entire time series within a user defined period; whereas, in SP we are classifying extracted segments of time series into taxonomic classifications. We use the same 10 archetype behavior classes as the Categorize Trends technique that were developed with the help of professional analysts. In both the Categorize Trend paper and this one, we are using the same dataset so the behavior classes remain relevant to our SP technique. For other domains, these behavior classes can be altered to suit the problem of interest although it is likely that our classes remain general enough to be applicable to many problems already. Note, we use the terms *behavior class* and *archetype shape* interchangeably throughout the rest of the paper. The 10 archetype shape names and their descriptions are described in Table 2.2. The archetype shapes are meant to be practical patterns of interest to analysts. For example, we have chosen behavior classes such as constant, down, up, wave, and other patterns which might indicate anomalies (e.g., spike-down, spike-up), sudden dramatic shifts (e.g., shift-up, shift-down), and recovery (e.g., smile) in socioeconomic time series data.

For each archetype shape or behavior class, we synthetically generate time series trends representing that class’s shape. The synthetically generated trends for each behavior class are presented in Figure 2.7. We then compare the real-world time series segments to each synthetically generated time series shape archetype using dynamic time warping (DTW). We chose to use DTW because it is a shape-based distance measure. Thus, DTW is invariant to phase shifts which allows us to capture the general shape of a time series segment with robustness to the time domain of the occurring pattern. The lower the DTW distance measure from segment to synthetically generated time series, the more closely that segment fits into the corresponding behavior class. For more information on how DTW works, see Appendix A.

Table 2.2: Archetype Shapes

Archetype	Description
Up	Values that are steadily increasing over the time segment
Down	Values that are steadily decreasing over the time segment
Smile	Values that are initially decreasing but with an upturn towards the end of the time segment
Frown	Values that are initially increasing but with a downturn towards the end of the time segment
Wave	Values that are showing erratic behavior over the time segment
Constant	Values that are showing little to no change over the time segment
Spike Up	Values that are showing little to no overall change except for a sudden rise during a limited time period in the segment, which returns to the previous values
Spike Down	Values that are showing little to no overall change except for a sudden drop during a limited time period in the segment, which returns to the previous values
Shift Up	Values that are initially steady, stepping upward in value and holding at the new value for the rest of the time period
Shift Down	Values that are initially steady, stepping downward in value and holding at the new value for the rest of the time period

In Table 2.3, we present the DTW distance results of comparing each segment to all synthetically generated trends for the Japan GDP example. The minimum value is highlighted and is chosen as the behavior class for the corresponding segment. For this particular example, we can see that there are clear cut minimums for each segment. While we are using hard classification in our technique, there is the possibility of using soft or fuzzy classification which could provide more versatile results when many behavior classes have similar distances to the same segment. Furthermore, Figure 2.8 shows the segments labeled on the time series plot. We can see that visually each segment fits into the behavior class we would expect. The first segment (1970 - 1983) is categorized as an *Up* shape, the second segment (1984 - 1992) is categorized as an *Up* shape, the third segment (1993 - 2008) is categorized as a *Wave* shape, and the fourth segment (2009 - 2019) is classified as a *Shift-Down* shape.

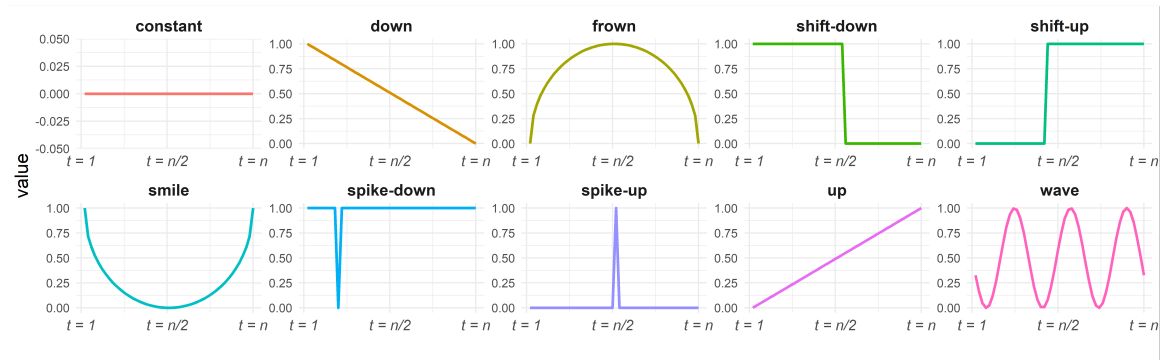


Figure 2.7: Template Trends

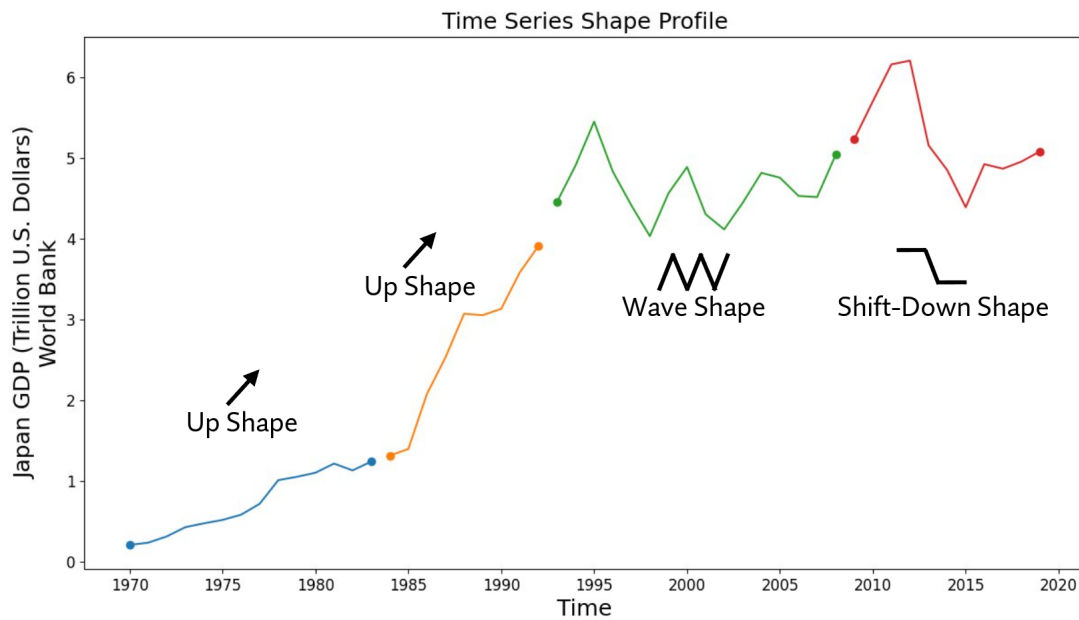


Figure 2.8: Japan's Gross Domestic Product (GDP) Shape Profile: A Shape Profile (SP) consists of a segmented time series along with the shape classifications for each segment.

Table 2.3: Japan’s Gross Domestic Product (GDP) Segment Behavior Class Dynamic Time Warping (DTW) Distances: Each segment’s highlighted value shows the minimum distance across 10 archetype shapes.

Segment	Start	End	Constant	Down	Frown	Shift-Down	Shift-Up	Smile	Spike-Down	Spike-Up	Up	Wave
1	1970	1983	0.47	1.44	0.51	0.68	0.19	0.51	0.72	0.63	0.08	0.45
2	1984	1992	0.50	1.33	0.50	0.68	0.28	0.59	0.54	0.69	0.12	0.55
3	1993	2008	0.49	0.65	0.60	0.47	0.48	0.45	0.55	0.44	0.56	0.38
4	2009	2019	0.40	0.48	0.50	0.25	0.55	0.40	0.68	0.40	0.86	0.54

The categorized segments yield a shape profile for a time series that we can use to answer time series related questions when comparing multiple time series. For example, we can see what other time series have an *up* shape that occurs in the same time period, or overlapping time period. We can also query for when the shapes are different in a time period.

2.5 Data

The dataset utilized in this study consists of geospatial time series data containing country-level socioeconomic indicators with annual resolution. This data was sourced from the database of the World SpatioTemporal Analytics and Mapping Project (WSTAMP), a geospatial analysis platform developed at Oak Ridge National Laboratory [142, 112, 95, 140]. The dataset incorporates economic indicators from various open-source data, with a substantial portion derived from organizations such as the World Bank, the World Health Organization, among others. The WSTAMP database includes over 16,000 attributes spanning 50 years of data across more than 200 nation states and entities, with support for data at multiple administrative levels [112].

2.6 Results

To assess the quality of Shape Profile results, we visually interpret the UFLUSS Matrix Profile semantic segmentation of individual time series along with the shape classification of each segment. Understanding the foundational performance of Shape Profiling is critical, as it establishes the baseline from which further query-by-content searches and overall capabilities can be accurately interpreted. We examine illustrative examples of outcomes in selected economic indicators for Japan.

Figure 2.9 shows the Shape Profile of the Japanese population growth time series. The semantic segmentation in this example performs effectively, mirroring our natural approach to identifying and modeling patterns. The time series is segmented into four regimes that exhibit distinct global behaviors while also maintaining the ability to capture more subtle nuances. The first segment from 1970 to the first half of the decade shows a period of sporadic population growth compared to the rest of the time series. From the mid-1970s to 2010 we see a relatively stable decline of population growth across the time period. However, of note is that our segmentation algorithm UFLUSS still splits the time series during this period into two different regimes. Although both regimes are classified as *downward* trends, we can see that the second segment from 1990 to 2010 (green) exhibits a noisier down trend than that of the first down trend from 1977 to 1989 (orange). This result illustrates a strong ability for our segmentation approach to capture local subtleties within a time series. Furthermore, this example showcases a successful classification of segments into shapes that a human would likely perceive as the best-fit among our 10 behavior classes. In the *spike-up* trend (blue), it might be argued that this segment could also be classified as a *wave* trend. Given that the peak of the spike is much higher than the trough of the valley, it makes sense that this characteristic is the driving force behind the *spike-up* classification. In Section 2.9 we discuss the possibility of including all the classification distances into a multidimensional distance vector for each segment to provide important secondary shape information when necessary.

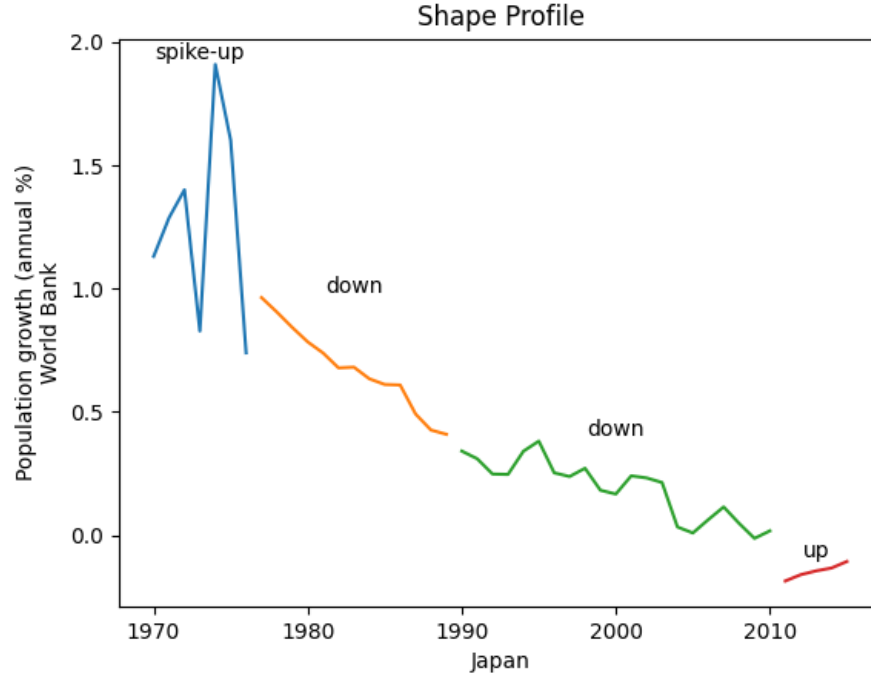


Figure 2.9: Japan's Population Growth Shape Profile

Figure 2.10 shows the Shape Profile for Japan's population growth among individuals aged 65 and older. Viewed as a whole, the time series exhibits a consistent upward trend. However, UFLUSS segments the series into multiple regimes, capturing variations in the steepness of the upward slope across different periods. This example highlights the ability to detect meaningful regime shifts even when the overall directional trend remains the same. Such segmentation is particularly valuable when analyzing subtle differences between segments within the same broader behavior class. Because our shape classification technique is largely agnostic to specific value magnitudes, this characteristic of UFLUSS allows for a more nuanced interpretation of local dynamics within a Shape Profile. Depending on the goals of downstream analysis, a user might instead prefer to classify this time series as a single *up* segment. This consideration raises the possibility of post-processing steps that merge adjacent segments with identical classifications to produce a simpler segmentation. However,

such merging would likely be appropriate only for certain shape types; for example, successive *spike-up* segments might retain important distinctions and would be better analyzed separately.

Figure 2.11 shows the Shape Profile of Japan’s personal remittances. The *shift-up* segment (blue) fits its classification well: the segment is relatively flat initially, followed by a noticeable upward trend, which together characterize the *shift-up* shape. The final two segments, both classified as *spike-down*, capture sharp single-year drops followed by returns to baseline values. The green *spike-down* segment exhibits some oscillation, which could arguably suggest a *wave* pattern. However, the sudden low value in 2013 strongly influences its classification as a *spike-down* archetype.

Figure 2.12 shows the Shape Profile of Japan’s livestock production. The first segment (blue) is classified within the *up* behavior class, accurately capturing the substantial increase in livestock production during the 1970s. Although production continues to rise in the following segment, it is classified as a *frown* (orange), reflecting a negative hyperbolic shape where values increase, round off, and then begin to decline by the end of the segment. The third segment (green) appears relatively flat and might arguably be better classified as a *constant* trend; however, two noticeable drops influence its classification as a *wave*. If only a single sharp drop had been present instead of two, this segment could have been classified as a *spike-down*, similar to the green segment observed in Figure 2.11.

Figure 2.13 shows the Shape Profile of Japan’s crop production. This example highlights a case where the data is noticeably noisier than in previous examples, with unstable oscillations throughout the time series. One might initially expect the algorithm to classify these segments as *wave* patterns. However, the Shape Profiling approach can still detect underlying trends despite the noise. The blue segment, while containing some oscillation, exhibits an overall *frown* shape and is classified accordingly. The orange segment is classified as a *down* trend, demonstrating the method’s robustness in capturing the dominant downward movement despite a large drop in the mid-1990s.

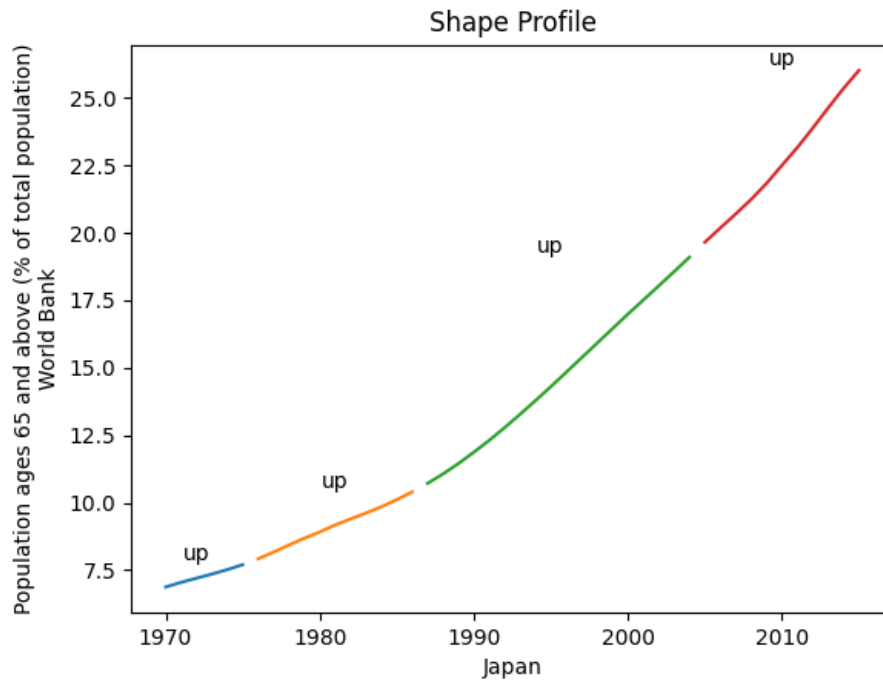


Figure 2.10: Japan's Population Growth (Ages 65 and Older) Shape Profile

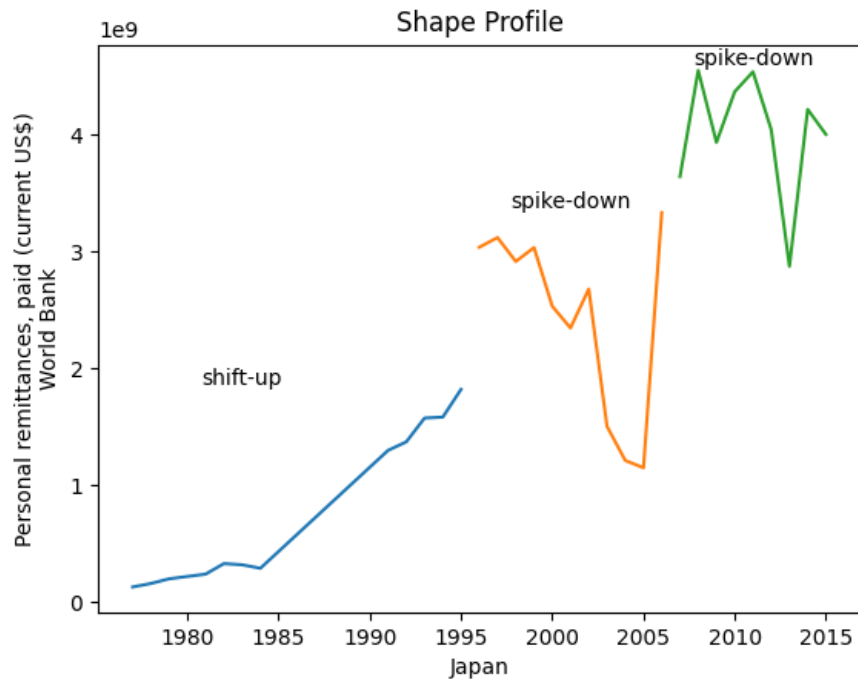


Figure 2.11: Japan's Personal Remittances Shape Profile

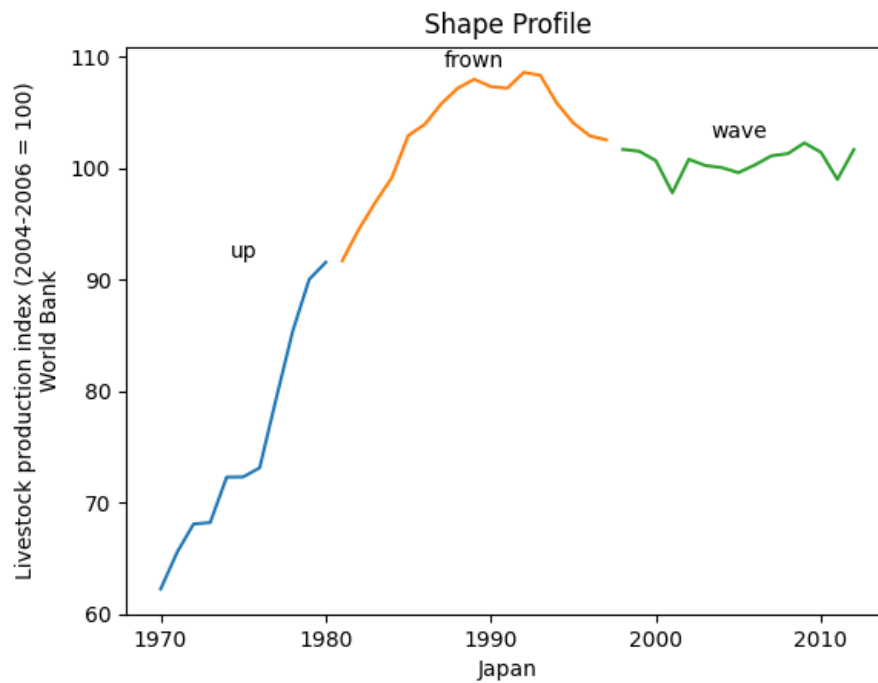


Figure 2.12: Japan's Livestock Production Shape Profile

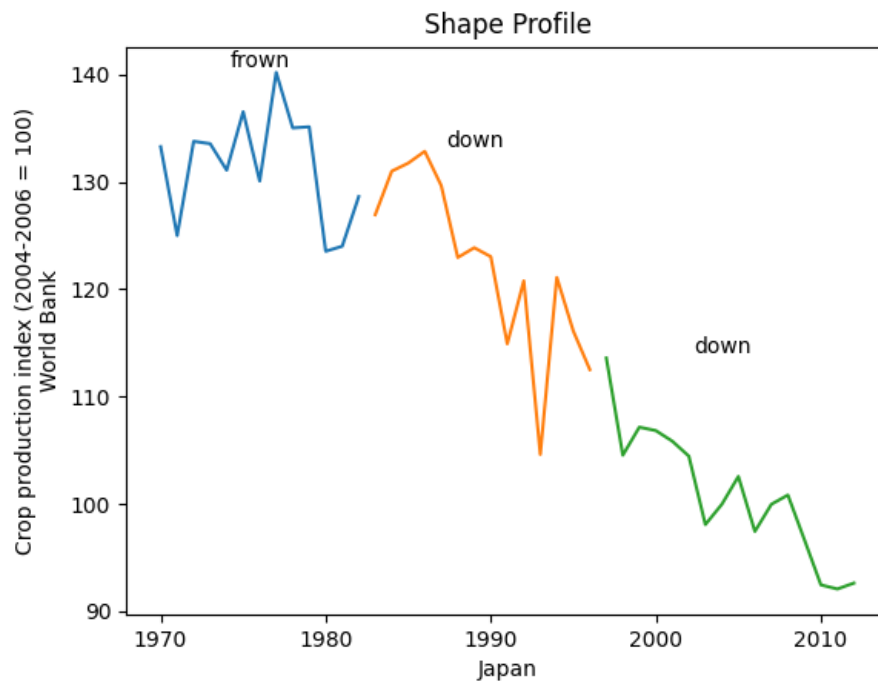


Figure 2.13: Japan's Crop Population Shape Profile

2.7 Limitations

While the FLUSS approach has been described as *parameter-lite* by Gharghabi et al. [42], it is not entirely parameter-free. This property also holds for UFLUSS, the method introduced in this paper. In particular, although the window size in the Matrix Profile is often treated as a hyperparameter, it plays a critical role in defining subsequence length and, consequently, the granularity of segmentation. The original FLUSS authors demonstrated that window size is generally robust by examining the results of using varying window sizes on labeled data [42], but their analysis focused on long time series datasets, typically containing thousands of points. In contrast, our analysis on much shorter series (containing fewer than 50 points) found UFLUSS to be more sensitive to window size, with segmentation boundaries shifting noticeably as the parameter varied.

For example, Figure 2.14 illustrates the Shape Profile for Japan’s health expenditure over 15 years, a much shorter series than those in Section 2.6. The combination of a short series and relatively large window size leads UFLUSS to detect only a single segment. A human analyst might reasonably perceive several distinct patterns—such as a *smile* (2000–2004), a *spike-up* (2004–2011), or a *frown* (2010–2013)—but the algorithm’s limited context causes it to classify the entire sequence as a single *up* segment. This example highlights an important limitation: UFLUSS may oversimplify time series when the window size is large relative to the overall time series length, missing meaningful sub-patterns that are apparent upon visual inspection.

Another practical consideration is the exclusion zone used in FLUSS, which defaults to five times the window size for simplicity [42]. While this setting is reasonable for long time series, we found it problematic for shorter series. The large exclusion zone reduced the number of viable boundary candidates, leading to suboptimal segmentation results. Consequently, we adjusted the exclusion zone in our experiments. For analysts working with shorter time series, this exclusion zone should be treated as an additional hyperparameter requiring consideration.

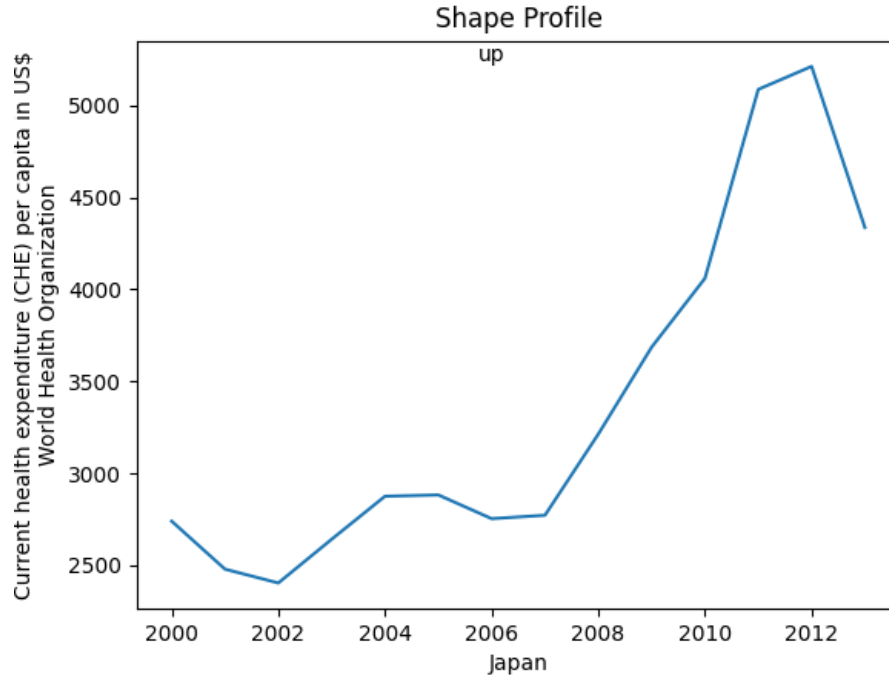


Figure 2.14: Japan's Health Expenditure Shape Profile

Since Shape Profiling relies on Dynamic Time Warping (DTW) distance (see Appendix A), our approach also inherits some limitations of DTW itself. A key concern is degenerate warpings, where many points in one series are aligned to just a few points in another. This effect can distort the influence of certain regions in a time series, suppressing important patterns. While DTW's flexibility allows it to handle temporal misalignments, a valuable feature in most cases, this same property can also lead to unintended distortions, especially in shorter time series.

Figure 2.15 illustrates this issue with the Shape Profile for Japan's consumer price inflation. The final segment (orange) is classified as a *smile*. While the period from 2008 onward clearly exhibits a smile-like pattern, the earlier values from 2002 to 2008 show a steady upward trend that may not fully align with this classification. A plausible explanation is that the 2002–2008 increase is mapped onto only a few points of the smile template in the DTW alignment, diminishing its impact on the overall

segment shape. Meanwhile, the latter portion of the segment aligns more consistently with the expected *smile* pattern. This example highlights how degenerate warpings can bias Shape Profiling results, particularly when segment lengths are short.

One potential way to mitigate this limitation is to move beyond single-label classification and represent segments using a distance vector to all shape templates. In such a scheme, degenerate warpings are less likely to influence all template distances simultaneously. For instance, in this example, the segment is classified as a *smile*, but the *wave* class may have the second-highest similarity. By incorporating multiple shape distances, the classification could reflect the uncertainty or ambiguity inherent in the alignment, allowing analysts to consider alternative shape interpretations rather than relying solely on a single best-fit label. This approach could reduce the impact of degenerate warpings by preserving more information about a segment’s similarity to all shape templates.

2.8 Conclusion

In conclusion, Shape Profiling advances shape-based time series representations by addressing critical gaps in existing methods. Many prior approaches rely on segmentation strategies developed for dimensionality reduction, such as Piecewise Linear Approximation or equal-length slicing, that fail to capture semantically meaningful transitions. Others require analysts to predefine pattern lengths or engage in extensive post-processing to extract and interpret shapes, limiting their scalability and usability. Shape Profiling overcomes these challenges through two key contributions: (1) the use of semantic segmentation via UFLUSS, which identifies regime boundaries without the need for labeled data or predefined segmentation counts, and (2) a classification framework that minimizes analyst intervention by automatically labeling each regime based on dynamic alignment with interpretable shape templates.

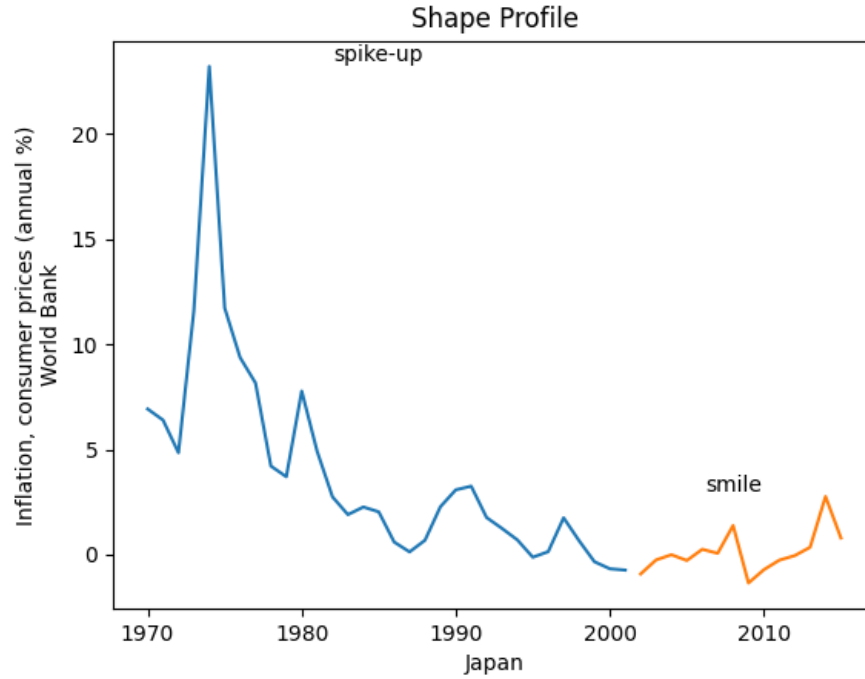


Figure 2.15: Japan’s Inflation of Consumer Prices Shape Profile

By uniting these components into a cohesive, automated pipeline, Shape Profiling produces compact, human-aligned representations of time series that are well-suited for large-scale exploration and discovery. The resulting Shape Profiles offer not only a high-level summary of structural changes in the data but also a practical foundation for content-based querying, comparative analysis, and pattern retrieval. This work lays the groundwork for further advancements in time series interpretation and indexing, particularly in domains where analysts need scalable, low-effort tools to surface and reason about temporal patterns.

2.9 Future Work

While we have presented a new approach for time series data mining that is useful in its own right, there is room for further exploration of our technique from several

angles. One possibility is building on our approach for larger time series datasets. While our dataset was large, it was admittedly large in a different capacity than what most practitioners consider large time series data. Our data was large in the sense that we had many attributes or dimensions for each country, but each attribute contained relatively few data points due to our primary resolution of yearly collected data during the last half-century or so. This precluded us from using approximate techniques, where data that has finer resolutions over larger time periods may require them. In particular, using the STAMP anytime algorithm variant of Matrix Profile on data of this nature could be explored.

In the Matrix Profile I paper [166], they present an additional algorithm called Scalable Time series Anytime Matrix Profile, or STAMP, that computes an approximate Matrix Profile. Instead of computing the distance from every subsequence to every other subsequence in order, distances are computed randomly and the Matrix Profile algorithm can be stopped at anytime. This enables the user to tune their tolerance for accuracy versus speed depending on the specific application and gives them an alternative to the slower but exact MASS algorithm used in traditional Matrix Profile computation. As with MASS, STAMP maintains the ability to be parallelizable by distributing the distance calculations across additional cores.

In terms of asymptotic time complexities, the Matrix Profile portion of our approach has the highest upper bound of $O(n \log n)$ given by its FFT operation. In practice, most FFT computation has been highly optimized and it tends to be closer to a linear time computation, especially when the data is smaller. Nonetheless, going forward the Matrix Profile implementation provides the most upside in reducing the overall Shape Profile runtime for larger time series datasets.

Our approach was presented with batch data as the primary consideration. That is, our algorithm expects to see all of the time series data points prior to calculating the Shape Profile. However, in many cases it might be desirable to use our approach on streaming applications that are acquiring data regularly at finer timescales. Upon first constructing a Matrix Profile, further data may be added to the time series. In

this case, the Matrix Profile can be updated incrementally as each new data arrives. The authors in the Matrix Profile I paper [166] provide an incremental algorithm for efficiently updating the Matrix Profile (STAMP Incremental) and go on to later provide an online semantic segmentation variant of their FLUSS algorithm called Fast Low-cost Online Semantic Segmentation (FLOSS) in their Matrix Profile VIII paper [42]. FLOSS computes the segmentation over a sliding window of the most recent streaming data instead of the whole time series. The key insight allowing FLOSS to be computationally tractable is maintaining only unidirectional arcs within the CAC, thus eliminating the need to recalculate the entire Matrix Profile since a new point could potentially be a nearest neighbor match to all past subsequences in the pathological case. In practice it is shown that a one-dimensional arc curve produces similar results to that of a bidirectional arc curve [42], thus ridding ourselves of the need to update every Matrix Profile entry. This would allow us to maintain a worst-case $O(n \log n)$ time complexity for re-segmenting with each ingressed data point. It should be noted that to the best of our knowledge FLOSS does not currently have an accompanying unparameterized version of the algorithm. Also, it remains to be seen within what domains maintaining a running Shape Profile might be useful.

Choosing to use segmentation before shape classification was a deliberate choice on our part. It made sense to first find reasonable semantic regimes, and then characterize those by labeling them with their best fit shape. This is in opposition to labeling all windows or a range of windows whether you know a pattern of interest might exist there or not. However, in some cases segmenting a time series may not produce as meaningful results (e.g. a really long time series). This could especially be the case when segments may contain many patterns, but shape profiling is classifying it as one pattern. It is likely this could happen when the dataset contains time series of many different lengths making it hard to tune the window size for UFLUSS. In some cases it is also plausible that having a full enumeration of all shapes and patterns is important for the specific domain application. Computing shape classifications over all possible window sizes or a range of window sizes would allow more fine-grained

data analysis such as considering patterns across entities within specific time periods, not just overlapping time periods as with the semantic segmentation approach. The one downside to this approach is the exponential increase in computational time and space. However, for offline data analysis this approach is feasible for those who want an ultimate enumeration of shapes and patterns in their dataset.

In the Matrix Profile and segmentation portion of the shape profiling technique, a window size must be chosen. We mentioned that we chose the window size based on visual inspection of the downstream classification results. While this worked well for our dataset, choosing the window size in this way could present issues when the time series have varying lengths. Manually choosing the window size also requires understanding of the full Shape Profile operation pipeline and could present challenges for non-technical users. Future work should explore automatically selecting the window size based on quantitative measures or scoring.

In the creation of our synthetic trends, we chose shapes that were intuitive to the geospatial community. These shapes would be helpful across many domains as they are fairly generic; however, there are domains that may require additional or a new set of synthetic trends based on the specific use case. For example, in the financial technology industry, it may be desired to generate synthetic trends representing common security trading patterns. While any set of trends or shapes can be substituted during our classification step, there is further room to explore DTW’s sensitivity to various synthetically generated trends. Sensitivity is likely dependent upon the specific DTW implementation optimizations. For example, if you have a *spike-up* trend generated by giving one data point a high value in an otherwise constant linear sequence, DTW could optimize out a mapping onto this point depending the real time series data. In this case, it might be better to have more than one point used to create the spike shape. While our results seem to show robustness to this type of issue, a thorough sensitivity analysis could aid in creating more robust synthetically generated trends as other datasets are explored.

Shape Profiling in this paper was presented as an unsupervised data analysis technique. Alternatively, Shape Profiling could be used in the context of supervised or labeled data. If time series data comes labeled with regimes, it would not be necessary to use our unparameterized version of FLUSS. We could simply train FLUSS to detect the number of regimes as originally laid out in the Matrix Profile VIII paper [42]. Given the learned number of regimes and the segmented time series, we could proceed forward with the Shape Profile trend classification. In the context of geospatial time series analysis, the labels might be known socioeconomic events. A Shape Profile would inform analysts of the trend leading up to and after an event and could be used as a data summary technique. Other domains with supervised time series data are presumed to have similar applications.

We presented Shape Profiling as a hard classification technique where each time series segment can belong to only one behavior class. In the future, it would be possible to convert the classification to a soft classification or fuzzy classification using the DTW distance. In cases where segments may have close matches to many behavior classes, or domains in which several similar patterns are used as shape archetypes, this could allow further development of data mining algorithms where multi-class classification problems may be desired.

UFLUSS segments time series based on distinct regimes. However, it is possible that adjacent segments get classified in the same behavior class. For example, see Figure 2.8, where adjacent segments are both classified into the *Up* behavior class. We see that the first segment (1970 - 1983) has a slight linear upward trend, whereas the second segment between 1984 and 1992 has a more aggressive upward linear trend. For some use cases, it may be desirable to combine these into a single *Up* behavior class from 1970 to 1992. In the future, it would be interesting to explore the idea of re-coarsening Shape Profiles by combining linked segments with the same behavior class. Re-coarsening could combine segments after classification and re-apply the classification technique to the new segment. It would be possible to perform many iterations of re-coarsening depending on the necessitated granularity of the results.

There is also the potential to transform a SP into a sequence of characters using an alphabet of letters representing corresponding behavior classes. For example, in the case of our 10 behavior classes, we could define an alphabet shown in Table 2.4. Since we are using non-uniform segments, we could easily represent all the sliding windows in a particular segment by the corresponding alphabet letter. This would open the door to using text matching techniques for analyzing and querying Shape Profiles.

While this work has focused on discovering shapes using more traditional computational and mathematical approaches, the potential of artificial intelligence (AI) methods warrants further exploration. One promising direction would be to use our Shape Profiling pipeline to generate high-quality, semantically meaningful training data for time series segmentation. This data could support the development of unsupervised or semi-supervised neural network models capable of automatically detecting regime boundaries and classifying shape patterns. Such models could offer significant advantages in scenarios requiring rapid, large-scale segmentation and pattern extraction, enabling near real-time analysis without extensive manual intervention.

Table 2.4: Symbolic Index Alphabet: Each letter corresponds to a behavior class or shape. Note: this alphabet can be extended if desired.

letter	shape
c	constant
d	down
f	frown
D	shift-down
U	shift-up
s	smile
k	spike-down
K	spike-up
u	up
w	wave

2.10 Author Contributions

Evan Ezell and Robert Stewart conceptualized the idea of a temporal fingerprint using generic shapes for time series data. Evan Ezell implemented the code. Evan Ezell wrote the manuscript. Robert Stewart oversaw the work, proofread and edited the final manuscript.

Chapter 3

Dynamic Network Matrix Profile: Uncovering Temporal Patterns in Dynamic Graphs

3.1 Abstract

The Matrix Profile is a powerful data structure for time series, compactly encoding similarity information that supports a wide range of data mining tasks by using the similarity join solution. While this concept has been extensively applied to time series, its potential in the context of dynamic networks remains largely unexplored. We introduce Dynamic Network Matrix Profile (DNMP), a novel framework that extends the similarity join paradigm to dynamic networks by leveraging graph embeddings and applying the Multivariate Matrix Profile. This approach transforms sequences of network snapshots into multidimensional time series, enabling the discovery of recurring and anomalous temporal patterns in evolving graphs. DNMP bridges the gap between time series analysis and dynamic graph mining, unlocking new capabilities for pattern discovery in complex systems. We demonstrate that DNMP can effectively analyze two dynamic network datasets: the Capital Bikeshare system and the Atlas global trade data.

3.2 Introduction

The Matrix Profile, originally introduced in a landmark paper [166], was proposed as a new solution to the all-pairs similarity search problem (i.e., similarity join) in time series data mining. Described by leading expert Dr. Eamonn Keogh as “the best idea in time series data mining in a decade,” the Matrix Profile has since proven to be a compact and efficient data structure for storing critical similarity information. Its design supports a wide range of time series analysis tasks—including motif discovery, anomaly detection, and semantic segmentation—and has demonstrated strong performance across numerous applied domains.

In Chapter 2, we applied semantic segmentation with the Matrix Profile data structure to partition time series into characteristic subsequences. In this chapter, we extend the Matrix Profile framework developed by the University of California,

Riverside (UCR) team to dynamic network analysis. In [166], it is stated that “nothing intrinsically precludes [Matrix Profile] generalizations to multidimensional data.” As such, following this statement [167] extended the one-dimensional Matrix Profile to the multidimensional case for finding time series motifs. We apply the ideas from both [166] and [167] to develop a Matrix Profile for dynamic networks. To the best of our knowledge, the Matrix Profile has not yet been applied to dynamic networks in this way.

Traditionally, dynamic networks have been analyzed by comparing individual snapshots in isolation. A key contribution of this work introduces a new perspective: we can mine dynamic networks by solving the dynamic network similarity join problem. The self-similarity join problem for dynamic networks tries to find the most similar subsequence of network snapshots to all other subsequences within the same dynamic network. The general similarity join problem can be applied to any type of discrete data objects forming a series. For example, this problem has been heavily studied for text data and time series. However, this data object can also be graph snapshots in a dynamic network. This perspective shift has several advantages, as demonstrated by the success of similar approaches in time series data mining. For example, given daily snapshots, we can identify other similar seven day sequences or detect the most anomalous seven day period—tasks that differ fundamentally from simply aggregating data into weekly snapshots and comparing them individually. Aggregating daily data into a single weekly snapshot discards the evolutionary information present between days. By instead operating on sequences of snapshots, our approach preserves the underlying dynamics of the network while enabling richer temporal comparisons.

Applying this perspective using traditional methods, such as graph distance measures, would be computationally expensive. Because these computations are intensive, extending them across sequences of snapshots can quickly become impractical. Our approach improves the feasibility of this sequence-based analysis by enabling efficient

similarity joins, allowing for continuous comparisons without sacrificing temporal resolution for larger data.

The Dynamic Network Matrix Profile (DNMP) framework first builds or learns a vector representation of each snapshot. These graph embeddings are used to transform snapshots of a dynamic network into signal carrying vectors. These embeddings, taken across multiple time steps, form a multidimensional time series across the embedding dimensions. This bridges Matrix Profile methods to dynamic networks, facilitating tasks such as similarity search, motif discovery, and anomaly detection over evolving graphs, and extending the reach of Matrix Profile methods beyond traditional time series analysis.

The remainder of this chapter is organized as follows. Section 3.3 reviews relevant background and related work necessary for understanding our method. Section 3.4 formally introduces the DNMP approach. Section 3.5 and Section 3.6 present experimental results, demonstrating DNMP’s effectiveness on two dynamic network datasets: the Capital Bikeshare system [19] and the Atlas global trade data [5]. Section 3.8 provides concluding remarks on our findings and contributions. Finally, Section 3.9 outlines potential directions for future research.

3.3 Background

Understanding the evolution of networks over time requires models that can capture both structural and temporal dynamics. Traditional approaches for comparing graphs often rely on distance measures such as graph edit distance, spectral distance, and kernel-based techniques, which quantify similarity based on structural properties [35]. However, these methods are primarily designed for direct comparisons between individual graph snapshots. Extending them to dynamic networks would require pairwise comparisons across multiple snapshots, resulting in significant computational overhead and making such approaches impractical for large dynamic graph sequences.

To address scalability and richer temporal modeling, research has shifted toward dynamic graph embedding methods, which aim to represent nodes, edges, or entire graphs in low-dimensional vector spaces that evolve over time. Unlike static embeddings, dynamic embeddings must account for both the network’s topology at each snapshot and its temporal progression, requiring stability across embeddings, efficiency in training, and the ability to capture evolving dynamics such as gradual changes, regime shifts, and sudden structural transitions [163, 8].

Dynamic networks differ from static ones in that nodes and edges can appear, disappear, or change attributes over time. Embedding methods for dynamic networks must therefore balance preserving local structural features with maintaining temporal consistency. Two main approaches have emerged [163]: structural-first and temporal-first methods. Structural-first methods extend static graph embeddings to dynamic settings, typically by retraining or fine-tuning over time. Whereas, temporal-first methods explicitly model the temporal dimension, often using recurrent models or point processes to predict future states.

Several dynamic embedding frameworks have been proposed. Among the most influential foundations for many methods is Node2Vec, a random-walk based embedding method for static graphs [48]. The skip-gram model, which learns embeddings by predicting the local context of a target token, was popularized and optimized for large-scale training by Mikolov et al. [90, 91] through the Word2Vec framework. Node2Vec [48] adapted these ideas to graphs by treating random walks over nodes as “sentences,” enabling the application of the skip-gram model to learn node embeddings that preserve both local and global graph structure. DynNode2Vec [86] extends the Node2Vec framework to dynamic graphs by incrementally updating embeddings over time, using changed regions of the graph to generate updated random walks while warm-starting from previous embeddings to maintain temporal consistency.

To enable snapshot-level comparisons, it is necessary to aggregate node embeddings into a compact representation of the entire graph. In this work, we use

a weighted variant of DynNode2Vec [86] with mean pooling to generate graph-level snapshot embeddings. To analyze the resulting embeddings, we leverage the Multidimensional Matrix Profile (MMP), a data mining primitive originally introduced for time series similarity joins [166, 167].

The similarity join problem asks: given a collection of objects (e.g., subsequences of a time series, subsequences of dynamic graph snapshots), find the most similar pair for each object. This problem can be challenging due to the large number of overlapping subsequences and the computational burden of exhaustive pairwise comparisons. Traditional approaches suffered from scalability issues, often requiring months of computation for modest-sized datasets. The Matrix Profile framework revolutionized this task by introducing an exact, parameter-free algorithm based on Fast Fourier Transforms (FFT) that can compute and maintain nearest neighbor relationships efficiently [166]. Importantly, the Matrix Profile is not merely a computational optimization—it unifies and accelerates a wide range of tasks, including motif discovery, discord (anomaly) detection, segmentation, shapelet mining, and density estimation in time series.

The Multidimensional Matrix Profile (MMP) extends the Matrix Profile to support full multidimensional time series while preserving the key advantages of exactness, scalability, and robustness with little to no parameter tuning [167]. The MMP also allows for analyzing a subset of dimensions when every dimension may not contain meaningful signal. In our work, the multidimensional time series of interest arises from graph embeddings, where every dimension is designed to capture meaningful structure. Therefore, we compute the full-dimensional Multidimensional Matrix Profile directly. This choice preserves the entire embedding signal across all dimensions and treats the graph snapshots as complete structural representations over time.

We build upon the work discussed in this section, by first embedding evolving snapshots of a network into vector spaces and then treating the sequence of embeddings over time as a multidimensional time series. Finally, we apply the MMP

to discover periods of structural similarity, transition, or novelty within dynamic networks. Our work bridges the gap between dynamic network embeddings and time series analysis, allowing us to leverage the efficiency and robustness of the Matrix Profile for dynamic graphs.

3.4 Methods

In this section, we formally describe the Dynamic Network Matrix Profile (DNMP) technique. Fundamentally, a dynamic graph can be viewed as a sequence of static graphs, with each static graph serving as a snapshot in time.

Definition 3.1. A static network or graph G is represented as $G = (V; E; w)$, where V is a set of vertices or nodes, E is the set of edges connecting nodes $E \subseteq \{(u, v) \mid (u, v) \in V \times V, u \neq v\}$, and w is a weight function $w : E \rightarrow \mathbb{R}$ such that $w(u, v) = c$ where $c \in \mathbb{R}$.

For DNMP, we consider the discrete model for representing our dynamic graph as a sequence of static snapshot graphs. Static snapshots may be built directly from a discrete data collection or can also be aggregated data from a particular time window. Figure 3.1 shows a node-link diagram of what this sequence of snapshots might look like.

Definition 3.2. A discrete or snapshot dynamic network $\mathbf{G}$ is a sequence of network snapshots within a given time period. $\mathbf{G} = [G_1, G_2, \dots, G_t, \dots, G_T]$ where T is the

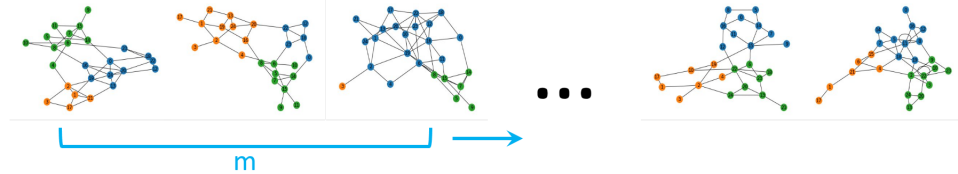


Figure 3.1: Dynamic Graph as a Sequence of Snapshots

number of snapshots and $t \in \{1, \dots, T\}$. Each snapshot G_t is the graph at time period t , $G_t = (V_t; E_t; w_t)$.

For a single graph (Definition 3.1) or each snapshot of a dynamic graph (Definition 3.2) we can define an embedding. The purpose behind an embedding is to encode a graph into a vector of some dimension d such that it is smaller than storing the entire graph while still preserving the desired properties and information of the graph. Typically, $d \ll |V|$.

Definition 3.3. A static graph embedding can be defined as a function $\phi : G \rightarrow \mathbb{R}^d$. Thus, a snapshot embedding can be defined as a function on each snapshot of a dynamic graph $\phi_t(G_t) \in \mathbb{R}^d$ where $G_t \in \mathbf{G}$ and $t \in \{1, \dots, T\}$.

Our approach can be easily used with most embedding techniques (e.g. kernel, feature, representation learning) so long as the resulting embedding method yields a single vector per snapshot. In practice, node-level embeddings (e.g. Node2Vec [48]) can be pooled, transforming them into a graph-level embedding ϕ . Furthermore, given all the graph-level embeddings, we can define an embedding matrix such that each column of the matrix represents the embedding at snapshot t .

Definition 3.4. Given a sequence of embeddings $\{\phi_t(G_t)\}_{t=1}^T$, the embedding matrix of a dynamic graph is defined as:

$$\Phi(\mathbf{G}) = [\phi_1(G_1), \phi_2(G_2), \dots, \phi_T(G_T)] \in \mathbb{R}^{d \times T},$$

where each column $\phi_t(G_t)$ corresponds to the embedding at time t .

The embedding matrix $\Phi(\mathbf{G})$ can also be interpreted row-wise as a multidimensional time series describing the dynamic graph. Each row of $\Phi(\mathbf{G})$ represents the temporal evolution of one embedding dimension across all time steps:

$$\Phi_i(\mathbf{G}) = [\phi_1(G_1)_i, \phi_2(G_2)_i, \dots, \phi_T(G_T)_i] \in \mathbb{R}^T, \quad \forall i \in \{1, 2, \dots, d\}.$$

Thus, $\Phi(\mathbf{G})$ simultaneously encodes temporal changes in each embedding dimension and the structural state of the graph at each time step. By treating the dynamic graph in a row-wise manner as a multidimensional time series, we can construct a Multidimensional Matrix Profile to analyze the dynamic graph. For simplicity, throughout the rest of this chapter we define the i -th embedding dimension's time series as:

$$X^i = \Phi_i(\mathbf{G}) \in \mathbb{R}^T, \quad \forall i \in \{1, 2, \dots, d\}.$$

Therefore, the full multidimensional embedding matrix can be compactly denoted as:

$$X = [X^1, X^2, \dots, X^d]^\top \in \mathbb{R}^{d \times T}.$$

Matrix Profile compares the individual subsequences of a time series to capture local properties. We can define the subsequence on the embedding dimension's time series. For clarity, we start by defining the univariate Matrix Profile (MP) and progress to the Multidimensional Matrix Profile (MMP) used for DNMP. The univariate definitions are equivalent to the Matrix Profile definitions defined in Section 2.4.1 and [166], but defined on the embedding dimension's time series instead of a generic time series.

Definition 3.5. A subsequence $X_{j,m}^i \in \mathbb{R}^m$ of a univariate time series embedding dimension X^i is a continuous subset of values from X^i of length m , starting from position j . Formally,

$$X_{j,m}^i = [x_j^i, x_{j+1}^i, \dots, x_{j+m-1}^i],$$

where $i \in \{1, 2, \dots, d\}$ and $j \in \{1, 2, \dots, T - m + 1\}$.

The subsequence length m , also known as the window size, is the only parameter that must be defined for the Matrix Profile algorithm. According to the authors, the choice of m can be determined by the desired length of pattern to search for. In the case of DNMP, if we have yearly snapshot graphs for a dataset, $m = 5$ would compare five year periods of network evolution to all other five year periods. Figure

3.1 illustrates how the sliding window m determines the length of snapshot sequences used in the similarity join.

Definition 3.6. The Distance Profile $D_j^i \in \mathbb{R}^{T-m+1}$ of a univariate time series embedding dimension X^i and its subsequence $X_{j,m}^i$ is a vector storing distances to every possible subsequence of length m :

$$D_j^i = [\text{dist}(X_{j,m}^i, X_{1,m}^i), \dots, \text{dist}(X_{j,m}^i, X_{T-m+1,m}^i)],$$

where $i \in \{1, 2, \dots, d\}$ and $j \in \{1, 2, \dots, T - m + 1\}$. dist is the z-normalized Euclidean distance between each subsequence.

Definition 3.7. The Matrix Profile $P^i \in \mathbb{R}^{T-m+1}$ of a univariate time series embedding dimension X^i is a meta time series that stores the z-normalized Euclidean distance between each subsequence and its nearest neighbor subsequence. In other words, the Matrix Profile stores the minimum distance from each subsequence's Distance Profile. Formally, the j -th position in P^i is defined using the distance profile D_j^i as:

$$P_j^i = \min(D_j^i)$$

The full Matrix Profile for the i -th embedding dimension is defined as:

$$\begin{aligned} P^i &= [P_1^i, P_2^i, \dots, P_{T-m+1}^i] \\ &= [\min(D_1^i), \min(D_2^i), \dots, \min(D_{T-m+1}^i)] \\ &= [\min(D_j^i)]_{j=1}^{T-m+1} \end{aligned}$$

In practice, to prevent *trivial matches*, where a subsequence's nearest neighbor is itself or a nearby subsequence, an exclusion region is enforced such that there are no such matches in the Distance Profile or Matrix Profile.

We now describe the Multidimensional Matrix Profile. As mentioned previously, we can compactly represent the full multidimensional embedding as follows.

Definition 3.8. A multidimensional time series embedding $X \in \mathbb{R}^{d \times T}$ is a set of d co-evolving univariate embedding dimensions $X^i \in \mathbb{R}^T$. Formally,

$$X = [X^1, X^2, \dots, X^d]^\top,$$

where d is the number of dimensions and T is the number of time steps.

Definition 3.9. A multidimensional subsequence $X_{j,m} \in \mathbb{R}^{d \times m}$ of a multidimensional time series embedding X is a continuous subset of values from each dimension of length m , starting from position j . Formally,

$$X_{j,m} = [X_{j,m}^1, X_{j,m}^2, \dots, X_{j,m}^d]^\top,$$

where $X_{j,m}^i = [x_j^i, x_{j+1}^i, \dots, x_{j+m-1}^i]$ for $i \in \{1, 2, \dots, d\}$ and $j \in \{1, 2, \dots, T - m + 1\}$.

Definition 3.10. The d -dimensional Distance Profile $D_j \in \mathbb{R}^{T-m+1}$ of a multidimensional time series embedding X and its subsequence $X_{j,m}$ is a vector storing the d -dimensional distances between $X_{j,m}$ and every other subsequence of length m in X . Formally,

$$D_j = [\text{dist}^{(d)}(X_{j,m}, X_{1,m}), \dots, \text{dist}^{(d)}(X_{j,m}, X_{T-m+1,m})],$$

where $j \in \{1, 2, \dots, T - m + 1\}$ and $\text{dist}^{(d)}$ is the z-normalized Euclidean distance across all d dimensions.

Definition 3.11. The d -dimensional Matrix Profile $P \in \mathbb{R}^{T-m+1}$ of a multidimensional time series embedding X is a meta time series that stores the minimum distance from each subsequence's d -dimensional Distance Profile. Formally, the j -th position in P is defined using the distance profile D_j as:

$$P_j = \min(D_j)$$

The full Matrix Profile for the multidimensional time series is defined as:

$$\begin{aligned}
P &= [P_1, P_2, \dots, P_{T-m+1}] \\
&= [\min(D_1), \min(D_2), \dots, \min(D_{T-m+1})] \\
&= [\min(D_j)]_{j=1}^{T-m+1}
\end{aligned}$$

Figure 3.2 conceptually illustrates the role of the MMP vector. Although each snapshot is ultimately represented by an embedding, the process still involves structurally comparing sequences of snapshots to one another to identify the best match.

As mentioned in Section 3.3, the original MMP for motif discovery [167] also computes the MMP for a subset of k -dimensions, whereas we compute the full MMP for all d embedding dimensions. The reasoning behind this is that in traditional multidimensional time series, some dimensions are less informative for the patterns of interest. For example, in motion capture data for a boxer, sensors on the feet may contribute less relevant information than sensors on the arms when detecting punches. To address this, [167] selects the best k dimensions to focus on the most meaningful signals.

However, in the case of DNMP, we are treating the dynamic network embedding itself as the multidimensional time series. Since each snapshot is represented by an embedding, all dimensions should contribute meaningful signal to the MMP. Thus, we ignore this special k -dimensional case and use the full d -dimensional MMP. We use the STUMPY implementation to compute the MMP [77], which internally uses a multidimensional version of the Scalable Time series Ordered-search Matrix Profile (STOMP) algorithm presented in [175].

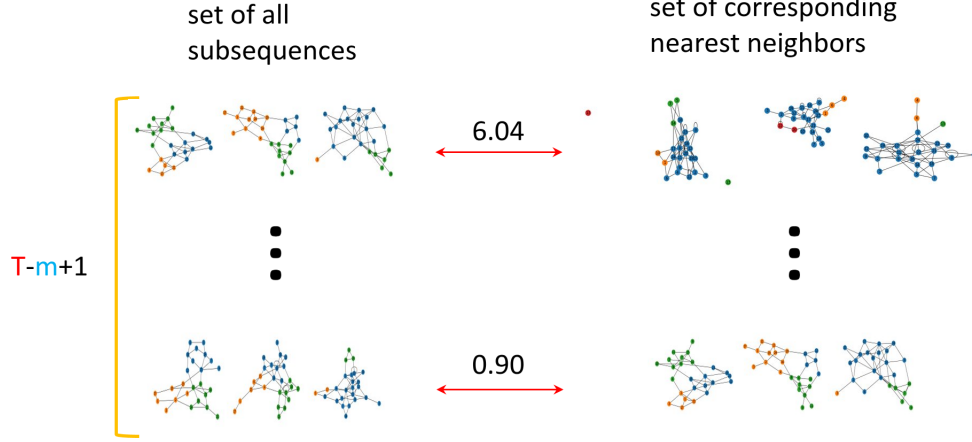


Figure 3.2: Dynamic Graph Matrix Profile Vector

3.5 Capital Bikeshare Results

3.5.1 Data

Bikesharing systems offer a lightweight and flexible alternative to traditional public and private transportation in urban areas. These services allow users to rent a bicycle from one station and return it to any other within the network, enabling efficient short-distance travel. Over the past decade, bikesharing has undergone significant transformation, with the introduction of mobile applications that provide real-time information on bike availability and station status. These advancements have greatly improved the system’s reliability and convenience.

Although the concept of bikesharing originated in Amsterdam and other European cities over 50 years ago [33], it has gained popularity in the United States, particularly in large metropolitan areas such as New York City (Citi Bike) and Washington, D.C. (Capital Bikeshare). This rise in adoption coincides with increasing urban congestion and longer commutes by private vehicle.

In this study, we examine the Capital Bikeshare system, which serves the Washington, D.C. metropolitan region. Capital Bikeshare publicly releases detailed

trip data for educational and research purposes [19]. The dataset includes information such as trip duration, start and end times, station locations, bike ID, and rider type. As of now, the system operates around 800 stations, as illustrated in Figure 3.3.

To analyze the temporal structure of the bikeshare network, we aggregate trips on a daily basis to construct a sequence of snapshot graphs. In each graph, nodes represent stations, and directed edges capture rides from start to end stations. The weight of each edge corresponds to the number of trips taken between the connected stations on that day. Using this approach, we generated daily snapshot graphs spanning from January 2018 through March 2025, resulting in over 2,500 individual network snapshots. An example of such a snapshot is shown in Figure 3.4, which visualizes station-level trip volumes on Saturday, March 15, 2025.

3.5.2 Model Training

We trained a DynNode2Vec-style graph embedding for each daily snapshot in the Capital Bikeshare dataset. While the original DynNode2Vec model is designed for unweighted and undirected graphs and produces node-level embeddings, we adapted it to operate on weighted directed graphs. Specifically, we modified the random walk process to respect edge direction and bias edge selection according to edge weight. These weights reflect daily rider counts, making high-traffic routes more likely to be traversed during walks. As a result, the embeddings better capture the relative importance of central stations with high ridership compared to peripheral stations.

DynNode2Vec reduces training time by updating random walks only for nodes that have changed between consecutive snapshots. A node is considered changed if it is incident to any edge that has been added or removed, as formally defined:

$$\Delta V_t = V_{\text{add}} \cup \{v_i \in V_t \mid \exists e_i = (v_i, v_j) \in (E_{\text{add}} \cup E_{\text{del}})\}$$

We retained this update policy in our implementation. However, the Capital Bikeshare network exhibits high edge turnover between daily snapshots, which

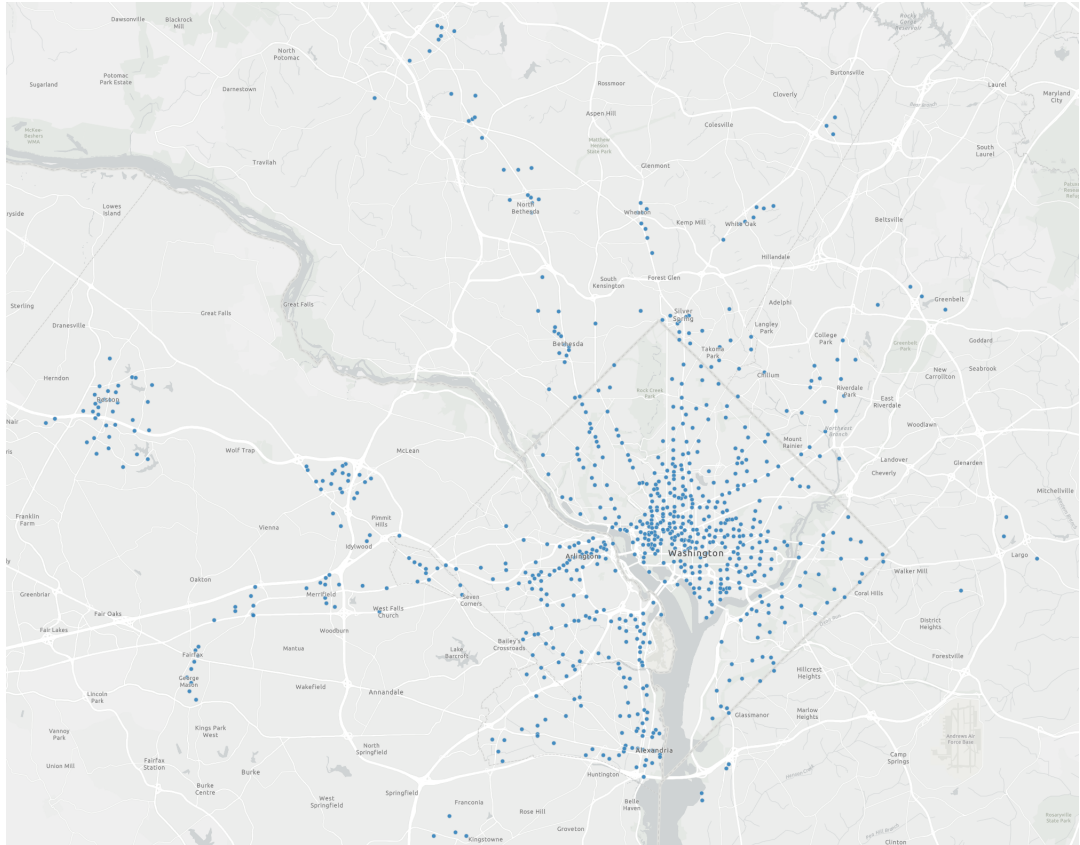


Figure 3.3: District of Columbia Capital Bikeshare Station Locations [19]

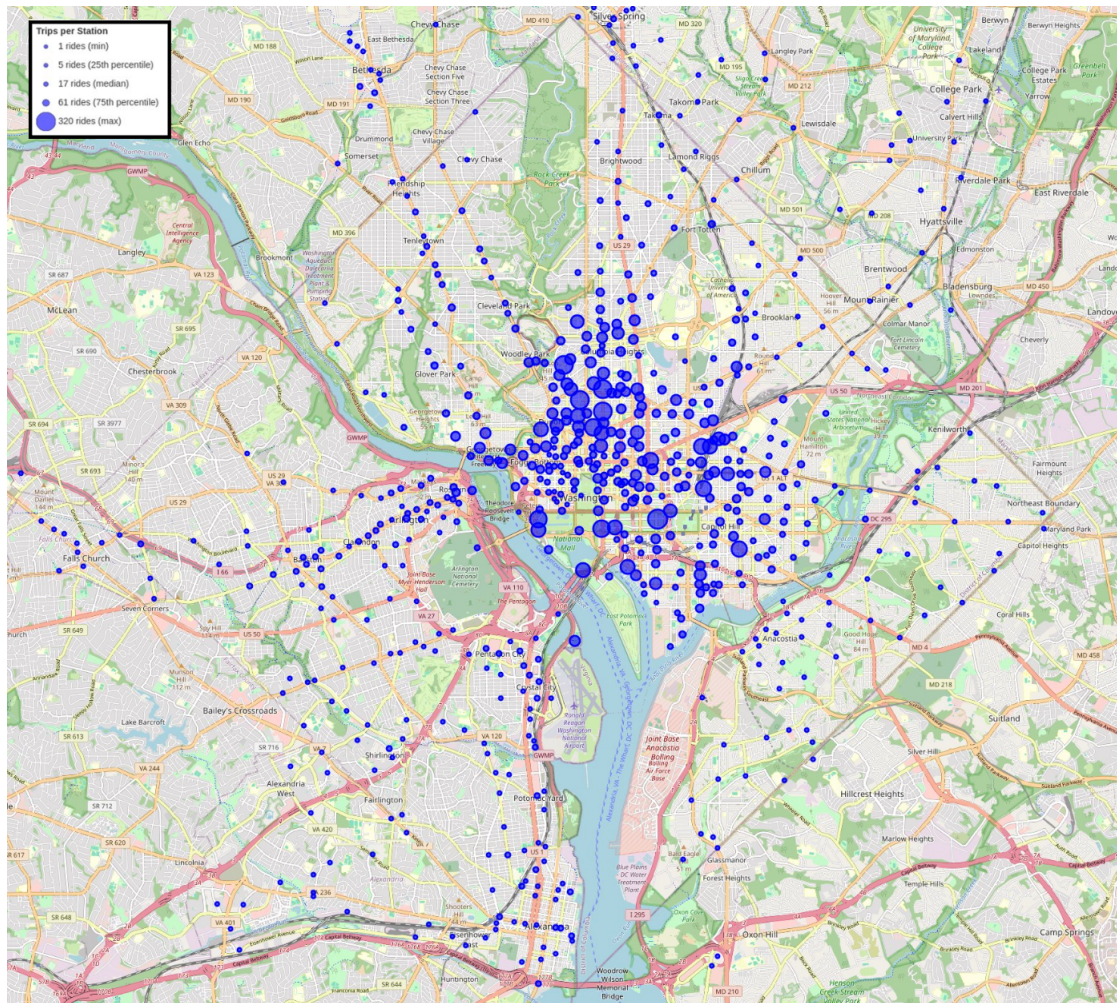


Figure 3.4: Capital Bikeshare Station Trip Volumes on Saturday, March 15, 2025

significantly increases the number of affected nodes. In particular, many rides represent one-off trips between station pairs that do not recur across days. This results in an average of 4,517 added and 4,513 deleted edges per snapshot, leading to updates in the random walks of approximately 99% of nodes per day. Although training remained feasible at this scale, such high churn may hinder scalability for larger datasets.

An alternative update policy based on changes in edge weights, rather than edge additions or deletions, may be more efficient. For this dataset, adopting a strategy based on weight change would reduce node updates by nearly half. Introducing a minimum weight change threshold could further decrease update frequency and improve training performance when speed is a priority.

To obtain graph-level embeddings from the node embeddings generated by DynNode2Vec, we apply a mean pooling strategy. For each daily snapshot, we average the embeddings of all nodes present in the graph to produce a fixed-length vector representation of the entire snapshot. This approach allows us to retain information about the overall structural and functional characteristics of the network, while remaining computationally efficient and easy to implement. Mean pooling also ensures that variations in graph size or density do not dominate the resulting representation, making it suitable for comparing graphs across time in a consistent embedding space.

Having constructed node-level embeddings for each daily snapshot, we next examine how well the model captures the evolving structure of the network over time. One way to assess this is by tracking the vocabulary size, the number of unique nodes encountered in random walks used for training. To monitor how the embedding model’s coverage evolves, we track the size of the vocabulary, the number of unique node identifiers observed in random walks, as a function of time in Figure 3.5. This vocabulary reflects the set of nodes that have been incorporated into the learned embedding space. As new stations appear or previously inactive ones reenter the system, the vocabulary grows accordingly.

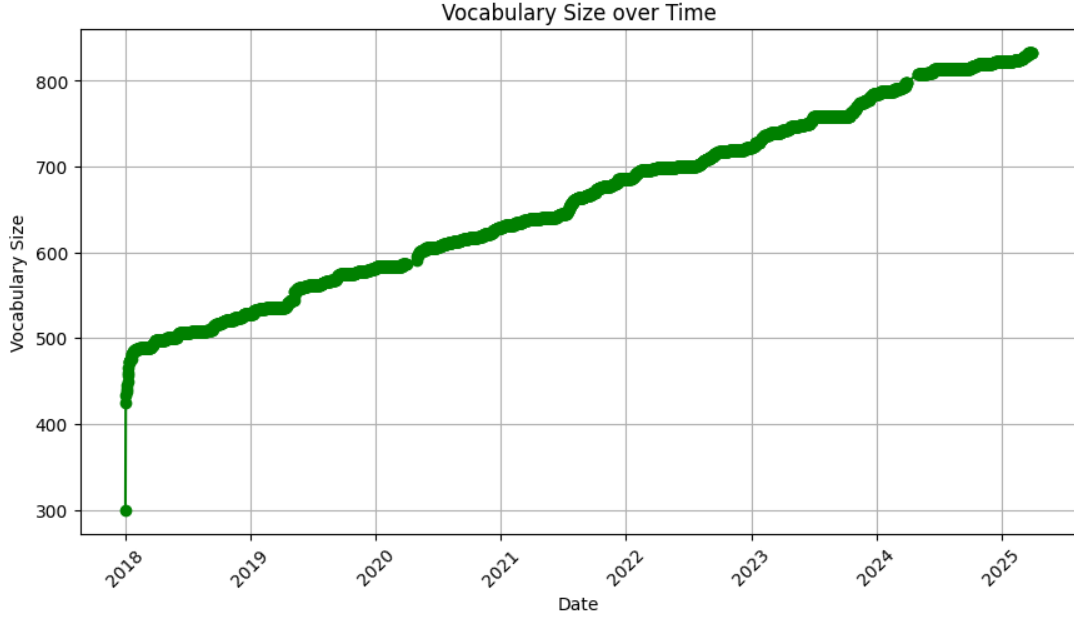


Figure 3.5: Vocabulary Size of DynNode2Vec Model on Capital Bikeshare

3.5.3 Embeddings

After training the DynNode2Vec model across all daily snapshots, we compute the final node embeddings for each snapshot using the fully trained model. These embeddings are then aggregated via mean pooling to produce a 128-dimensional vector representation per snapshot. When visualized as a heatmap in Figure 3.6, the resulting graph-level embeddings exhibit relatively little variation over time. This visual uniformity suggests that the the overall structural properties between snapshots, at least as captured by the embedding model, are relatively stable. However, since the heatmap aggregates all 128 dimensions and reflects stability across most dimensions, it may obscure more subtle temporal dynamics present in the data.

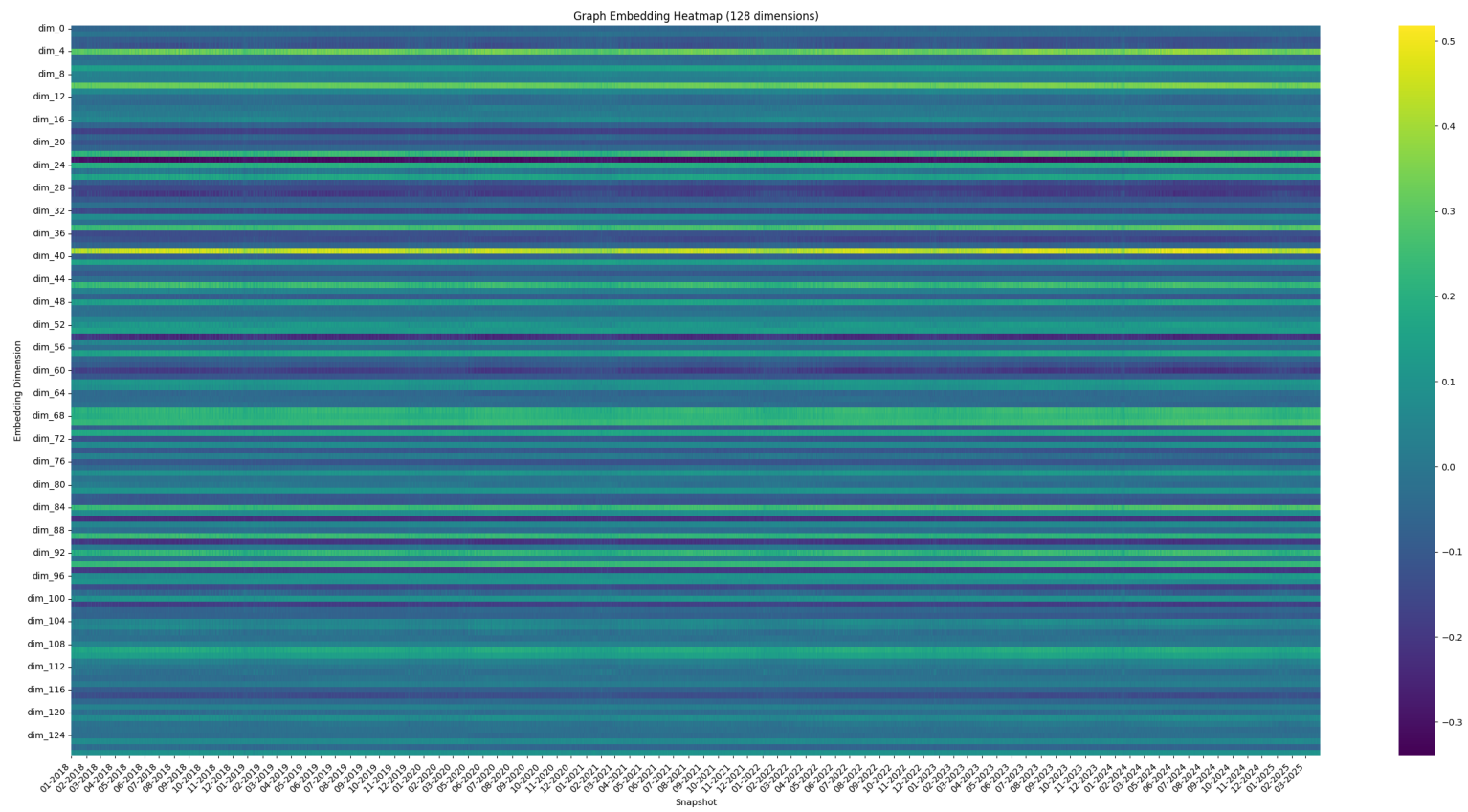


Figure 3.6: DynNode2Vec Graph Embedding Heatmap

To explore potential latent structure in the embedding space, we apply dimensionality reduction using t-SNE and project the 128-dimensional embeddings into three dimensions. As shown in the 3D t-SNE plot in Figure 3.7, no sharply separated clusters or anomalous groupings are apparent. Still, broader regions emerge where snapshots from similar time periods tend to cluster, and the embeddings exhibit a smooth temporal progression across the plot. These gradual transitions and loosely defined groupings likely reflect sustained network conditions or seasonal patterns that influence the structure of the bikeshare system over extended periods. That said, it is important to keep in mind that projecting high-dimensional data into three dimensions can distort both distances and spatial relationships. The t-SNE plot should therefore be interpreted with caution as it may not fully preserve the geometric structure of the original embedding space.

3.5.4 Multidimensional Matrix Profile

Figure 3.8 presents the Multidimensional Matrix Profile (MMP) computed on the DynNode2Vec graph embeddings for the Capital Bikeshare dataset, using a subsequence length of 30 days. The MMP captures, for each snapshot window, the z-normalized Euclidean distance to its nearest neighbor window in the embedding space. This profile provides a powerful lens for exploring dynamic network behavior, enabling a range of data mining tasks such as motif discovery, anomaly detection, and regime segmentation. Low MMP values indicate that a given snapshot sequence closely resembles another in the dataset, while high values suggest dissimilarity and potential anomalies. In time series analysis, these are commonly referred to as motifs and discords, respectively. We annotate the global minimum or motif which we will use as a reference query for further analysis.

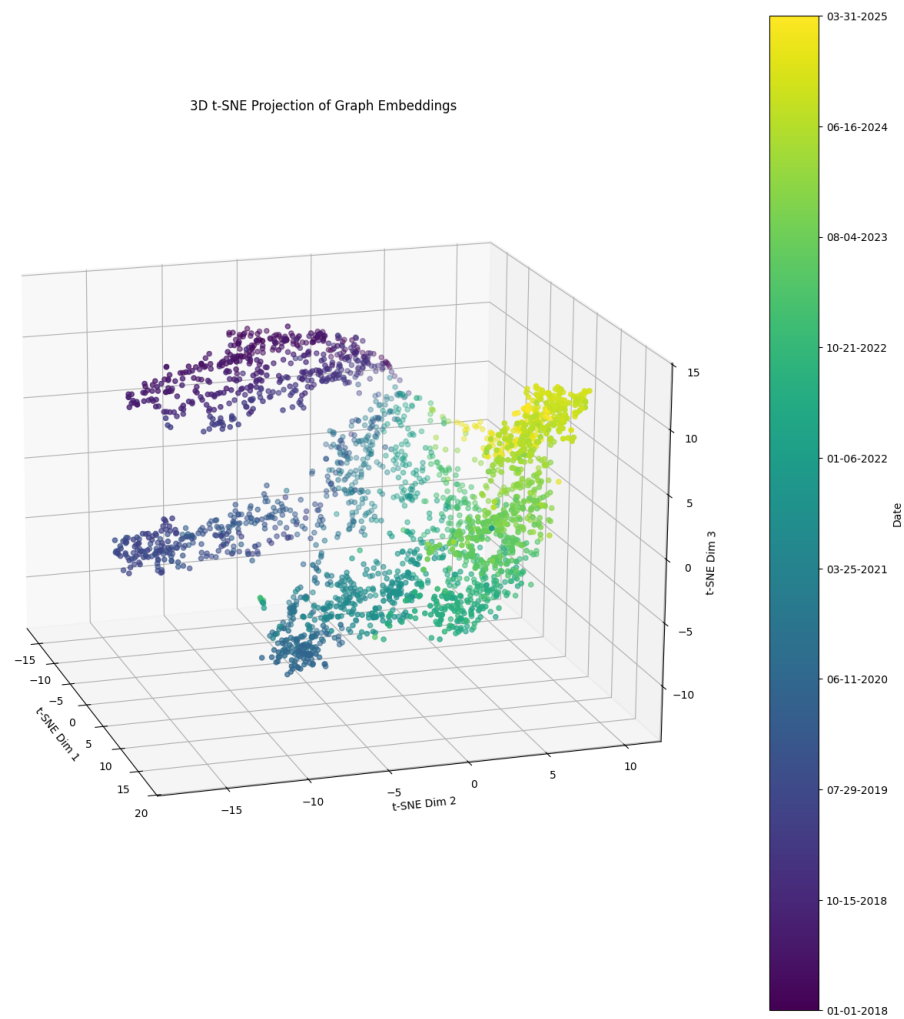


Figure 3.7: 3D t-SNE Plot of Graph Embedddings

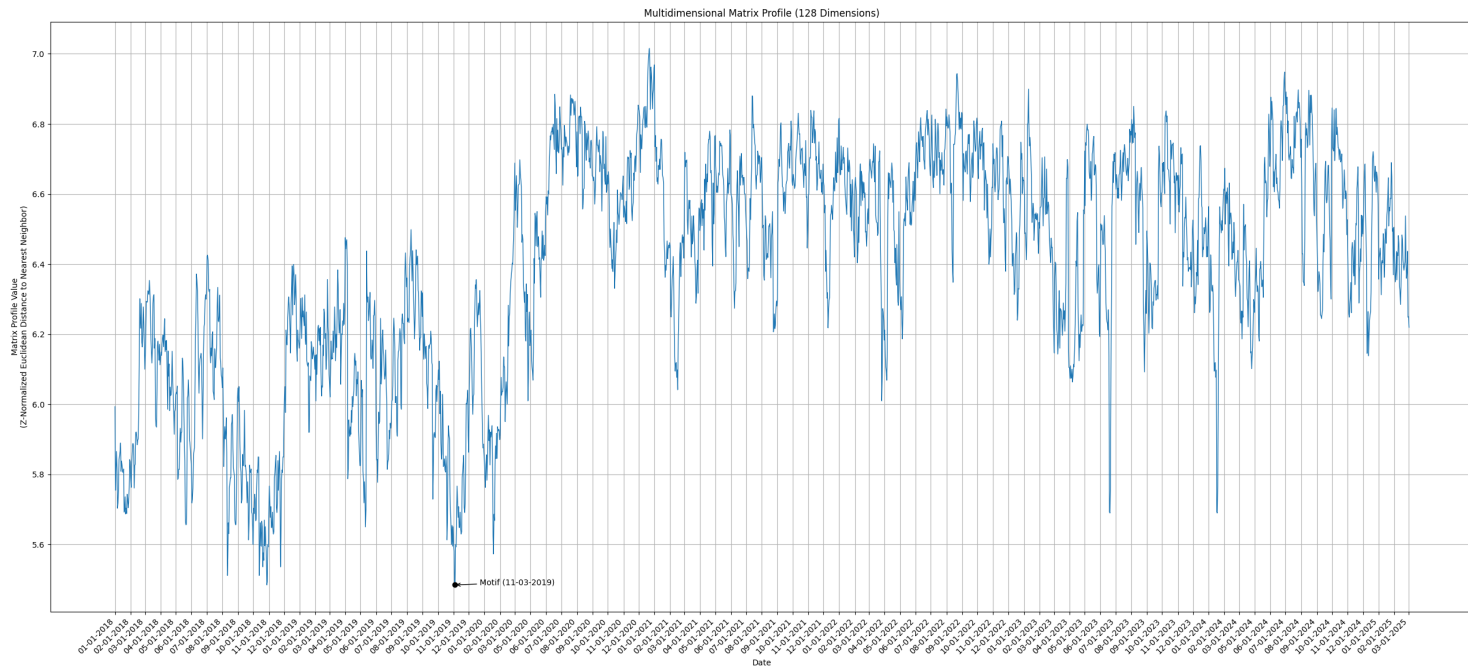


Figure 3.8: Multidimensional Matrix Profile on DynNode2Vec Graph Embeddings

To more explicitly illustrate how each value in the MMP is computed, we present the full distance profile for the labeled motif in Figure 3.9*. The distance profile represents the distance between a fixed query subsequence and all other candidate subsequences in the dataset. The black dashed line indicates the location of the query, and the surrounding region with missing values corresponds to the exclusion zone described in Section 3.4. This zone prevents trivial self-matches by ignoring comparisons too close in time. The green dashed line marks the nearest neighbor match, which corresponds to the minimum distance in the profile. This single value is what is retained and stored in the MMP at the query’s position (as shown in Figure 3.8).

*Reference sequences are commonly referred to a *query* and their nearest neighbor is commonly referred to as a *match*.

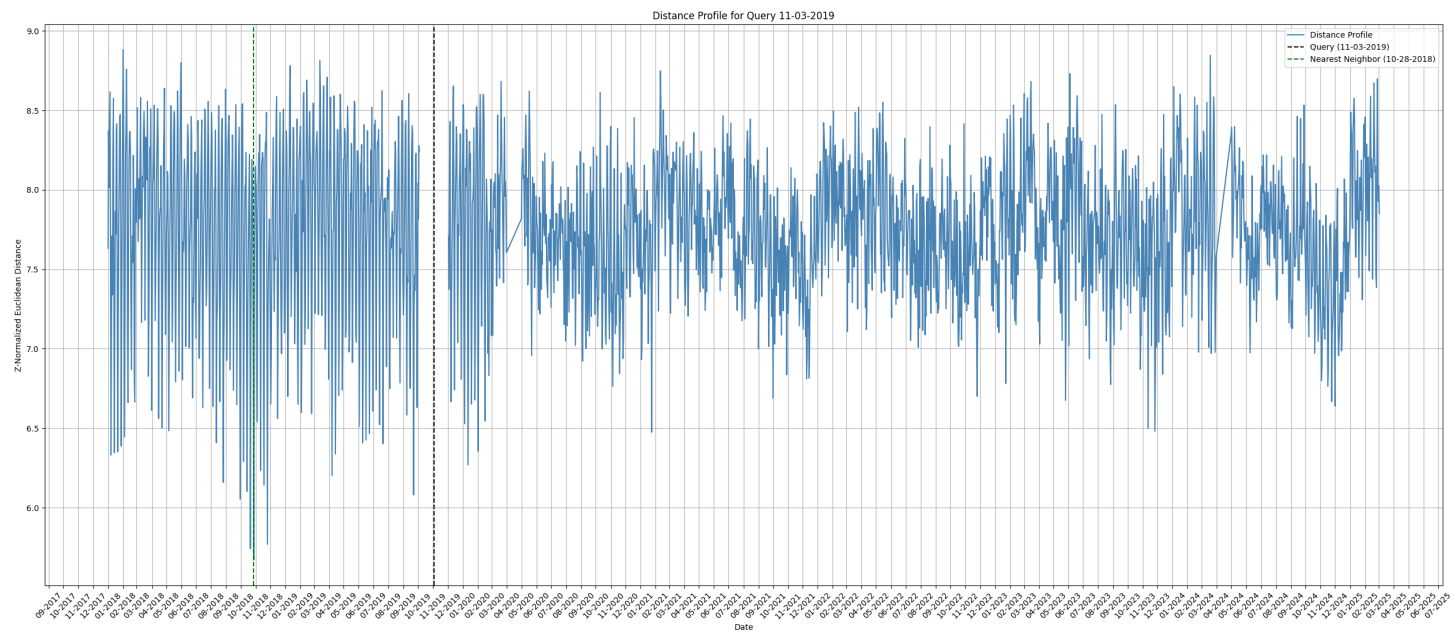


Figure 3.9: Distance Profile for Query

Thus far, we have presented the output of the Dynamic Network Matrix Profile (DNMP), but we must also assess whether applying the MMP to DynNode2Vec embeddings yields meaningful results. To evaluate the quality of embeddings, we analyze the evolution of structural graph metrics over time. Our objective is to demonstrate that the DNMP identifies genuinely similar periods in the dynamic network and captures significant structural patterns.

Measuring similarity between sequences of dynamic graphs is inherently challenging due to their complex evolving structure. A practical way to assess this similarity is by examining how key graph metrics—such as edge density, clustering coefficient, and component sizes—evolve over the life of a subsequence. In Figure 3.10, we plot the graph metrics for the motif (i.e. the query subsequence referenced in Figure 3.9 and annotated in Figure 3.8) and its nearest neighbor match, over the 30-day window used in DNMP. The query is shown in blue, and its nearest neighbor in orange. The strong alignment in metric trends across these sequences supports the claim that our embedding-based similarity is capturing meaningful structural relationships.

To further validate this, we compare the same query to its median and worst matches in Figure 3.11. The median match (orange) and worst match (green) deviate substantially from the query (blue), with particularly noticeable divergence in the worst match. This contrast reinforces our confidence that the embedding space induced by DynNode2Vec and used in DNMP reflects a meaningful notion of structural similarity across temporal snapshots.

As an additional perspective on how dynamic network sequences align over time, we examine the confusion matrix of nearest neighbor matches derived from the MMP, shown in Table 3.1. This matrix compares the weekday of a query sequence to the weekday of its nearest neighbor match. We observe strong temporal alignment: 47.03% of nearest neighbors occur on the same weekday, substantially higher than the random chance of $1/7$. This result is consistent with the nature of urban bikeshare usage, which tends to follow weekly commuting and activity patterns. For example, weekday rider behavior is often shaped by work-related travel, while weekends may

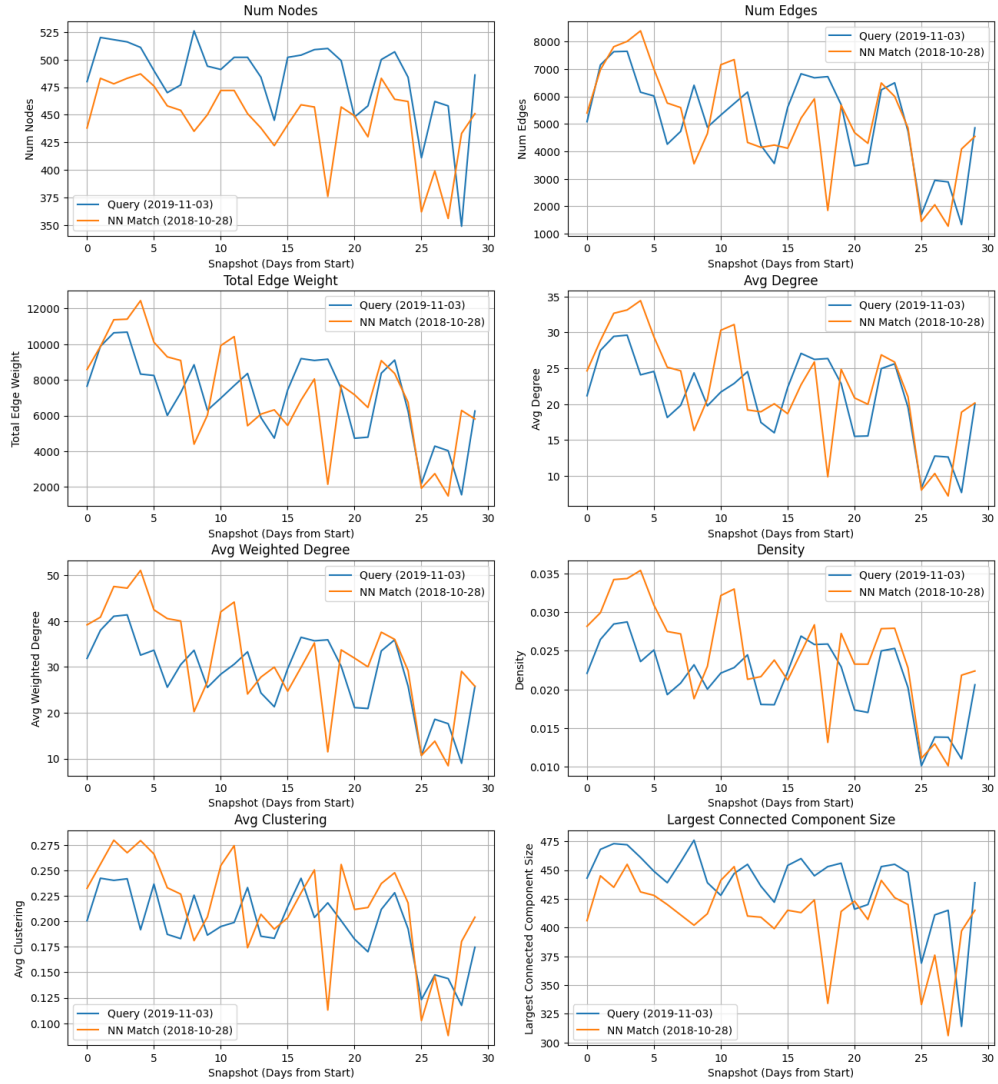


Figure 3.10: Snapshot Graph Metrics of Capital Bikeshare Motif

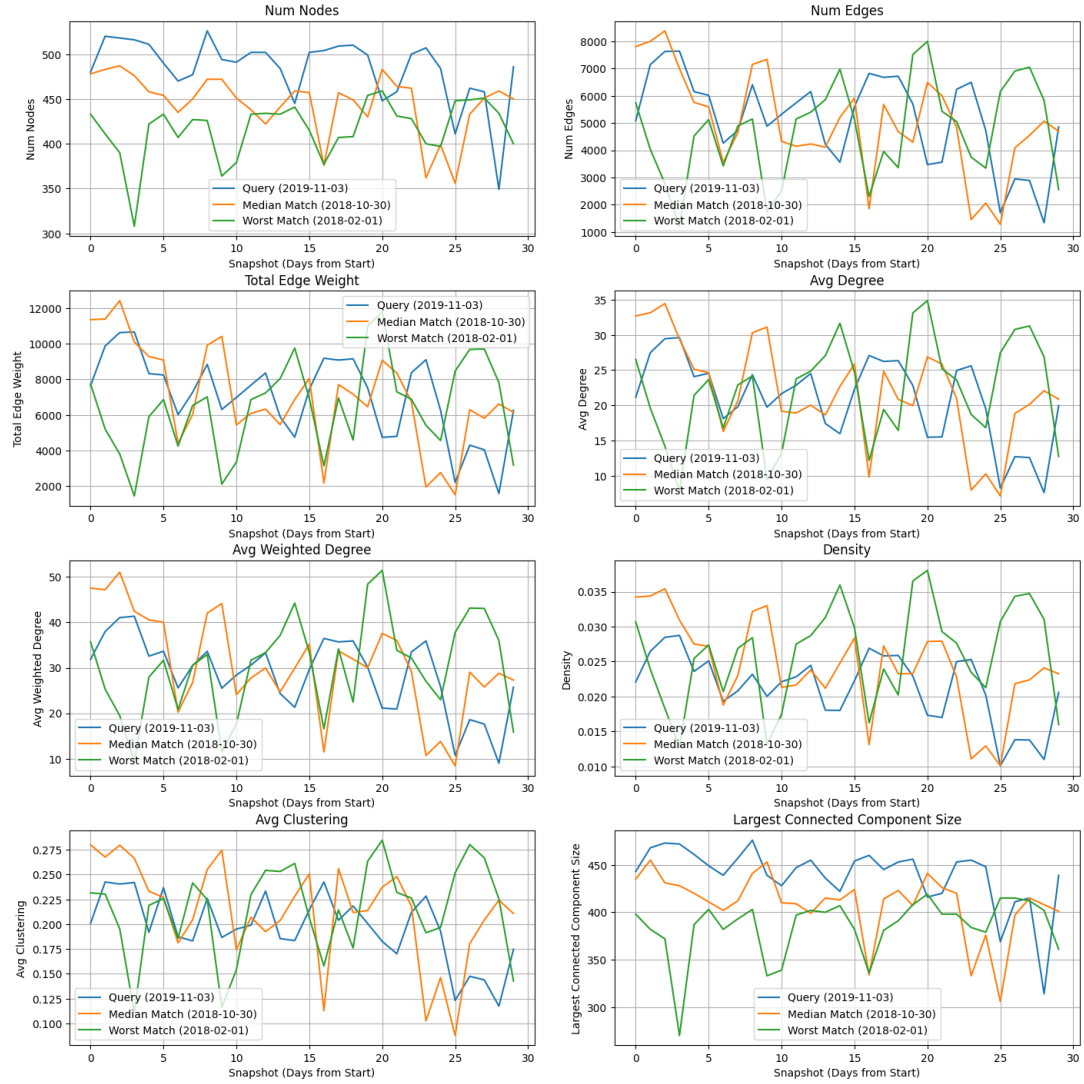


Figure 3.11: Query's Median and Worst Match Snapshot Graph Metrics

exhibit different usage driven by leisure or recreational trips. Matches are also more likely between adjacent weekdays, reflecting the likelihood that rider behavior persists across nearby days.

Table 3.1: Nearest Neighbor Weekday Confusion Matrix

Query	Match						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Mon	173	41	45	25	22	28	31
Tue	31	170	50	34	27	29	24
Wed	34	43	161	49	35	20	23
Thu	21	26	35	178	42	44	19
Fri	24	25	21	38	174	41	43
Sat	39	24	21	21	44	179	38
Sun	37	44	22	27	26	42	168

The DNMP framework enables more than just motif discovery and anomaly detection; it also supports other data mining tasks such as semantic segmentation and change point detection. Similar to its application in time series analysis in Chapter 2, the MMP can be used to segment dynamic networks by identifying structural shifts over time. Figure 3.12 displays the Corrected Arc Curve (CAC) derived from the DNMP of the Capital Bikeshare dataset. Each point in the CAC represents a normalized count of arcs, connections between a query index and its nearest neighbor, that cross over that index[†]. High arc densities typically occur in periods of behavioral stability, where snapshots have nearest neighbors in close temporal proximity. Conversely, valleys in the CAC indicate reduced arc crossings and suggest shifts in network behavior, making them useful candidates for change points or segmentation boundaries.

In the Capital Bikeshare CAC, a prominent dip appears in March 2020, signaling a significant behavioral transition. This corresponds to the onset of the COVID-19 pandemic in the United States, when stay-at-home orders and remote work policies disrupted regular commuting patterns. The structural change in rider behavior

[†]See Section 2.4.1 for a detailed discussion on arc curve normalization.

captured by this low CAC value highlights the utility of DNMP for detecting meaningful real-world transitions in dynamic networks.

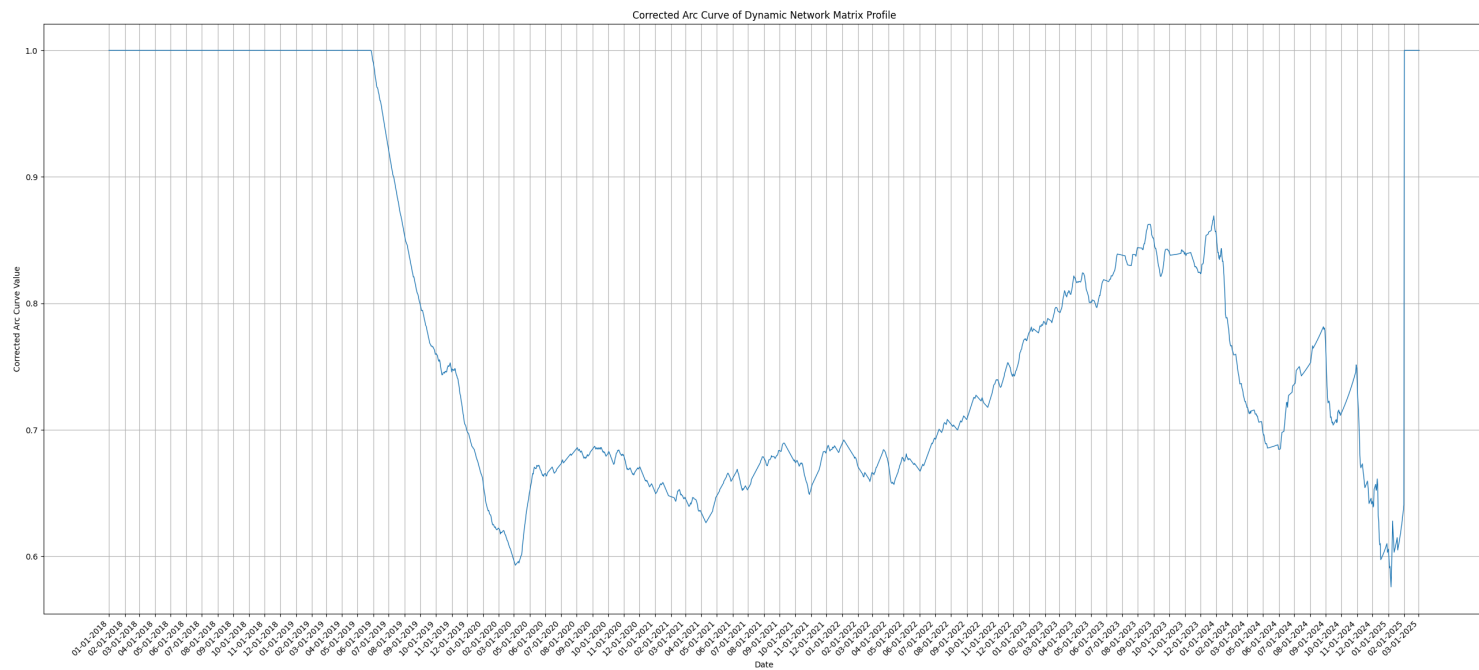


Figure 3.12: Corrected Arc Curve of Capital Bikeshare DNMP

3.6 Atlas Coffee Trade Results

3.6.1 Data

The Atlas dataset provides detailed international trade information, documenting over 6,000 products exchanged among approximately 250 countries and territories from 1962 to 2023 [5]. Derived from United Nations Comtrade data and curated by the Harvard Growth Lab, the dataset is cleaned and standardized for open research use. In this study, we focus on the Standard International Trade Classification (SITC) variant, Atlas’s most extensive and historically complete version, and restrict our analysis to SITC category “07,” which includes coffee, tea, cocoa, spices, and related manufactured goods. To analyze temporal patterns in global trade, we construct a sequence of yearly snapshot graphs in which nodes represent countries or territories and directed edges capture the value of exports from one country to another for category “07” products. Edge weights correspond to the reported export value in U.S. dollars for each year, producing a directed weighted dynamic network that evolves annually and supports the study of long-term structural changes in global trade for coffee and related commodities.

3.6.2 Model and Embeddings

As with the Capital Bikeshare analysis (Section 3.5), we train a weighted variant of the DynNode2Vec [86] model to generate yearly graph snapshot embeddings for the trade network. Node-level embeddings are aggregated using mean pooling to produce a fixed-length vector representation for each year.

Figure 3.13 illustrates the evolution of the model’s vocabulary size over time, where vocabulary refers to the number of unique geographic entities participating in trade. Notable increases occur in 1970, 1992, and 2000, corresponding to expansions in the set of countries and territories reporting trade data to the United Nations. Importantly, our downstream Matrix Profile results remain stable despite these shifts,

suggesting that the method is robust to changes in node count. This robustness is likely a result of the mean pooling strategy, which normalizes across nodes and emphasizes structural trends rather than raw graph size.

Similar to the embeddings discussed in Section 3.5.3, the heatmap of yearly trade embeddings exhibits limited visual variation. However, unlike the Capital Bikeshare low-dimensional projection, the PCA plot of the trade embeddings in Figure 3.14 reveals more distinct clustering, suggesting stronger temporal structure and separability in the trade network dynamics.

To highlight distinct temporal regimes in the embedding space, we applied k -means clustering ($k = 5$) to the PCA-reduced trade network embeddings. The resulting clusters, summarized in Table 3.2, reveal that most groupings correspond to contiguous ranges of years.

3.6.3 Multidimensional Matrix Profile

Figure 3.15 presents the Corrected Arc Curve (CAC) of the DNMP with a window size of 10 years. A prominent global minimum appears in 1976, indicating a potential change point in global coffee trade dynamics. This timing aligns with the aftermath of the 1975 “Black Frost” in Brazil, which severely damaged coffee crops and led to a global supply shortage [107]. At the time, Brazil accounted for 13.45% of the world’s coffee exports by value [5], and the resulting supply disruption drove a sharp increase in coffee prices the following year.

Cluster	Years
1	1962–1964, 1976
2	1965–1969
3	1970–1975, 1977–1991
4	1992–1999
5	2000–2023

Table 3.2: Yearly Snapshot Clusters from Trade Network Embeddings

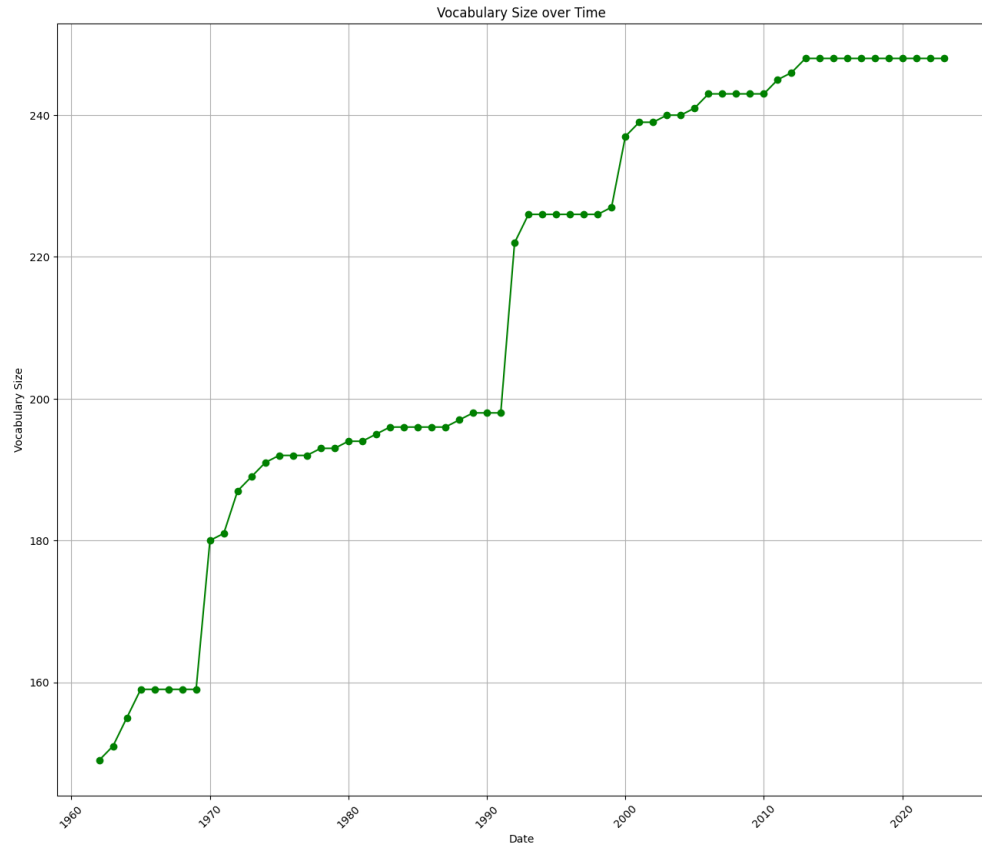


Figure 3.13: Vocabulary Size of DynNode2Vec Model on Coffee Trade

The minimum in 2001 indicates another change point in the dynamics of world coffee trade. This aligns with the well-documented global coffee crisis of 2001 [117, 100]. The crisis was driven by a significant oversupply of coffee, particularly from Vietnam, which expanded production rapidly after receiving development assistance and World Bank support [7]. During the late 1990s, global coffee production outpaced demand growth, while the collapse of the International Coffee Agreement in 1989 meant there were no export quotas to stabilize prices [118]. This oversupply led to

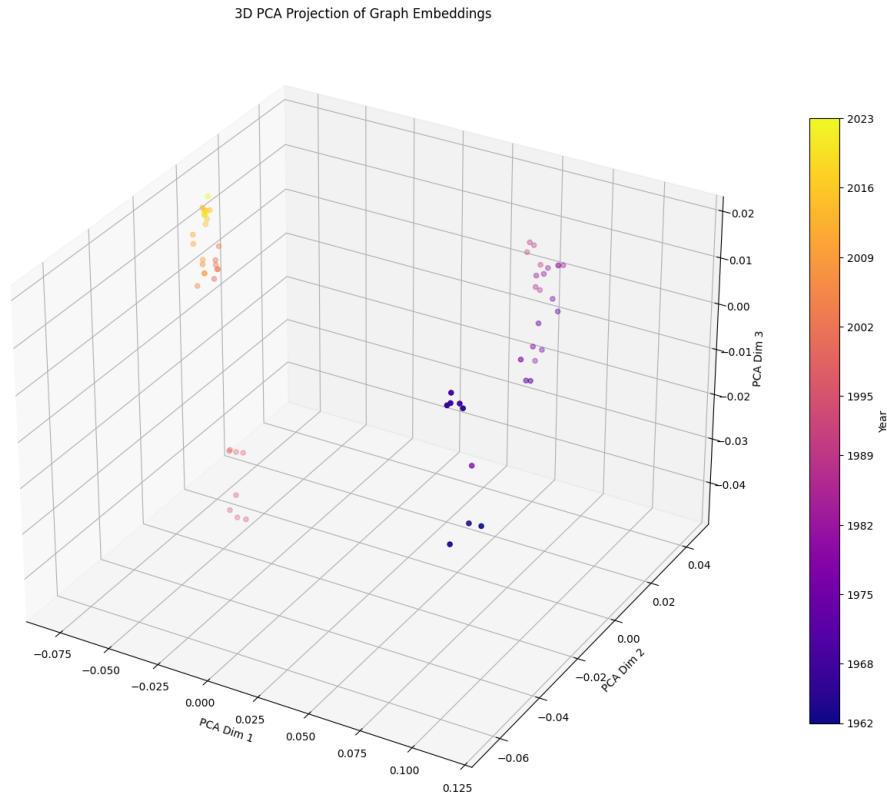


Figure 3.14: 3D PCA Plot of Graph Embeddings on Coffee Trade

a steady decline in international prices, reaching historic lows in 2001 and below the cost of production for many smallholder farmers [100]. As a result, coffee farming became unprofitable for millions of producers, causing widespread economic distress, abandonment of farms, and even destruction of surplus coffee [79].

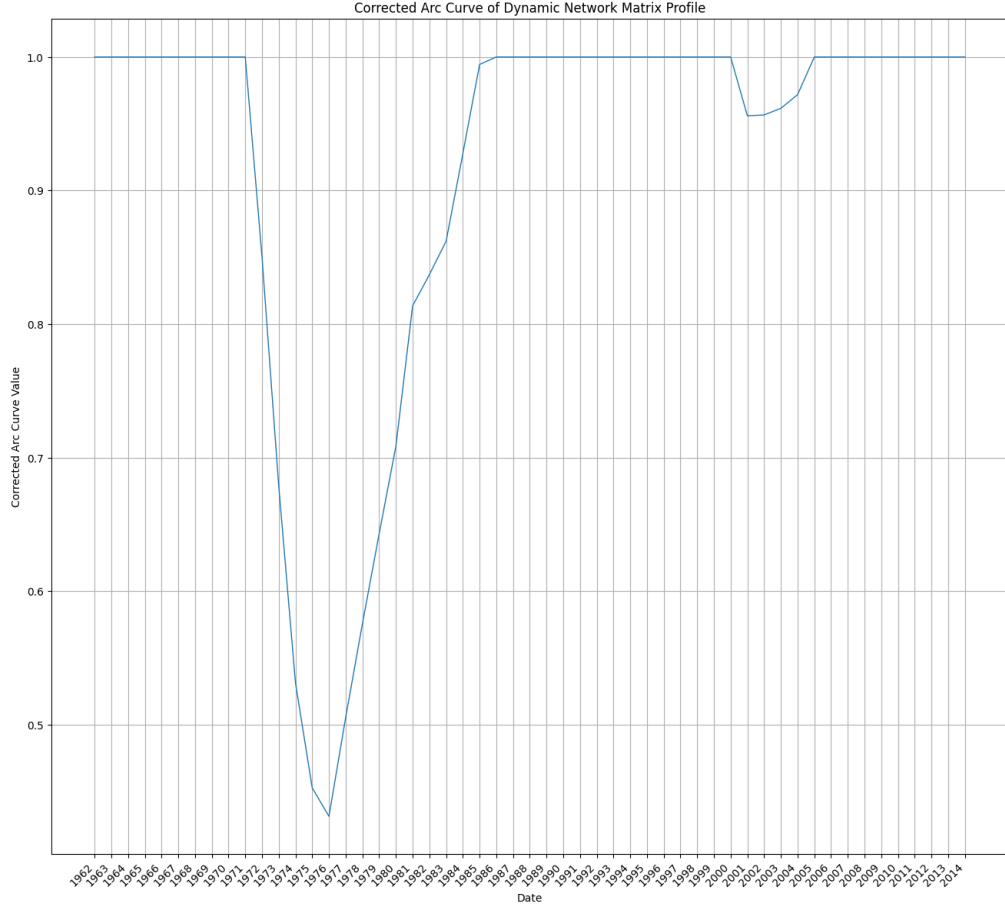


Figure 3.15: Corrected Arc Curve of Coffee Trade DNMP

3.7 Limitations

A fundamental limitation of our approach lies in its reliance on node embeddings that abstract away specific node identities. In our model, the embeddings capture structural similarity at the graph level, but do not preserve who occupies which position in the network. As a result, if nodes swap roles between snapshots, such as one country becoming the major exporter while another takes over as the

major importer, the overall graph embedding may remain nearly unchanged despite meaningful underlying changes.

This limitation is can be relevant in applications like trade networks, where the identity of specific nodes (i.e. countries) carries critical meaning. While our approach effectively detects broad structural similarities and changes over time, it may overlook localized or role-based shifts in node behavior. In other words, the method prioritizes global structural patterns over node-level dynamics, and is therefore less suited for analyses where tracking specific entities across time is essential.

Future work could address this limitation by incorporating node identity information directly into the embedding process, or by designing hybrid models that jointly consider both structural similarity and the persistence or change of individual node roles over time.

3.8 Conclusion

In this chapter, we introduced the Dynamic Network Matrix Profile (DNMP), a novel extension of the Matrix Profile framework to dynamic graph analysis. By combining graph embeddings with Multidimensional Matrix Profile techniques, DNMP enables scalable, temporally-aware similarity search over evolving network structures. Applied to the Capital Bikeshare dataset, our approach successfully uncovered temporal motifs, anomalies, and structural transitions across over 2,500 daily network snapshots. The Multidimensional Matrix Profile revealed consistent motifs, strong temporal alignment, and clear regime changes—most notably the sharp shift in March 2020 corresponding to the onset of the COVID-19 pandemic. We extended this analysis to global trade data on coffee and related commodities, constructing yearly snapshot graphs spanning over six decades. In this setting, DNMP identified a prominent change point in 1976, coinciding with a global coffee shortage caused by the 1975 “Black Frost” in Brazil, a major exporter. This result further demonstrates DNMP’s ability to detect significant real-world transitions in dynamic

networks. Taken together, these findings validate DNMP as a general-purpose tool for time-aware graph mining, enabling motif discovery, semantic segmentation, and change point detection in evolving network data. More broadly, this work highlights the versatility of the Matrix Profile paradigm and its applicability beyond traditional time series, offering a new lens through which to study temporal structure in complex network data.

3.9 Future Work

This work opens several promising directions for extending the DNMP framework and improving its scalability, flexibility, and interpretability in dynamic network analysis.

One avenue for improvement lies in the scalability of the DynNode2Vec training process. In its current form, the model updates random walks for all nodes incident to any added or removed edge, which can become computationally expensive in high-churn networks such as Capital Bikeshare. Adapting the update policy—for example, by triggering walk regeneration only for nodes experiencing significant edge weight changes—could reduce training overhead without sacrificing embedding quality.

In addition to improving training efficiency, future work could explore the integration of feature selection mechanisms into the embedding pipeline. While our approach leverages embedding-based similarity, it may be beneficial to identify and emphasize embedding dimensions that correlate with structural properties derived from traditional graph distance metrics. This could help disentangle which aspects of the graph structure contribute most to the observed similarities in the DNMP.

Because we do not rely on a supervised downstream task, we are unable to directly train attention mechanisms to learn pooling weights [150]. However, exploring unsupervised heuristics—such as degree, betweenness centrality, or PageRank—as proxies for attention-based pooling could improve representation quality. This may be particularly impactful in settings like bikesharing networks, where a small number of central stations disproportionately influence overall flow patterns. Mean pooling

treats all nodes equally, which can cause smaller, peripheral stations to exert undue influence over the graph-level embedding. Weighting node embeddings by importance could yield more faithful representations of the network’s operational structure.

An alternative to weighted embeddings is to simply reduce the influence of uninformative dimensions altogether. The k -dimensional Multidimensional Matrix Profile provides a principled way to select the most relevant subset of embedding dimensions for motif and discord detection [167]. Using only the top- k dimensions may improve robustness to noise and enhance the ability to detect meaningful patterns in high-dimensional embedding spaces.

The Matrix Profile framework also supports heterogeneous similarity joins across different time series. Extending this idea to dynamic networks could allow us to identify similar subsequences between two different dynamic graphs—for example, comparing bikeshare usage patterns across cities. This would require consistent or aligned embedding spaces across datasets, but could offer new insights into the comparative evolution of coevolving dynamic networks.

Finally, while the DNMP currently identifies motifs based on the best-matching subsequence pair, recent work in the Matrix Profile literature has introduced methods for discovering motif groups via index linking [18]. Incorporating group motif discovery could enhance the interpretability of recurring network states and reveal richer patterns in dynamic graph behavior.

3.10 Author Contributions

Evan Ezell conceptualized the idea of a Matrix Profile for dynamic networks. Evan Ezell implemented the code. Evan Ezell trained the node embeddings model. Evan Ezell wrote the manuscript. Seung-Hwan Lim oversaw the work, proofread, and edited the final manuscript.

Chapter 4

Community Fabric: Visualizing Communities and Structure in Dynamic Networks

4.1 Abstract

We present Community Fabric, a novel visualization technique for simultaneously displaying communities and structure within dynamic networks. In dynamic networks, the structure of the network is continuously evolving throughout time and these underlying topological shifts tend to lead to communal changes. Community Fabric helps the viewer more easily interpret and understand the interplay of structural change and community evolution in dynamic graphs. To achieve this, we take a new approach, hybridizing two popular network and community visualizations. Community Fabric combines the likes of the Biofabric static network visualization method with traditional community alluvial flow diagrams to visualize communities in a dynamic network while also displaying the underlying network structure. Our approach improves upon existing state-of-the-art techniques in several key areas. We describe the methodologies of Community Fabric, implement the visualization using modern web-based tools, and apply our approach to three example data sets.

4.2 Introduction

Community detection is a major branch of studies in the understanding of static and dynamic networks. A network can be decomposed of its communities or substructures, where these substructures are made up of nodes with similar relationships and characteristics. Thus, in a dynamic network, the structural decomposition may change over time. In order to understand this change, we must first find the structures, through optimization algorithms or heuristics, and connect them across the network. Algorithmically detecting structures and connecting them in dynamic networks can be performed independently or dependently of the change occurring. Rosetti et. al. provide an excellent survey in various community detection techniques for dynamic networks [121]. By partitioning the nodes into related groups and tracking different

characteristics of these groups, we can empirically learn through the visual analysis of the evolutionary patterns of communities in a network.

In this paper, we deal with techniques in the visual analysis of communities in dynamic networks. The visual analysis of network communities is important because it allows us to verify community detection methods and find patterns in data that we otherwise might not observe.

Currently, there is a lack of visualization techniques which display communities in dynamic networks [121]. Displaying communities in dynamic networks is a challenging task because an effective visualization technique must integrate evolving community representation with the difficulties of dynamic networks [121]. Beck et al. provides a comprehensive survey of dynamic graph visualization techniques [9] while Vehlow et al. narrows in on specific approaches to visualizing communities in dynamic networks [151].

Communities in dynamic networks evolve and exhibit different behavioral actions through time. These community actions: birth, death, growth, contraction, merging, splitting, continuity, and resurgence [101, 21, 20] are fundamentally driven by changes in the underlying network topology. While it is interesting to visualize community evolution and network topology on their own, there is often a need to visualize each in coordination together to further realize network characteristics. Visualizing the structure of networks in addition to community evolution is important in understanding overall network dynamics and the concrete underpinnings for specific changes in individual communities. We present a novel visualization technique for viewing the changes in communities while simultaneously visualizing their relationship with the underlying network structure of a dynamic graph.

Figure 4.1 shows the current state-of-the-art techniques in visualizing communities and structure in dynamic networks. Juxtaposed node-link diagrams with nodes colored by their community (see Figure 1a) show the structure of the dynamic network at each time step. The disadvantage to this approach is the structure is hard to understand if the network is too large. Also, it is hard to track how individual

nodes and communities change when the network has a significant number of time steps. Community alluvial flow diagrams (see Figure 1b), similar to flowcharts, show how communities change over time in a dynamic network [123]. Alluvial diagrams represent the change across a dynamic network as a continuous flow using colors to track different network partitions across time and smoothly transitioning the colors when the partitions change. In the case of community alluvial diagrams, the partitions are communities where each color represents a distinct community. A community alluvial diagram is better at tracking community actions when a dynamic network has a larger number of time steps and provides an excellent at a glance view of a network. However, this approach does not show the underlying structure of the network. This can be an issue when the user needs to verify if a particular community detection algorithm is selecting reasonable partitions of the network. Also, it does not allow the user to investigate how a particular node/community may have changed across time steps. Addressing these concerns, [151] adjoined node-link diagrams to each community at each time step of the alluvial flow community diagram (see Figure 1c). This approach preserved the aforementioned benefits of the alluvial flow community diagram while adding the display of structure to the visualization. To the best of our knowledge, this approach is the current state-of-the-art in visualizing both community evolution and the structure of a dynamic network simultaneously.

While alluvial flow community diagrams with adjoined node-link diagrams are the current state-of-the-art technique, still limitations exist. Node-link diagrams typically do not scale well to large networks, creating a “hairball” effect of edge crossings and node placement. This will likely cause the observer to be overwhelmed, especially in cases when individual communities are large. Using adjoined node-link diagrams beside communities requires the use of limited force-directed layouts by restricting the horizontal axis on which a node can appear, since nodes must be placed collinearly to their respective position in the flow diagram. This may create suboptimal node positions leading to a less interpretable node-link diagram. Having individual node-link diagrams for each respective community makes it difficult to

show inter-community edges because each node-link diagram’s layout is oblivious to other communities. This makes connecting all of the nodes with relationships between communities difficult without cluttering the visualization. This is especially apparent in such cases where the network is dense or has a large number of inter-community edges. The current solution to remedy this issue is by displaying inter-community edges via overlaid arcs or curved links separately from the node-link diagrams [151]. While these curved links allow the user to see inter-community relationships, they are viewed orthogonally to intra-community edges. This may create visual dissonance between the interpretation of inter-community edges and intra-community edges.

Our proposed technique, Community Fabric, attempts to alleviate these drawbacks by employing a novel strategy for simultaneously visualizing communities and structure within dynamic networks in an offline manner. The Community Fabric visualization technique superimposes a static Biofabric-like diagram [85] atop each time step of a traditional alluvial community flow diagram. This technique improves upon existing work by allowing the visualization of communities in larger networks all while preserving the awareness of the underlying structure of the graph topology. By using the Biofabric technique to visualize structure, we remove the difficulty of dealing with restrictive node positioning algorithms for node-link diagrams. We are able to naturally leverage advantages of the Biofabric and alluvial flow diagram techniques because both approaches fix nodes to horizontal lines. Community Fabric consolidates visualizing inter-community edges and intra-community edges by displaying all types of edges in a congruent manner, eliminating the need to employ two different strategies to visualize each edge type, creating a more cohesive display of network characteristics. This new visualization approach allows researchers to more adequately verify different community detection algorithms, locate noisy nodes and anomalous data, and find useful trends and patterns in the dynamics of an evolving network.

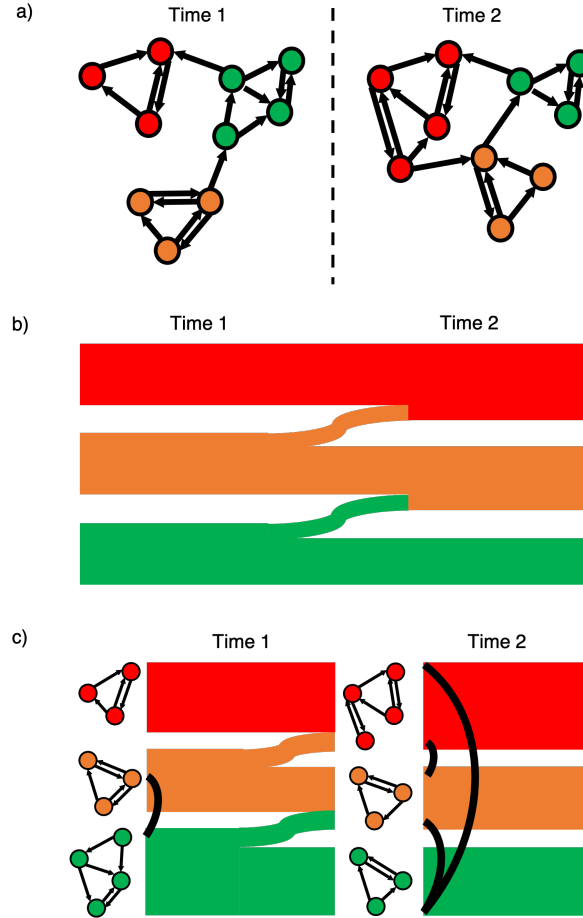


Figure 4.1: Current state-of-the-art for visualizing communities in dynamic networks on an example network. Figure a is juxtaposed node-link diagrams with the nodes colored by their respective communities. Figure b is a community alluvial flow diagram. Figure c is a community alluvial flow diagram with adjointed node-link diagrams.

4.3 Methods

The Community Fabric visualization technique merges the Biofabric diagram, a static structural network visualization approach, with the alluvial flow community diagram for dynamic networks. We chose to use the Biofabric diagram to show structure because of its scalability advantages over a traditional node-link diagram and its compatibility with the alluvial flow community diagram.

The Biofabric diagram is a static network visualization technique which aims to eliminate the sporadic spaghetti like edge crossings, also known as the “hairball” effect, resulting from a node-link diagram displaying a large network. The Biofabric approach uniquely assigns nodes to horizontal lines or rows and assigns edges to vertical lines creating a separate axis for each component. This contrasting placement of nodes and edges produces a large number of orthogonal edge crossings but these crossings are much easier to follow due to consistency across the entire visualization. By ordering the nodes and edges functionally, the Biofabric visualization displays critical network characteristics through individual node edge wedges. Figure 4.2 contrasts the differences between a node-link diagram and the Biofabric visualization on the *Les Miserables*’ character network.

The alluvial flow community diagram depicts the change of nodes in and out of communities throughout a dynamic network. An alluvial flow diagram is superior to a juxtaposed colored node-link diagram for showing community evolution when the number of time steps of the network is large (see Figure 4.1). The alluvial flow approach easily shows community actions across the entire network. An example of the alluvial flow community diagram on a sample data set is presented in Figure 4.3. This is a very useful visualization technique if you are concerned about the overall understanding of the evolution of abstract structures or communities, but are not concerned with focus on the intricacies of local behavior near individual nodes.

By merging Biofabric and alluvial flow community diagrams, the Community Fabric visualization technique provides a new way to show the change in communities

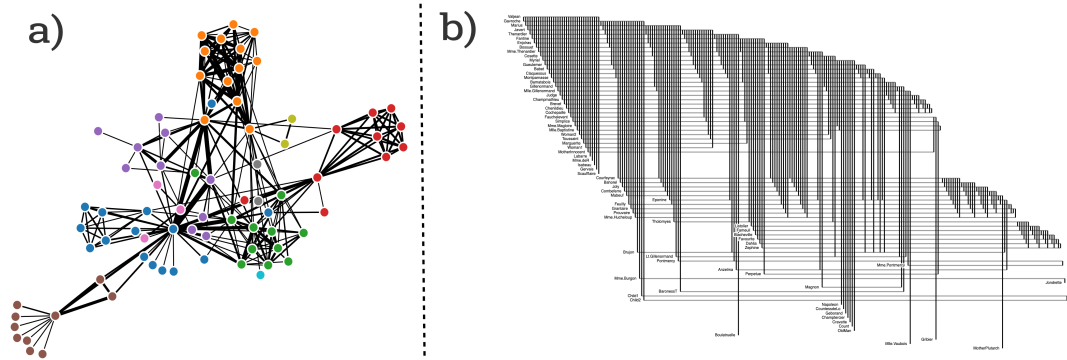


Figure 4.2: Node-link diagram versus Biofabric diagram of the *Les Misérables*' character network. Figure a is the node-link diagram and Figure b is the Biofabric diagram.

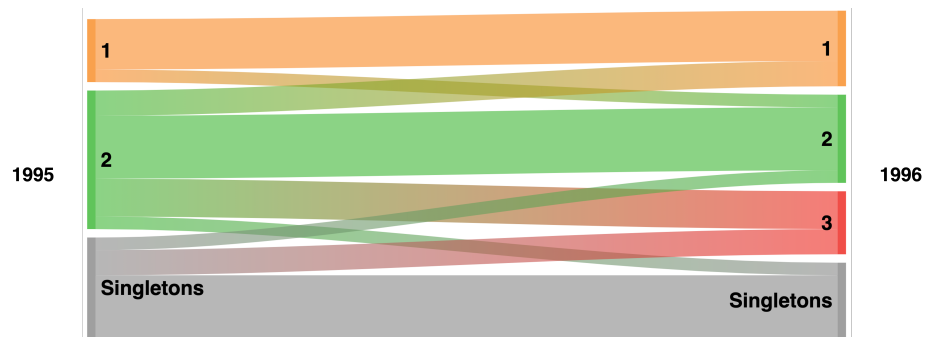


Figure 4.3: Alluvial flow community diagram.

of a dynamic network while simultaneously displaying the structure of the network. Community Fabric leverages the advantages of the two techniques, creating a visualization that is more well suited to understanding the network topology behind changes in communities across a dynamic network. Since both the Biofabric diagram and the alluvial flow community diagram fix nodes to rows, we are able to overlay the Biofabric diagram atop each flow time step in Community Fabric without significant manipulation. This is an advantage over using adjoined node-link diagrams when showing structure alongside community evolution.

The Community Fabric visualization technique is presented in Figure 4.4, showing the changing dynamics of a vehicle trade network from the ATLAS data set between two time steps. Each horizontal line is assigned a node, in this case a country. The vertical lines represent a vehicle trade with another country. For example, in the 1995 time step Cypress (CYP) exported vehicles to Israel (ISR). Communities are highlighted behind horizontal node lines with a unique color. In this particular example, we used the infomap community detection algorithm [122]; however, any non-overlapping community detection algorithm can be used with Community Fabric. Singleton nodes, nodes that do not have any edges or are not in a community, are shown at the bottom of the diagram without a colored underlay. This helps illuminate when nodes enter and exit participation in the network. For example, Guadeloupe (GLP) has imports in 1995 but does not have any imports or exports in 1996. The transition between time steps elicits how individual nodes are moving in and out of communities, displaying the changing network. A Community Fabric diagram can display any number of time steps in a dynamic network. It may be used to show a static network or single time step, change between sequential or non-sequential time steps, or all time steps of dynamic network. The Community Fabric diagram is suited for dynamic networks that have discrete time steps or can be aggregated into time steps across an interval.

In the design of Community Fabric, we drew on seminal research in visual perception psychology incorporating the principles of continuity, similarity, and

proximity [71]. We optimize node orderings to minimize the number of crossing edges between time steps, creating less visual disturbance making transitions easier to understand. We use smooth transition edge curves reducing the effort needed to follow individual nodes. We position community nodes adjacently and denote each community with a unique colored underlay.

In addition to using the above design principles, we have included the ability to interactively focus on specific nodes within the Community Fabric diagram (see Figure 4.13). By clicking a node’s corresponding horizontal line, it is brought into a focused state. The focused state has the following characteristics:

- The selected node line is highlighted so you can follow it throughout the dynamic network.
- The links incident on the selected node (i.e. in-links and out-links) are opaque whereas other links in the network are transparent.
- The selected node’s community is opaque at each time step whereas other communities are transparent.

This helps practitioners zero in on a specific node to understand its role within the overall network.

4.3.1 Crossing Minimization

Below, we discuss the formal method for minimizing the transition edge crossings. In the rest of the paper we refer to the edges between time steps as *transition edges* to distinguish them from the vertical edges that represent node-to-node relationships.

Reducing the number of transition edge crossings in the Community Fabric visualization is important to make the diagram more readable and interpretable. If a transition between time steps has many nodes changing positions, there will likely be many transition edge crossings. The need to reduce these transition edge crossings is apparent when dealing with noisy and large networks that may span many time

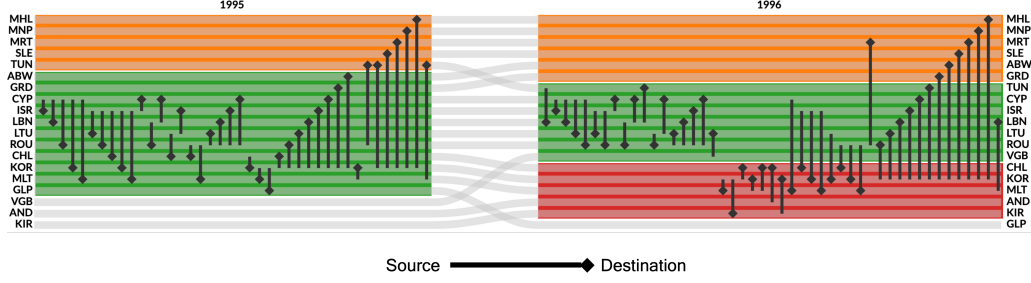


Figure 4.4: Community Fabric visualization technique on an example data set. This data is a selection of countries’ vehicle trade data from 1995 to 1996. On the far left and right of the visualization are the country codes (ISO alpha-3) showing which horizontal line are assigned to each country. Communities are shown with colored underlays at each time step. Singleton nodes at the bottom have no colored underlay because they are not participating in vehicle trade for these years.

steps. An abundance of transition edge crossings can make it difficult to find patterns and follow specific nodes of interest throughout the diagram.

The optimization problem of reducing transition edge crossings in the Community Fabric diagram is related to the long studied crossing minimization problem in computer science. We use the same sweeping heuristic approach described by Vehlow et al. [151] for reducing the edge crossings in an alluvial flow community diagram. This heuristic internally uses a barycentric Sugiyama style one-sided crossing minimization algorithm and builds upon traditional layout algorithms for visualizing hierarchical structures [146, 55]. The Sugiyama algorithm reorders nodes in a hierarchy to reduce the number of edge crossings, creating a more readable drawing.

In order to utilize the crossing minimization heuristic appropriately, we must transform the Community Fabric diagram into hierarchical structures. We represent each transition between time steps as its own hierarchical network called a *transition network*. A Community Fabric diagram has T time steps ranging from 1 to T . For each t where $1 \leq t \leq T - 1$, there exists a transition network G_t that represents the transition between time steps t and $t + 1$. Thus, there are $T - 1$ transition networks in

a Community Fabric diagram. The entire Community Fabric hierarchical structure can be described by the sequence of all transition networks $G = G_1, G_2, \dots, G_{T-1}$.

A transition network is denoted by a 6-level proper hierarchy $G_t = (V, E, n = 6, \sigma)$, where V is the vertex set and E is the edge set. n is the number of levels, which is always 6 in a Community Fabric diagram as shown in Figure 4.5. σ is the vertex ordering for all levels of the hierarchy. G_t satisfies the following 3 conditions:

1. V is partitioned into 6 subsets, that is

$$V = V_1 \cup V_2 \cup \dots \cup V_6 \quad (V_i \cap V_j = \emptyset, i \neq j)$$

where V_i is called the i th level.

2. Every edge $e = (v_i, v_j) \in E$, where $v_i \in V_i$ and $v_j \in V_j$. Each edge in E is unique.

3. E is partitioned into 5 subsets, that is

$$E = E_1 \cup E_2 \cup \dots \cup E_5 \quad (E_i \cap E_j = \emptyset, i \neq j)$$

where $E_i \subset V_i \times V_{i+1}, i = 1, \dots, 5$.

The ordering of all vertices is given by $\sigma = \sigma_1, \sigma_2, \dots, \sigma_6$. The ordering of the i th-level vertices V_i is given by the vertex sequence $\sigma_i = v_1, v_2, \dots, v_{|V_i|}$. The six total levels in a transition network correspond to a time level, community level, and node level for each side of the transition. Figure 4.5 shows the transition network from the Community Fabric diagram in Figure 4.4.

To minimize the overall number of transition edge crossings in Community Fabric, we must minimize the edge crossings between all transitions. The total number of transition edges in a Community Fabric diagram is directly related to the ordering of the vertices in the community levels (i.e. σ_2 and σ_5) and the node levels (i.e. σ_3

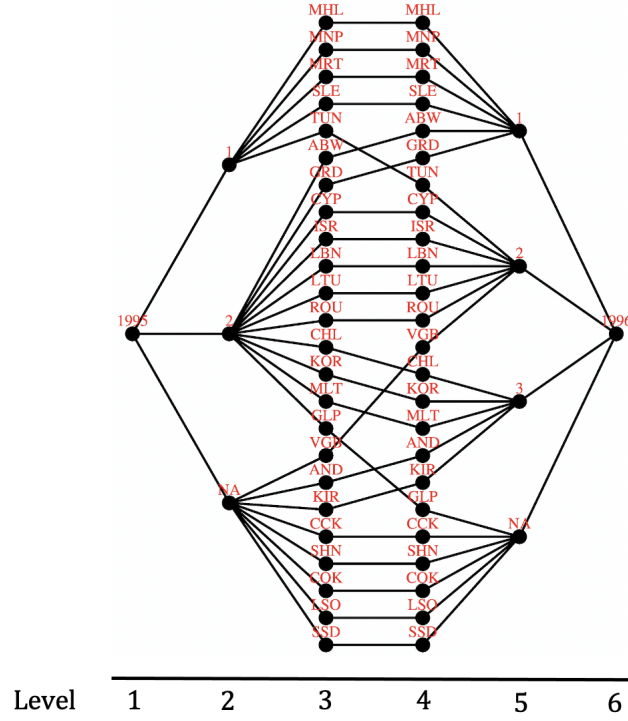


Figure 4.5: Transition network corresponding to the Community Fabric diagram shown in Figure 4.4. There are six total levels in a transition network, a time level (Level 1 and Level 6), community level (Level 2 and Level 5), and node level (Level 3 and Level 4) for each time step of the transition. The edges between Level 3 and Level 4 are the transition edges seen in the Community Fabric diagram.

and σ_4) of each transition network. Let us consider the matrix formulation of our problem.

The ordered adjacency matrix of the node layers of G_t is given by $M^t = M(\sigma_3, \sigma_4)$. M^t is a matrix where the rows and columns are ordered according to σ_3 and σ_4 , respectively. If $\sigma_3 = v_1, \dots, v_k, \dots, v_{|V_3|}$ and $\sigma_4 = w_1, \dots, w_l, \dots, w_{|V_4|}$, then the (v_k, w_l) element of M^t , denoted by m_{kl}^t , is given by

$$m_{kl}^t = \begin{cases} 1 & \text{if } (v_k, w_l) \in E_3 \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

representing the existence of an edge.

As an example, the adjacency matrix M^1 of the transition network in Figure 4.5 is shown in Figure 4.6. We can represent an entire Community Fabric diagram G by $M = M^1, M^2, \dots, M^{T-1}$ (i.e. the adjacency matrix of each transition). Using M^t , we can calculate the number of transition edge crossings from the formulas given in [159]. If we let $r(v)$ be the row vector of M^t where $v \in V_3$, then we can denote the crossings between two vertices as

$$k(r(v_j), r(v_k)) = \sum_{\alpha=1}^{q-1} \sum_{\beta=\alpha+1}^q m_{j\beta} m_{k\alpha} \quad (4.2)$$

where $q = |V_4|$. We can then compute the total number of crossings of single transition $K(M^t)$ with the following formula ,

$$K(M^t) = \sum_{j=1}^{p-1} \sum_{k=j+1}^p k(r(v_j), r(v_k)) \quad (4.3)$$

where $p = |V_3|$. This double summation counts the number of crossings between all pairs of vertices in Layer 3 of their edges to Layer 4. Thus, the total number of

crossings in G is given by

$$K(G) = \sum_{t=1}^{T-1} K(M^t) \quad (4.4)$$

which is also the total number of crossings in the Community Fabric diagram.

Since the node orderings of all transition networks directly determines the number of Community Fabric transition edge crossings, our goal is to find a node ordering which significantly reduces the number of crossings. More formally, if we let σ_3^t and σ_4^t be the orderings of the node layers for the t th transition, then our objective is to find σ_3^t and σ_4^t for all $t \in 1, \dots, T-1$ that minimizes $K(G)$.

To find suitable node orderings, we use the barycentric Sugiyama style one-sided sweeping heuristic [151]. The one-sided Sugiyama algorithm holds one side of the transition fixed and reorders the adjacent side. For example, we hold σ_4^t fixed while reordering σ_3^t . The one-sided approach is used because we are dealing with multiple interconnected hierarchical structures where the ordering of nodes in one transition network affects the number of crossings of the adjacent transition networks. This is because σ_4^t must be the same ordering as σ_3^{t+1} and σ_3^t must be the same ordering as σ_4^{t-1} . The sweeping heuristic iterates through each transition in the forward or backward direction, holding σ_3^t fixed in the forward direction and σ_4^t fixed in the backward direction, attempting to improve the node ordering upon each iteration .

To update node orderings of each transition during a sweep, we use the barycenter measure as in [146, 151]. The barycenter measure has been shown to be effective and efficient when compared to other measures [60, 87, 153]. Given a transition matrix M^t , the barycenter of the k th node in V_3 is given by

$$B_{tk} = \frac{\sum_{l=1}^q l \cdot m_{kl}^t}{\sum_{l=1}^q m_{kl}^t} \quad (4.5)$$

calculating the position of a node with respect to its neighbors in the adjacent layer. During an iteration of the sweeping algorithm we order nodes according to their barycenters, in ascending order, while also keeping nodes in the same community

	MHL	MNP	MRT	SLE	ABW	GRD	TUN	CYP	ISR	LBN	LTU	ROU	VGB	CHL	KOR	MLT	AND	KIR	GLP	CCK	SHN	COK	LSO	SSD
MHL	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MNP	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MRT	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SLE	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TUN	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ABW	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GRD	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CYP	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ISR	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LBN	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LTU	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
ROU	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CHL	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
KOR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
MLT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
GLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
VGB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
AND	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
KIR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
CCK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
SHN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
COK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
LSO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SSD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 4.6: Adjacency matrix, M^1 , of the transition network in Figure 4.5.

in adjacent positions. We must keep nodes in the same community adjacent to preserve one colored underlay for each community when drawing the Community Fabric diagram.

The Community Fabric crossing minimization pseudocode is presented in Algorithm 3 and Algorithm 4. Algorithm 3 presents the helper method *sweep*, which traverses the transition networks forward and backward trying to improve the node orderings of σ_3^t and σ_4^t for all t and returns the new ordering. In the *sweep* method presented here, we take a *SWEEP_ITS* parameter which determines how many forward and backward passes will be made through the transition networks. This method can be further optimized to halt on other conditions if desired.

The sweeping heuristic is presented in Algorithm 4 and determines the best node ordering for a Community Fabric diagram. In our sweeping heuristic, we randomize the community positions (σ_2^t and σ_5^t) and node positions (σ_3^t and σ_4^t) at the start of each sweep. During a sweep, we do not alter the order of the community positions, only the node positions. If a random start community ordering, random start node

Algorithm 3 *sweep*($G, SWEEP_ITERS$)

```
1:  $t \leftarrow$  randomly choose number between 1 and  $T - 1$ 
2:  $direction \leftarrow$  randomly choose forward or backward
3:  $iters \leftarrow 0$ 
4: while  $iters < SWEEP\_ITERS$  do
5:   while  $1 \leq t \leq T - 1$  do
6:     Optimize  $K(M_t)$  using barycentric measure
7:     increment  $t$  in  $direction$ 
8:   end while
9:   flip  $direction$ 
10:  increment  $iters$ 
11: end while
12: return  $G'$  {new Community Fabric transition networks}
```

ordering, and sweep produce a solution that has less crossings than we have seen before, then we save this ordering as the new solution. The sweeping heuristic is run a certain number of times determined by the *MAX_ITERS* parameter. These algorithms may be further optimized based upon preexisting knowledge of a data set. Each sweep may also be worked on by a separate process and run in parallel for faster computation of large networks.

4.4 Results

We visualized three network data sets across different domains using Community Fabric to evaluate the efficacy of our technique. We visualized the interactions of people in a hospital ward across two hours, showing how the communities and structure of the network changed across each hour. We visualized a synthetic dynamic network across three time steps. We also visualized five years of cotton trade between central globe countries.

Algorithm 4*sweeping_heuristic*($G, MAX_ITERS, SWEEP_ITERS$)

```
1:  $iters \leftarrow 0$ 
2:  $MIN\_CROSSINGS \leftarrow K(G)$ 
3:  $solution \leftarrow ordering$ 
4: while  $iters < MAX\_ITERS$  do
5:   randomize community and node positions
6:    $ordering \leftarrow sweep(G, SWEEP\_ITERS)$ 
7:   if  $K(ordering) < MIN\_CROSSINGS$  then
8:      $MIN\_CROSSINGS \leftarrow K(ordering)$ 
9:      $solution \leftarrow ordering$ 
10:  end if
11:  increment  $iters$ 
12: end while
13: return  $solution$  {best node ordering}
```

4.4.1 Hospital Ward Interaction Network

We use a real temporal network of interactions among health-care workers and patients in a hospital ward. This data was collected for a study to estimate infectious transmission routes [149] and distributed via <http://www.sociopatterns.org/>. Each participant wore an RFID tag which recorded interactions when two participants were within close proximity of each other. The interactions were recorded at a resolution of seconds and we chose to aggregate the network into hourly snapshots for analysis. In this paper, we present two time steps from the hospital interaction network. The first time step is the aggregated network showing the interactions from 5 p.m. to 6 p.m. and the second time step is the aggregated network showing the interactions from 6 p.m. to 7 p.m. For visualization purposes, we label each participant by their role abbreviation hyphenated with their unique ID. The following roles are present in the network: administrator (ADM), medical doctor (MED), nurse (NUR), and patient (PAT).

In order to understand the usefulness of our approach, we compared the traditional adjointed node-link diagram (Figure 4.7) and Community Fabric with reduced edge

crossings (Figure 4.8). We detected communities in both time steps using the edge betweenness algorithm [99]. We then matched communities across time steps using the Jaccard similarity index Backward Greedy algorithm [57, 147]. Figure 4.7 presents the adjoined node-link diagram of the two aggregated networks each with nodes colored by their respective community. The node positions are laid out using Fruchterman-Reingold force-directed placement [40]. Figure 4.8 presents our approach, Community Fabric, on the same dynamic network for comparison.

Using the sweeping heuristic for crossing minimization we were able to reduce the number of transition edge crossings from 211 to 58 with five iterations of the sweeping heuristic and halting each sweep when we could not reduce the crossings any further. Figure 4.9 shows the unoptimized Community Fabric diagram as a reference. It is evident that minimizing the number of transition edge crossings improves the readability of the diagram.

When comparing the adjoined node-link diagram (Figure 4.7) and our approach (Figure 4.8) we notice several major differences. With the adjoined node-link diagram it can be hard to identify singleton nodes (colored gray) from one time step to the next. You can see that there are a similar amount of singleton nodes in each time step but it is not clear whether it is the same nodes or different nodes without closely looking at all the node labels. In the Community Fabric diagram, you can clearly see the nodes that become active and become dormant by locating the transition edges at the bottom of the visualization. It is obvious that all of the nodes that were singletons in the first time step become active in the second time step. This allows the viewer to more quickly understand the structural changes with regards to active and dormant nodes in a dynamic network.

Similarly to the above point, it is difficult to identify which nodes are moving in and out of communities without deeply looking at the individual labels in the adjoined node-link diagram. In these two time steps, the orange community has limited change keeping all but two members. This is easy to spot in Community Fabric as most of the transition edges are straight lines, keeping their ordering and

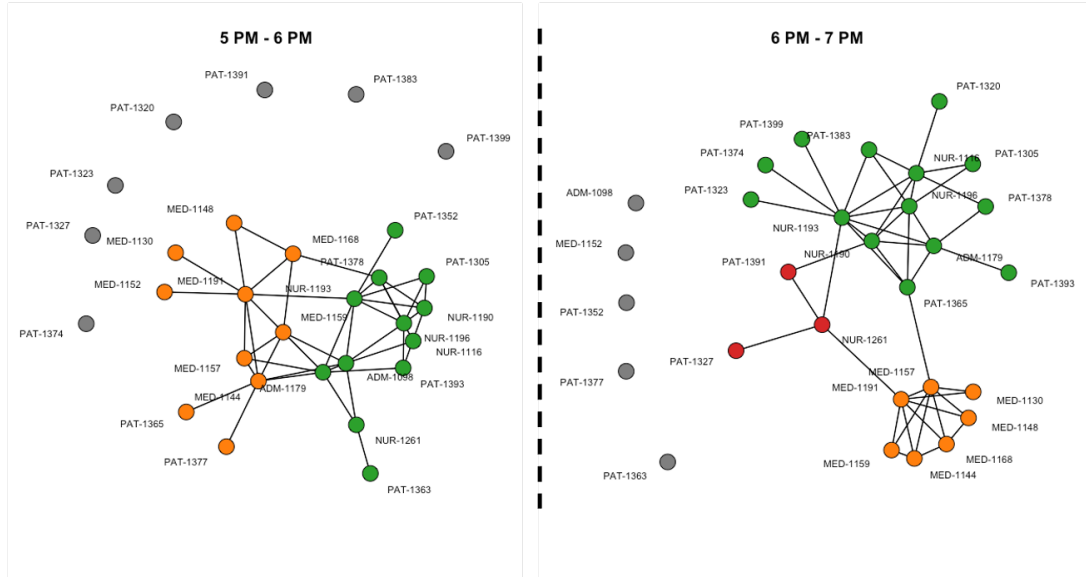


Figure 4.7: Adjoined node-link diagram of two hours in the hospital interaction network. Nodes represent people and links represent interaction between people. Nodes are colored by their respective communities. Nodes are labeled with their respective participant role and id.

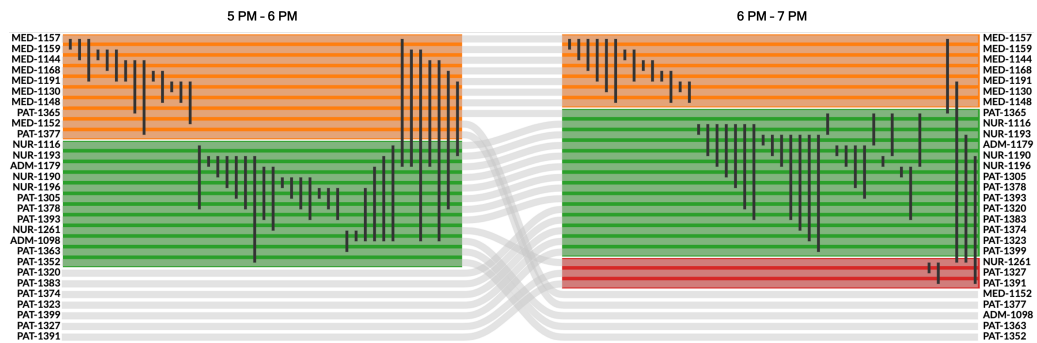


Figure 4.8: Community Fabric visualization of two hours in the hospital interaction network. This is the same network from Figure 4.7.

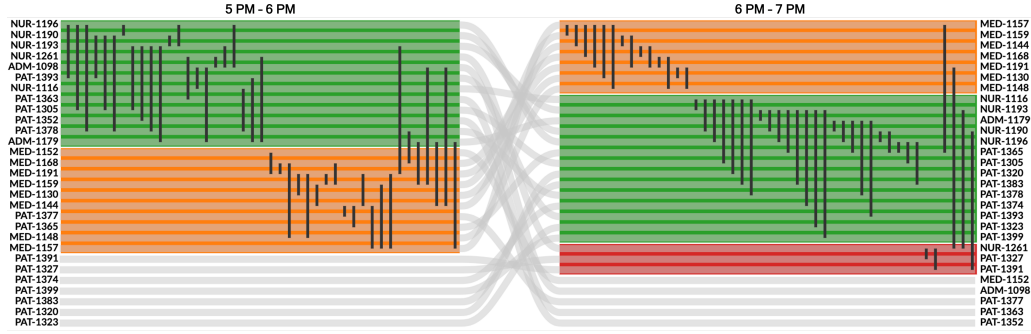


Figure 4.9: Unoptimized Community Fabric diagram. The optimized version of this visualization is shown in Figure 4.8.

position in the next time step, and two of the nodes have transition edges that curve out of the community, becoming dormant nodes in the next time step. Using the adjoined node-link diagrams, this information is not as easy to gather at a glance. You may be able to see that a few nodes dropped out of the orange community but locating where they went is more tedious than using Community Fabric where transitions are streamlined together.

While the orange community does not change much, the green community's structure changes more drastically. Several dormant nodes become active and join the green community, a few nodes drop out becoming dormant, and one node is annexed from the orange community.

Another interesting thing to note is the emergence of a new red community between 6 pm and 7 pm. We can see this clearly in both the adjoined node-link diagram and in Community Fabric. However, we would like to understand the structural changes that led to this new community. Looking at the node-link diagram, it is difficult to tell where the nodes that formed the red community came from. In Community Fabric, we can see that this new community is made up of two nodes that were dormant in the previous time step and one node that came from the green community.

When analyzing communities in dynamic networks it is sometimes important to understand the size of each community. In a node-link diagram, it is not always easy to detect the size of each community, particularly when the node layout algorithm does not match the communities detected (i.e. the same colored nodes are scattered across the diagram). In Community Fabric, each node gets the same amount of horizontal space and nodes in the same community are placed adjacent to each other. This yields a colored underlay for each community that is directly proportional to the size of the community in the network at each time step. This makes it less demanding to see how the size of each community is changing throughout the dynamic network.

Thus far, we have discussed tracking community evolution in this network. However, another important aspect of analyzing a network is understanding the underlying structure of the network, the individual node and edges contributing to the formation of each community. In Community Fabric, the underlying network structure is understood through analyzing the vertical edges in the diagram. Like traditional Biofabric diagrams, we sort the vertical edges from shortest to longest which create edge wedges making it easier to analyze a node's incident edges. For example, the large edge wedge of MED-1157 in the second time step denotes that it is the most connected member of the orange community. If we look at MED-1157's edge wedge in the first time step, it is not nearly as big. Thus, we quickly see that MED-1157 becomes much more highly connected member of the orange community in the second time step. Looking deeper at the edge structure of the network we see that patients tend to only interact with their nurse and not with other patients. Doctors are mostly in a community with other doctors and tend to interact with doctors.

Another interesting aspect of analyzing dynamic networks is looking at the modularity of communities, how densely connected they are within but how sparsely connected they are with other communities. In Community Fabric, we intentionally place inter-community edges first over the community underlay and intra-community edges after. This allows the viewer to understand the modularity of the community

visually. For example, in this dynamic network the orange and green community are highly connected in the first time step. In the second time step, there is only one connection between the orange and green community. We can quickly see that MED-1157 and PAT-1365 are the only connected nodes from the green community to the orange community in the second time step.

4.4.2 Synthetic Dynamic Network

We generated a synthetic dynamic network for visualization. Our synthetic network contains three time steps. The first time step was built using the Lancichinetti-Fortunato-Radicchi (LFR) graph generation technique [76] which builds a network with known communities. Our LFR parameter configuration is shown in Table 4.1. To create the second time step of the dynamic network, we randomly removed edges. We chose to remove half of the edges from about a third of the nodes. In the third and final time step, we randomly added edges. We chose to add up to 5 edges (number of edges randomly selected) to about a third of the nodes. After constructing the dynamic network we performed independent static community detection with matching. As in the previous example, we used edge betweenness community detection [99] with Jaccard similarity matching [57].

We visualize both the adjoined node-link diagram (Figure 4.10) and the Community Fabric diagram (Figure 4.11) of this created network. The purpose of synthetically generating this network is to showcase the ability of Community Fabric

Parameter	Description	Value
N	number of nodes in the network	25
γ	power law exponent for node degree distribution	3
β	power law exponent for community size distribution	1.5
μ	mixing parameter: $1 - \mu$ links will be intra-community links	0.15

Table 4.1: Lancichinetti-Fortunato-Radicchi Graph Configuration Parameters [76].

to visualize many time step dynamic networks. We only use three time steps for this paper. However, dynamic networks with many more time steps can be viewed on a display device where space constraints are not an issue. See the next example for a network with more time steps.

Using the sweeping heuristic for crossing minimization we were able to reduce the number of transition edge crossings from 439 to 48 with five iterations of the sweeping heuristic. In this particular dynamic network, there are three communities in the first time step and five communities in the subsequent time steps. From time step one to time step two the red community is split into two communities forming a new purple community. Two nodes from the orange community break off to form a brown community which persists throughout the rest of the network. The emergence of new communities in time step two makes sense because we removed edges. The removal of edges causes communities to become less densely connected yielding the splitting of communities. In time step three, the number of communities remain the same from time step two. In this time step, we added edges to random nodes. These edges may or may not be intra-community edges and are more likely to be inter-community edges because the partner node was chosen at random. Ultimately, this makes the network less modular than previous time steps. We can quickly see the difference between modularity of each time step by looking at the right most edges.

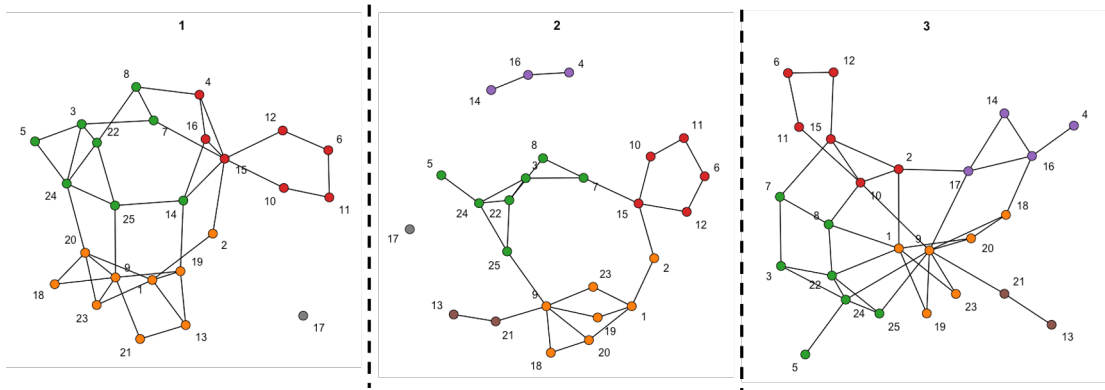


Figure 4.10: Node-link diagram of a synthetically generated dynamic network.

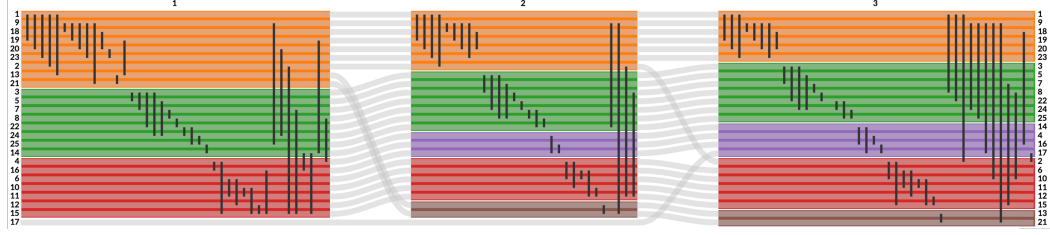


Figure 4.11: Community Fabric diagram of a synthetically generated dynamic network.

There are the least amount of inter-community edges in time step two, whereas the most number of inter-community edges are in the last time step.

4.4.3 Atlas Economic Trade Data

We use yearly aggregated economic world trade data between countries [4]. This data is based upon the United Nations Comtrade data and is cleaned up by the Harvard Growth Lab for open research. In this example, we selected cotton trade among 18 central globe countries from five consecutive years, 2014 to 2018. The Community Fabric diagram is presented in Figure 4.12. As in the previous examples, we used edge betweenness community detection [99] with Jaccard similarity matching [57].

Looking at this network from a macro level, we see that there are several changes taking place throughout this network. The communities are changing frequently and are not as stable as the previous two networks. The longest persisting community is the brown community made up of Uzbekistan and Kyrgyzstan from 2015 to 2018, driven by Uzbekistan’s cotton export to Kyrgyzstan in those years. The green community is present from 2014 to 2016 but splits in 2017. Other than the brown and green communities, no other community persists for more than two time steps in this network.

Observing the network from a micro view, we can gain insight into the specific node level relationship changes taking place. In 2014, the United Arab Emirates and

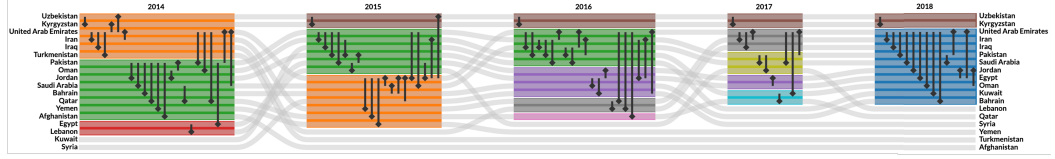


Figure 4.12: Community Fabric diagram of cotton trade from 2014 to 2018 between central globe countries.

Pakistan are the major exporters. However, in 2018 Pakistan is only exporting to one country while the United Arab Emirates is still trading with many countries.

In Figure 4.13, we present the same dynamic network with the United Arab Emirates selected for focus. In this focused mode, it is easier to follow one specific node throughout the network. We can more clearly see that the United Arab Emirates is involved throughout every time step as the major exporter of cotton among central globe countries.

4.5 Conclusion

We have presented a novel approach for visualizing evolving communities in a dynamic network, together with evolving structure of the network. Community Fabric improves on existing state-of-the-art techniques via a hybrid approach, which superimposes a Biofabric-like [85] structural graph atop each time step of a traditional alluvial flow community diagram [151]. By using Biofabric style horizontal node placement and vertical edge placement, we alleviate the need to deal with restricted node-link layouts to show network structure. This allows us to display inter-community edges and intra-community edges as identical edges, which was not possible before.

We use a hierarchical edge crossing minimization algorithm to reduce transition edge crossings in the Community Fabric diagram, making the visualization more palatable to viewers. Our results demonstrated a significant decrease in the number of transition edge crossings among all example networks.

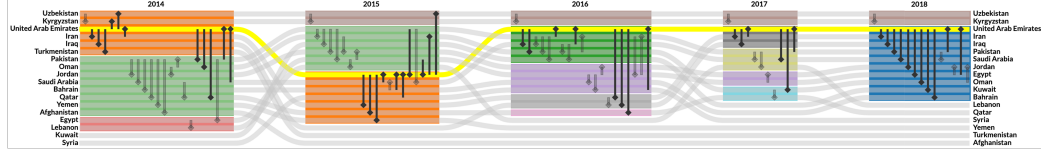


Figure 4.13: Community Fabric diagram with the United Arab Emirates interactively selected to focus on. This dynamic network shows cotton trade from 2014 to 2018 between central globe countries, the same data used in Figure 4.12.

The Community Fabric diagram was developed using modern web-based visualization tools, showing it is possible to implement such a visualization. We showcased our visualization on three different example datasets: a human interaction network, a synthetic network, and a trade network. Our results indicate that Community Fabric can be used to more easily interpret and understand the interplay of structural change and community evolution in dynamic graphs.

4.6 Future Work

For future work, we suggest possible directions to advance the Community Fabric visualization technique. Adding more interactivity to our visualization and allowing the user to select different node/edge orderings based on their data and preferences may advance the usefulness of Community Fabric. It is possible to provide a more horizontally compact version of our visualization by modifying the edge assignment algorithm to allow the stacking of edges where there is no overlap. We have not yet considered enhanced coloring techniques for coloring the nodes and edges, such as stability-based coloring techniques and coloring based on other network metrics [151]. Future directions include providing an implementation of the sweeping crossing minimization heuristic to the open source and research community for further study. A framework specific implementation, such as integration of the algorithm into the popular Open Graph Drawing Framework [28] would provide easy accessibility. We

would like to further compare and rigorously evaluate different community detection algorithms using Community Fabric.

4.7 Author Contributions

Evan Ezell and Seung-Hwan Lim conceptualized the new visualization technique. Evan Ezell with the help of Seung-Hwan Lim developed the crossing minimization algorithm. Evan Ezell implemented the crossing minimization algorithm and ran the experiments. David Anderson and Evan Ezell developed the web visualization. Seung-Hwan Lim and Robert Stewart oversaw the work. Robert Stewart provided suggestions for practical applications and example data sets. Evan Ezell wrote the manuscript. Seung-Hwan Lim proofread and edited the final manuscript.

4.8 Acknowledgements

Content from this chapter has been originally published in two locations. An extended abstract, developing the ideas and framework was published in the 2020 Graph Drawing and Network Visualization Conference Proceedings:

Evan Ezell, Seung-Hwan Lim, David Anderson, and Robert Stewart. Visualizing communities and structure in dynamic graphs. In David Auber and Pavel Valtr, editors, *Graph Drawing and Network Visualization: 28th International Symposium, GD 2020*, volume 12590 of *Lecture Notes in Computer Science*, pages 532–534. Springer, February 2021.

Following the above extended abstract, a more thorough work was completed fully developing the methods and application. This work was published as a full journal article in the SAGE Information Visualization Quarterly Journal:

Evan Ezell, Seung-Hwan Lim, David Anderson, and Robert Stewart. Community fabric: Visualizing communities and structure in dynamic networks. *Information Visualization*, 21:130–142, April 2022.

This manuscript was authored by UT-Battelle LLC under Contract No. DE-AC05-00OR22725 with DOE. The US government retains—and the publisher, by accepting the article for publication, acknowledges that the US government retains—a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript or allow others to do so for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<http://energy.gov/downloads/doe-public-access-plan>).

Bibliography

- [1] Lahreche Abdelmadjid and Bachir Boucheham. A comparison study of dynamic time warping’s variants for time series classification. *International Journal of Informatics and Applied Mathematics*, 4:56–71, 01 2021. ISSN 2667-6990. [25](#), [169](#)
- [2] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In David B. Lomet, editor, *Foundations of Data Organization and Algorithms*, pages 69–84, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. ISBN 978-3-540-48047-1. [28](#)
- [3] Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zaït. Querying shapes of histories. In *Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB ’95, page 502–514, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603794. [10](#), [12](#), [18](#), [20](#)
- [4] The Growth Lab at Harvard University. International trade data (hs, 92), 2019. URL <https://doi.org/10.7910/DVN/T4CHWJ>. [130](#)
- [5] The Growth Lab at Harvard University. International trade data (sitc, rev. 2), 2025. URL <https://doi.org/10.7910/DVN/H8SFD2>. [71](#), [98](#), [99](#)
- [6] Cyril Banderier and Sylviane Schwer. Why Delannoy numbers? *Journal of Statistical Planning and Inference*, 135(1):40–54, 2005. ISSN 0378-3758. doi:

- <https://doi.org/10.1016/j.jspi.2005.02.004>. URL <https://www.sciencedirect.com/science/article/pii/S0378375805000534>. Special issue on lattice path combinatorics and discrete distributions. 167
- [7] World Bank. *Vietnam Coffee Sector Report*. World Bank, Washington, DC, jun 2004. 100
- [8] Claudio D. T. Barros, Matheus R. F. Mendonça, Alex B. Vieira, and Artur Ziviani. A survey on embedding dynamic graphs. *ACM Computing Surveys*, 55(1), November 2021. ISSN 0360-0300. doi: 10.1145/3483595. URL <https://doi.org/10.1145/3483595>. 72
- [9] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum*, 36(1): 133–159, January 2017. ISSN 0167-7055. doi: 10.1111/cgf.12791. URL <https://doi.org/10.1111/cgf.12791>. 108
- [10] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952. ISSN 0027-8424. doi: 10.1073/pnas.38.8.716. URL <https://www.pnas.org/content/38/8/716>. 165
- [11] Richard Ernest Bellman. *The Theory of Dynamic Programming*. RAND Corporation, Santa Monica, CA, 1954. 165
- [12] Richard Ernest Bellman. *Dynamic programming*. Princeton University Press, Princeton, 1957. 165
- [13] Richard Ernest Bellman and Stuart E. Dreyfus. *Applied dynamic programming*. Princeton University Press, Princeton, N.J, 1962. 165
- [14] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994. 25, 165

- [15] Alessandro Camerra, Themis Palpanas, Jin Shieh, and Eamonn Keogh. isax 2.0: Indexing and mining one billion time series. In *2010 IEEE International Conference on Data Mining*, pages 58–67, 2010. doi: 10.1109/ICDM.2010.124. [17](#), [18](#), [29](#)
- [16] Alessandro Camerra, Jin Shieh, Themis Palpanas, Thanawin Rakthanmanon, and Eamonn Keogh. Beyond one billion time series: Indexing and mining very large time series collections with isax2+. *Knowledge and Information Systems*, 39, 04 2014. doi: 10.1007/s10115-012-0606-6. [17](#), [18](#), [29](#)
- [17] K. Selcuk Candan, Rosaria Rossini, Maria Luisa Sapino, and Xiaolan Wang. sdtw: Computing dtw distances using locally relevant constraints based on salient feature alignments. *Proc. VLDB Endow.*, 5:1519–1530, 2012. [25](#), [170](#)
- [18] Danyang Cao and Zifeng Lin. Multidimensional time series motif group discovery based on matrix profile. *Knowledge-Based Systems*, 304:112509, 2024. ISSN 0950-7051. doi: <https://doi.org/10.1016/j.knosys.2024.112509>. URL <https://www.sciencedirect.com/science/article/pii/S0950705124011432>. [105](#)
- [19] Capital Bikeshare. Capital bikeshare trip history data. <https://s3.amazonaws.com/capitalbikeshare-data/index.html>, 2025. [71](#), [81](#), [82](#)
- [20] Rémy Cazabet and Frédéric Amblard. Dynamic community detection. In Reda Alhajj and Jon Rokne, editors, *Encyclopedia of Social Network Analysis and Mining*, pages pp 404–414. Springer, New York, NY, 09 2014. doi: 10.1007/978-1-4614-6170-8_383. [108](#)
- [21] Rémy Cazabet, Hideaki Takeda, Masahiro Hamasaki, and Frédéric Amblard. Using dynamic community detection to identify trends in user-generated content. *Social Network Analysis and Mining*, 2(4):361–371, 2012. doi: 10.1007/s13278-012-0074-8. URL <http://dblp.uni-trier.de/db/journals/snam/snam2.html#CazabetTHA12>. [108](#)

- [22] Kaushik Chakrabarti and Sharad Mehrotra. The hybrid tree: an index structure for high dimensional feature spaces. *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*, pages 440–447, 1999. URL <https://api.semanticscholar.org/CorpusID:5856151>. [28](#)
- [23] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, jun 2002. ISSN 0362-5915. doi: 10.1145/568518.568520. URL <https://doi.org/10.1145/568518.568520>. [17](#)
- [24] Pimwadee Chaovalit, Aryya Gangopadhyay, George Karabatis, and Zhiyuan Chen. Discrete wavelet transform-based time series analysis and mining. *ACM Comput. Surv.*, 43(2), feb 2011. ISSN 0360-0300. doi: 10.1145/1883612.1883613. URL <https://doi.org/10.1145/1883612.1883613>. [16](#), [26](#)
- [25] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings 2004 VLDB Conference*, pages 792–803. Elsevier Inc, 2004. ISBN 9780120884698. [26](#)
- [26] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 491–502, 2005. [26](#)
- [27] V. M. Chhieng and Raymond K. Wong. Adaptive distance measurement for time series databases. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4443, pages 598–610, 2007. ISBN 9783540717027. [26](#)
- [28] Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (ogdf). *Handbook of graph drawing and visualization*, 2011:543–569, 2013. [132](#)

- [29] Gautam Das, Dimitrios Gunopulos, Heikki Mannila, Jan Komorowski, and Jan Zytkow. Finding similar time series. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1263 of *Lecture Notes in Computer Science*, pages 88–100. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. ISBN 9783540632238. [26](#)
- [30] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. [24](#)
- [31] Stuart Daw, Charles Finney, and Eugene Tracy. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74:915–930, 02 2003. doi: 10.1063/1.1531823. [18](#)
- [32] Shohreh Deldari, Daniel V. Smith, Amin Sadri, and Flora Salim. Espresso: Entropy and shape aware time-series segmentation for processing heterogeneous sensor data. *Proceedings of ACM on interactive, mobile, wearable and ubiquitous technologies*, 4(3):1–24, 2020. ISSN 2474-9567. [31](#)
- [33] Paul DeMaio. Bike-sharing: History, impacts, models of provision, and future. *Journal of public transportation*, 12(4):41–56, 2009. ISSN 1077-291X. [80](#)
- [34] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, aug 2008. ISSN 2150-8097. doi: 10.14778/1454159.1454226. URL <https://doi.org/10.14778/1454159.1454226>. [165](#)

- [35] Claire Donnat and Susan Holmes. Tracking network dynamics: A survey using graph distances. *The Annals of Applied Statistics*, 12(2):971–1012, jun 2018. doi: 10.1214/18-AOAS1176. [71](#)
- [36] S R Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 01 1998. ISSN 1367-4803. doi: 10.1093/bioinformatics/14.9.755. URL <https://doi.org/10.1093/bioinformatics/14.9.755>. [31](#)
- [37] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1), dec 2012. ISSN 0360-0300. doi: 10.1145/2379776.2379788. URL <https://doi.org/10.1145/2379776.2379788>. [14](#), [15](#), [24](#), [32](#)
- [38] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’94, page 419–429, New York, NY, USA, 1994. Association for Computing Machinery. ISBN 0897916395. doi: 10.1145/191839.191925. URL <https://doi.org/10.1145/191839.191925>. [16](#), [26](#)
- [39] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’94, page 419–429, New York, NY, USA, 1994. Association for Computing Machinery. ISBN 0897916395. doi: 10.1145/191839.191925. URL <https://doi.org/10.1145/191839.191925>. [28](#)
- [40] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi: <https://doi.org/10.1002/spe.4380211102>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380211102>. [124](#)

- [41] Tak-Chung Fu, F-L Chung, Robert Luk, and C-M Ng. Financial time series indexing based on low resolution clustering. In *Proceedings of International Conference on Data Management*, pages 5–14, 2004. [15](#)
- [42] Shaghayegh Gharghabi, Yifei Ding, Chin-Chia Michael Yeh, Kaveh Kamgar, Liudmila Ulanova, and Eamonn Keogh. Matrix profile viii: Domain agnostic online semantic segmentation at superhuman performance levels. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 117–126, 2017. doi: 10.1109/ICDM.2017.21. [10](#), [11](#), [30](#), [32](#), [34](#), [38](#), [42](#), [44](#), [58](#), [63](#), [65](#)
- [43] Fatma Ezzahra Gmati, Salem Chakhar, Wided Lejouad Chaari, and Mark Xu. Shape-based representation and abstraction of time series data along with a dynamic time shape wrapping as a dissimilarity measure. In *2021 26th International Conference on Automation and Computing (ICAC)*, pages 1–8, 2021. doi: 10.23919/ICAC50006.2021.9594127. [10](#), [12](#), [20](#), [22](#)
- [44] Tomasz Gorecki and Maciej Luczak. Using derivatives in time series classification. *Data mining and knowledge discovery*, 26(2):310–331, 2013. ISSN 1384-5810. [27](#)
- [45] Tomasz Górecki and Paweł Piasecki. A comprehensive comparison of distance measures for time series classification. In Ansgar Steland, Ewaryst Rafajłowicz, and Ostap Okhrin, editors, *Stochastic Models, Statistics and Their Applications*, pages 409–428, Cham, 2019. Springer International Publishing. ISBN 978-3-030-28665-1. [24](#), [26](#), [165](#), [169](#)
- [46] Machon Gregory. Shape identificaiton and ranking in temporal data sets. Master’s thesis, University of Maryland, College Park, College Park, Maryland, 2009. [18](#)

- [47] Machon Gregory and Ben Shneiderman. *Shape Identification in Temporal Data Sets*, pages 305–321. Springer London, London, 2012. ISBN 978-1-4471-2804-5. doi: 10.1007/978-1-4471-2804-5_17. URL https://doi.org/10.1007/978-1-4471-2804-5_17. [10](#), [12](#), [18](#), [20](#)
- [48] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939754. URL <https://doi.org/10.1145/2939672.2939754>. [72](#), [75](#)
- [49] Xudong Guan, Chong Huang, Gaohuan Liu, Xuelian Meng, and Qingsheng Liu. Mapping rice cropping systems in vietnam using an ndvi-based time-series similarity measurement based on dtw distance. *Remote Sensing*, 8(1):19, 2016. ISSN 2072-4292. doi: 10.3390/rs8010019. [34](#)
- [50] Francesco Gullo, Giovanni Ponti, Andrea Tagarelli, and Sergio Greco. A time series representation model for accurate and fast similarity detection. *Pattern Recogn.*, 42(11):2998–3014, November 2009. ISSN 0031-3203. doi: 10.1016/j.patcog.2009.03.030. URL <https://doi.org/10.1016/j.patcog.2009.03.030>. [10](#), [20](#), [21](#)
- [51] Tomasz Górecki. Classification of time series using combination of dtw and lcss dissimilarity measures. *Communications in statistics. Simulation and computation*, 47(1):263–276, 2018. ISSN 0361-0918. [27](#)
- [52] Tomasz Górecki, Maciej Łuczak, and Paweł Piasecki. An exhaustive comparison of distance measures in the classification of time series with 1nn method. *Journal of Computational Science*, 76:102235, 2024. ISSN 1877-7503. doi: <https://doi.org/10.1016/j.jocs.2024.102235>. URL <https://www.sciencedirect.com/science/article/pii/S1877750324000280>. [24](#), [27](#)

- [53] Yuanduo He, Xu Chu, and Yasha Wang. Neighbor profile: Bagging nearest neighbors for unsupervised time series mining. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, volume 2020-, pages 373–384. IEEE, 2020. ISBN 1728129036. [31](#)
- [54] Joseph M. Hellerstein, Elias Koutsoupas, and Christos H. Papadimitriou. On the analysis of indexing schemes. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems - PODS*, pages 249–256, 1997. [29](#)
- [55] Danny Holten and Jarke J. van Wijk. Visual comparison of hierarchically organized data. In *Proceedings of the 10th Joint Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis’08, pages 759–766, Geneva and New York, May 2008. The Eurographs Association and John Wiley & Sons, Ltd. doi: 10.1111/j.1467-8659.2008.01205.x. URL <https://doi.org/10.1111/j.1467-8659.2008.01205.x>. [116](#)
- [56] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975. doi: 10.1109/TASSP.1975.1162641. [25](#), [169](#)
- [57] Paul Jaccard. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 01 1901. doi: 10.5169/seals-266450. [124](#), [128](#), [130](#)
- [58] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. Time series management systems: A survey. *IEEE transactions on knowledge and data engineering*, 29(11):2581–2600, 2017. ISSN 1041-4347. [28](#)
- [59] Matthew J. Johnson and Alan S. Willsky. Bayesian nonparametric hidden semi-markov models. *Journal of machine learning research*, 14(1):673–701, 2013. ISSN 1532-4435. [31](#)

- [60] Michael Junger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.*, 1, 01 1997. doi: 10.7155/jgaa.00001. [120](#)
- [61] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of arima time-series. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 273–280, 2001. doi: 10.1109/ICDM.2001.989529. [16](#), [27](#)
- [62] Kouki Kawabata, Yasuko Matsubara, and Yasushi Sakurai. Streamscope: Automatic pattern discovery over data streams. In *Proceedings of the 1st International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM 2018*, 2018. ISBN 9781450358514. [31](#)
- [63] Eamonn Keogh and Michael Pazzani. Derivative dynamic time warping. *Proceedings of the First SIAM International Conference on Data Mining*, 1: 5–7, 01 2002. doi: 10.1137/1.9781611972719.1. [25](#), [169](#)
- [64] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005. ISSN 0219-1377. [25](#)
- [65] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, Aug 2001. ISSN 0219-1377. doi: 10.1007/PL00011669. URL <https://doi.org/10.1007/PL00011669>. [16](#), [29](#), [32](#)
- [66] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. *SEGMENTING TIME SERIES: A SURVEY AND NOVEL APPROACH*, pages 1–21. World Scientific Publishing Company, 2003. doi: 10.1142/9789812565402_0001. URL https://www.worldscientific.com/doi/abs/10.1142/9789812565402_0001. [17](#), [33](#)

- [67] Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. Towards parameter-free data mining. In *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, 2004. ISBN 9781581138887. [27](#)
- [68] Eamonn J. Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7:349–371, 2002. URL <https://api.semanticscholar.org/CorpusID:6072819>. [28](#)
- [69] Eamonn J. Keogh and Michael J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD’98, page 239–243. AAAI Press, 1998. [33](#)
- [70] Sang Kim, Hee Lee, Han Ko, Seung Jeong, Hyun Byun, and Kyong Oh. Pattern matching trading system based on the dynamic time warping algorithm. *Sustainability*, 10(12):4641, 2018. ISSN 2071-1050. doi: 10.3390/su10124641. [33](#)
- [71] Kurt Koffka. *Principles of Gestalt Psychology*. Harcourt, Brace, New York, NY, 1935. [115](#)
- [72] Haridimos Kondylakis, Niv Dayan, Kostas Zoumpatianos, and Themis Palpanas. Coconut: A scalable bottom-up approach for building data series indexes. *Proceedings of the VLDB Endowment*, 11(6):677–690, 2018. ISSN 2150-8097. [30](#)
- [73] Haridimos Kondylakis, Niv Dayan, Kostas Zoumpatianos, and Themis Palpanas. Coconut: sortable summarizations for scalable indexes over static and streaming data series. *The VLDB journal*, 28(6):847–869, 2019. ISSN 1066-8888. [30](#)

- [74] Haridimos Kondylakis, Kostas Zoumpatianos, Niv Dayan, and Themis Palpanas. Coconut palm: Static and streaming data series exploration now in your palm. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1941–1944, 2019. ISBN 1450356435. [30](#)
- [75] Zhichen Lai, Huan Li, Dalin Zhang, Yan Zhao, Weizhu Qian, and Christian S. Jensen. E2usd: Efficient-yet-effective unsupervised state detection for multivariate time series. In *WWW 2024 - Proceedings of the ACM Web Conference*, pages 3010–3021, 2024. ISBN 9798400701719. [31](#)
- [76] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4): 046110, Oct 2008. ISSN 1550-2376. doi: 10.1103/physreve.78.046110. URL <http://dx.doi.org/10.1103/PhysRevE.78.046110>. [128](#)
- [77] Sean M. Law. STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *The Journal of Open Source Software*, 4(39):1504, 2019. [44](#), [79](#)
- [78] Oleksandra Levchenko, Boyan Koley, Djamel-Edine Yagoubi, Dennis Shasha, Themis Palpanas, Patrick Valduriez, Reza Akbarinia, Florent Massegli, Elisa Fromont, Andreas Hotho, Arno Knobbe, Ulf Brefeld, Marloes Maathuis, and Céline Robardet. Distributed algorithms to find similar time series. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11908 of *Lecture Notes in Computer Science*, pages 781–785. Springer International Publishing, Cham, 2020. ISBN 9783030461324. [30](#)
- [79] Bryan Lewin, Daniele Giovannucci, and Panos Varangis. Coffee markets: New paradigms in global supply and demand. (Agriculture and Rural Development Discussion Paper No. 3), 2004. [101](#)

- [80] Chung-Sheng Li, Philip S. Yu, and Vittorio Castelli. Malm: a framework for mining sequence database at multiple abstraction levels. In *Proceedings of the Seventh International Conference on Information and Knowledge Management, CIKM '98*, page 267–272, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 1581130619. doi: 10.1145/288627.288666. URL <https://doi.org/10.1145/288627.288666>. [10](#), [12](#), [20](#), [21](#), [33](#)
- [81] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03*, pages 2–11, 2003. [27](#)
- [82] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, Oct 2007. ISSN 1573-756X. doi: 10.1007/s10618-007-0064-z. URL <https://doi.org/10.1007/s10618-007-0064-z>. [15](#), [17](#), [29](#), [32](#)
- [83] Michele Linardi and Themis Palpanas. Scalable, variable-length similarity search in data series: The ulisse approach. *Proceedings of the VLDB Endowment*, 11(13):2236–2248, 2018. ISSN 2150-8097. [30](#)
- [84] Michele Linardi and Themis Palpanas. Ulisse: Ultra compact index for variable-length similarity search in data series. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1356–1359. IEEE, 2018. ISBN 9781538655207. [30](#)
- [85] William Longabaugh. Combing the hairball with biofabric: A new approach for visualization of large networks. *BMC bioinformatics*, 13:275, 10 2012. doi: 10.1186/1471-2105-13-275. [110](#), [131](#)

- [86] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3762–3765, 2018. doi: 10.1109/BigData.2018.8621910. [72](#), [73](#), [98](#)
- [87] Erkki Makinen. Experiments on drawing 2-level hierarchical graphs. *International Journal of Computer Mathematics*, 36(3-4):175–181, 1990. doi: 10.1080/00207169008803921. URL <https://doi.org/10.1080/00207169008803921>. [120](#)
- [88] P.-F. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):306–318, 2009. ISSN 0162-8828. [26](#)
- [89] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. Autoplait: Automatic mining of co-evolving time sequences. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 193–204, 2014. ISBN 9781450323765. [31](#)
- [90] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013. [72](#)
- [91] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf. [72](#)
- [92] Saptarshi Mondal and C. Jeganathan. Mountain agriculture extraction from time-series modis ndvi using dynamic time warping technique. *International*

- Journal of Remote Sensing*, 39(11):3679–3704, 2018. ISSN 0143-1161. doi: 10.1080/01431161.2018.1444289. [34](#)
- [93] Pablo Montero and José A. Vilar. Tslust: An r package for time series clustering. *Journal of Statistical Software*, 62(1):1–43, 2014. doi: 10.18637/jss.v062.i01. URL <https://www.jstatsoft.org/index.php/jss/article/view/v062i01>. [24](#)
- [94] Geoffrey H. Moore, editor. *Business Cycle Indicators*, volume 1 of *National Bureau of Economic Research Studies in Business Cycles*. Princeton University Press, Princeton, NJ, January 1961. ISBN 0-87014-093-0. [14](#)
- [95] A. Morton, R. Stewart, E. Held, J. Piburn, M. R. Allen, R. McManamay, J. Sanyal, A. Sorokine, and B. L. Bhaduri. The World Spatiotemporal Analytics and Mapping Project (WSTAMP): Discovering, Exploring, and Mapping Spatiotemporal Patterns Across Heterogenous Space-Time Data. In *AGU Fall Meeting Abstracts*, volume 2017, pages IN41B–0030, December 2017. [52](#)
- [96] Muhammad Marwan Muhammad Fuad, Pierre-François Marteau, Roland Wagner, Josef Küng, Sourav S Bhowmick, Roland Wagner, Josef Küng, and Sourav S. Bhowmick. Extending the edit distance using frequencies of common characters. In *Database and Expert Systems Applications*, volume 5181 of *Lecture Notes in Computer Science*, pages 150–157. Springer Berlin / Heidelberg, Germany, 2008. ISBN 9783540856535. [26](#)
- [97] C. Myers, L. Rabiner, and A. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):623–635, 1980. doi: 10.1109/TASSP.1980.1163491. [165](#)

- [98] Cory S. Myers. A comparative study of several dynamic time warping algorithms for speech recognition. Master’s thesis, Massachusetts Institute of Technology, Cambridge, MA, February 1980. [165](#)
- [99] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004. doi: 10.1103/PhysRevE.69.026113. URL <https://link.aps.org/doi/10.1103/PhysRevE.69.026113>. [124](#), [128](#), [130](#)
- [100] Néstor Osorio. The global coffee crisis: a threat to sustainable development. *International Coffee Organization, London*, 21, 2002. [100](#), [101](#)
- [101] Gergely Palla, Albert-Laszlo Barabasi, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446:664–7, 05 2007. doi: 10.1038/nature05670. [108](#)
- [102] Themis Palpanas. Data series management: The road to big sequence analytics. *SIGMOD record*, 44(2):47–52, 2015. ISSN 0163-5808. [28](#)
- [103] Themis Palpanas. Evolution of a data series index. In Giorgos Flouris, Dominique Laurent, Dimitris Plexousakis, Nicolas Spyratos, and Yuzuru Tanaka, editors, *Information Search, Integration, and Personalization*, pages 68–83, Cham, 2020. Springer International Publishing. ISBN 978-3-030-44900-1. [18](#)
- [104] Themis Palpanas, Nicolas Spyratos, Yuzuru Tanaka, Giorgos Flouris, Dimitris Plexousakis, and Dominique Laurent. Evolution of a data series index: The isax family of data series indexes: isax, isax2.0, isax2+, ads, ads+, ads-full, paris, paris+, messi, dpisax, ulisse, coconut-trie/tree, coconut-lsm. In *Communications in Computer and Information Science*, volume 1197 of *Communications in Computer and Information Science*, pages 68–83. Springer International Publishing, Cham, 2020. ISBN 9783030448998. [28](#), [29](#), [30](#)

- [105] Antonello Panuccio, Manuele Bicego, and Vittorio Murino. A hidden markov model-based approach to sequential data clustering. In Terry Caelli, Adnan Amin, Robert P. W. Duin, Dick de Ridder, and Mohamed Kamel, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 734–743, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-70659-5. [16](#)
- [106] S. Papadimitriou, Jimeng Sun, and P.S. Yu. Local correlation tracking in time series. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 456–465. IEEE, 2006. ISBN 9780769527017. [26](#)
- [107] Mark Pendergrast. *Uncommon Grounds: The History of Coffee and How It Transformed Our World*, chapter 16, pages 303–319. Basic Books, 2010. ISBN 9780465024049. [99](#)
- [108] Botao Peng, Panagiota Fatourou, and Themis Palpanas. Paris: The next destination for fast data series indexing and query answering. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 791–800. IEEE, 2018. ISBN 9781538650356. [30](#)
- [109] Botao Peng, Panagiota Fatourou, and Themis Palpanas. Messi: In-memory data series indexing. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, volume 2020-, pages 337–348. IEEE, 2020. ISBN 1728129036. [30](#)
- [110] Botao Peng, Panagiota Fatourou, and Themis Palpanas. Paris+: Data series indexing on multi-core architectures. *IEEE transactions on knowledge and data engineering*, 33(5):2151–2164, 2021. ISSN 1041-4347. [30](#)
- [111] Chang Shing Perng, Haixun Wang, Sylvia R. Zhang, and D. Stott Parker. Landmarks: A new model for similarity-based pattern querying in time series

- databases. In *Proceedings - International Conference on Data Engineering*, pages 33–42, 2000. [15](#)
- [112] J. Piburn, R. Stewart, A. Myers, A. Sorokine, E. Axley, D. Anderson, J. Burdette, C. Biddle, A. Hohl, R. Eberle, J. Kaufman, and A. Morton. The world spatiotemporal analytics and mapping project (wstamp): Further progress in discovering, exploring, and mapping spatiotemporal patterns across the world’s largest open source data sets. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W2:199–205, 2017. doi: 10.5194/isprs-annals-IV-4-W2-199-2017. URL <https://isprs-annals.copernicus.org/articles/IV-4-W2/199/2017/>. [5](#), [6](#), [23](#), [52](#)
- [113] Jesse Piburn, R. Stewart, Aaron Myers, Alexandre Sorokine, E. Axley, D. Anderson, J. Burdette, C. Biddle, Alexander Hohl, R. Eberle, J. Kaufman, and A. Morton. The world spatiotemporal analytics and mapping project (wstamp): Further progress in discovering, exploring, and mapping spatiotemporal patterns across the world’s largest open source data sets. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W2:199–205, October 2017. doi: 10.5194/isprs-annals-IV-4-W2-199-2017. [6](#)
- [114] Jesse Piburn, Robert Stewart, and April Morton. Attribute portfolio distance: A dynamic time warping-based approach to comparing and detecting common spatiotemporal patterns among multiattribute data portfolios. In Daniel A. Griffith, Yongwan Chun, and Denis J. Dean, editors, *Advances in Geocomputation*, pages 197–205, Cham, 2017. Springer International Publishing. ISBN 978-3-319-22786-3. [6](#)
- [115] Jesse O. Piburn, Robert N. Stewart, and April M. Morton. An approximate entropy based approach for quantifying stability in spatio-temporal data with limited temporal observations. In *Proceedings of the International Conference*

- on *GeoComputation*, volume 2017, page 55, Leeds, England, October 2017. University of Leeds. Conference held Sep 4–7, 2017. [6](#)
- [116] Domenico Piccolo. A distance measure for classifying arima models. *Journal of time series analysis*, 11(2):153–164, 1990. ISSN 0143-9782. [27](#)
- [117] Stefano Ponte. Behind the coffee crisis. *Economic and Political Weekly*, 36(46/47):4410–4417, 2001. ISSN 00129976, 23498846. URL <http://www.jstor.org/stable/4411387>. [100](#)
- [118] Stefano Ponte. The ‘latte revolution’? regulation, markets and consumption in the global coffee chain. *World Development*, 30(7):1099–1122, 2002. doi: 10.1016/S0305-750X(02)00032-3. [100](#)
- [119] Kevin Pratt and Eugene Fink. Search for patterns in compressed time series. *International Journal of Image and Graphics*, 02:89–106, 11 2001. doi: 10.1142/S0219467802000482. [15](#)
- [120] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’12, page 262–270, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314626. doi: 10.1145/2339530.2339576. URL <https://doi.org/10.1145/2339530.2339576>. [165](#)
- [121] Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: a survey. *CoRR*, abs/1707.03186, 2017. URL <http://arxiv.org/abs/1707.03186>. [107](#), [108](#)

- [122] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, Nov 2009. ISSN 1951-6401. doi: 10.1140/epjst/e2010-01179-1. URL <http://dx.doi.org/10.1140/epjst/e2010-01179-1>. [114](#)
- [123] Martin Rosvall and Carl T. Bergstrom. Mapping change in large networks. *PLOS ONE*, 5:1–7, 01 2010. doi: 10.1371/journal.pone.0008694. URL <https://doi.org/10.1371/journal.pone.0008694>. [109](#)
- [124] Amin Sadri, Yongli Ren, and Flora D. Salim. Information gain-based metric for recognizing transitions in human activities. *Pervasive and mobile computing*, 38:92–109, 2017. ISSN 1574-1192. [31](#), [33](#)
- [125] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978. doi: 10.1109/TASSP.1978.1163055. [25](#), [165](#), [169](#)
- [126] Hiroaki Sakoe. Dynamic-programming approach to continuous speech recognition. In *1971 Proc. the International Congress of Acoustics, Budapest*, 1971. [11](#), [13](#), [25](#), [165](#)
- [127] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent data analysis*, 11(5):561–580, 2007. ISSN 1088-467X. [25](#)
- [128] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 0123694469. [28](#)
- [129] Patrick Schäfer, Arik Ermshaus, and Ulf Leser. Clasp - time series segmentation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21*, page 1578–1587, New York, NY, USA,

2021. Association for Computing Machinery. ISBN 9781450384469. doi: 10.1145/3459637.3482240. URL <https://doi.org/10.1145/3459637.3482240>. 31, 32
- [130] Paola Sebastiani, Marco Ramoni, Paul Cohen, John Warwick, and James Davis. Discovering dynamics using bayesian clustering. In David J. Hand, Joost N. Kok, and Michael R. Berthold, editors, *Advances in Intelligent Data Analysis*, pages 199–209, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48412-7. 16, 27
- [131] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, pages 1–23, 2008. 165
- [132] Joan Serrà and Josep Ll. Arcos. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67:305–314, 2014. ISSN 0950-7051. doi: <https://doi.org/10.1016/j.knosys.2014.04.035>. URL <https://www.sciencedirect.com/science/article/pii/S0950705114001658>. 24, 165
- [133] Hagit Shatkay and Stanley B. Zdonik. Approximate queries and representations for large data sequences. In *Proceedings of the Twelfth International Conference on Data Engineering, ICDE '96*, page 536–545, USA, 1996. IEEE Computer Society. ISBN 0818672404. 10, 15, 17, 21, 22, 32, 33
- [134] Jin Shieh and Eamonn Keogh. *i2i/i2sax*: Indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08*, page 623–631, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581934. doi: 10.1145/1401890.1401966. URL <https://doi.org/10.1145/1401890.1401966>. 17, 18, 29
- [135] Gota Shirato, Natalia Andrienko, and Gennady Andrienko. Identifying, exploring, and interpreting time series shapes in multivariate time intervals.

- Visual Informatics*, 7(1):77–91, 2023. ISSN 2468-502X. doi: <https://doi.org/10.1016/j.visinf.2023.01.001>. URL <https://www.sciencedirect.com/science/article/pii/S2468502X23000013>. [10](#), [12](#), [20](#), [23](#)
- [136] Eugen Slutsky. The summation of random causes as the source of cyclic processes. *Econometrica*, 5(2):105–146, 1937. ISSN 00129682, 14680262. URL <http://www.jstor.org/stable/1907241>. [14](#)
- [137] Padhraic Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems*, pages 648–654, 1997. ISBN 0262100657. [27](#)
- [138] Alexandra Stefan, Vassilis Athitsos, and Gautam Das. The move-split-merge metric for time series. *IEEE transactions on knowledge and data engineering*, 25(6):1425–1438, 2013. ISSN 1041-4347. [26](#)
- [139] Sveinn Steinarsson. Downsampling time series for visual representation. Master’s thesis, University of Iceland, Reykjavik, Iceland, June 2013. [23](#)
- [140] R. Stewart, J. Piburn, A. Sorokine, A. Myers, J. Moehl, and D. White. World spatiotemporal analytics and mapping project (wstamp): Discovering, exploring, and mapping spatiotemporal patterns across the world’s largest open source data sets. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-4/W2:95–102, 2015. doi: 10.5194/isprsannals-II-4-W2-95-2015. URL <https://isprs-annals.copernicus.org/articles/II-4-W2/95/2015/>. [5](#), [52](#)
- [141] Robert Stewart, Jesse Piburn, Alexandre Sorokine, Aaron Myers, and Devin A. White. World spatiotemporal analytics and mapping project (wstamp): Discovering, exploring, and mapping spatiotemporal patterns across the world’s largest open source geographic data sets. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-4/W2:95–102, July 2015.

ISSN 2194–9050. doi: 10.5194/isprsannals-II-4-W2-95-2015. URL <https://www.osti.gov/biblio/1236577>. 6

- [142] Robert Stewart, Jesse Piburn, **Evan Ezell**, David Anderson, Eric Axley, Jason Kaufman, Sarah Walters, Alexandre Sorokine, Gregory Simpson, Josh Grant, and Brian Eaton. Wstamp 2.7 update. In *Geocomputation 2021*, October 2021. 5, 23, 52
- [143] Robert Stewart, Jesse Piburn, Sarah Walters, Jason Kaufman, **Evan Ezell**, David Anderson, Eric Axley, Josh Grant, Bryan Eaton, Alex Sorokine, and Greg Simpson. A taxonomic classification approach for global spatio-temporal data. In *Spatial Data Science Symposium 2021 (SDSS 2021) Short Paper Proceedings*. University of California Open Access, December 2021. doi: 10.25436/E22S3H. 6, 10, 12, 18, 20, 23, 25, 33, 34, 49
- [144] Zbigniew R. Struzik and Arno Siebes. Measuring time series’ similarity through large singular features revealed with wavelet transformation. In *10th International Workshop on Database & Expert Systems Applications, Florence, Italy, September 1-3, 1999, Proceedings*, pages 162–166. IEEE Computer Society, 1999. doi: 10.1109/DEXA.1999.795160. URL <https://doi.org/10.1109/DEXA.1999.795160>. 16
- [145] Zbigniew R. Struzik, Arno Siebes, Jan Zytkow, Jan Rauch, Jan van Leeuwen, Jan M. Zytkow, and Jan Rauch. The haar wavelet transform in the time series similarity paradigm. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1704 of *Lecture Notes in Computer Science*, pages 12–22. Springer Berlin / Heidelberg, Germany, 1999. ISBN 3540664904. 26
- [146] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981. doi: 10.1109/TSMC.1981.4308636. 116, 120

- [147] Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 717–726, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936097. doi: 10.1145/1281192.1281269. URL <https://doi.org/10.1145/1281192.1281269>. [124](#)
- [148] Prodromos Tsinaslanidis and Francisco Guijarro. What makes trading strategies based on chart pattern recognition profitable? *Expert Systems*, 38(5), 2021. ISSN 0266-4720. doi: 10.1111/exsy.12596. [33](#)
- [149] Philippe Vanhems, Alain Barrat, Ciro Cattuto, Jean-Francois Pinton, Nagham Khanafer, Corinne Regis, Byeul-a Kim, Brigitte Comte, and Nicolas Voirin. Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. *PLOS ONE*, 8(9):1–9, 09 2013. doi: 10.1371/journal.pone.0073970. URL <https://doi.org/10.1371/journal.pone.0073970>. [123](#)
- [150] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. [104](#)
- [151] C. Vehlou, F. Beck, P. Auwarter, and D. Weiskopf. Visualizing the evolution of communities in dynamic graphs. *Computer Graphics Forum*, 34(1):277–288, 2015. doi: 10.1111/cgf.12512. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12512>. [108](#), [109](#), [110](#), [116](#), [120](#), [131](#), [132](#)
- [152] V.M. Velichko and N.G. Zagoruyko. Automatic recognition of 200 words. *International Journal of Man-Machine Studies*, 2(3):223–234, 1970. ISSN 0020-7373. doi: [https://doi.org/10.1016/S0020-7373\(70\)80008-6](https://doi.org/10.1016/S0020-7373(70)80008-6). URL [https://doi.org/10.1016/S0020-7373\(70\)80008-6](https://doi.org/10.1016/S0020-7373(70)80008-6). URL [https://doi.org/10.1016/S0020-7373\(70\)80008-6](https://doi.org/10.1016/S0020-7373(70)80008-6).

[//www.sciencedirect.com/science/article/pii/S0020737370800086](https://www.sciencedirect.com/science/article/pii/S0020737370800086). 11, 13, 25, 165

- [153] Luca Vismara, Giuseppe Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, and Francesco Vargiu. Experimental studies on graph drawing algorithms. *Softw., Pract. Exper.*, 30:1235–1284, 09 2000. doi: 10.1002/1097-024X(200009)30:11<1235::AID-SPE339>3.0.CO;2-B. 120
- [154] Michail Vlachos, Philip Yu, and Vittorio Castelli. On periodicity detection and structural periodic similarity. In *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, pages 449–460, 2005. 26
- [155] Chengyu Wang, Kui Wu, Tongqing Zhou, and Zhiping Cai. Time2state: An unsupervised framework for inferring the latent states in time series data. *Proc. ACM Manag. Data*, 1(1), May 2023. doi: 10.1145/3588697. URL <https://doi.org/10.1145/3588697>. 31
- [156] Chengyu Wang, Xionglve Li, Tongqing Zhou, and Zhiping Cai. Unsupervised time series segmentation: A survey on recent advances. *Computers, Materials and Continua*, 80(2):2657–2673, 2024. ISSN 1546-2218. doi: <https://doi.org/10.32604/cmc.2024.054061>. URL <https://www.sciencedirect.com/science/article/pii/S1546221824005617>. 30, 31, 32
- [157] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, Mar 2013. ISSN 1573-756X. doi: 10.1007/s10618-012-0250-5. URL <https://doi.org/10.1007/s10618-012-0250-5>. 24
- [158] Zeyu Wang, Peng Wang, Themis Palpanas, and Wei Wang. Graph- and tree-based indexes for high-dimensional vector similarity search: Analyses,

- comparisons, and future directions. *IEEE Data Eng. Bull.*, 46:3–21, 2023. URL <https://api.semanticscholar.org/CorpusID:265509541>. 30
- [159] J. Warfield. Crossing theory and hierarchy mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, 7:505–523, 1977. 119
- [160] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases, VLDB '98*, page 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605665. 29
- [161] Daniel Wu, Divyakant Agrawal, Amr El Abbadi, Ambuj K. Singh, and Terence R. Smith. Efficient retrieval for browsing large image databases. In *CIKM '96, Proceedings of the Fifth International Conference on Information and Knowledge Management, November 12 - 16, 1996, Rockville, Maryland, USA*, pages 11–18. ACM, 1996. doi: 10.1145/238355.238365. URL <https://doi.org/10.1145/238355.238365>. 16
- [162] Jinfei Xie and Wei-Yong Yan. Pattern-based characterization of time series. *International Journal of Information and Systems Sciences*, 3(3):479–491, 2007. 10, 12, 20, 21
- [163] Guotong Xue, Ming Zhong, Jianxin Li, Jia Chen, Chengshuai Zhai, and Ruochen Kong. Dynamic network embedding survey. *Neurocomputing*, 472:212–223, 2022. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2021.03.138>. URL <https://www.sciencedirect.com/science/article/pii/S0925231221016234>. 72
- [164] Djamel Edine Yagoubi, Reza Akbarinia, Florent Masseglia, and Themis Palpanas. Dpisax: Massively distributed partitioned isax. In *2017 IEEE*

- International Conference on Data Mining (ICDM)*, volume 2017-, pages 1135–1140. IEEE, 2017. ISBN 9781538638347. [30](#)
- [165] Djamel-Edine Yagoubi, Reza Akbarinia, Florent Massegla, and Themis Palpanas. Massively distributed time series indexing and querying. *IEEE transactions on knowledge and data engineering*, 32(1):108–120, 2020. ISSN 1041-4347. [30](#)
- [166] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1317–1322, 2016. doi: 10.1109/ICDM.2016.0179. [11](#), [15](#), [31](#), [35](#), [38](#), [62](#), [63](#), [69](#), [70](#), [73](#), [76](#)
- [167] Chin-Chia Michael Yeh, Nickolas Kavantzias, and Eamonn Keogh. Matrix profile vi: Meaningful multidimensional motif discovery. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 565–574, 2017. doi: 10.1109/ICDM.2017.66. [70](#), [73](#), [79](#), [105](#)
- [168] Byoung-Kee Yi and Christos Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB '00*, page 385–394, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607153. [16](#), [24](#)
- [169] Daren Yu, Xiao Yu, Qinghua Hu, Jinfu Liu, and Anqi wu. Dynamic time warping constraint learning for large margin nearest neighbor classification. *Inf. Sci.*, 181:2787–2796, 07 2011. doi: 10.1016/j.ins.2011.03.001. [25](#), [170](#)
- [170] Shun-Zheng. Yu. *Hidden Semi-Markov Models : Theory, Algorithms and Applications*. Elsevier, San Francisco, 2015. ISBN 9780128027714. [31](#)

- [171] G. Udny Yule. On a method of investigating periodicities in disturbed series, with special reference to wolfer’s sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226:267–298, 1927. ISSN 02643952. URL <http://www.jstor.org/stable/91170>. 14
- [172] Masoomeh Zameni, Amin Sadri, Zahra Ghafoori, Masud Moshtaghi, Flora D. Salim, Christopher Leckie, and Kotagiri Ramamohanarao. Unsupervised online change point detection in high-dimensional time series. *Knowl. Inf. Syst.*, 62(2):719–750, February 2020. ISSN 0219-1377. doi: 10.1007/s10115-019-01366-x. URL <https://doi.org/10.1007/s10115-019-01366-x>. 30
- [173] Zheng Zhang, Ping Tang, and Rubing Duan. Dynamic time warping under pointwise shape context. *Information Sciences*, 315:88–101, 2015. ISSN 0020-0255. doi: <https://doi.org/10.1016/j.ins.2015.04.007>. URL <https://www.sciencedirect.com/science/article/pii/S0020025515002753>. 25, 170
- [174] Zheng Zhang, Romain Tavenard, Adeline Bailly, Xiaotong Tang, Ping Tang, and Thomas Corpetti. Dynamic time warping under limited warping path length. *Information Sciences*, 393:91–107, 2017. ISSN 0020-0255. doi: <https://doi.org/10.1016/j.ins.2017.02.018>. URL <https://www.sciencedirect.com/science/article/pii/S0020025517304176>. 25, 170
- [175] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth Funning, Abdullah Mueen, Philip Brisk, and Eamonn Keogh. Exploiting a novel algorithm and gpus to break the ten quadrillion pairwise comparisons barrier for time series motifs and joins. *Knowledge and information systems*, 54(1):203–236, 2018. ISSN 0219-1377. 79
- [176] Kostas Zoumpatianos, Stratos Idreos, and Themis Palpanas. Indexing for interactive exploration of big data series. In *Proceedings of the ACM SIGMOD*

International Conference on Management of Data, pages 1555–1566, 2014.
ISBN 9781450323765. [30](#)

- [177] Kostas Zoumpatianos, Stratos Idreos, and Themis Palpanas. Rinse: Interactive data series exploration with ads. In *Proceedings of the VLDB Endowment*, volume 8, pages 1912–1915, 2015. [30](#)
- [178] Kostas Zoumpatianos, Stratos Idreos, and Themis Palpanas. Ads: the adaptive data series index. *The VLDB journal*, 25(6):843–866, 2016. ISSN 1066-8888.
[30](#)

Appendices

A Dynamic Time Warping (DTW)

A.1 Background

Dynamic programming was a strategy to solve optimization problems developed by Richard Bellman in the 1950s while he was working for the RAND Corporation. He gave several invited talks about dynamic programming which included a talk given to the National Academy of Sciences [10] and to the American Mathematical Society [11] in the early 1950s. Bellman later wrote two books on the topic [12, 13] and in the 1960s the usefulness of dynamic programming to applications of many kind became apparent to others. In the 1970s, the dynamic programming algorithm was heavily studied and was successfully used for spoken word recognition by Sakoe, Chiba, Itakura, Myers, Rabiner, Rosenberg, Levinson, and others. [126] and [152] first showed the ability for dynamic programming to detect spoken words, an approach that came to be called Dynamic Time Warping (DTW) in the data mining community. While more popular DTW speech recognition papers were developed later [125, 97, 98], DTW began being used for more than just speech recognition and proved itself useful in many other time series domains.

Using DTW for time series data mining was popularized by [14] and research on the topic has since exploded. Another driving factor in DTW's use has been its ability to perform well compared to other similarity measures on a variety of data sets [34, 132, 45]. Furthermore, the highly cited pioneering work [120] developed an innovative approach to performing exact similarity searches using DTW faster than the gold standard Euclidean distance searches on large datasets. This advancement has garnered further interest in the continued use of DTW as a time series analysis technique due to the feasibility to perform exact and efficient subsequence searches.

For a complete overview of DTW, see [131].

A.2 Method

Let us define time series to be a real-valued sequence where each element represents a measurement at a point in time. More formally, a time series is $T = (t_1, t_2, \dots, t_N)$ where $t_i \in \mathbb{R}$ and $1 \leq i \leq N$. To generate a measure of similarity, we will be comparing a time series query $T_q = (t_1, \dots, t_i, \dots, t_N)$ to a reference time series $T_r = (t'_1, \dots, t'_j, \dots, t'_N)$ determining how similar T_q is to T_r .

DTW is ultimately proposed as an optimization problem which computes the minimum cost temporal alignment between two time series. The cost to align two values is most commonly calculated using Euclidean distance. That is, the cost to align an element $t_i \in T_q$ with an element $t_j \in T_r$ can be defined by the following equation.

$$dist(t_i, t'_j) = \sqrt{(t_i - t'_j)^2} \quad (6)$$

Subsequently, DTW can be defined by the following minimizing optimization equation:

$$DTW(T_q, T_r) = \min_{\pi \in \mathcal{A}(T_q, T_r)} \left(\sum_{(i,j) \in \pi} dist(t_i, t'_j) \right) \quad (7)$$

where an alignment path π (i.e. warping path) is a sequence of index pairs $((i_1, j_1), \dots, (i_k, j_k), \dots, (i_K, j_K))$. The warping path length is K and each pair $\pi_k = (i_k, j_k)$ represents the temporal alignment of the query element t_i to the reference element t'_j at the k^{th} position in the warping path. $\mathcal{A}(T_q, T_r)$ is the universal set of all admissible paths. For a path to be admissible, it must satisfy the following three conditions:

1. **Boundary Condition:** The starting and ending points of the query and reference must be aligned.

$$\pi_1 = (1, 1) \quad (8)$$

$$\pi_K = (N, N) \quad (9)$$

2. **Monotonicity Condition:** The alignment path is monotonically increasing on both indexes, i and j .

$$i_{k-1} \leq i_k \leq i_{k+1} \quad (10)$$

$$j_{k-1} \leq j_k \leq j_{k+1} \quad (11)$$

3. **Continuity Condition:** Every index i and j must appear at least once. This limits the alignment path to only a single step in time, thus preventing long jumps.

$$\pi_k - \pi_{k-1} \in \{(1, 1), (1, 0), (0, 1)\} \quad (12)$$

It is worth noting that the number of possible admissible paths $|\mathcal{A}|$ is finite and equivalent to the Delannoy number, which describes the number of paths between diagonal corners of any arbitrary sized rectangular grid [6]. For our practical purposes, the dynamic programming table or cost matrix is always square and the Delannoy numbers of square matrices are known as the central Delannoy numbers*. The closed formula for computing the central Delannoy number and subsequently the number of admissible paths is given by:

$$Central\text{-}Delannoy(n) = |\mathcal{A}(T_q, T_r)| = \sum_{k=1}^N \binom{N}{k}^2 2^k \quad (13)$$

With the number of temporal alignments increasing exponentially in the size of N , it becomes intractable to compute all possible alignments and compare their costs. This rules out the possibility of determining the minimal temporal alignment in a brute-force fashion for even small sizes of N . Thus, we need a technique such as dynamic programming often coupled with additional optimization strategies to

*The central Delannoy numbers - OEIS A001850. The on-line encyclopedia of integer sequences. oeis.org/A001850.

compute the minimum temporal alignment or warping path in a reasonable amount of time (i.e. at least polynomial time).

Dynamic programming solves the DTW optimization equation (Equation 6) by breaking it down into recursively defined sub-problems which can be exploited to make computation more efficient. The DTW algorithm formulates the sub-problems as prefixes of the optimization equation. Let's consider the following sub-problem definition:

$$\delta_{i,j} = DTW(T_q \rightarrow i, T_r \rightarrow j) \quad (14)$$

where $T_q \rightarrow i = (t_1, \dots, t_i)$ and $T_r \rightarrow j = (t'_1, \dots, t'_j)$. Thus, $\delta_{i,j}$ is the best alignment of the prefix sequences of the query and reference time series up to and including t_i and t'_j , respectively. Relating the sub-problems, we can see that

$$\delta_{i,j} = \min_{\pi \in \mathcal{A}(T_q \rightarrow i, T_r \rightarrow j)} \left(\sum_{(m,n) \in \pi} \text{dist}(t_m, t'_n) \right) \quad (15)$$

$$= \text{dist}(t_i, t'_j) + \min_{\pi \in \mathcal{A}(T_q \rightarrow i, T_r \rightarrow j)} \left(\sum_{(m,n) \in \pi_{\hat{K}}} \text{dist}(t_m, t'_n) \right) \quad (16)$$

$$= \text{dist}(t_i, t'_j) + \min \begin{cases} \delta_{i-1,j} & \text{cost to align } T_q \rightarrow (i-1) \text{ and } T_r \rightarrow j \\ \delta_{i,j-1} & \text{cost to align } T_q \rightarrow i \text{ and } T_r \rightarrow (j-1) \\ \delta_{i-1,j-1} & \text{cost to align } T_q \rightarrow (i-1) \text{ and } T_r \rightarrow (j-1) \end{cases} \quad (17)$$

Equation 16 follows from the Boundary Condition (Equation 9), since we must align the last elements of the query and reference. $\pi_{\hat{K}} = (\pi_1, \dots, \pi_{K-1})$ is a path π with the last alignment removed from the sequence or the path up to but not including π_K . Equation 17 follows from the Continuity Condition (Equation 12) restricting the penultimate alignment of π to three possibilities.

We can further define the warping path as a matrix (A_π) , such that:

$$(A_\pi)_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \pi \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

An example warping path matrix is shown in Figure A.1. In this example, the first point of the reference time series gets mapped onto the first four points of the query time series. The ending of the warping path aligns the last point of the query to the last four points of the reference. This example demonstrates how the comparison of two time series with the same shape (sine wave period), albeit occurring at different times, get aligned properly.

A.3 Variants and Optimization

Since the popularization of DTW, there have been many variants of DTW proposed in the literature [1]. These variants have generally been developed for specific purposes or to strive for improvement over the original DTW algorithm. While different variants are shown to outperform the original DTW algorithm on a case-by-case basis, the original DTW algorithm with a Sakoe-Chiba warping band constraint does not perform statistically worse than other techniques on a wide variety of data sets [45]. Other concerns about DTW include the possibility of a pathological alignment between two time series. A pathological warping path presents itself when a relatively small number of features of one time series get mapped onto a large number of features of the other time series, producing an unintended correspondence [63].

Furthermore, there have been many additional techniques proposed for how to restrict the warping path and help reduce the the number of pathological warpings. The most widely used warping path constraints are shape-based constraints. The Sakoe-Chiba band [125] imposes that the warping path not stray too far from the diagonal. The Itakura parallelogram [56] is similar to the Sakoe-Chiba band, but

[†]Graphics adapted from Romain Tavenard under CC BY-SA.

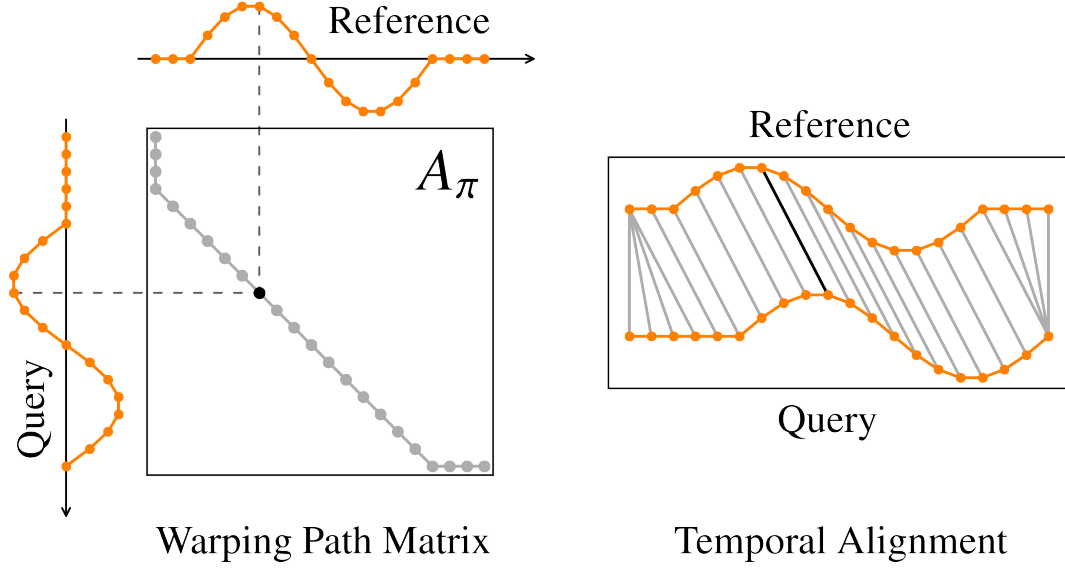


Figure A.1: Warping Path Matrix (Left) and Corresponding Temporal Alignment (Right)[†]

allows the warping path to map to further points in the center of the warping path matrix. The shape constraint can also be generated through machine learning [17, 169, 173]. Figure A.2 shows these approaches visually on the warping path matrix. Another approach to restrict warping path is by controlling the length of the overall path [174]. In addition to conciliating illogical warping paths, the constraint of a warping band inherently reduces computational complexity.

While pathological warpings are a legitimate concern in most practice, the environment in which we are using DTW is more controlled – we are calculating DTW over shorter length windows and our reference time series are synthetically produced eliminating noise (see Section 2.4.2). The use of the Sakoe-Chiba warping band and our controlled environment should be sufficient in allaying most situations in which a pathological alignment might occur. However, in certain situations such as a short time series segment, degenerate warpings can still happen (see Figure 2.15).

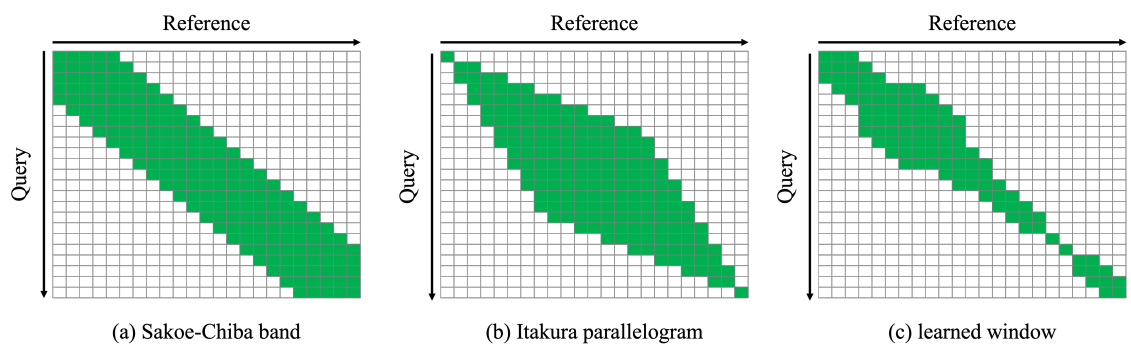


Figure A.2: Warping Path Constraints

Vita

Evan Ezell, a native of Georgia raised in Canton, developed an interest in computer science and mathematics early in life. After high school, he earned a Bachelor of Arts in Computer Science from Maryville College in 2018. Ezell continued his academic journey at the University of Tennessee, where he received a Master of Science in Computer Science in 2020 and a Doctor of Philosophy in Data Science and Engineering in 2025. During his graduate studies, he served as a Graduate Research Assistant at Oak Ridge National Laboratory in the Geographic Data Science Section for four years and worked as a Senior Research Engineer at Graf Research Corporation for two years.