Katrine Lie Holm

Surveying Similarly Distance for Trajectory Data

Master's thesis in Computer Science
Supervisor: Professor Svein Erik Bratsberg
February 2022

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Katrine Lie Holm

Surveying Similarly Distance for Trajectory Data

Master's thesis in Computer Science
Supervisor: Professor Svein Erik Bratsberg
February 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Surveying Similarly Distance for Trajectory Data

Katrine Lie Holm

February 7, 2022

Abstract

In tandem with an increase in devices that can detect and store motion data, the interest in analyzing this type of data has grown. Computing similarity distance is crucial for all types of analysis, and for trajectory data, there are several algorithms that compute it.

In this thesis, we theoretically and experimentally investigate seven different similarity distance measures. The theoretical part of the thesis examines different notions of trajectory similarly and relates it back to the trajectory data format. The experimental part of the thesis is focused on how the theory unfolds in practice. We analyze a data set of arbitrary trajectories. The analysis emphasizes how the different algorithms use different notions of similarity.

Sammendrag

I takt med en økning av enheter som kan registrere og lagre bevegelsesdata har interessen for å analysere denne dataen vokst. Å begrene likhetsavstand mellom observasjoner er avgjørende for å kunne gjennomføre all type analyse. For tidsrekkedata finnes det mange ulike algoritmer som måler deres likhetsavstand

Denne oppgaven består av teoretiske og eksperimentelle undersøkelser av syv likhetsavstandsmål. Den teoretiske delen greier ut om ulike mål av sporlikhet, og hvordan disse er inspirert av dataformatet til tidsrekker. Den eksperimentelle delen av oppgaven handler om hvordan teorien utspiller seg i praksis. Vi analyserer et datasett bestående av tilnærmet vilkårlige spor. Analysen tydeliggjør hvordan de ulike algoritmene bruker forskjellige definisjoner av likhet.

Preface

This thesis was written during the fall of 2021 and completed at the beginning of 2022 as part of the M.Sc. degree in Computer Science at the Norwegian University of Science and Technology (NTNU) in Trondheim.

I would like to my supervisor Professor Svein Erik Bratsberg for his guidance, advice, and patience. The feedback and suggestions he provided have been indispensable for the development and completion of this project.

I would also like to thank my friends for supporting me through many late nights and stress-filled days as the thesis deadline crept closer.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

Several modern devices can collect location data which is usually stored as trajectories. With the rise of available data, the interest in analyzing it has been increasing as well. In order to achieve a meaningful analysis, we depend on the similarity operator, but challenges arise when narrowing now what it means for trajectories to be similar.

In computer vision, object outlines can be mapped as trajectories and thus their shape similarity becomes essential for recognition. Furthermore, the ability to determine the similarity between trajectories is essential to trajectory database management. Pattern mining, classification, outlier detection, observational uncertainty, and forecasts are examples of queries that depend on a similarity measure[1–4].

Trajectories have a compound data format; they are a series of observations that vary over time and location. The location data may itself be multi-dimensional. Complex geometric shapes do not have a simple notion of shortest distance. This gives rise to different interpretations of how to quantify trajectory similarity and in turn the development of different algorithms.

In this thesis, seven different similarity measures are used to perform a comparative study. In contrast to the majority of existing work, we do not seek an ordered ranking of the most efficient or accurate measures. Rather, the aim is to accurately describe the properties of the selected measures discussed and guide someone to pick the most appropriate measure of them for their applications.

1.2 Research Objectives

Given the purpose of this thesis, we formulate the research objectives as the following:

Research Objective 1: Describe qualities and traits of both conventional and newer similarity measures.

Research Objective 2: Determine how different similarity algorithms treat trajectory features and display these contrasts visually.

Research Objective 3: Shed light on how similarity algorithm selection matters for a specific application.

1.3 Thesis Outline

The rest of the thesis is structured as follows:

Chapter 2 presents concepts and notations that will be used throughout the thesis. We narrow down the definition of a trajectory, discuss how to concretize similarity between two trajectories, and review trajectory mining as a comparative technique for trajectory similarity.

Chapter 3 pitches seven trajectory similarity algorithms. We describe how they quantify alikeness and bring up their associated advantages and shortcomings.

Chapter 4 briefly recounts the contents and conclusions of existing comparative studies of similarity algorithms.

Chapter 5 proceeds to define the implementation details of the experiments. It also describes the methodology, how we evaluated the results, and discloses the simplifications that were made.

Chapter 6 is a presentation of the results. They are presented in three parts: the similarity scores, the results from the mining tasks, and evaluation of those results.

Chapter 7 discusses the results that were presented in the preceding chapter. We emphasize what each similarity measure's result was like in comparison to its theoretical characterization.

Chapter 8 contains the concluding remarks for this thesis. We reflect on what has been achieved and revisit the research objectives. Finally, we discuss how this set up could be advanced for future work.

Chapter 2

Theory

This section is largely based on the work done for the course *TDT4501 Computer Science, Specialization Project*[5]. It is assumed that the readers of this thesis will not have read the final report and thus some of its content is restated. This chapter aims to summarize the related to trajectories and the manners of computing their similarity. We define terms and notation that will be used throughout this work and close the chapter by providing background for trajectory similarity usages.

2.1 Movement Data

Collecting data on how *something* moves is done in a wide variety of fields. Some of the applications of movement data analysis are found in zoology, finance, forecasts, navigation, and medicine[6–9]. The fields process the movement data differently and there different types of moment data that each process.

The type of movement data we will examine here are trajectories, a data format whose prevalence is unmistakable.

2.1.1 Collection Techniques

A trajectory is a record of movement data commonly represented as a timestamped sequence of positions. N. Andrienko et al. noted that there are four ways to observe and record movement data[6]:

- *Time-based* collection means that the position is recorded at equidistant time intervals. For instance, if we were to record the location of an entity every other minute it would be time-based collection.
- *Change-based* collection means that a new location is recorded when an entity has moved a set distance from its previous location. An example of this would be sensors in smartwatches that count steps.

- *Location-based* collection means that records are created when an entity enters a specific area. This is useful for tracking location changes over large distances and when precision is not vital.
- For *event-based* data collection the data is only recorded if a set of predetermined events occur. In particular, the moving entity itself should trigger the data sampling. A triggering event could be a user electing to share their current location.

Lastly, the authors discussed a *mixed*-solution. As the name indicates, this collection method acknowledges that the listed techniques are not mutually exclusive. In fact, depending on the application it may be desirable to combine them. In the case of tracking taxi trips, data collection may be done via time-based techniques during the ride. However, the only time the data is of interest is when there is a passenger in the vehicle. A passenger getting into a taxi would then become the triggering event for movement data collection.

2.1.2 Notation

As stated, a trajectory is a time-stamped sequence of positions. Phrased another way, it is a sequence of measurements with a temporal component which is the definition of a *time-series*, also referred to as simply *series*. Consequently terms and techniques from time-series analysis will be used to characterize trajectories in this work.

In particular, we will use the terms time-series and series when referring to the trajectories. We shall use T_A to denote trajectory A and T_B denotes trajectory B. A trajectory contains a several elements that each encompass temporal and spatial information, see Equation (2.1)

Table 2.1: Notation

Notation	**Description**
T_A or A	Trajectory A
a	An element of T_A
a_i	The ith element of T_A
n_A	The length of T_A
$Dist_X(A,B)$ or $X(A,B)$	Distance between trajectory A and B with measure X
$d(a_i,b_j)$ or $d(a,b)$	Distance between given trajectory elements of A and B
$Rest(A)$	Trajectory A without its leading element

$$T_A = [a_1, \cdots, a_n] \quad \text{where } a_i = (\mathrm{t}_i,\ a_{i,\mathrm{LOC}}) \tag{2.1}$$

Where n is the number of trajectory elements in T_A, t_i the timestamp, and $a_{i,\mathrm{LOC}}$ is the location. The dimensionality of this component is most commonly either two or three[10].

In this thesis, trajectories have been simplified to consist of a series of locations in 2-d space and the temporal aspect is implicit. See Equation (2.2) for the format of the trajectory elements. This simplification is common where time-based collection is used to gather data.

$$a_i = [a_{i,X},\ a_{i,Y}] \tag{2.2}$$

where X is the longitudinal aspect and Y is the latitudinal one. See Table 2.1 for additional notation used throughout in this thesis.

2.2 Trajectory Similarity Distance

Informally, a measure is something that numerically quantifies something. The mathematical definition of a measure requires precise terminology; studying that definition is far beyond the scope of this thesis. Consequently, we present a simplified definition of a measure that complies with the proper one[11, 12]:

Definition 1 *A measure on the set S is a function μ that assigns a non-negative numeric value to each subset of S such that for $E, F \subset S$:*

$$\begin{aligned} \mu(E) &\geqslant 0 && \textit{Non-negativity} \\ \mu(\emptyset) &= 0 && \textit{Null empty set} \\ E, F \textit{ disjoint } \Rightarrow \mu(E \cup F) &= \mu(E) + \mu(F) && \textit{Countable additivity} \end{aligned}$$

Another concept from mathematics we wish to bring forth is the notion of a *metric*[13].

Definition 2 *A function $f(a, b)$ on S is a metric if for $a, b, c \subset S$ the following requirements are met:*

$$\begin{aligned} f(a,b) &\geqslant 0 && \textit{Non-Negativity} \\ f(a,b) = 0 &\Leftrightarrow a = b && \textit{Identity} \\ f(a,b) &= f(b,a) && \textit{Symmetry} \\ f(a,c) &\leqslant f(a,b) + d(b,c) && \textit{Triangle Inequality} \end{aligned}$$

In mathematics, a metric is an abstraction of what distance means between members of a set. There are a number of generic methods that are designed to work with metric distance functions. Two examples of applications where a measure's metricity is required are sub-linear time approximation for clustering and index-based search and pruning[14–16]. About half of the similarity algorithms we discussed in Chapter 3 are not proper metrics. Nevertheless, a distance function does not have to be a metric to be applicable for some selected data mining tasks. [17].

Now that we have established what a measure is, we turn our focus to what similarity means for trajectory data. We refer to work done by Golding and Kanellakis who formalized the notion of two series' similarity distance.[18]. They did so by modifying a version of *The Matching Problem* which we have simplified and restated below:

> *Given a query series* $\mathbf{Q}$ *of length* $\mathbf{n}$ *and another series* $\mathbf{S}$ *of length* $\mathbf{N}$*, where* $\mathbf{N}$ *is much larger than* $\mathbf{n}$*, we are searching for a contiguous subsequence within* $\mathbf{S}$ *that is identical to* $\mathbf{Q}$

Their work defined term *Similarity Distance* as seen in Definition 3.

Definition 3 *The similarity distance, D, between* T_A *and* T_B *is a numerical value produced by a given distance measure X:*

$$Dist_X(T_A, T_B) = D$$

In essence, this thesis is a comparative study of how different measures affect similarity distance. It is generally understood that a short distance signifies closeness; in other words, the measure X has to be a dissimilarity measure.

The distinction between similarity and dissimilarity measures is generally is useful as they are opposite maximizing functions. The similarity function is useful for detecting duplicates and automatic grouping whereas the dissimilarity function is more appropriate for identifying outliers in a data set. [19]. Nevertheless, the term similarity will be used to describe both types of alikeness quantification. This is both due to being a natural shorting of *similarity distance* and it being being used as such in the literature[20–23].

2.2.1 Elasticity of Methods and Time Shifting

Although the trajectories in this thesis are defined with implicit temporal components, there are characteristics of time-series, and thereby time-series similarity, which are easier understood when the temporal value is explicitly stated.

Definition 4 *A similarity function is* ***elastic*** *if trajectory elements can be compared one-to-many or one-to-none. In contrast, a* ***lock-step*** *function compares* a_i *to* b_i *at every step. It requires the input trajectories to be of equal length.*

Definition 5 ***Global time shifting*** *is a distortion or relative lag between trajectories[24]. Formally we say that B is* ***time shifted*** *by δ from A if the following holds:*

$$A = [(t_1, a_1), \cdots, (t_n, a_n)]$$
$$B = [(t_1, b_1), \cdots, (t_n, b_n)]$$
$$B = Shift(A, \delta) = [(t_1 + \delta, a_1), \cdots, (t_n + \delta, a_n)]$$

Definition 6 ***Local time shifting*** *refers to time-shifts that occur non-uniformly and locally.*

In order to account for similarity under *local time shifts*, the measure has to be *elastic*. This is because these measures can realign trajectories so that corresponding trajectory sections can line up which maximizes shape similarity. We note that it is possible to eliminate the effects of global time shifting by statistical normalization. When a measure adapts to local time shifts, we say that it has *warped* time. The warping could refer to single trajectory elements or whole trajectory segments.

2.2.2 Sectioning Trajectories

With the aim of defining how similar whole trajectories are, we have to determine to decompose the trajectories into *sections*. Next, we must determine how to quantify the distance between those. The manner in which this is done varies according to the different trajectory similarity algorithms.

A straightforward idea is to simply use the raw observations in trajectories as the basis; indeed that is a popular choice. All but one of the algorithms discussed in this thesis uses this approach. This is not the only approach, we observe that there are measures that remap the trajectories to direction vectors or sets before computing their similarity [1, 25].

If the choice has been made to evaluate individual points as trajectory sections, it is possible to use any number of point-to-point distance functions. The most common choices are the L_p*- metrics* and the *Haversine distance*. If a trajectory spans a great distance, we need to account for the curvature of the Earth. The Haversine distance is was developed so that distances could be accurately computed at a scale. However, in this thesis, we assume that the trajectories span a small area, allowing for the assumption is of a flat geometry. This lets us safely use the L_p-metrics, of which L_1 and L_2 are the most used ones. The latter of them is known as the *Euclidean distance*, and almost all of the similarity algorithms examined use it to quantify the distance between trajectory elements. Its definition is seen in Equation (2.3) and for the remainder of this thesis, $d(a, b)$ denotes the Euclidean distance between trajectory elements unless otherwise specified.

$$d(a, b) = \sqrt{(a_X - b_X)^2 + (a_Y - b_Y)^2} \tag{2.3}$$

2.2.3 Classes of Trajectory Similarity

With both spatial and temporal components or trajectories explicitly expressed, it becomes evident that weighing them differently gives rise to different types of similarity. Pelekis et al. categorized four classes of trajectory similarity which represent four different ways to account for the trajectories' components[26]. Below is a summary of each of the classes, each with an example of application.

1. When analyzing trajectories, we could value their *Spatio-temporal similarity*. If so, we require trajectories to be alike in shape and to have occurred at around the same time for them to be considered similar.

   An example of where one would need this type of similarity would be when analyzing traffic networks and attempting to detect the roads that have the most congestion during rush hours.

2. *Time-relaxed*, or *Spatial-only, Similarity* as the name implies considers the shape similarity of trajectory. Under this type of similarity the temporal, component does not necessarily affect the final similarity score. A hybrid approach primarily values the shape and then examines the temporal aspect afterward. This differs from *Spatio-Temporal* similarity as the temporal component is treated as a secondary aspect rather than a primary one.

   An example of where one would prefer this kind of similarity is "identifying highways at sea". The routes themselves are the subjects of to interest while the time of which a vessel was in a specific location is not.

3. *Speed-Pattern based spatial similarity* can be conceptualized as an extension of *Spatial-Only similarity*. As the name indicates, the shape of a trajectory and the velocities of the entities are the components that are important for this class.

   This similarity class is instrumental when analyzing data sets to classify entries that move at different velocities. If velocity data have been collected, the temporal component can be disregarded. On the other hand, if the velocity is not recorded the temporal data is implicitly needed so that the rate of change of location, the velocity, can be derived.

4. The last class is *directional similarity*. With this type of similarity, entities that moved in the same directions in the same order are considered to be similar, regardless of the entities' origin. For instance, two entities will have a high similarity score if they both did some movement east and then north-west. As with spatial-only similarity, the temporal component of the trajectory does not affect the similarity score.

   This kind of similarity is used when one wants to examine how something is moving. In particular, it is useful for looking into weather phenomena such as cyclones, hurricanes, and typhoons.

The class of similarity this thesis will focus on is the Spatial-Only class. In other words, we will weigh the geometrical resemblance and disregard the temporal and directional aspects of the trajectories. This leads us back the the notation we laid out in Table 2.1 where $T_A = [a_1, \dots a_n]$ and $a_i = [a_{i,X}, a_{i,Y}]$.

2.3 Trajectory Clustering

Data mining is a field that describes how one can process large data sets so that underlying patterns become apparent. The field is broad and encompasses clustering, classification, outlier detection as well as other tasks[27]. In this thesis, we use trajectory mining, and specifically clustering, as an application for the similarity measures.

To balance out the biases of specific clustering techniques, we have chosen to utilize two of them. It is a well-known issue that there is no ground truth to how to precisely define a cluster[28]. Optimizing trajectory clustering is itself an interesting research area, and great efforts are being put into it[29, 30]. Nevertheless, we deemed that a proper exploration of trajectory clustering methods was out of scope for this thesis. We have settled using well-known clustering techniques. We briefly explain the theory of the two clustering methods we chose and an evaluation index for cluster results.

2.3.1 Affinity Propagation (AP)

Affinity Propagation (AP) was first introduced in 2007 and it creates clusters by passing messages between the data elements[31]. The passing of the messages is an iterative process and the information passed relates to how well-suited the elements are for being the cluster representative. Whether or not a given element is suited to be the representative changes as both the number of clusters and the member count of each cluster change. The algorithm completes either when it has converged, meaning a new iteration does not update the clusters, or when an iteration limit has met.

An advantage of AP is that it does not require the number of final clusters to be known at the start.

2.3.2 Hierarchical Clustering Analysis (HCA)

Hierarchical Clustering Analysis (HCA), or just Hierarchical Clustering, refers to a family of algorithms wherein there are different variation[32]. The version we focus on in this thesis is *agglomerative complete-linkage* clustering. The agglomerative part of the name means that each data point starts out as a cluster, then the most similar clusters are grouped into a new cluster. This happens recursively until all data points are in the same cluster. The second part of the name specifies how one determines which clusters should be merged. Under complete-linkage,

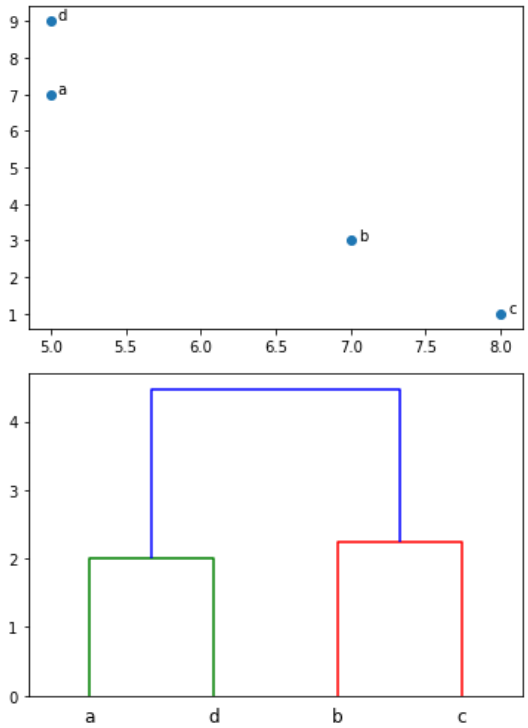


Figure 2.1: Top: A simple scatter plot of some arbitrary data points
Bottom: A dendrogram illustrating the agglomerative process.
Notice that (b, c) are merged after (a, d) indicating that latter are more similar

each cluster finds the farthest neighbors of the other clusters. The clusters that get merged are ones that have the closest "farthest neighbors". This iterative merging can be displayed in a dendrogram, see Figure 2.1

Under hierarchical clustering, any number of final clusters can be set. This means that the number of final clusters is determined up to human discretion, thereby not guaranteeing clearly defined clusters.

2.3.3 Evaluating Clusters

After the clusters have been generated, it is useful to evaluate the result. One way to do this is with the Davis-Bouldin index (DB-index or DB-criterion) is defined as the ratio of the "within-cluster"-distance and "between-cluster"-distance. "Within-cluster"-distance is a description of dispersed elements of a cluster is and "between-cluster"-distance is a description of how far apart the cluster are from each other. This means that a low DB-index is an indication of a good clustering result.

In the context of this, the information gained from the DB-index is minimal. This is because it as well as all other numerical evaluations, depend on a ground truth of how to correctly quantify the similarity distance. The index may provide some in-

sight, however, we expect the true information gain to be minimal and its ranking to be biased.

As a closing remark, we acknowledge that there is no general standard for what constitutes a “reasonable clustering” on account of it being highly domain-dependent. Human inspection remains a vital tool for evaluating final clusterings[33]. In order to discuss the clustering results from a visual perspective, we define the terms *Fuzzy* and *Crisp* as seen in Definitions 7 and 8.

Definition 7 *A cluster is **fuzzy** if its members appear spread out and with seemly little relation to each other. For a "good" clustering result, we wish to minimize the number of fuzzy clusters.*

Definition 8 *A cluster is **crisp** if its members create a clear silhouette with seemly a clear correspondence. For a "good" clustering result, we wish to have as many crisp clusters as possible*

Chapter 3

Similarity Distance Algorithms

In this chapter, we present seven different similarity distance measures. First, we describe four that are parameter-free, then we describe three algorithms that require a parameter. By chance, all of the parametric methods are based on Edit Distance, so we briefly explain how its construction as well.

The methods have largely been chosen due to their prominence in the existing literature[34–38]. However, we present two newer algorithms, one parameter-free and one that requires a parameter. They are presented at the end of their respective groupings.

As was the case in the previous chapter, large portions here are based on the report created for *TDT4501 Computer Science, Specialization Project*

3.1 Parameter-Free Measures

3.1.1 Euclidean Distance (Ed)

As noted in Chapter 2, the Euclidean distance (Ed) is a way to quantify the distance between two points in space. There is a naive extension of that principle that lays the foundation for a Euclidean Distance measure for trajectories[36, 39–41]. This method would be to compute the mean point-point distances in a lock-step manner as seen in Equation (3.1).

$$Ed(A,B) = \frac{1}{n} \cdot \sum_{i=1}^{n} d(a_i, b_i) \tag{3.1}$$

where the distance between corresponding points is the L_2-norm as defined in Equation (2.3)

Its simplicity comes with a couple of disadvantages, one notable one is that it requires the trajectories to be of equal length. In the cases where the number

of points does not match one would have to adapt the data, potentially altering the original observations. Furthermore, computing distance in a lock-step manner means that it is sensitive to local time shifts and noise[42].

Even with these limitations, this measure is included for its simplicity. Previous work has shown that it holds remarkably well up to more advanced methods[43], further encouraging us to examine this metric.

3.1.2 Dynamic Time Warping (DTW)

As stated in Section 2.2.1 an elastic measure is needed in order to handle local time shifts and Dynamic Time Warping (DTW) is precisely that. DTW was originally designed for speech recognition which means that it handles relative lag seamlessly[44, 45]. This makes it a prevalent choice for examining similarity under temporal drift. The idea behind this algorithm is to stretch and contract time such that a favorable alignment of the input trajectories can be found.

Figure 3.1 illustrates the implementation of DTW. The computation begins by taking input trajectories A and B and constructing a full cost matrix C. This matrix keeps all pairwise distances between the trajectories' elements. We use the L_2 norm for point-to-point distance computation as is common for DTW[46]. After the matrix has been constructed, the next step is to iteratively search through C and find the optimal warping path, p. The optimal warping path is the set of point-point pairings, entries of C, that minimizes the accumulated cost. The accumulated cost along the warping path is the DTW distance between the trajectories. Figure 3.2 exemplifies how the Euclidean distance and DTW differ with respect to time-shift.

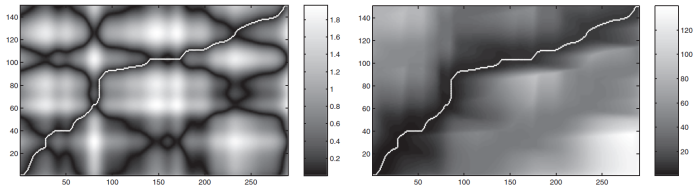


Figure 3.1: Cost matrix C (left) and accumulated cost matrix C' which the optimal warping path p (right) . Figure copied from [45]

The main drawback of DTW is that it weights all points of both trajectories equally and thus it is not robust to noise. A naive implementation of DTW would forcefully align the start- and endpoints of the two trajectories which can disproportionately punish trajectories that are overall similar[47]. Generating the cost matrix quickly becomes expensive due to pairwise comparing all points of the input trajectories which makes it less suited for large data sets [45].

Its shortcomings have inspired iterations of DTW that seek to address them. Some remark that constructing a full cost matrix may not be needed, and some seek to

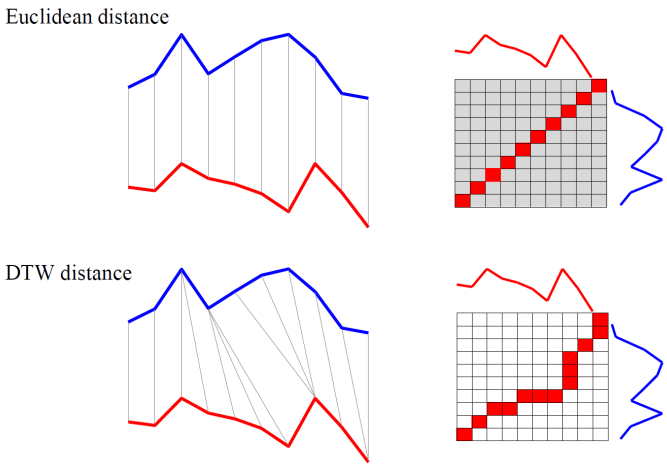


Figure 3.2: An illustration of how the Euclidean forces a lock-step computation whereas DTW realigns the trajectories. Figure copied from [34]

speed up how to iterate through C[46]. We note that a popular choice is **FastDTW**, which is a parameterized version of DTW. As the name indicates, it is much faster than the naive implementation yet gives comparably accurate results[20]. Nevertheless, this thesis shall only focus on the parameter-free version.

3.1.3 Hausdorff Distance (Hd)

This technique is often discussed in relation to polygons and sets[48] however it is applicable for time-series data as well. A consequence of its geometric origin is that the trajectory direction no longer affects the final result, the notion of a first or last trajectory element is not kept[41]. The Hausdorff distance from A to B is the maximum of the minimum of all pairwise distances, see Equation (3.2)[49]

$$\widetilde{Hd}(A,B) = max_{a\in A}\{\, min_{b\in B}\ \ d(a,b)\,\} \tag{3.2}$$

where $d(a,b)$ is any metric distance function, here the L_2-norm is used. The distance function described in Equation (3.2) is not symmetric and thereby not a metric[50]. The function is referred to as the directed Hausdorff distance. To make the Hausdorff metric, it is defined as the maximum of the two directed results. This makes the function symmetric, so that it fulfills the requirements for a metric, see Equation (3.3).

$$Hd(A,B) = max\{\, \widetilde{Hd}(A,B),\ \widetilde{Hd}(B,A)\}\tag{3.3}$$

Since we can compare any point in A to any point in B this measure is elastic. All points of the trajectories are weighted equally which makes it is sensitive to

noisy data. Additionally, the effect of reducing the similarity score to the distance between two points from each trajectory is that information about the trajectories' overall shape is lost. The metric is highly sensitive to noise[4]. Lastly, we need to work out both of the directed distances to get the similarly score. This means more work per trajectory pair than naturally symmetric measures such as Ed and DTW.

3.1.4 Symmetrized Segment-Path Distance (SSPD)

As the Hausdorff distance, Symmetrized Segment-Path Distance (SSPD) method is a spatial only algorithm that largely disregards the direction of the trajectories. It is was developed from the same principles as the Hausdorff distance, but with the addition of being able to account for the trajectories as a whole.[23, 51]. The algorithms explained so far have used the trajectory elements directly as the basis for the computations. For SSPD, this is no longer the case, and thus we define notation and terms to describe this algorithm.

In line with Table 2.1 a, b are elements of Trajectories A, B respectively . A *segment* is the line between two consecutive points, and we use $\breve{a}$ to denote a segment of A. Using indexes, $\breve{a}_i$ is the line between a_i and a_{i+1}.

Next we define the *point-to-segment distance* as shown in Equation (3.4). This distance function requires us to find the point's orthogonal projection onto the segment. If it is within the segment, the distance will be the L_2-norm between the original point a and its projection $\dot{a}$. Otherwise, the L_2- norm to segment's end points is computed nearest those distances becomes the point-to-segment distance.

$$d_{ps}(a_i, \breve{b}_j) = \begin{cases} d(a, \dot{a}) & \text{if } \dot{a} \in \breve{b}_j \\ min\{d(a_i, b_j), d(a_i, b_{j+1})\} & \text{otherwise} \end{cases} \tag{3.4}$$

Where $\dot{a}$ is the orthogonal projection of point a onto segment $\breve{b}_j$ and d is the L_2-norm.

From point-to-segment distance, *point-to-trajectory distance* is defined as shown in Equation (3.5). The point-to-segment distance is computed for all the segments of the trajectory and the lowest one becomes the point-to-trajectory distance.

$$D_{pT}(a_i, B) = min_{\breve{b} \in B}\{\, d_{ps}(a_i, \breve{b}) \,\} \tag{3.5}$$

The *Segment-Path-Distance* (SPD) is directed. It is defined to be the mean of the all point-to-trajectory distances from points of the first trajectory to the second trajectory, Equation (3.6).

$$SPD(A, B) = \frac{1}{n_A} \sum_{i=1}^{n_A} D_{pT}(a_i, B) \tag{3.6}$$

As with the Hd, this distance measure is made symmetric by computing both directed versions first. However unlike Hd, the final result is the mean of the directed results. The name comes from this last step and the final formula is shown in Equation (3.7).

$$SSPD(A,B) = \frac{\big(SPD(A,B) + SPD(B,A)\big)}{2} \tag{3.7}$$

The creators of SSPD note that if one were to use the max function rather than computing the mean upon symmetrification this algorithm would be identical to the Hausdorff function[51]. They note that with their method, this distance function becomes more robust to noise when calculating the mean, as opposed to maximizing or minimizing. Moreover, they commend SSPD for being parameter-free as well as not relying on interpolation between two observed point, preferring to strengthen the importance of observed data. As a closing remark, we note that SSPD does not fulfill the requirements of a metric.

3.2 Parameterized Measures

In this section, we have described measures that require a parameter. All of them are based on String Edit Distance (SED) which is a similarity measure designed for strings. The idea is to count the number of *edits* needed to convert one string into the other using the edit operations are insert, delete and replace. Due to the strings' natural discretization, it is trivial to determine whether or not two symbols are matching. There are variations of implementation but a common choice is to let the cost of an *edit* be 1, creating a formula as seen in Equation (3.8):

$$\text{SED}(S1,S2) = \begin{cases} |S1| & \text{if } |S2| = 0 \\ |S2| & \text{if } |S1| = 0 \\ \text{SED}(S1,\ S2) & \text{if } |S1| = |S2| \\ 1 + min\big\{\text{SED}(Rest(S1),\ Rest(S2)) & \\ \qquad \text{SED}(Rest(S1),\ S2) & \text{otherwise} \\ \qquad \text{SED}(S1,\ Rest(S2)) & \end{cases} \tag{3.8}$$

Real data does not let itself be discretized as fortuitously as strings, thereby leading to the creation of the algorithms below. The measures differ in how they have adapted SED for time series data, and the role of the parameter value changes as well.

3.2.1 Edit Distance on Real Sequence (EDR)

Edit Distance on Real Sequence (EDR)[42] introduces a threshold parameter ϵ that determines if two trajectory elements are "matching". Crucially, there can be no partial matches so the point-to-point distance is either 1 or a 0 as shown in Equation (3.9).

$$d_{edr}(a,b) = \begin{cases} 0 & \text{if} \quad |a_{LNG} - b_{LNG}| \leqslant \epsilon \text{ and } |a_{LAT} - b_{LAT}| \leqslant \epsilon \\ 1 & \text{otherwise} \end{cases} \tag{3.9}$$

where ϵ is the threshold parameter. After determining whether or not two points match with the subcost function we get the full EDR is shown algorithm as seen in Equation (3.10)

$$EDR(A,B) = \begin{cases} n_A & \text{if } n_B = 0 \\ n_B & \text{if } n_A = 0 \\ min\{EDR(\text{ Rest}(A),\ \text{Rest}(B)) + d_{edr}(a,b), & \\ \qquad EDR(\text{ Rest}(A), B) + 1, & \text{otherwise} \\ \qquad EDR(A,\ \text{Rest}(B)) + 1\} & \end{cases} \tag{3.10}$$

where a, b are the leading elements of A, B.

The main advantages of EDR are its resistance to noise and the ability to handle local time shifts. It is resistant to noise because of how it maps the distance between elements to a binary 1 or 0. Toohey and Duckham asserted that EDR performed well in spite of variance in the sampling rate[10].

EDR is not a metric measure as it does not fulfill triangle inequality, but it meets other requirements of metrics. Furthermore, it evaluates similarity as trajectory elements, not taking into account the trajectories' overall shape. We have chosen to include this algorithm as it is a well-studied algorithm. Its prevalence in the literature has led it to be studied alongside DTW and Euclidean Distance.

3.2.2 Edit Distance with Real Penalty (ERP)

Edit Distance with Real Penalty (ERP) was designed to bridge the gap between metrics distance functions and local time shifting tolerant ones. [52]. The design of EDR began with the L_1-norm Equation (3.11)

$$Dist_{L1}(A,B) = \sum_{i=1}^{n} d(a_i, b_i) \qquad \text{where} \qquad d_{L_1}(a_i, b_i) = \sum_{j=1}^{p} |a_{i,j} - b_{i,j}| \tag{3.11}$$

From SED, the notion of a *gap element* is introduced. A gap element is a symbol that could have been deleted from string $S1$ but instead is inserted into string $S2$. The point-to-point distance function would then become what is shown in Equation (3.12).

$$d_{ed}(a_i, b_i) = \begin{cases} 0 & \text{if } a_i = b_i \\ 1 & \text{if } a_i \text{ or } b_i \text{ is a gap} \\ 1 & \text{otherwise} \end{cases} \tag{3.12}$$

Rather than using a constant value to penalize all edit operations uniformly like EDR, ERP differentiates between gap elements and non-gap elements. This distinction is important so that it will be tolerant to local time sifting. Gap elements are penalized with a constant value, while non-gap elements have a real-value cost based on their value. The parameter of ERP, g, is the reference value for cost computation, and the point-to-point distance function is seen in Equation (3.13)

$$d_{epr}(a_i, b_i) = \begin{cases} |a_i - b_i| & \text{if neither is a gap} \\ |a_i - g| & \text{if } b_i \text{ is a gap} \\ |b_i - g| & \text{if } a_i \text{ is a gap} \end{cases} \tag{3.13}$$

Again, the trajectory distance function is an adaption of SED, and the full algorithm for ERP is shown in Equation (3.14)

$$ERP(A,B) = \begin{cases} \sum_{i=1}^{n_A} |a_i - g| & \text{if } n_B = 0 \\ \sum_{i=1}^{n_B} |b_i - g| & \text{if } n_A = 0 \\ min\{ERP(Rest(A),\ Rest(B)) + d_{erp}(a_1,\ b_1), & \\ \quad ERP(Rest(A),\ B) + d_{erp}(a_1,\ g), & \text{otherwise} \\ \quad ERP(A,\ Rest(B)) + d_{erp}(b_1,\ g)\} & \end{cases} \tag{3.14}$$

The main drawback of this method stems from the characteristic that makes it a metric, using differences of real values as costs. This method is a metric as long as g is constant, making the cost of an edit vary the location of the trajectory element. However, this makes ERP sensitive to noise[42].

3.2.3 Move-Split-Merge (MSM)

Move-Split-Merge (MSM) [53] is similar to the aforementioned to the other SED-based approaches in that it calculated the similarity score from how many operations are needed to transform one trajectory into another one. This algorithm is tolerant to temporal misalignments and translation invariant[37] as are EDR and ERP. What distinguishes this algorithm from them is how insertions and deletions are handled. The MSM cost model uses both the value of the element being modified as well the adjacent one whereas ERP only uses the element being modified and EDR uses a constant cost for all operations.

As the name of this algorithm indicates, the possible operations are Move, Split, and Merge, which are then used to emulate the established Edit Distance operations Insert, Delete, and Substitute. The Move operation is a renaming of does the same as Substitute. The Split operation inserts a copy of a given value directly after itself, and the Merge operation deletes a value if it is immediately followed by a matching value[37]. In other words, Split and Merge are each others' inverse. The Insert operation becomes a Split followed by a Move, while Delete becomes a Move followed by a Merge. The MSM parameter, c, is a non-negative value that determines the cost of every Split and Merge operation. See Equations (3.15) to (3.20) for details.

$$A = [a_1, \cdots, a_{i-1}, a_i, a_{i+1}, \cdots, a_{n_A}]$$

$$\text{Move}_{a_i \to b_j}(A) = [a_1, \cdots, a_{i-1}, b_j, a_{i+1}, \cdots, a_{n_A}] \tag{3.15}$$

$$\text{Cost}\big\{\text{Move}_{a_i \to b_j}(A)\big\} = d(a_i, b_j) \tag{3.16}$$

$$\text{Split}_{a_i}(A) = [a_1, \cdots, a_{i-1}, a_i, a_i, a_{i+1}, \cdots, a_{n_A}] \tag{3.17}$$

$$\text{Cost}\big\{\text{Split}_{a_i}(A)\big\} = c \tag{3.18}$$

$$\text{Merge}_{a_i}(A) = [a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_{n_A}] \tag{3.19}$$

$$\text{Cost}\big\{\text{Merge}_{a_i}(A)\big\} = c \tag{3.20}$$

Recall that the $\text{Merge}_{a_i}(A)$ operation is only permitted if $a_i = a_{i+1}$. This algorithm fulfills the requirements of a metric and thus it can be used for indexing and clustering techniques that are designed to function in metric spaces. In terms of computational cost and accuracy, we refer to work done by Bagnall et. al. which alludes to MSM being comparably efficient to ERP and more tolerant to noise DTW[54].

Table 3.1: Quick view of some aspects the similarity distance measures

Name	**Metric**	**Single Element**	**Time Shifts**	**Parameter**	**Noise Tolerant**
ED	Yes	Yes	No	No	No
DTW	No:	Yes	Yes	No	No
Hd	Yes	Yes	Yes	No	No
SSPD	No	No	Yes	No	Yes
EDR	No	Yes	Yes	Yes	Yes
ERP	Yes	Yes	Yes	Yes	No
MSM	Yes	No	Yes	Yes	Yes

3.3 Summary of Algorithms

Table 3.1 summarizes some trajectory similarity features. Remark that none of the measures have matching rows, further indicating that the meaning of trajectory similarly varies with technique implementation.

We reiterate that large portions of this chapter stem from the report that was mentioned at the beginning. Still, there is not a complete overlap of the measures discussed in that report and this thesis.

Chapter 4

Existing Surveys of Time-Series Similarly Distance

With there being a large number of time-series similarity distance functions, it follows that there is an accompanying body of work which seeks to consolidate the scattered information. In this chapter we go over some of the those studies, and provide further motivation for the work conduced in this thesis.

We identify two varieties of studies: those that rank the measures according to some predetermined standard and those that provide comprehensive feature descriptions.

4.1 Ranking Studies

4.1.1 Trajectory Clustering

P. Besse et.al, the creators of SSPD, prefaced their work by examining existing distance functions for trajectories[51]. The measures were categorized into "shape-based" ones and “warping-based” ones, referring to how a measure accounted for the temporal component of the trajectories.

Their work studied the different clusterings that were obtained through clustering tasks where the distance function varies. They determined that the “shape-based" distance measures gave better results than the “warping-based" ones. However, they concluded this based on the cluster quality criteria they defined. The criterions resembled the Davies-Bouldin Index, but it was deconstructed so that the “between-like” and “within-like” evaluations were presented individually.

4.1.2 Time Series Mining

H. Din et.al did an experimental comparison of both time-series representations and similarity distance measures in an effort to categorize mining techniques [43]. Their motivation was that other studies had been too narrowly focused on a specific measure, or that the conclusions were too optimistic.

Their experiments used a nearest neighbor classifier to evaluate nine different similarity measures and eight different data representations. The thoroughness of their setup was further emphasized as they ran their experiments on 38 different time-series data sets. While data sets originated from a variety of real sources, they were all suited to the mining task classification.

Under classification, labels are given to each time-series based on their similarity are to other series. Typically the observations receive one label each. One key way classification differs from clustering is that in that classification results rend to be simpler to evaluate. They are distinct usages of similarity distance and the conclusion drawn in this work are oriented towards the classification results. The features of the similarity measures themselves were not emphasized.

4.2 Non-Ranking Studies

4.2.1 Review of Trajectory Measues

Work done by N. Magdy et.al examined 13 different trajectory similarity measures[40] in a theoretical manner. Of the seven discussed in Chapter 3, five of them were examined by the authors.

Their work did not include an experimental component. The measures were categorized based on implicit similarity definition, and the traits of each measure. Namely if whether or not they were noise-tolerant, could handle locally time-shifted data, if they fulfilled the requirements of metricity and their computational cost.

Furthermore, the researchers contrasted how the required data format varied, and how that affected the notion of similarity. The work concluded by underlining that there is no measure that is the most appropriate for any given data set. They expressed a wish for a generic trajectory similarity measure which could then be adapted to accommodate the desired specific type of similarity depending on the application.

4.2.2 Effectiveness Study

H. Wang et.al conducted an effectiveness study on six trajectory similarity measures[36] and half of them that were also discussed in Chapter 3. After a theoretical summary of the measures, they set up an experiment to test their *effectiveness*. The study was realized on a data set where the researchers had intentionally altered

some of the entries. This was done so that the advantages and drawbacks of their selected measures would be highlighted.

One of the aims of this study was to lay the foundation for a new benchmark that could objectively quantify the effectiveness of similarity measures. They tested the measures under the following trajectory transformations: a simulated sampling rate change, added noise, and shifts of the trajectory, both temporal and spatial. The benchmark compared the original trajectory to a transformed one, changing which similarity measures were used. The measures that recognized them as the same trajectory, or as similar trajectories, were classified as “passing" under that transformation. Using the experimental results they created a table that summarized each measures’ effectiveness under the various transformations.

4.2.3 Vehicle Trajectory Survey

Abas et.al did a comprehensive survey of trajectory data[14] wherein they focused on vehicular data. Their work investigated how the inherent inaccuracies of GPS data affect the quality of trajectory analysis. They surveyed different trajectory representations, noting the advantages and disadvantages of each. Next, they discussed processing techniques that are used to handle the drawbacks of a given representation. Some of the trajectory representations they brought up were road-network constrained trajectories, binary-encoded trajectories, and hash-based trajectories.

A selection of similarity measures was specified for each type of representation and they noted whether or not a measure fulfilled certain properties. The properties in question were metricity, parameter dependency, ability to handle local time shifts, and ability to handle noise. Four of the measures that were were brought up in their work are also discussed in this thesis. One of the research gaps they remarked was how one would process "big trajectory data".

4.3 Our Contribution

This thesis seeks to add the body of work that compares and contrasts trajectory similarity distance measures. The experiment we conducted here is based on various elements of the aforementioned studies.

We did not see a selection of measures matching ours; notably, few studies examine MSM and SSPD which is in all likelihood a testament to their novelty. We remark that the efforts described in “Trajectory Clustering” and “Vehicle Trajectories” have had major influences on our work. We hope to provide a comprehensive comparison wherein feature description and clustering results play equally important roles.

Chapter 5

Set Up and Method

5.1 Technical Environment

The experiments carried out in this thesis were executed in Google Colaboratory (Colab) which is a Google-hosted implementation of Jupyter Notebook[55, 56]. The base components of a Notebook are cells that can be grouped thematically. The cells can contain code or markdown, smoothing out the transition between coding and writing which is conducive to a smooth workflow. Another key advantage of this setup is the ability to work from anywhere. This is possible as the notebook is being executed by Google's cloud servers; there were no specific hardware dependencies for the computations.

Given the choice to work in the Jupyter environment, Python becomes the natural choice of programming language[57]. This choice is further substantiated by the prevalence of existing modules made for data analysis. There are significant computational gains from using an extension of Python called Cython[58].

We used the module `trajectory_distance`[23] to compute almost all the distance functions. There were no implementations of Euclidean Distance and Move-Split-Merge, so we coded our own. The former was trivial to implement. As for the latter, the creators of MSM had published a JAVA implementation of their algorithm. Using that code as a guide we created a "Pure-Python"-Cython cell in the Jupyter Notebook which was used for its similarity distance computation. Due to the differences in implementation, we cannot evaluate the algorithms' time complexity fairly.

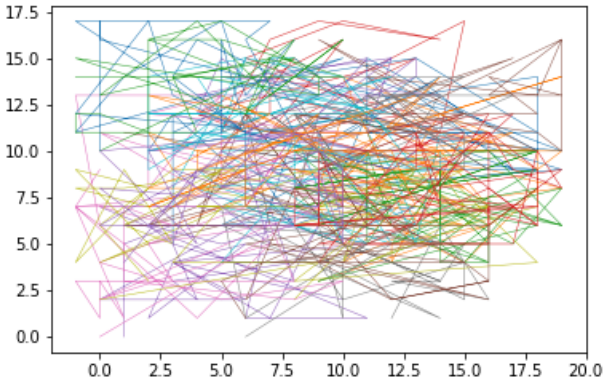


Figure 5.1: Synthetic data set of randomly generated trajectories.

5.2 Data Set

In this thesis, the aim is to gain information about various similarity measures and consequently, the setup should be as data set agnostic as possible. With this in mind, we first generated synthetic data for initial test runs and then modified real trajectory data which we could use for more extensive tests.

5.2.1 Synthetic Trajectory Generation

The generation of synthetic data was the first step of this experiment. The aim of doing this was so that we could determine the architecture of the experiment validate that the distance functions were producing reasonable results. The generated data set can be seen in Figure 5.1. It quickly became evident that the data was too random to serve as a basis for the similarity algorithms. Rather than using more advanced methods for creating synthetic data such that it could potentially accurately represent a complex system, we chose to examine trajectory real data.

5.2.2 Real Trajectory Preprocessing

The data set that was selected for this thesis is a taxi trajectory data set with over 1.7 million rows[59]. In addition to a large number of rows, there were superfluous columns such as what type of taxi ride it was and whether or not the trip was on a weekend. The first step of the data preprocessing was to remove these attributes from the rows. Next, a random subset consisting of just over 500 trajectories was selected. They were given unique IDs their routes can be seen in Figure 5.2a.

As mentioned in the Chapter 3, the Euclidean distance may only evaluate trajectories of the same length. Rather than interpolate to add data points– or compute the information gain of the trajectory elements and then get rid of the least informative ones, we truncated all trajectories to the same length. Moreover, initial tests indicated that a set of 500 trajectories was still too voluminous and time consuming to analyze. ' As a result, we created two subsets of those 500 raw trajectories, one significantly smaller than the other so that tests could be run quickly. The truncated trajectory sets are depicted in Figures 5.2b and 5.2c

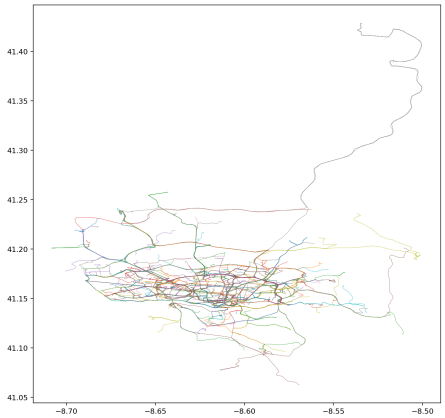
As seen in Figure 5.2 the trajectories are fairly spread out, and so we normalized the range of the coordinates of the data set. We scaled the data so that they would be in the same value range and we could examine similarity on a seemly arbitrary set of lines. ' Without this step, trajectories spanning a small identical geographical area would artificially receive a higher similarity score by being near each other, regardless of shape similarity. This is the re-scaling disconnected data from their real world source while maintaining a general similarity shape from trajectories that followed the same roads. Finally, we remark that the re-scaling was done with simplification of independently scaling the latitudinal and longitudinal dimensions, further distorting their relation to the underlying road network. The final scaled data set is graphed in Figure 5.3.

5.3 Setting Parameter Values

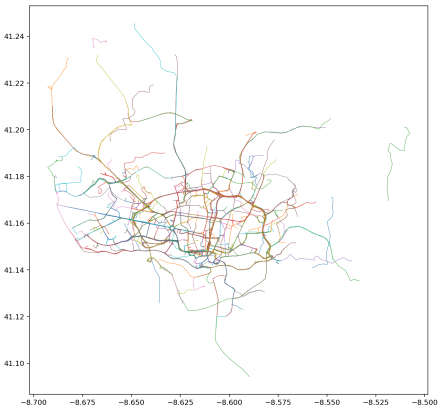
Three of the measures required a parameter value. For each of the measures, we followed the guidelines that the original creators set for determining a fitting value. While the value may have been usable for our tests, we did not fully optimize any of the parameters for this specific data set. This may have negatively impacted their performance, and as a counter, we would test more than one value where we could reasonably determine another value. For details regarding parameters themselves, refer to Chapter 2.

5.3.1 EDR

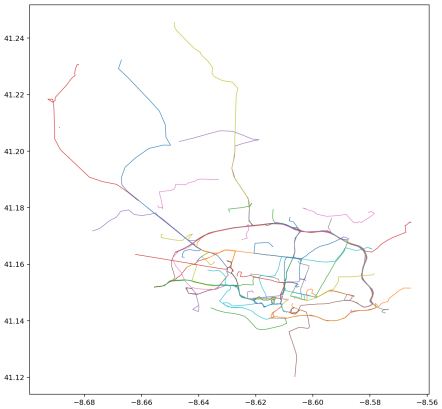
The creators of this method explicitly stated that they achieved the best clustering results when the matching threshold ϵ was set to a quarter of the maximum standard deviation[42]. The time-series data that were used in their research were one-dimensional, thus finding the standard deviations of each series was a straightforward task. Akin to how to how the re-scaling was done, a naive process that isolated the two dimensions was used to estimate a sensible value for ϵ. For each trajectory, the standard deviation of each dimension was calculated. Then the mean of the largest standard deviations from the data set was computed. A quarter of that value was used as the first value of ϵ, while the second value was half; the two values we used for EDR were $\{0.101, 0.203\}$



(a) Subset with 551 entries

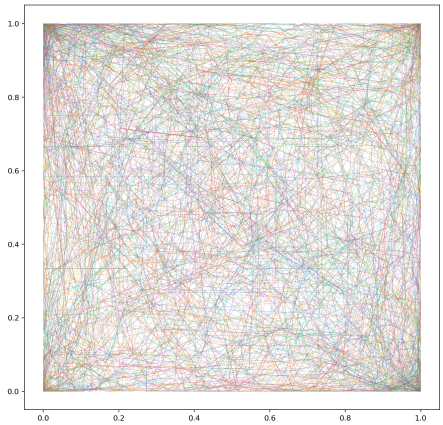


(b) Truncated subset, 353 entries

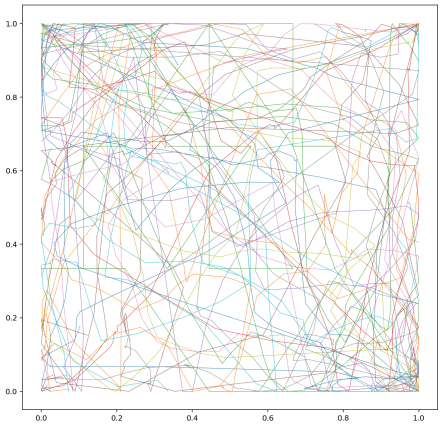


(c) Subset of the truncated data with 53 entries

Figure 5.2: Real data set with trajectories collected from taxis



(a) Scaled version of the data set in Figure 5.2b



(b) Scaled version of the data set in Figure 5.2c

Figure 5.3: The trajectories in the data set after re-scaling

5.3.2 ERP

The researchers that developed ERP stated that the algorithm would be a metric as long as the reference value, g, was fixed[52].

In their paper, they asserted that fixing g at the origin was a natural choice. Consequently, we set $g = (0,0)$ for the tests in this set up. We did not try another value for g due to there being a lack on information of how approximate another suitable value.

5.3.3 MSM

Much like the creators of the algorithm– and subsequent experiments of MSM [38, 53, 60], we let the cost, c, vary over different orders of magnitude. Concretely c took the values $\{1, 0.1, 0.01\}$. These orders of magnitude were chosen based on the results from initial tests executed on the subset of truncated trajectories.

5.4 Clustering Analysis

The first step was to compute all pairwise trajectory similarity scores using each of the similarity distance functions. The next step was to create the clusters and as specified in Chapter 2 two different models were used for this. Details on the clustering models were given in that chapter as well.

The Python's module `sklearn` contains functionality for creating for both types of clustering, as well as models for even more options. The module can take parameters which affect how the model creates the clusters. An example would be setting an iteration limit for affinity propagation. However, in keeping this stage of the experiment as simple as possible, all but one parameter value were kept their defaults.

The value we specified was the number of clusters that the hierarchical model should create. This was necessary as the stopping criterion for agglomerative clustering is to have one cluster that contains all elements. The number chosen was for all intents and purposes arbitrary, but we set it to be the least number of clusters found by affinity propagation. Our motivation for picking this number, and keeping it equal for all measures, was that we wanted to have clusters of meaningful sizes while still being able to spot the variances from a given measure.

Finally, we implemented a version of the Davies-Bouldin Index. Again, the underlying theory is elaborated upon on Chapter 2. The DB-index depends on having a representative observation of each cluster. Commonly this is either the center of a cluster or an exemplar observation that is the one nearest to other ones. Unfortunately, the center of a cluster of series data does not let itself be computed easily– especially in the cases where the trajectories are of unequal length. Furthermore,

determining the exemplar observation would require us to pick a similarity to be the proper one.

Yet again simplifications were made, and the center trajectory of a cluster was set to be the series of mean positions of all trajectories in that cluster. The mean was computed in a lock-step manner using the L_2-norm. For the similarity distance computations necessary for the DB-index calculation, the Euclidean distance was used because of if the simplest one.

5.5 Simplifications

As stated at various points throughout the chapter, we made simplifications while setting up our experiment. The simplifications were made in order to limit the scope, and in some cases due to the time constraints surrounding the thesis. We close the chapter by summarizing them below.

- The trajectories were truncated to equal length and then re-scaled so that they did not remain true to their original relative location.
- At several points, longitude and latitude were treated as if they were independent series. This becomes particularly important with respect to re-scaling and cluster evaluation.
- At the evaluation stage, point-point distances, and trajectory-trajectory were calculated using the Euclidean distance. This further exaggerated the biases of the DB-index.
- Clustering itself lacks a well-defined standard, meaning it is not necessarily the most appropriate application comparison.

The effect of simplification and the importance of the cluster evaluation is further elaborated upon in Chapter 7.

Chapter 6

Results

In this chapter, we will present the results of the analysis we conducted. First, we briefly look at how the algorithms scored the trajectories. For the remainder of the chapter, we focus on the clusters and meaningfully interpret them.

6.1 Similarity Scores

Even though they lay the basis for further analysis, there is no meaningful way to directly compare the similarity scores. However, it could be interesting to look at their ranking of trajectory pairs. We are present the ten pairs that each measure deemed the most similar in Table 6.1.

We observe that there is variance with respect to where the pairs are placed, yet there are a handful of recurrent pairs. We noticed that one of the trajectory pairs appeared in the top-10 ranks for all the measures. Furthermore, two pairs appeared for all but one measure, and one pair that was in all but two. These four frequent pairs have been highlighted in Table 6.1 and are displayed in Figure 6.1. The table has 27 distinct trajectory pairs, indicating that the measures generally agree on which pairs are the most similar ones.

Table 6.1: Overview of most similar trajectory pairs according to each measure. A larger version of the table is in**??**

Euclidean	DTW	SSPD	Hausdorff	ERP	MSM; cost=1	MSM; cost=.1	MSM; cost=.01	EDR; $\epsilon = .203$	EDR; $\epsilon = .101$
QREOR-IHNMX	*LFQDH-ACJIG*	*LFQDH-ACJIG*	WBNSL-SATNK	*QREOR-IHNMX*	*QREOR-IHNMX*	*QREOR-IHNMX*	*LFQDH-ACJIG*	*LFQDH-ACJIG*	*LFQDH-ACJIG*
LFQDH-ACJIG	*NVZCQ-TGKDX*	WBNSL-SATNK	*NVZCQ-TGKDX*	*LFQDH-ACJIG*	*LFQDH-ACJIG*	*LFQDH-ACJIG*	*NVZCQ-TGKDX*	LVFYZ-LENWX	*QREOR-IHNMX*
BZCYT-VXLBC	*QREOR-IHNMX*	KQQUG-OVNBK	NHNMB-PSVVR	BZCYT-VXLBC	BZCYT-VXLBC	*NVZCQ-TGKDX*	KQQUG-OVNBK	XXVQK-LOBQA	KQQUG-OVNBK
LHTXJ-TGKDX	HXVAH-RPCOW	*QREOR-IHNMX*	HXVAH-RPCOW	*LHTXJ-TGKDX*	*LHTXJ-TGKDX*	BZCYT-VXLBC	WBNSL-SATNK	BLHVW-ATPTV	LFQDH-ASYEZ
PSVVR-XSJJE	KQQUG-OVNBK	NHNMB-PSVVR	*QREOR-IHNMX*	PSVVR-XSJJE	PSVVR-XSJJE	KQQUG-OVNBK	*QREOR-IHNMX*	WBTAG-QMRPP	*NVZCQ-TGKDX*
NVZCQ-TGKDX	JRDIM-OVNBK	*NVZCQ-TGKDX*	JRDIM-OVNBK	*NVZCQ-TGKDX*	*NVZCQ-TGKDX*	LFQDH-ASYEZ	HXVAH-RPCOW	NKEGR-VXLBC	BZCYT-VXLBC
TNPJQ-LRZLR	WBNSL-SATNK	HXVAH-RPCOW	*LFQDH-ACJIG*	TNPJQ-LRZLR	TNPJQ-LRZLR	*LHTXJ-TGKDX*	*LHTXJ-TGKDX*	TNPJQ-LRZLR	PSVVR-XSJJE
HUSJF-UHLAI	NHNMB-PSVVR	JRDIM-OVNBK	KQQUG-OVNBK	HUSJF-UHLAI	HUSJF-UHLAI	PSVVR-XSJJE	JRDIM-OVNBK	PSVVR-XSJJE	KQQUG-GRPNX
ZQZSA-NVZCQ	*LHTXJ-TGKDX*	*LHTXJ-TGKDX*	DADNP-LOBQA	ZQZSA-NVZCQ	ZQZSA-NVZCQ	HXVAH-RPCOW	LFQDH-ASYEZ	*LHTXJ-TGKDX*	CWCGU-ACJIG
NVZCQ-TNPJQ	BZCYT-VXLBC	SATNK-FIVOK	OAAYX-OKPGU	NVZCQ-TNPJQ	NVZCQ-TNPJQ	HUSJF-UHLAI	NHNMB-PSVVR	HUSJF-UHLAI	EBYMF-LENWX

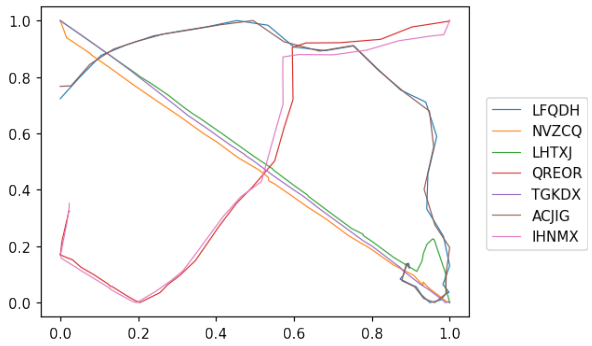


Figure 6.1: The trajectory pairs that appeared the most based on the similarity scores.

6.2 Davies-Bouldin Index

In Table 6.2 we have presented the Davies-Bouldin results for each cluster per type of clustering method. As was brought up in Chapter 5 and further elaborated upon in Chapter 7, this is not a decisive ranking.

Three observations stand out in Table 6.2. Firstly, we note that SSPD and HD have received the worst ranks under both AP and HCA. Next, we remarked that Ed and MSM- which had identical rankings of trajectory pair, also got the same DB-index under both types of clustering. ERP, which also had an identical top-10 raking, received a DB-index under AP-clustering, but did so under HCA. The last thing that stands out is that the HCA gave a worse result for almost all methods, except for DTW and MSM with the lowest cost value. The difference is not significant, but it is consistent.

Table 6.2: Davies-Bouldin Indices for each cluster result and how each algorithm rank against each other according to this evaluation

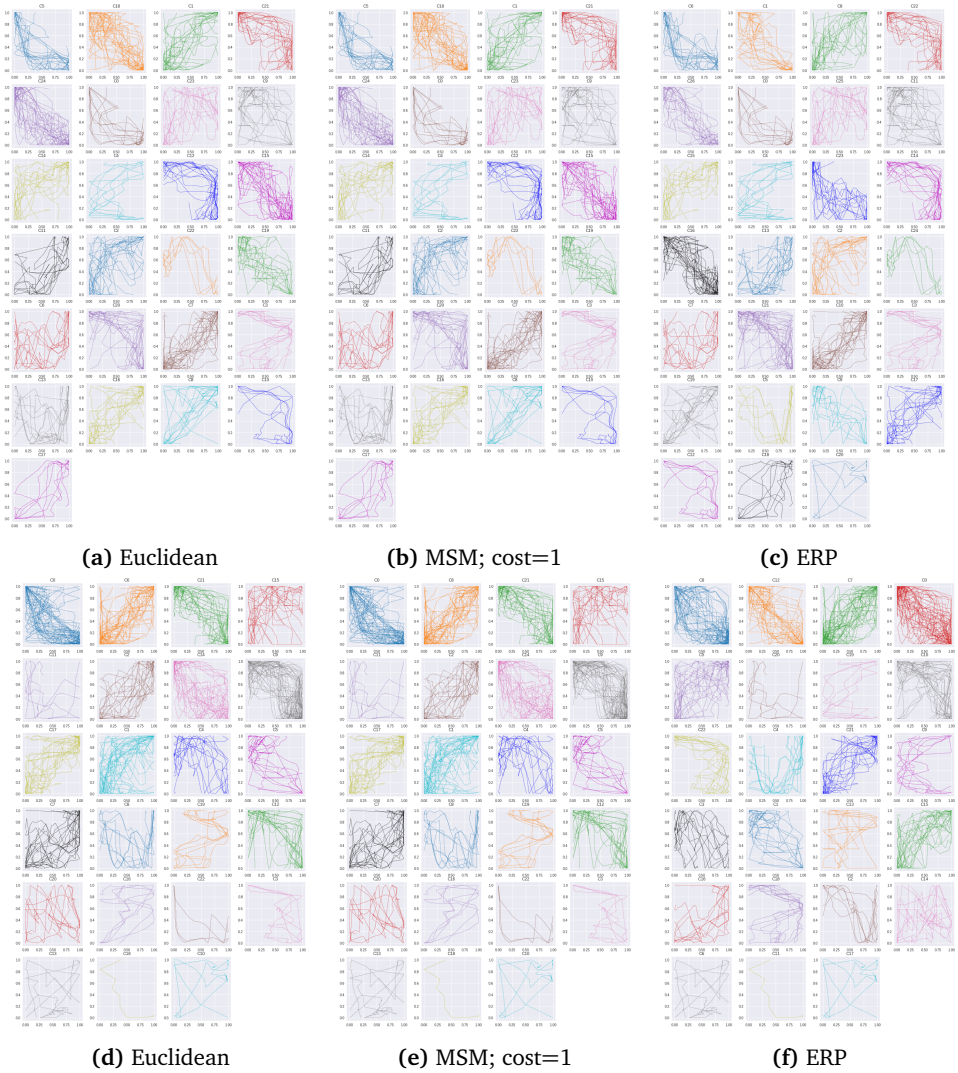
	Affinity Propagation		**Hierarchical Clustering**	
	Score	*Rank*	*Score*	*Rank*
Ed	0.091	3	0.095	2
DTW	0.156	8	0.140	5
Hd	0.294	9	0.459	9
SSPD	0.431	10	0.531	10
ERP	0.083	2	0.090	1
EDR $\epsilon = 0.203$	0.082	1	0.162	6
EDR $\epsilon = 0.101$	0.091	3	0.217	7
MSM cost= 0.01	0.150	6	0.137	4
MSM cost= 0.1	0.155	7	0.222	8
MSM cost= 1	0.091	3	0.095	2

6.3 Clustering

We restate that the area of interest for this thesis is the similarity distance measures and not various clustering techniques. The clusters are presented grouped by how they evaluated the trajectories' similarity, or by the underlying idea trajectory similarity.

The two models used to create the clusters will be presented alongside each other. The number of clusters produced under affinity propagation varied from 23 to 48, and as explained in Chapter 5 the number of hierarchical clusters was set to 23 for all measures. See Appendices B and C for higher resolution depictions of all clusters.

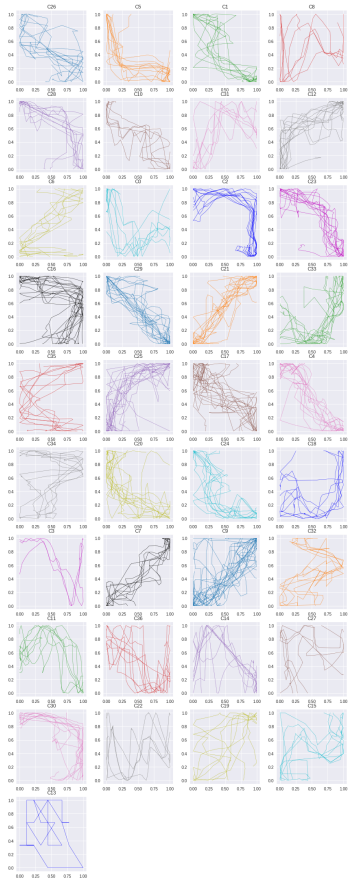
The first three measures' clusters we present are the ones with the identical ranking of most similar trajectory pairs; Euclidean distance, ERP, and MSM with cost parameter to 1. The clusters are depicted in Figure 6.2. Next, Figure 6.3 displays the clusters obtained from the remaining parameter values of Move-Split-Merge. Continuing with the parameters measured, Figure 6.4 shows the clusters that EDR generated with both parameter values. Finally, Figure 6.5 shows the rest of the measures, namely clusters created with DTW, Hausdorff distance, and SSPD.



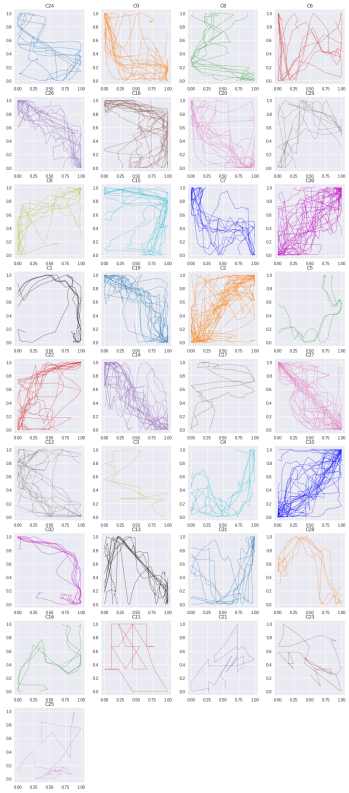
(a) Euclidean **(b)** MSM; cost=1 **(c)** ERP

(d) Euclidean **(e)** MSM; cost=1 **(f)** ERP

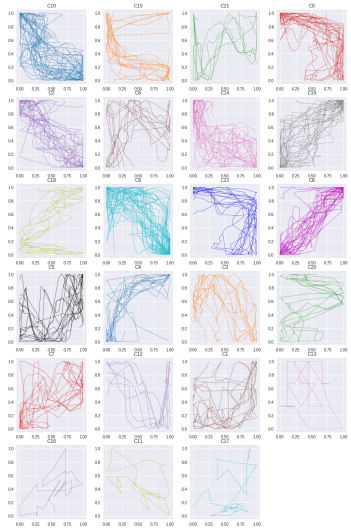
Figure 6.2: Clusters of the measures that had the same raking of most similar pairs. The top row is the affinity propagation Clusters and the bottom ones are from hierarchical clustering



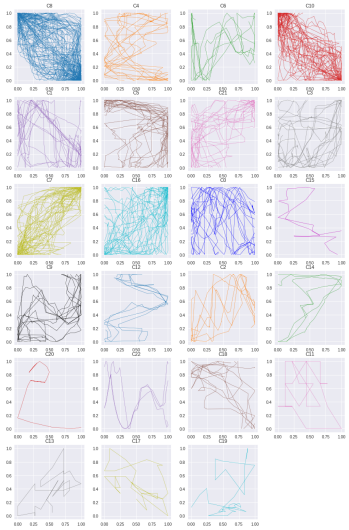
(a) MSM; Cost = 0.1



(b) MSM; Cost = 0.01

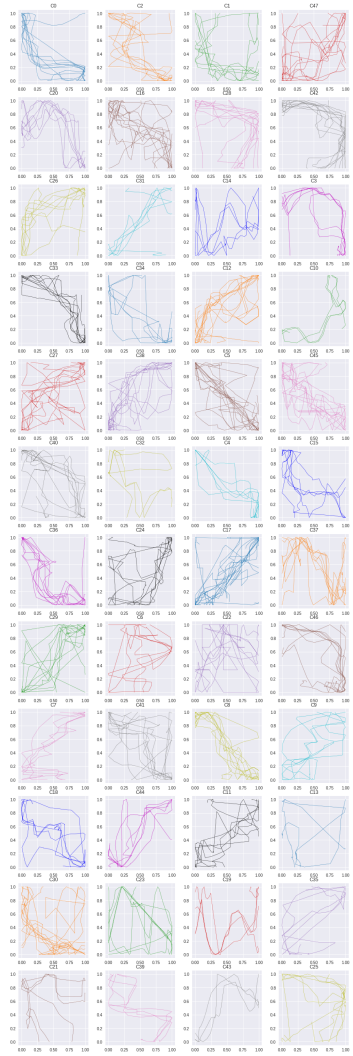


(c) MSM; Cost = 0.1

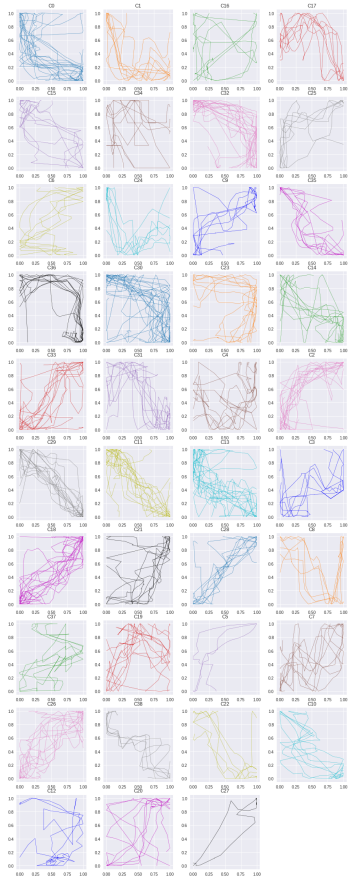


(d) MSM; Cost = 0.01

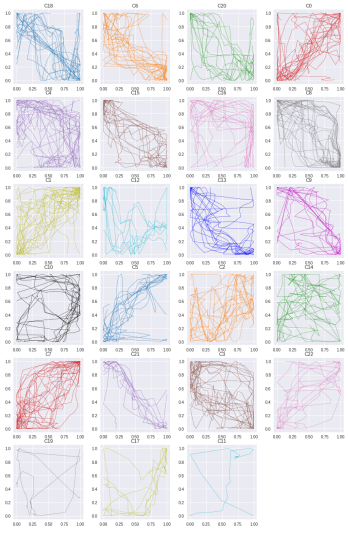
Figure 6.3: Clusters generated by Move-Split-Merge with different parameter values.The top row is the affinity propagation Clusters and the bottom ones are from hierarchical clustering



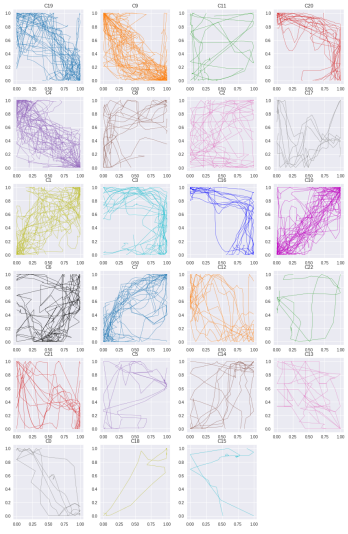
(a) EDR; $\epsilon = .101$



(b) EDR; $\epsilon = .203$

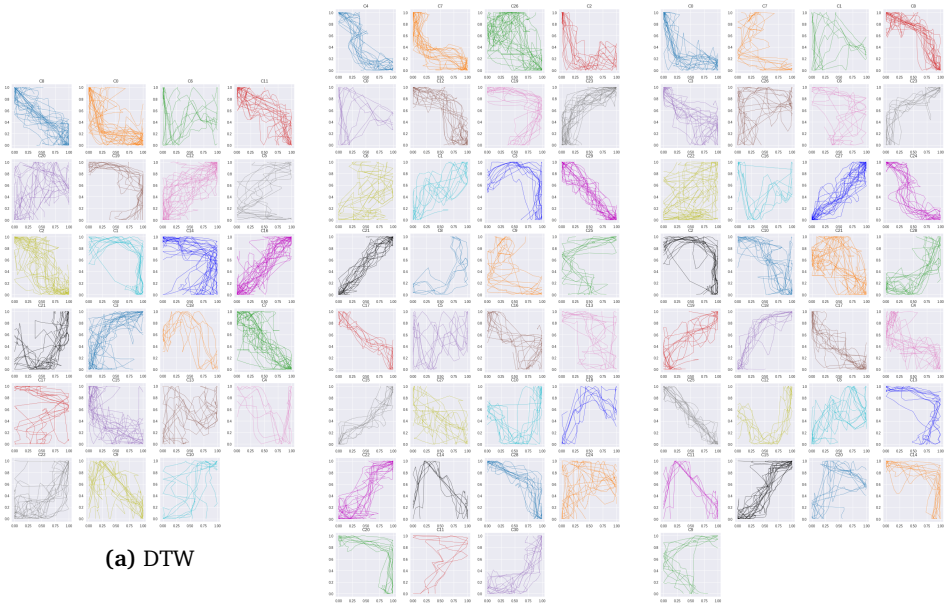


(c) EDR; $\epsilon = .101$

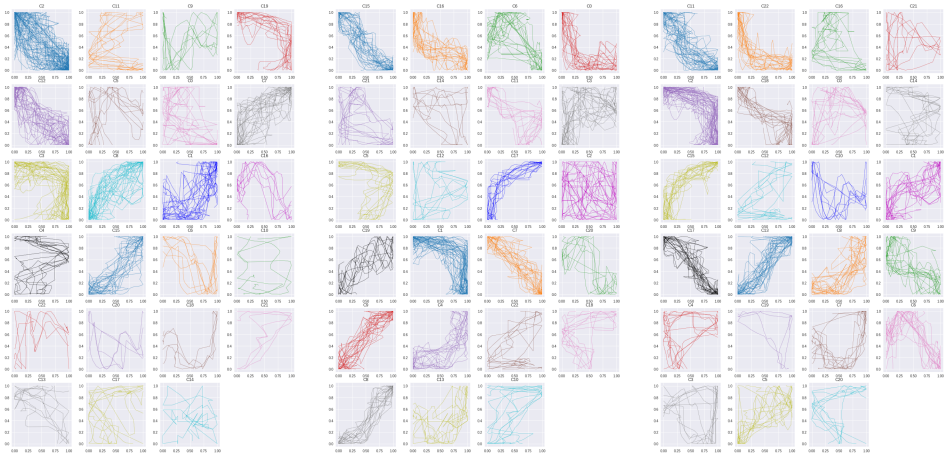


(d) EDR; $\epsilon = .203$

Figure 6.4: Clusters generated by Edit Distance on Real Sequence with different parameter values.The top row is the affinity propagation clusters and the bottom ones are from hierarchical clustering



(a) DTW **(b)** Hausdorff **(c)** SSPD



(d) DTW **(e)** Hausdorff **(f)** SSPD

Figure 6.5: Clusters created with DTW, Hausdorff and SSPD. The top row is the affinity propagation clusters and the bottom ones are from hierarchical clustering

Chapter 7

Discussion

In this chapter, we seek to contextualize the results from the preceding chapter. We apply theory from Chapter 2 and compare the result to the expected behavior

7.1 Similarity Scores

We have established that the measures define what makes trajectories similar. Some of the measures have a similar base idea, and thus we expect to see this reflected in the results. From the rankings of most similar pairs, there are a few things that stand out and we will go over them here.

First of all, EDR with ϵ set to 0.203 was the column in Table 6.1 which had the least number of frequent "most similar pairs". When adjusting the parameter down to 0.101, our approximation of the recommend parameter value, all of its eight most pairs were found elsewhere in the table. Recall that EDR's parameter is the threshold distance that determines if points are "matching". The similarity ranking with the lower parameter value aligns more with the other measures, and this could indicate that the other threshold was set too high. We would presume that EDR would give a similar ranking as the other noise-tolerant edit-distance inspired method, MSM. From what can be seen in the table, this appears to hold, but it varies with its parameter value too.

Next, we note that the Hausdorff distance generated similar pairs which were quite different from the other measures. As the Hausdorff metric is parameter-free, we reckon that the reason its results stood out was that it defined trajectory similarity in a different manner than the rest. We established that the Hausdorff distance does not take into account the direction of a trajectory and the only other measure that is invariant to direction is SSPD. Indeed they share many of the top 10 ranked scores, however, SSPD distinguishes itself Hd as it reduces the similarity score to a single element-element pair making altering their internal rankings again.

The final observation from Table 6.1 we noted was that Euclidean distance, ERP, and one of the MSM measures gave the exact same top 10 pairs. Upon closer examination, it turned out that their ranking remained the same until the 44th pair, and the similitude between the measures continued beyond that. MSM with cost value 1 and the Euclidean distance's ranking match for 9456 of the 10 thousand most similar trajectory pairs. We reason that this occurred as a consequence of the cost of Splits and Merges being set too high, making MSM default to the Move operation whose cost is the L_2 norm between trajectory elements. As for ERP, we reason that setting the reference point, g, to the origin, increased the cost for trajectory elements that were far away. Again, making the method depend on the L_2 norm between trajectory elements as the cost of an edit.

7.2 Davies-Bouldin Results

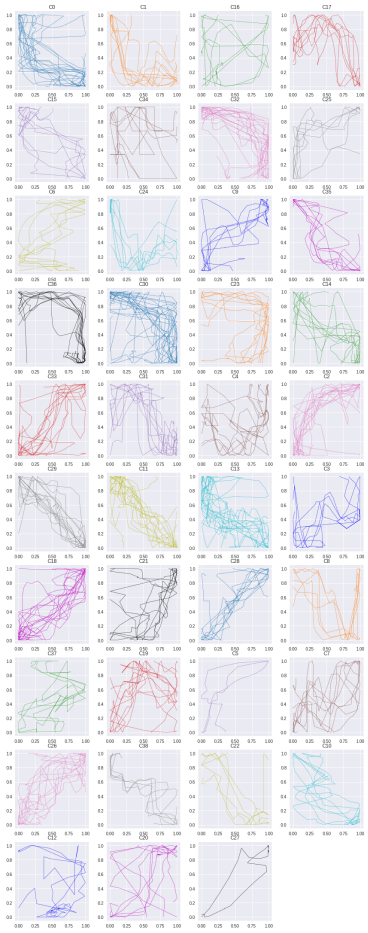
The essence of this thesis is to examine how the definition of similarity varies with different measures. Clustering is a manner of grouping the trajectories such that traits which are seen as the most defining characteristics under various definitions get highlighted.

As stated in Chapter 2, there are a number of cluster evaluation techniques that aim to numerically rank the quality of clustering results. One of these is the DB-index however we will not an assumption regarding which type of type similarity was the most "correct" one. The ranking created by this criterion is not invalid, but the insight it adds to is limited. Nevertheless, we have chosen to leave in this stage of the experiment and in the report on the account of the efforts that were put into computing it as well as its natural role as a starting point for analyzing the clusters.

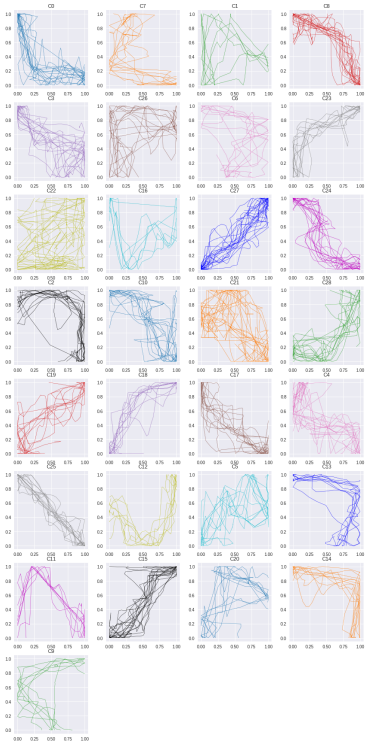
Our version of Davis-Bouldin favors clusterings where each trajectory element differs as little as possible from the mean of all trajectory elements when computed in a. A consequence of this is that the cluster can appear fuzzy since the placement of the surrounding points does not affect the index. From Table 6.2 it appears as if the lock-step method and EDR with threshold parameter $\epsilon = 0.203$ were the best performing methods, and this is as expected.

Moreover, methods that do not take into account order observations, or methods that would reorder the trajectory elements received low DB-index rankings. In Figures 7.1 and 7.2 the clusters that were generated by the measures which received highest and lowest DB-indexes are drawn. These figures illustrate the bias of our DB-index implementation.

As a consequence of the unreliability of the DB-index, the main tool for examining the clusters will be visual inspection. This will allow us to account for the different aspects of trajectory similarity and formulate conclusions accordingly. Re-examining Figures 7.1 and 7.2 with visual inspection it would appear that SSPD produced cleaner and fewer clusters than both EDR and ERP. This is in stark



(a) EDR;$\epsilon = 0.203$, highest rank



(b) SSPD, lowest rank

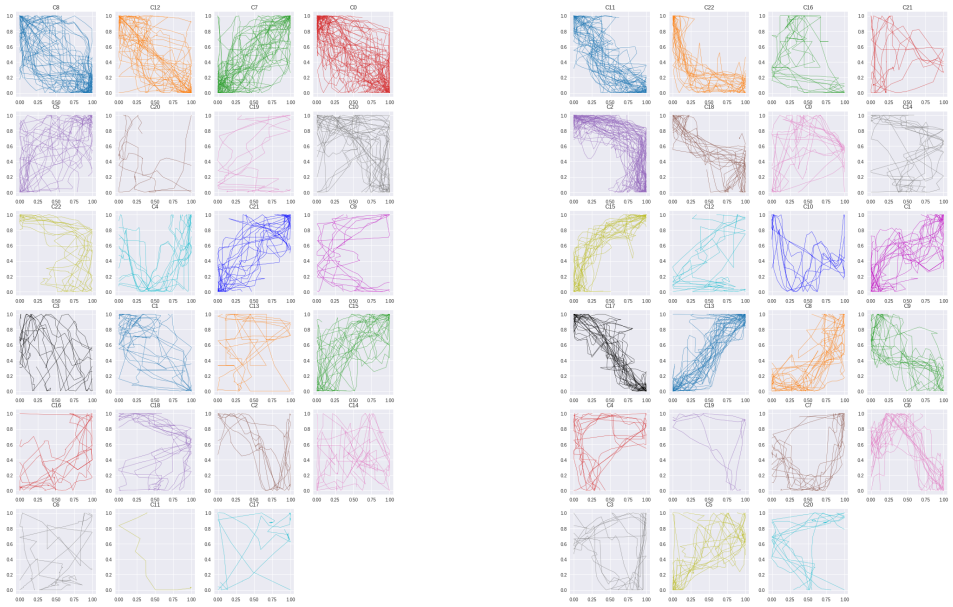
Figure 7.1: The AP-clusters of the measures the best and wort Davies-Bouldin indexes.

contrast to the ranking created by the Davies-Bouldin criteria under both HCA and AP analysis.

This evaluation alternative comes with its own drawbacks, one of which is the matter of subjectivity. At the same time, it has been stated that human assessment is an important component for determining the quality of clusters[33]. We will not be declaring any measure better than the others as our results do not provide support for an accurate ranking.

7.3 Cluster Behavior

First, we describe what we expected the clusters to look like based on the properties of the specific similarly distance function. For the methods that operate with



(a) ERP, $\epsilon = 0.203$, highest rank

(b) SSPD, lowest rank

Figure 7.2: The HCA-clusters of the measures the best and wort Davies-Bouldin indexes.

a parameter value, we discuss how changing that value would affect the cluster results. Then we compare our expectations to the clusters themselves. Affinity propagation is a was the parameter-free method, thus its results are weighted more. On the other hand, the number of hierarchical clusters was arbitrarily set, leading us to use its results as a supplementary evaluation.

7.3.1 Expectations

In general, there are three factors that will affect how we expect the final clusters to appear. Measures that are context-aware should calculate fewer fuzzy clusters. Next, measures that account for local time shifts should group trajectories that have similar sub-sections but with an offset creating a wider band of similarity. Lastly, we expect the noise-sensitive trajectories to create fuzzier trajectories.

Euclidean Distance is the only lock-step measure, and trajectories that are in the same region might receive an artificially high similarly score by nature of being near each other. A trajectory pair will be rewarded if they have elements at the same index which are near each. This would create fuzzy clusters as each point evaluated isolation and taking the average of the pairwise element-distances will artificially smooth out the distances. An extra amount of fuzziness is expected to come from the fact that this measure is sensitive to noise.

Dynamic Time Warping optimizes for local shape similarity and warps trajectories to reflect similarity after shape-preserving transformations. We expect the final clusters to appear fuzzy at first glance as the local similarities would be out of sync. Yet upon closer examination, we should be able to spot bands of cohesive trajectory sections. DTW is a noise-sensitive method, so we would expect some level of fuzziness to be present, however, the clusters should be crisper than that of the Ed.

From **Hausdorff Distance**, we would expect crisper clusters than both Ed and DTW if we could guarantee the data free of noise. The maximizing over a minimum of all element-element pairings means that the overall shape similarity is weighted in a way that the other measures do not account for. In other words, the global resemblance matters more under this definition. However, the data set we used is not noise-free so while we expect some crispness, there will be fuzziness of the clusters due to its noise sensitivity.

One of the key features of **Symmetrized Segment-Path Distance** is that it accounts for whole trajectory shape similarity. This means that we expect it to generate the crispest looking clusters. The expectation of crisp clusters is further substantiated by its tolerance to noise This method aimed to be invariant to the physical locations, meaning that trajectories of similar shapes with different origins could be grouped together. This could result in some broader bands of similarity.

Recall that **Edit Distance on Real Sequences** uses the parameter ϵ to as the matching threshold for how close how two trajectory elements are. Decreasing the parameter value leads to an increase in strictness in how close elements have to be, and in turn, would lead to crisper clusters. It goes without saying that a too restrictive ϵ would no longer be accurate; if no trajectory elements are matching, the only basis for similarity would be the trajectory lengths. Clusters are expected to have some level of fuzziness as EDR considers the trajectory element in isolation.

Edit Distance with Real Penalty was computed with one parameter value. We have established that this metric is sensitive to noise and that it accommodates local time shifts. As EDR, this measure does isolates the trajectory elements when computing the similarity distance. This means that there are three factors that contribute to fuzzy clusters, thus we expect the fuzziest clusters to be generated from this metric.

Move-Split-Merge handles local time shifts in the same manner as DTW and the other Edit Distance-based measures. We expect to observe trajectory sections that match and create wider bands. MSM distinguishes itself by accounting for the values that surround a given element when computing the similarity score. This means that we expect the clusters are crisper than those of generated by EDR and ERP. The parameter of MSM determines to which degree Splits and Merges are favored over Moves. As the cost increases, they will increasingly be evaded in favor of directly substituting the element. The cost of a Move is pairwise element-

element distance, and in turn, the clusters would resemble those created by a Euclidean distance based measure. On the other hand, if the parameter is set too low we expect more trajectories that do not resemble each other to get clustered together.

7.3.2 Visual Inspection

The visual inspection inspects the apparent fuzziness of the clusters as well as how scatted the trajectories within a cluster are. We will refer to the latter of these characteristics as the *band* of similarity which describes the general *trend* of the cluster. Where is it possible, we remark how larger trajectory sections were processed. Unfortunately, spotting tendencies like that is not something visual inspection excels at.

In the case of affinity propagation, we take note of how many clusters it generated. The corresponding evaluation for the hierarchical clusters is taking note of how balanced the final clusters are. With how hierarchical clusters are created, some imbalance is expected as the most distinct trajectories will be connected last.

Euclidean Distance:

- AP: In terms of overall shape, the clusters appear very fuzzy. There are some clusters where that have a clearer contour and some where it is possible to spot a trend for the trajectories. Nevertheless the general impression remains fuzzy; we observe trajectories that seem to oscillate freely around the apparent trajectory band. While oscillations make the clusters appear fuzzier, the clusters do not come across as randomly grouped observations.
- HCA: There appears to be a bias and clusters are unbalanced. Some of the clusters contained a few trajectories while some of them encompassed a large number of them. In those clusters the bands were obvious, but the oscillations were even greater than those observed under AP. This makes sense as the new clusters are created by merging similar sub-clusters. We would expect a clearer band, but a fuzzier contour.

Dynamic Time Warping:

- AP: We observed two clusters that were very crisp and even more clusters that exhibited clear trends. The trajectories do not seem to oscillate around the band in the clusters as much as they did under Ed. We can spot some trajectory sections that appear as if they have been re-aligned– leading to a less messy expression. Still, there are some oscillations and noisy clusters that make it hard to tell exactly why some trajectories were clustered together.

- HCA: The clusters are unbalanced, however less so than the ones created by Ed. They illustrate how shape similarity is persevered under this measure by displaying a clear trend in the clusters. Yet, the clusters are fuzzy in that their trajectories still deviate from the central band. From these clusters it appears as if some of the trajectories are outliers; they appear to be distinct from the rest of the trajectories and are placed in their own clusters.

Hausdorff Distance

- AP: Compared to DTW and Ed, the final number of clusters increased and the clusters created were both crisper and fuzzier. The clusters display both the most advantageous and most disadvantageous aspects of the Hausdorff metric. However, the number of crisp clusters outweighs the fuzzy ones, and the contours of the clusters are crisp enough to highlight more intricate trajectory details.
- HCA: Again, we see both crisp and intricate clusters as well as very fuzzy ones. The observations from affinity propagation hold for these clusters as well; both the advantages and disadvantages of setting the similarity score as the distance between two trajectory elements are highlighted. There is one cluster that is so fussy that we were quite puzzled by how it was formed.

Symmetrized Segment-Path Distance:

- AP: This measure created fewer clusters than Hd, yet the clusters it created appear to be at least equally as crisp. We observed that the clusters had such tight bands that curves of the trajectories were accentuated. In contrast, DTW clustered the trajectories by their general shape and direction. Whereas Hd was sensitive to noise, it becomes clear that SSPD addressed that weakness. Two clusters stand out as more fuzzy than the others, but this is likely due to noisy data.
- HCA: The clusters are about as balanced as those created by Hd. As was demonstrated in the AP-clustering, there is a trend towards showing the intricacies of shape similarity at a larger scale. The bands of similarity are thinner than both Ed end DTW and more detailed than Hd.

Edit Distance with Real Penalty:

- AP: The clusters created by ERP are significantly more fuzzy than that of Hd and SSPD. This is expected as it gives similarity scores based on edits before and then the distance between trajectory elements. However, it is unexpected that the resulting clusters were as fuzzy as those created by DTW and Ed, especially as it created far more clusters than either of them did. The increased number of clusters should indicate that we would have more distinguished clusters, but this does not appear to be the case. Even still,

we observe that clusters were not randomly put together. Looking closer at specific clusters, it is possible to spot turns and segments that are just shifted from each other.

- HCA: These clusters are unbalanced, there is a cluster that only has one trajectory. Yet, that trajectory does not appear significantly distinct from the ones existing clusters. From visual inspection it challenging to get more insight. The bands across the clusters are quite general and there is not a consistent show of shifted segments.

Edit Distance on Real Sequence: In discussing this method, we examine the clusters obtained from both parameter values in parallel.

- AP: This measure generated the two largest number of AP-clusters. The most restrictive parameter led to 48 clusters while the other value resulted in 39 clusters. In our subjective opinion, both of the parameters resulted in too many clusters for a data set of 300 observations. However, with fewer trajectories in each cluster analysis through visual inspection becomes is easier. We observed that the clusters contained trajectories whose segments were shifted from each other. Both parameter values resulted in crisp clusters, and our approximation of the recommended value gave seemingly more defined contours. While increasing the threshold value led to fuzzier clusters, these clusters were crisper than those of ERP. This is an expected result as EDR is noise-tolerant whereas ERP is not.
- HCA: For both parameter values, there were bands that obscured the intricacies of shape similarity. Once more we observed that increasing the parameter value resulted in fuzzier clusters. However, the increase in fuzziness was less than expected. The largest parameter value led to a more unbalanced distribution of trajectories, further signifying that the smaller one remains the most suited value for ϵ.

Move-Split-Merge:

As we did for EDR, the clusters created by MSM and the effect of varying the parameter are discussed in parallel. We first note that the clusters MSM created with the cost was set to 1 and were indistinguishable from those of Ed. This was not an entirely unexpected result given what we know about MSM, and that the two methods had identical rankings of the most similar trajectory pairs. By reason of that cost parameter being set too high and the accompanying clusters already being described, we will focus on the two remaining parameter values.

- AP: We observed that the lowest parameter value resulted in fewer clusters, but both of the parameter values led to a larger number of clusters than expected. As with EDR, the large number of clusters makes it easier to visually analyze them. Given that this measure is both moderately noise-tolerant

and context-aware we expect the clusters to be crisp. Indeed, the majority of the clusters have crisp contours wherein the intricacies of the trajectories are preserved. Some clusters contain what appear to be outlier trajectories. This would suggest that the discrimination degree of this measure was high enough to isolate them in a way that no measures did. The smaller parameter value resulted in both fuzzier clusters and an increase in clusters with few trajectories. This may be an indication of that value being too low, suggesting that 0.1 was the most appropriate cost for this data set.

- HCA: The clusters created by the lowest cost parameter were more unbalanced and fuzzier than those generated by the middle one. This further supports the argument that the middle of the cost value is the most fitting one. Interestingly, there were still trajectories that were clustered by themselves as if they were outliers. It looks as if these trajectories are consistent across both parameter values and clustering techniques.

7.4 Reflections

The measures created clusters whose qualities coincide with the anticipated results. We observed fuzzy clusters where the underlying measure was less tolerant to noise and crisper clusters where the measure accounted for whole trajectory similarity. Almost all of the measures were elastic, and we observed several clusters with trajectories that were time-shifted.

By visual inspection, it was hard to tell if metricity affected the clusters although this may have been a result of picking clustering techniques that did not require a metric distance function.

The clusters' appearance remained consistent between AP and HCA for a given measure, even as the number of clusters changed. This observation was to be expected as the models that created the clusters used the same similarity distances scores.

Lastly, we comment on a few algorithm-specific observations. The fuzziness of the clusters created by the Euclidean distance align with its lock-step design just as the crispness of SSPD's clusters aligns with its design purposes. It was initially unexpected that ERP would have the fuzziest looking clusters. However, upon closer examination, we reasoned that it made sense that a non-context aware, noise sensitive method that adapts to time shifts would lead to quite fuzzy clusters.

The way MSM could consistently filter out trajectories that did not behave like the others would imply that it would be well suited for outlier detection.

7.5 Inconsistencies from Simplifications

As noted in Chapter 5, several simplifications were made for this experiment. We close this chapter by reflecting on how these simplifications have affected the results.

7.5.1 Data format

The first simplification that was made was truncating the trajectories to equal length. In doing this we removed the option to study how measures would have handled cases like this. There would be insight to be gained from examining measures at different trajectory lengths. In particular, it is not reasonable to assume that real data would have this property. The clusters could have ended up looking quite different, possibly highlighting the difference between trajectory section and whole trajectory similarity.

Of the algorithms in this thesis, only the Euclidean distance lacked a definition for trajectories of unequal length. It is possible to design a comparative study where lock-step measures can give scores for these trajectories. An option would be to artificially add more trajectories elements by interpolation. However, we decided against it as we felt confident there would be enough properties to examine after the simplification.

The next data-format simplification we did was the re-scaling. The manner in which it was done meant that the data would lose its connection to the real world. To exemplify this we refer to Figure 7.3. The clusters were created based on the scaled data, thus those clustered were quite crisp. However, when displaying those same clusters with their raw coordinates it becomes clear how scattered the trajectories are. It becomes clearer that trajectories were clustered together based on shape similarity. We reiterate that the intent was to study measures themselves, thus the data set selection— and thereby the re-scaling was inconsequential.

7.5.2 Distance Computation and Evaluation

We tested three measures that required a parameter and we did not do any proper parameter tuning. ERP would likely have had more agreement with the other measures in Table 6.1 if different values for the reference point had been tested out.

The application we chose for the similarity distance measures, clustering, is itself an area of active research. The interconnectedness of clustering techniques and distance algorithms could have been studied in more detail before settling on affinity propagation and hierarchical clustering analysis as the evaluation basis. We acknowledge that there may be traits of the selected measures that have been obscured or misrepresented.

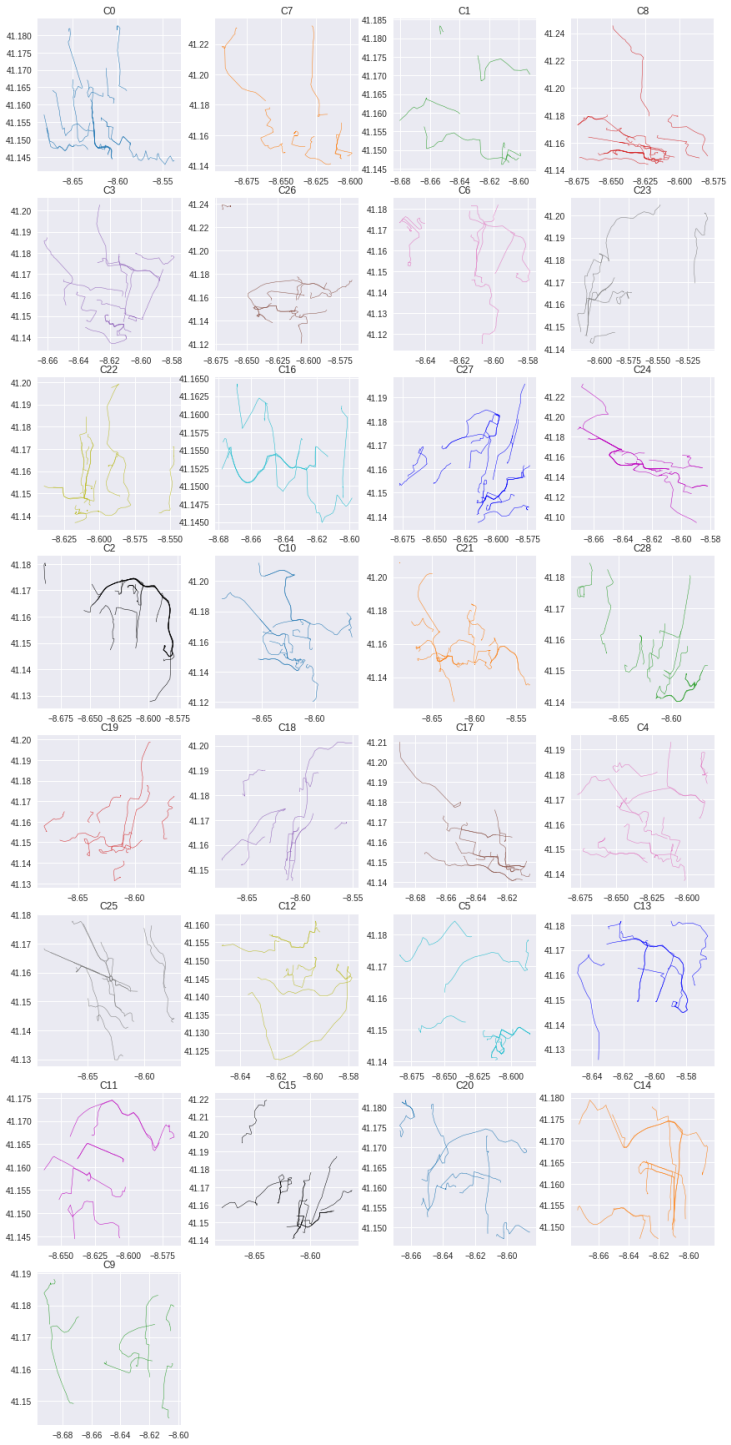


Figure 7.3: Affinity propagation clusters created with SSPD, but showing the raw trajectory locations

Chapter 8

Conclusion and Further Work

8.1 Conclusion

In this thesis, we studied seven similarity measures for trajectory data, five conventional ones and two recent ones. First, we clarified what constitutes a trajectory, and then we discussed how the notion of similarity of them varies. The measures we reviewed vary in how they defined similarity and we elaborated upon some of their advantages and disadvantages.

In order to experimentally compare the algorithms, we defined an application wherein the only thing that varied was the similarity distance computation. The application we decided on was clustering and we used two techniques for creating them. The clusters gave us a vantage point for further description of the measures, and from the clusters, we were able to observe the theoretical characteristics of the measures in practice. We summarize what has been achieved in this thesis by referring back to the research objectives:

Research Objectives 1 and 2:

The conventional similarity measures we examined were the Euclidean distance, Dynamic Time Warping, Hausdorff Distance, Edit Distance on Real Sequence, and Edit distance with Real Penalty. The latter two are newer than the other ones, however, their prominence in the literature establishes them as conventional methods. A commonality between these measures is that they are not context-aware. In terms of noise tolerance, there was no consistency, and as for elasticity all but Ed are could adapt to local time shifts.

Both Move-Split-Merge and Symmetric Segment Path Distance are methods that were developed in order to address the shortcomings of the conventional methods. Based on this we categorized them as the newer distance measures.

The development of MSM was a response to DTW and ERP while SSPD was an advancement of Hd. A feature that is shared between MSM and SSPD is that they both were designed to account for whole trajectory similarity. Additionally, they are more tolerant to noise than their respective conventional method inspirations. We saw this experimentally as MSM producing crisper clusterings than DTW and ERP and SSPD producing crisper clusterings than Hd.

We were not able to determine any effects of metricity, neither for the conventional methods nor for the newer ones. For the visual representation of how trajectory features were treated by the we refer to Appendices B and C

Research Objective 3:

In our set up, there was only one application that tested the algorithms' performance against each other. The lack of diverse applications and their removal from real-world observations are limiting how much insight we could gain regarding broader uses for the measures. Nevertheless, the variance of the cluster we generated demonstrates that similarity distance measure choice matters.

We could have constructed arguments that would have framed a given measure as the most suited one for shape-based clustering. However, we reiterate could not have generalized its applicability to other tasks.

Optimizing for global shape-similarity should be prioritized if the intended application is computer vision or migration analysis. On the other hand, locally shifted similarities are needed for database management tasks such as outlier detection and trajectory uncertainty. An outlier in this context could be either a trajectory element that stands out from its surrounding elements or a trajectory that stands out from the rest. Trajectory uncertainty management seeks to increase the utility of trajectories by approximating locations between observations. In both cases, a reference model of local resemblance is more apt.

Consequently, we remark that the accuracy and reliability of analysis or management tasks depend on whether or not an appropriate algorithm was chosen.

8.2 Further Work

The work we have done for this thesis explored just one class of trajectory similarity and one particular application for similarity measures. There are aspects of trajectory similarity we did not inquire into. Thus we suggest two extensions of our experiment which would facilitate a more comprehensive study.

As we noted in Chapter 5, our set up was not conducive to examining the differences in computational complexity. If it were possible, we would have liked to include an evaluation of measures' performances; both for the data set used here and sets where the lengths of the trajectories had not been artificially shortened. As with most algorithms, it is good to be informed about how they will perform as the size of the data or the data set increases.

It would make sense to include more applications for the measures. Along with clustering, classification is a commonly selected application for testing the measures. However, we argue it would be interesting to run either an outlier detection or noise removal task. This could hopefully highlight peculiarities of how the measures distinguish the trajectories.

Bibliography

[1] T. Nakamura, K. Taki, H. Nomiya, K. Seki and K. Uehara, 'A shape-based similarity measure for time series data with ensemble learning,' *Pattern Analysis and Applications*, vol. 16, no. 4, pp. 535–548, Nov. 2013, ISSN: 1433-7541, 1433-755X. DOI: 10.1007/s10044-011-0262-6.

[2] Y. Zheng, 'Trajectory Data Mining: An Overview,' *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 3, pp. 1–41, 20th May 2015, ISSN: 2157-6904, 2157-6912. DOI: 10.1145/2743025.

[3] K. Deng, K. Xie, K. Zheng and X. Zhou, 'Trajectory Indexing and Retrieval,' in *Computing with Spatial Trajectories*, 2011. DOI: 10.1007/978-1-4614-1629-6_2.

[4] B. J. Maiseli, 'Hausdorff Distance with Outliers and Noise Resilience Capabilities,' *SN Computer Science*, vol. 2, no. 5, p. 358, 26th Jun. 2021, ISSN: 2661-8907. DOI: 10.1007/s42979-021-00737-y.

[5] K. L. Holm, 'Similarity Measures for Trajectory Data,' Norwegian University of Science and Technology, Report, 2021.

[6] N. Andrienko, G. Andrienko, N. Pelekis and S. Spaccapietra, 'Basic Concepts of Movement Data,' in *Mobility, Data Mining and Privacy: Geographic Knowledge Discovery*, F. Giannotti and D. Pedreschi, Eds., Springer, 2008, pp. 15–38, ISBN: 978-3-540-75177-9. [Online]. Available: https://doi.org/10.1007/978-3-540-75177-9_2 (visited on 04/10/2021).

[7] A. Chakraborti, M. Patriarca and M. S. Santhanam, 'Financial Time-series Analysis: A Brief Overview,' 2007. DOI: 10.1007/978-88-470-0665-2_4.

[8] J. M. Dow, R. E. Neilan and C. Rizos, 'The International GNSS Service in a changing landscape of Global Navigation Satellite Systems,' *Journal of Geodesy*, vol. 83, no. 3, pp. 191–198, 1st Mar. 2009, ISSN: 1432-1394. DOI: 10.1007/s00190-008-0300-3.

[9] E. J. Keogh and M. J. Pazzani, 'An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback,' p. 9,

[10] K. Toohey and M. Duckham, 'Trajectory similarity measures,' *SIGSPATIAL Special*, vol. 7, no. 1, pp. 43–50, 20th May 2015, ISSN: 1946-7729. DOI: 10.1145/2782759.2782767.

[11] E. W. Weisstein and A. Cortzen. (2016). 'Measure,' mathworld - wolfram, [Online]. Available: https://mathworld.wolfram.com/Measure.html (visited on 13/10/2021).

[12] W. Contributors, *Measure (mathematics)*, in *Wikipedia*, 8th Oct. 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Measure_(mathematics)&oldid=1048807597 (visited on 13/10/2021).

[13] E. W. Weisstein. (2021). 'Metric,' [Online]. Available: https://mathworld.wolfram.com/Metric.html (visited on 14/10/2021).

[14] R. S. D. Sousa, A. Boukerche and A. A. F. Loureiro, 'Vehicle Trajectory Similarity: Models, Methods, and Applications,' *ACM Computing Surveys*, vol. 53, no. 5, 94:1–94:32, 28th Sep. 2020, ISSN: 0360-0300. DOI: 10.1145/3406096.

[15] G. R. Hjaltason and H. Samet, 'Index-driven similarity search in metric spaces (Survey Article),' *ACM Transactions on Database Systems*, vol. 28, no. 4, pp. 517–580, 1st Dec. 2003, ISSN: 0362-5915. DOI: 10.1145/958942.958948.

[16] P. Indyk, 'A Sublinear Time Approximation Scheme for Clustering in Metric Spaces,' in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, ser. FOCS '99, IEEE Computer Society, 17th Oct. 1999, p. 154, ISBN: 978-0-7695-0409-4.

[17] D. Jacobs, D. Weinshall and Y. Gdalyahu, 'Classification with nonmetric distances: Image retrieval and class representation,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 583–600, Jun. 2000, ISSN: 01628828. DOI: 10.1109/34.862197.

[18] D. Q. Goldin and P. C. Kanellakis, 'On similarity queries for time-series data: Constraint specification and implementation,' in *Principles and Practice of Constraint Programming — CP '95*, U. Montanari and F. Rossi, Eds., red. by G. Goos, J. Hartmanis and J. Leeuwen, vol. 976, Springer Berlin Heidelberg, 1995, pp. 137–153, ISBN: 978-3-540-60299-6 978-3-540-44788-7. [Online]. Available: http://link.springer.com/10.1007/3-540-60299-2_9 (visited on 04/10/2021).

[19] L. K. Rhodes. (23rd Jul. 2021). 'Similarity and Dissimilarity,' [Online]. Available: http://jcsites.juniata.edu/faculty/rhodes/ml/simdissim.htm (visited on 12/10/2021).

[20] S. Salvador and P. Chan, 'FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space,' p. 11,

[21] H.-C. Kuo, T.-L. Lee and J.-P. Huang, 'Cluster analysis on time series gene expression data,' *IJBIDM*, vol. 5, pp. 56–76, 1st Jan. 2010. DOI: 10.1504/IJBIDM.2010.030299.

[22] M. Vlachos, G. Kollios and D. Gunopulos, 'Discovering similar multidimensional trajectories,' in *Proceedings 18th International Conference on Data Engineering*, Feb. 2002, pp. 673–684. DOI: 10.1109/ICDE.2002.994784.

[23] B. Guillouet, *Trajectory_distance*, 5th Nov. 2021. [Online]. Available: https://github.com/bguillouet/traj-dist (visited on 08/11/2021).

[24] Z. Wang, 'Time Series Matching: A Multi-filter Approach,' p. 109,

[25] J. Peng, H. Wang, J. Li and H. Gao, 'Set-based Similarity Search for Time Series,' in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16, Association for Computing Machinery, 26th Jun. 2016, pp. 2039–2052, ISBN: 978-1-4503-3531-7. DOI: 10.1145/2882903.2882963.

[26] N. Pelekis, I. Kopanakis, G. Marketos, I. Ntoutsi, G. Andrienko and Y. Theodoridis, 'Similarity Search in Trajectory Databases,' in *14th International Symposium on Temporal Representation and Reasoning (TIME'07)*, Jun. 2007, pp. 129–140. DOI: 10.1109/TIME.2007.59.

[27] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, 'From Data Mining to Knowledge Discovery in Databases,' p. 18,

[28] V. Estivill-Castro, 'Why so many clustering algorithms: A position paper,' *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 1, pp. 65–75, 1st Jun. 2002, ISSN: 1931-0145. DOI: 10.1145/568574.568575.

[29] D. Wang, T. Miwa and T. Morikawa, 'Big Trajectory Data Mining: A Survey of Methods, Applications, and Services,' *Sensors (Basel, Switzerland)*, vol. 20, no. 16, p. 4571, 14th Aug. 2020, ISSN: 1424-8220. DOI: 10.3390/s20164571. eprint: 32824028.

[30] C. C. Aggarwal, 'Mining Spatial Data,' in *Data Mining: The Textbook*, C. C. Aggarwal, Ed., Springer International Publishing, 2015, pp. 531–555, ISBN: 978-3-319-14142-8. [Online]. Available: https://doi.org/10.1007/978-3-319-14142-8_16 (visited on 15/12/2021).

[31] B. J. Frey and D. Dueck, 'Clustering by Passing Messages Between Data Points,' *Science*, vol. 315, no. 5814, pp. 972–976, 16th Feb. 2007. DOI: 10.1126/science.1136800.

[32] The Pennsylvania State University. (2018). 'Lesson 10: Clustering,' [Online]. Available: https://online.stat.psu.edu/stat555/print/book/export/html/11/ (visited on 11/01/2022).

[33] G. Drakos. (4th Mar. 2020). 'Silhouette Analysis vs Elbow Method vs Davies-Bouldin Index: Selecting the optimal number of clusters for KMeans clustering,' GDCoder, [Online]. Available: https://gdcoder.com/silhouette-analysis-vs-elbow-method-vs-davies-bouldin-index-selecting-the-optimal-number-of-clusters-for-kmeans-clustering/ (visited on 11/01/2022).

[34] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana and E. Keogh. (8th Sep. 2019). 'The UCR Time Series Archive.' arXiv: 1810.07758, [Online]. Available: http://arxiv.org/abs/1810.07758 (visited on 01/12/2021).

[35] C. C. Aggarwal, 'Similarity and Distances,' in *Data Mining: The Textbook*, C. C. Aggarwal, Ed., Springer International Publishing, 2015, pp. 63–91, ISBN: 978-3-319-14142-8. [Online]. Available: https://doi.org/10.1007/978-3-319-14142-8_3 (visited on 16/12/2021).

[36] H. Wang, H. Su, K. Zheng, S. Sadiq and X. Zhou, 'An Effectiveness Study on Trajectory Similarity Measures,' *Database Technologies*, vol. 137, p. 10, 2013.

[37] J. Lines, 'Time Series classification through transformation and ensembles,' 2015.

[38] A. Kotsifakos, 'Online Efficient And Effective Search In Large And Noisy Sequence Databases,' *undefined*, 2014. [Online]. Available: https://www.semanticscholar.org/paper/Online-Efficient-And-Effective-Search-In-Large-And-Kotsifakos/21c3db65abf7cef992abd1b862326845411f8e0e (visited on 30/11/2021).

[39] C. Faloutsos, M. Ranganathan and Y. Manolopoulos, *Fast Subsequence Matching in Time-Series Databases*, 1994.

[40] N. Magdy, M. A. Sakr, T. Mostafa and K. El-Bahnasy, 'Review on trajectory similarity measures,' in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, Dec. 2015, pp. 613–619. DOI: 10.1109/IntelCIS.2015.7397286.

[41] G. Rote, 'Computing the minimum Hausdorff distance between two point sets on a line under translation,' *Information Processing Letters*, vol. 38, no. 3, pp. 123–127, 17th May 1991, ISSN: 0020-0190. DOI: 10.1016/0020-0190(91)90233-8.

[42] L. Chen, M. T. Özsu and V. Oria, 'Robust and fast similarity search for moving object trajectories,' in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data - SIGMOD '05*, ACM Press, 2005, p. 491, ISBN: 978-1-59593-060-6. DOI: 10.1145/1066157.1066213.

[43] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang and E. Keogh, 'Querying and mining of time series data: Experimental comparison of representations and distance measures,' *PVLDB*, vol. 1, pp. 1542–1552, 1st Aug. 2008.

[44] W. Abdulla, D. Chow and G. Sin, *Cross-Words Reference Template for DTW-based Speech Recognition Systems*. 15th Nov. 2003, 1579 Vol.4, 1576 pp., ISBN: 978-0-7803-8162-9. DOI: 10.1109/TENCON.2003.1273186.

[45] 'Dynamic Time Warping,' in *Information Retrieval for Music and Motion*, M. Müller, Ed., Springer, 2007, pp. 69–84, ISBN: 978-3-540-74048-3. [Online]. Available: https://doi.org/10.1007/978-3-540-74048-3_4 (visited on 13/10/2021).

[46] T. Giorgino, 'Computing and Visualizing Dynamic Time Warping Alignments in *R* : The **dtw** Package,' *Journal of Statistical Software*, vol. 31, no. 7, 2009, ISSN: 1548-7660. DOI: 10.18637/jss.v031.i07.

[47] L. J. Latecki, V. Megalooikonomou, Q. Wang and D. Yu, 'An elastic partial shape matching technique,' *Pattern Recognition*, vol. 40, no. 11, pp. 3069–3080, 1st Nov. 2007, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2007.03.004.

[48] N. Grégoire and M. Bouillot. (Aut. 1998). 'Hausdorff distance,' Hausdorff distance between convex polygons, [Online]. Available: http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html (visited on 23/11/2021).

[49] H. Alt, 'The Computational Geometry of Comparing Shapes,' in *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, ser. Lecture Notes in Computer Science, S. Albers, H. Alt and S. Näher, Eds., Springer, 2009, pp. 235–248, ISBN: 978-3-642-03456-5. DOI: 10.1007/978-3-642-03456-5_16.

[50] A. A. Taha and A. Hanbury, 'An Efficient Algorithm for Calculating the Exact Hausdorff Distance,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2153–2163, Nov. 2015, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2015.2408351.

[51] P. Besse, B. Guillouet, J.-M. Loubes and F. Royer, 'Review & Perspective for Distance Based Trajectory Clustering,' 17th Aug. 2015.

[52] L. Chen and R. Ng, 'On The Marriage of Lp-norms and Edit Distance,' p. 12, 2004.

[53] A. Stefan, V. Athitsos and G. Das, 'The Move-Split-Merge Metric for Time Series,' *IEEE Transactions on Knowledge and Data Engineering*, 2012. DOI: 10.1109/TKDE.2012.88.

[54] A. Bagnall, J. Lines, A. Bostrom, J. Large and E. Keogh, 'The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances,' *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 1st May 2017, ISSN: 1573-756X. DOI: 10.1007/s10618-016-0483-9.

[55] Google. (). 'Google Colaboratory,' [Online]. Available: https://colab.research.google.com/ (visited on 17/12/2021).

[56] (). 'Project Jupyter,' [Online]. Available: https://www.jupyter.org (visited on 17/12/2021).

[57] (). 'Welcome to Python.org,' Python.org, [Online]. Available: https://www.python.org/ (visited on 17/12/2021).

[58] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn and K. Smith, 'Cython: The Best of Both Worlds,' *Computing in Science & Engineering*, vol. 13, no. 2, pp. 31–39, Mar. 2011, ISSN: 1521-9615. DOI: 10.1109/MCSE.2010.118.

[59] C. Cross. (). 'Taxi Trajectory Data,' [Online]. Available: https://kaggle.com/crailtap/taxi-trajectory (visited on 30/08/2021).

[60] A. Bagnall, A. Bostrom, J. Large and J. Lines. (4th Feb. 2016). 'The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version.' arXiv: 1602.01711, [Online]. Available: http://arxiv.org/abs/1602.01711 (visited on 02/11/2021).

Appendix A

Most Similar Trajectory Pairs

Table A.1: Raking of most similar trajectory pairs

Rank	Euclidean	DTW	SSPD	Hausdorff	ERP	MSM; cost=1	MSM; cost=.1	MSM; cost=.01	EDR; $\epsilon = .203$	EDR; $\epsilon = .101$
1	QREOR-IHNMX	LFQDH-ACJIG	LFQDH-ACJIG	NVZCQ-TGKDX	QREOR-IHNMX	QREOR-IHNMX	QREOR-IHNMX	LFQDH-ACJIG	LFQDH-ACJIG	LFQDH-ACJIG
2	LFQDH-ACJIG	NVZCQ-TGKDX	KQQUG-OVNBK	QREOR-IHNMX	LFQDH-ACJIG	LFQDH-ACJIG	LFQDH-ACJIG	NVZCQ-TGKDX	WBTAG-QMRPP	QREOR-IHNMX
3	BZCYT-VXLBC	QREOR-IHNMX	QREOR-IHNMX	LFQDH-ACJIG	BZCYT-VXLBC	BZCYT-VXLBC	NVZCQ-TGKDX	KQQUG-OVNBK	BLHVW-ATPTV	KQQUG-OVNBK
4	LHTXJ-TGKDX	KQQUG-OVNBK	NVZCQ-TGKDX	KQQUG-OVNBK	LHTXJ-TGKDX	LHTXJ-TGKDX	BZCYT-VXLBC	QREOR-IHNMX	TNPJQ-LRZLR	NVZCQ-TGKDX
5	PSVVR-XSJJE	LHTXJ-TGKDX	LHTXJ-TGKDX	OVNBK-GRPNX	PSVVR-XSJJE	PSVVR-XSJJE	KQQUG-OVNBK	LHTXJ-TGKDX	UHLAI-WFZIE	PSVVR-XSJJE
6	NVZCQ-TGKDX	BZCYT-VXLBC	LVFYZ-SYVIQ	OKPGU-FZIOY	NVZCQ-TGKDX	NVZCQ-TGKDX	LHTXJ-TGKDX	LVFYZ-SYVIQ	XXVQK-LOBQA	BZCYT-VXLBC
7	TNPJQ-LRZLR	OVNBK-GRPNX	BZCYT-VXLBC	PSVVR-XSJJE	TNPJQ-LRZLR	TNPJQ-LRZLR	PSVVR-XSJJE	PSVVR-XSJJE	XLLLQ-OKPGU	HUSJF-UHLAI
8	HUSJF-UHLAI	LVFYZ-SYVIQ	OVNBK-GRPNX	WBTAG-QMRPP	HUSJF-UHLAI	HUSJF-UHLAI	HUSJF-UHLAI	BZCYT-VXLBC	OKPGU-FZIOY	LHTXJ-TGKDX
9	ZQZSA-NVZCQ	PSVVR-XSJJE	HUSJF-UHLAI	BZCYT-VXLBC	ZQZSA-NVZCQ	ZQZSA-NVZCQ	OVNBK-GRPNX	NVZCQ-LHTXJ	HUSJF-UHLAI	IQXOG-LENWX
10	NVZCQ-TNPJQ	HUSJF-UHLAI	PSVVR-XSJJE	HUSJF-UHLAI	NVZCQ-TNPJQ	NVZCQ-TNPJQ	NVZCQ-LHTXJ	OVNBK-GRPNX	NKEGR-VXLBC	OVNBK-GRPNX
11	OVNBK-GRPNX	LRZLR-BFFJE	LRZLR-BFFJE	UHLAI-WFZIE	OVNBK-GRPNX	OVNBK-GRPNX	NVZCQ-TNPJQ	LRZLR-BFFJE	LHTXJ-TGKDX	GELAB-TVLNM
12	BLHVW-ATPTV	NVZCQ-LHTXJ	NVZCQ-LHTXJ	IQXOG-LENWX	BLHVW-ATPTV	BLHVW-ATPTV	ZKNAC-QMRPP	HUSJF-UHLAI	QREOR-IHNMX	BLHVW-ATPTV
13	BZCYT-NKEGR	ZKNAC-QMRPP	ZKNAC-QMRPP	NVZCQ-LRZLR	BZCYT-NKEGR	BZCYT-NKEGR	BLHVW-ATPTV	NVZCQ-LRZLR	PSVVR-XSJJE	ZKNAC-QMRPP
14	IQXOG-LENWX	NVZCQ-BFFJE	IQXOG-LENWX	LHTXJ-TGKDX	IQXOG-LENWX	IQXOG-LENWX	LVFYZ-SYVIQ	NVZCQ-TNPJQ	LRZLR-JKBML	NVZCQ-LHTXJ
15	UHLAI-WFZIE	NVZCQ-TNPJQ	NVZCQ-TNPJQ	NVZCQ-LHTXJ	UHLAI-WFZIE	UHLAI-WFZIE	IQXOG-LENWX	LDASI-HUSJF	OVNBK-GRPNX	ZQZSA-NVZCQ
16	XLLLQ-OKPGU	IQXOG-LENWX	LDASI-HUSJF	NVZCQ-BFFJE	XLLLQ-OKPGU	XLLLQ-OKPGU	NVZCQ-LRZLR	ZKNAC-QMRPP	GPQBK-JBHAY	NVZCQ-LRZLR
17	IXPBX-QQOCG	NVZCQ-LRZLR	NVZCQ-BFFJE	XXVQK-LOBQA	IXPBX-QQOCG	IXPBX-QQOCG	ZQZSA-NVZCQ	BZCYT-NKEGR	VZJTC-PEICU	LDASI-UHLAI
18	LRZLR-JKBML	BLHVW-ATPTV	BLHVW-ATPTV	ZKNAC-QMRPP	LRZLR-JKBML	LRZLR-JKBML	LDASI-HUSJF	NVZCQ-BFFJE	BZCYT-VXLBC	GPQBK-JBHAY
19	KQQUG-OVNBK	WBTAG-QMRPP	XXVQK-LOBQA	VZJTC-PEICU	KQQUG-OVNBK	KQQUG-OVNBK	GPQBK-JBHAY	IQXOG-LENWX	MSSZC-GNPOB	LVFYZ-SYVIQ
20	WBTAG-QMRPP	BZCYT-NKEGR	NVZCQ-LRZLR	LRZLR-JKBML	WBTAG-QMRPP	WBTAG-QMRPP	TNPJQ-LRZLR	HUSJF-WFZIE	ZQZSA-TGKDX	LRZLR-JKBML
21	VZJTC-PEICU	TNPJQ-LRZLR	MSSZC-GNPOB	BLHVW-ATPTV	VZJTC-PEICU	VZJTC-PEICU	IXPBX-QQOCG	OKPGU-FZIOY	BZCYT-NKEGR	NVZCQ-TNPJQ
22	BXHSU-DRZGY	GPQBK-JBHAY	ZJFVA-VXLBC	LVFYZ-SYVIQ	BXHSU-DRZGY	BXHSU-DRZGY	BZCYT-NKEGR	ZQZSA-NVZCQ	NVZCQ-LRZLR	TNPJQ-LRZLR
23	GPQBK-JBHAY	IXPBX-QQOCG	HUSJF-WFZIE	MSSZC-GNPOB	GPQBK-JBHAY	GPQBK-JBHAY	LDASI-UHLAI	VZJTC-PEICU	NVZCQ-LHTXJ	LDASI-HUSJF
24	NVZCQ-LRZLR	XXVQK-LOBQA	TNPJQ-LRZLR	XLLLQ-OKPGU	NVZCQ-LRZLR	NVZCQ-LRZLR	XLLLQ-OKPGU	XLLLQ-OKPGU	HUSJF-WFZIE	VZJTC-PEICU
25	MSSZC-GNPOB	LDASI-HUSJF	IXPBX-QQOCG	WBTAG-VIBJH	MSSZC-GNPOB	MSSZC-GNPOB	UHLAI-WFZIE	NKEGR-VXLBC	LDASI-UHLAI	XLLLQ-OKPGU
26	LDASI-UHLAI	UHLAI-WFZIE	WBTAG-QMRPP	IQXOG-LVFYZ	LDASI-UHLAI	LDASI-UHLAI	LRZLR-BFFJE	IXPBX-QQOCG	ZKNAC-QMRPP	WBTAG-QMRPP
27	GELAB-TVLNM	ZQZSA-NVZCQ	ZBJPA-BMJXU	NKEGR-VXLBC	GELAB-TVLNM	GELAB-TVLNM	HUSJF-WFZIE	WBTAG-VIBJH	TSOBU-MIRVW	IXPBX-QQOCG
28	ZQZSA-TGKDX	XLLLQ-OKPGU	UHLAI-WFZIE	ZQZSA-NVZCQ	ZQZSA-TGKDX	ZQZSA-TGKDX	NVZCQ-BFFJE	BXHSU-DRZGY	IXPBX-QQOCG	ZBJPA-BMJXU

Continued on next page

Table A.1: Raking of most similar trajectory pairs

Rank	Euclidean	DTW	SSPD	Hausdorff	ERP	MSM; cost=1	MSM; cost=.1	MSM; cost=.01	EDR; $\epsilon = .203$	EDR; $\epsilon = .101$
29	WBTAG-VIBJH	MSSZC-GNPOB	BZCYT-NKEGR	LRZLR-BFFJE	WBTAG-VIBJH	WBTAG-VIBJH	XXVQK-LOBQA	XXVQK-LOBQA	WBTAG-VIBJH	UTNLD-XSJJE
30	IQXOG-LVFYZ	HUSJF-WFZIE	LDASI-UHLAI	TNPJQ-LRZLR	IQXOG-LVFYZ	IQXOG-LVFYZ	VZJTC-PEICU	UHLAI-WFZIE	BXHSU-DRZGY	HUSJF-WFZIE
31	OKPGU-FZIOY	OKPGU-FZIOY	OKPGU-FZIOY	BXHSU-DRZGY	OKPGU-FZIOY	OKPGU-FZIOY	BXHSU-DRZGY	TNPJQ-LRZLR	IQXOG-LVFYZ	BZCYT-NKEGR
32	LDASI-HUSJF	VZJTC-PEICU	BZCYT-ZJFVA	BZCYT-NKEGR	LDASI-HUSJF	LDASI-HUSJF	LRZLR-JKBML	BLHVW-ATPTV	NVZCQ-TNPJQ	BXHSU-DRZGY
33	XXVQK-LOBQA	LRZLR-JKBML	GPQBK-JBHAY	HUSJF-WFZIE	XXVQK-LOBQA	XXVQK-LOBQA	MSSZC-GNPOB	ZBJPA-BMJXU	IQXOG-LENWX	UHLAI-WFZIE
34	ZJFVA-VXLBC	LDASI-UHLAI	VKGQV-BLHVW	AEPAP-XLLLQ	ZJFVA-VXLBC	ZJFVA-VXLBC	ZQZSA-TGKDX	WBTAG-QMRPP	NVZCQ-TGKDX	ZQZSA-TGKDX
35	ZKNAC-QMRPP	WBTAG-VIBJH	XLLLQ-OKPGU	VKGQV-BLHVW	ZKNAC-QMRPP	ZKNAC-QMRPP	WBTAG-QMRPP	LDASI-UHLAI	LRZLR-BFFJE	MSSZC-GNPOB
36	NKEGR-VXLBC	ZBJPA-BMJXU	WBTAG-VIBJH	BZCYT-ZJFVA	NKEGR-VXLBC	NKEGR-VXLBC	ZBJPA-BMJXU	QQOCG-FIJTA	KQQUG-OVNBK	TSOBU-MIRVW
37	VKGQV-BLHVW	NKEGR-VXLBC	VZJTC-PEICU	GPQBK-JBHAY	VKGQV-BLHVW	VKGQV-BLHVW	WBTAG-VIBJH	LRZLR-JKBML	LVFYZ-SYVIQ	NVZCQ-BFFJE
38	NVZCQ-LHTXJ	BXHSU-DRZGY	ZQZSA-NVZCQ	IXPBX-QQOCG	NVZCQ-LHTXJ	NVZCQ-LHTXJ	VKGQV-BLHVW	GPQBK-JBHAY	ZQZSA-NVZCQ	VKGQV-BLHVW
39	LRZLR-BFFJE	ZQZSA-TGKDX	AEPAP-XLLLQ	ZQZSA-TGKDX	LRZLR-BFFJE	LRZLR-BFFJE	GELAB-TVLNM	MSSZC-GNPOB	NVZCQ-BFFJE	AEPAP-XLLLQ
40	AEPAP-XLLLQ	VKGQV-BLHVW	LRZLR-JKBML	GELAB-TVLNM	AEPAP-XLLLQ	AEPAP-XLLLQ	BZCYT-ZJFVA	ZJFVA-VXLBC	QQOCG-FIJTA	BZCYT-ZJFVA
41	HUSJF-WFZIE	BZCYT-ZJFVA	BLHVW-VPCTJ	QQOCG-FIJTA	HUSJF-WFZIE	HUSJF-WFZIE	OKPGU-FZIOY	AEPAP-XLLLQ	AEPAP-XLLLQ	XXVQK-LOBQA
42	NVZCQ-BFFJE	UTNLD-XSJJE	BXHSU-DRZGY	BLHVW-VPCTJ	NVZCQ-BFFJE	NVZCQ-BFFJE	UTNLD-XSJJE	VKGQV-BLHVW	LDASI-HUSJF	LRZLR-BFFJE
43	QQOCG-FIJTA	AEPAP-XLLLQ	NKEGR-VXLBC	NVZCQ-TNPJQ	QQOCG-FIJTA	QQOCG-FIJTA	BLHVW-VPCTJ	BZCYT-ZJFVA	GELAB-TVLNM	WBTAG-VIBJH
44	TSOBU-MIRVW	ZJFVA-VXLBC	ZQZSA-TGKDX	ZJFVA-VXLBC	TSOBU-MIRVW	TSOBU-MIRVW	NKEGR-VXLBC	ZQZSA-TGKDX	BZCYT-LPNRG	BLHVW-VPCTJ
45	BZCYT-ZJFVA	BLHVW-VPCTJ	QQOCG-FIJTA	BZCYT-LPNRG	LVFYZ-SYVIQ	BZCYT-ZJFVA	IQXOG-LVFYZ	IQXOG-LVFYZ	ZBJPA-BMJXU	ZJFVA-VXLBC
46	LVFYZ-SYVIQ	QQOCG-FIJTA	GELAB-TVLNM	LDASI-UHLAI	BZCYT-ZJFVA	LVFYZ-SYVIQ	ZJFVA-VXLBC	BLHVW-VPCTJ	UTNLD-XSJJE	IQXOG-LVFYZ
47	BZCYT-LPNRG	IQXOG-LVFYZ	IQXOG-LVFYZ	TSOBU-MIRVW	BZCYT-LPNRG	BZCYT-LPNRG	TSOBU-MIRVW	GELAB-TVLNM	VKGQV-BLHVW	OKPGU-FZIOY
48	UTNLD-XSJJE	GELAB-TVLNM	UTNLD-XSJJE	ZBJPA-BMJXU	UTNLD-XSJJE	UTNLD-XSJJE	QQOCG-FIJTA	BZCYT-LPNRG	BZCYT-ZJFVA	QQOCG-FIJTA
49	BLHVW-VPCTJ	TSOBU-MIRVW	TSOBU-MIRVW	LDASI-HUSJF	BLHVW-VPCTJ	BLHVW-VPCTJ	AEPAP-XLLLQ	TSOBU-MIRVW	BLHVW-VPCTJ	BZCYT-LPNRG
50	ZBJPA-BMJXU	BZCYT-LPNRG	BZCYT-LPNRG	UTNLD-XSJJE	ZBJPA-BMJXU	ZBJPA-BMJXU	BZCYT-LPNRG	UTNLD-XSJJE	ZJFVA-VXLBC	NKEGR-VXLBC

Appendix B

Affinity Propagation Clusters

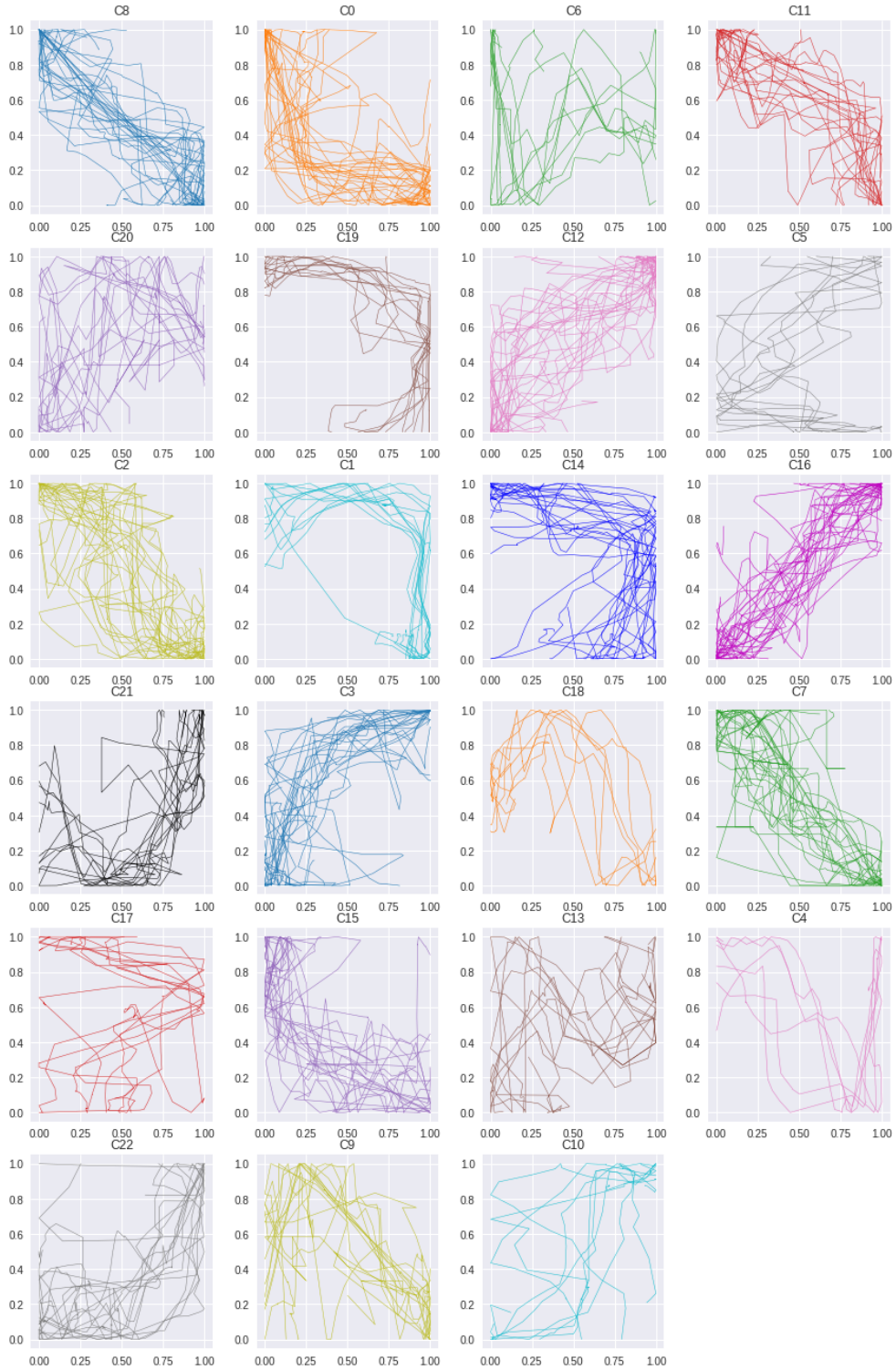


Figure B.1: Affinity Propagation: Dynamic Time Warping

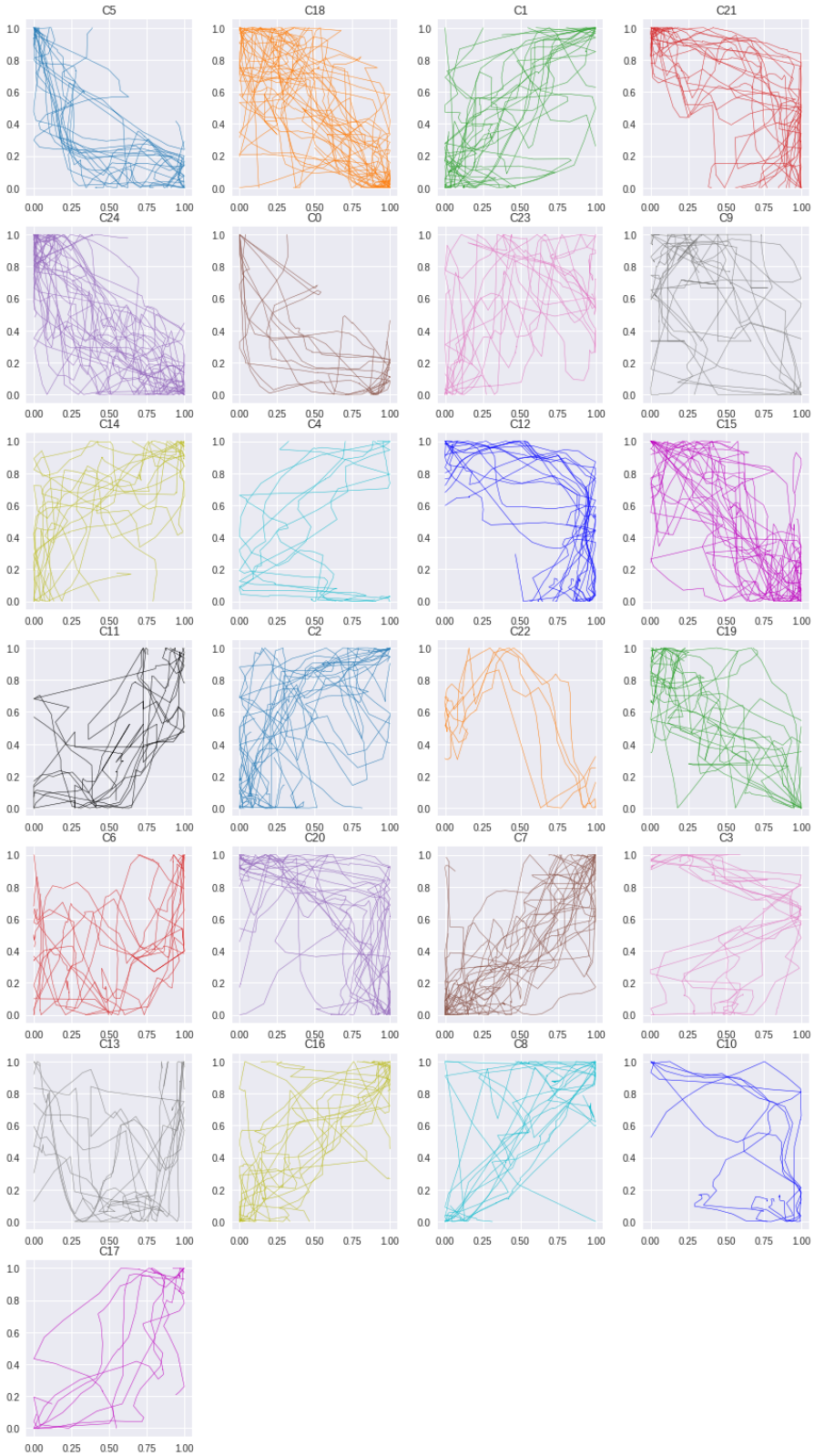


Figure B.2: Affinity Propagation: Euclidean Distance

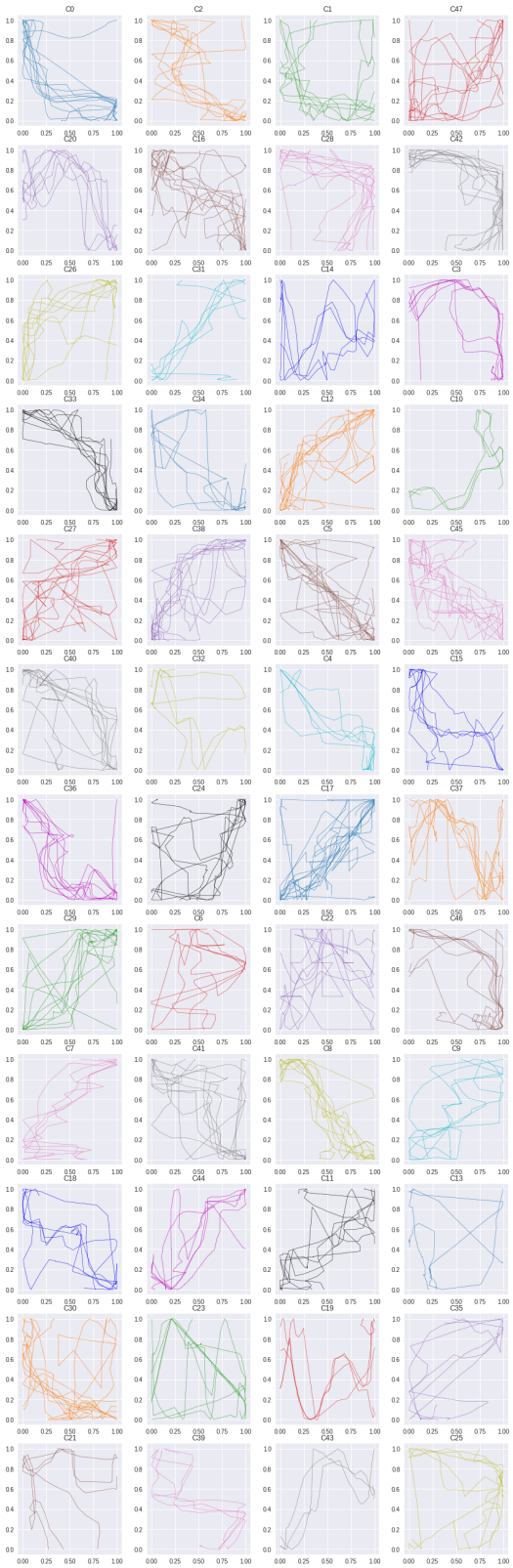


Figure B.3: Affinity Propagation: Edit Distance on Real Sequence, $\epsilon = 0.101$

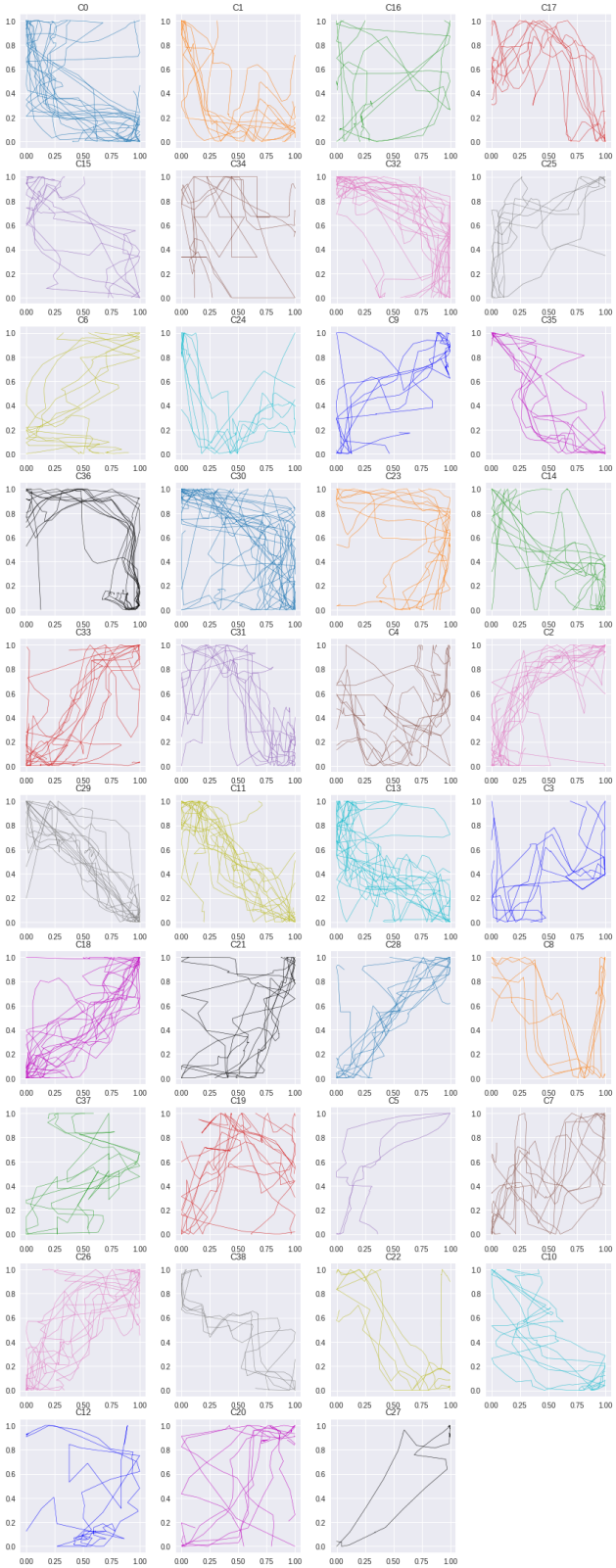


Figure B.4: Affinity Propagation: Edit Distance on Real Sequence, $\epsilon = 0.203$

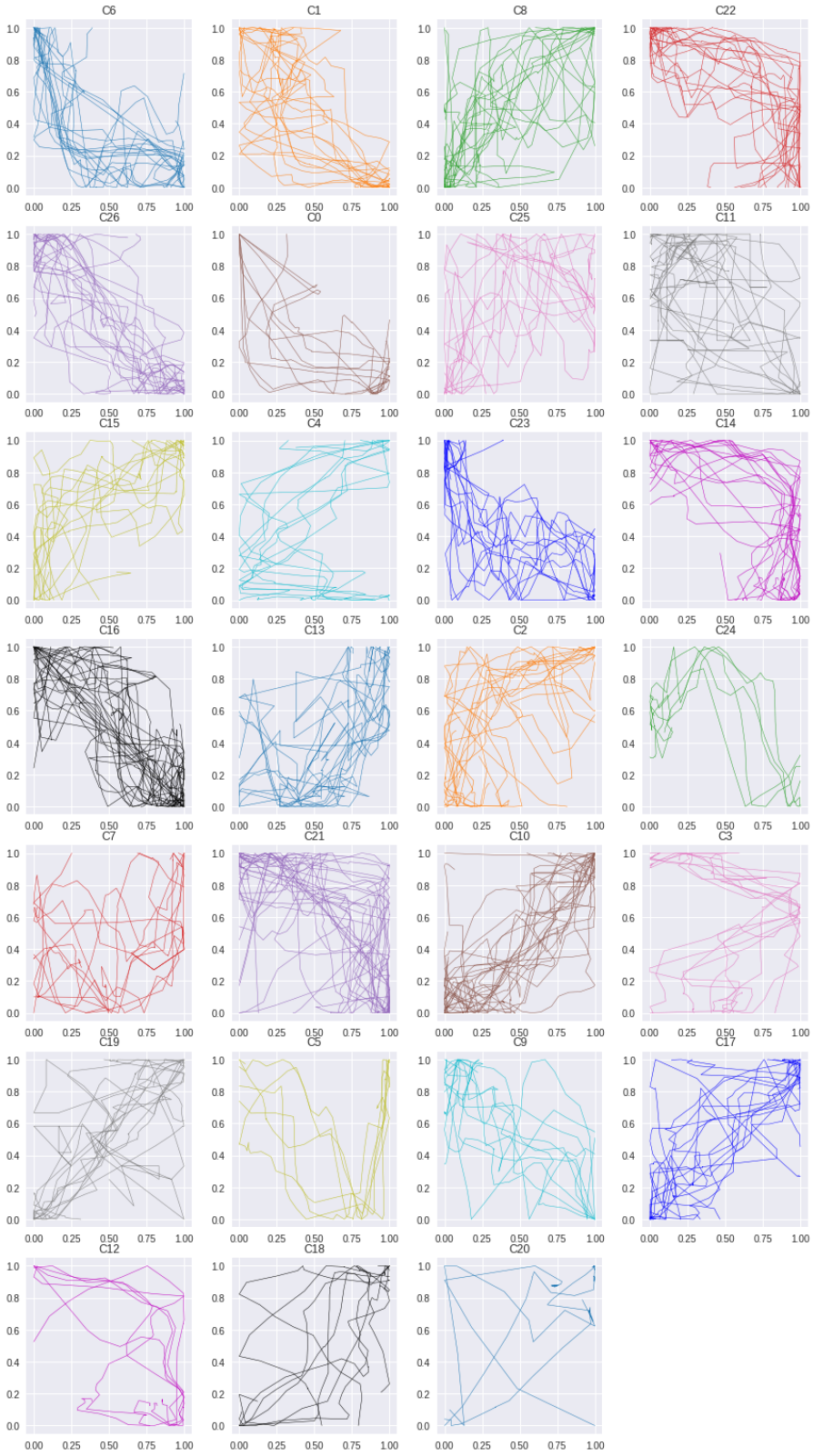


Figure B.5: Affinity Propagation: Edit Distance with Real Penalty

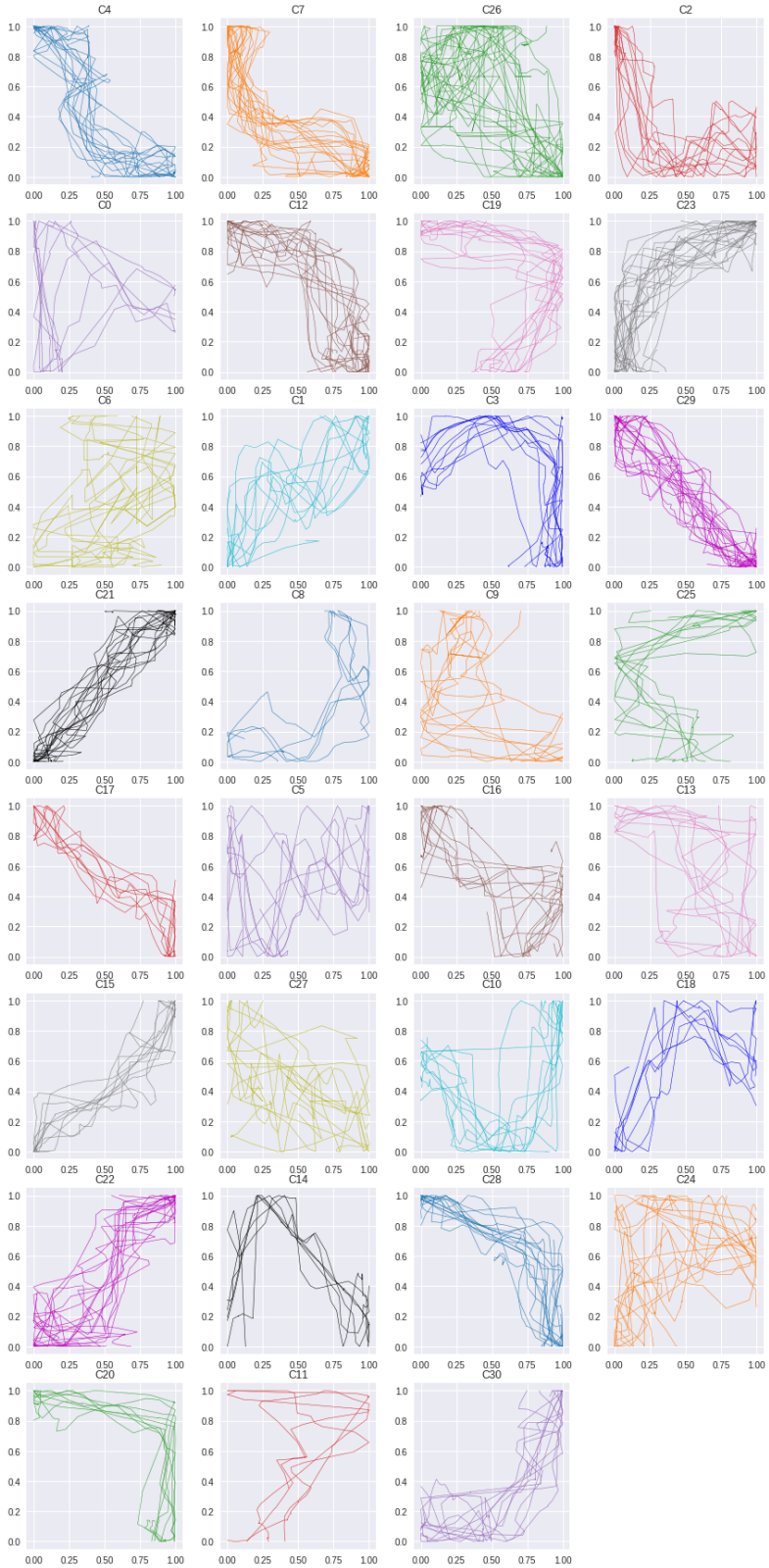


Figure B.6: Affinity Propagation: Hausdorff Distance

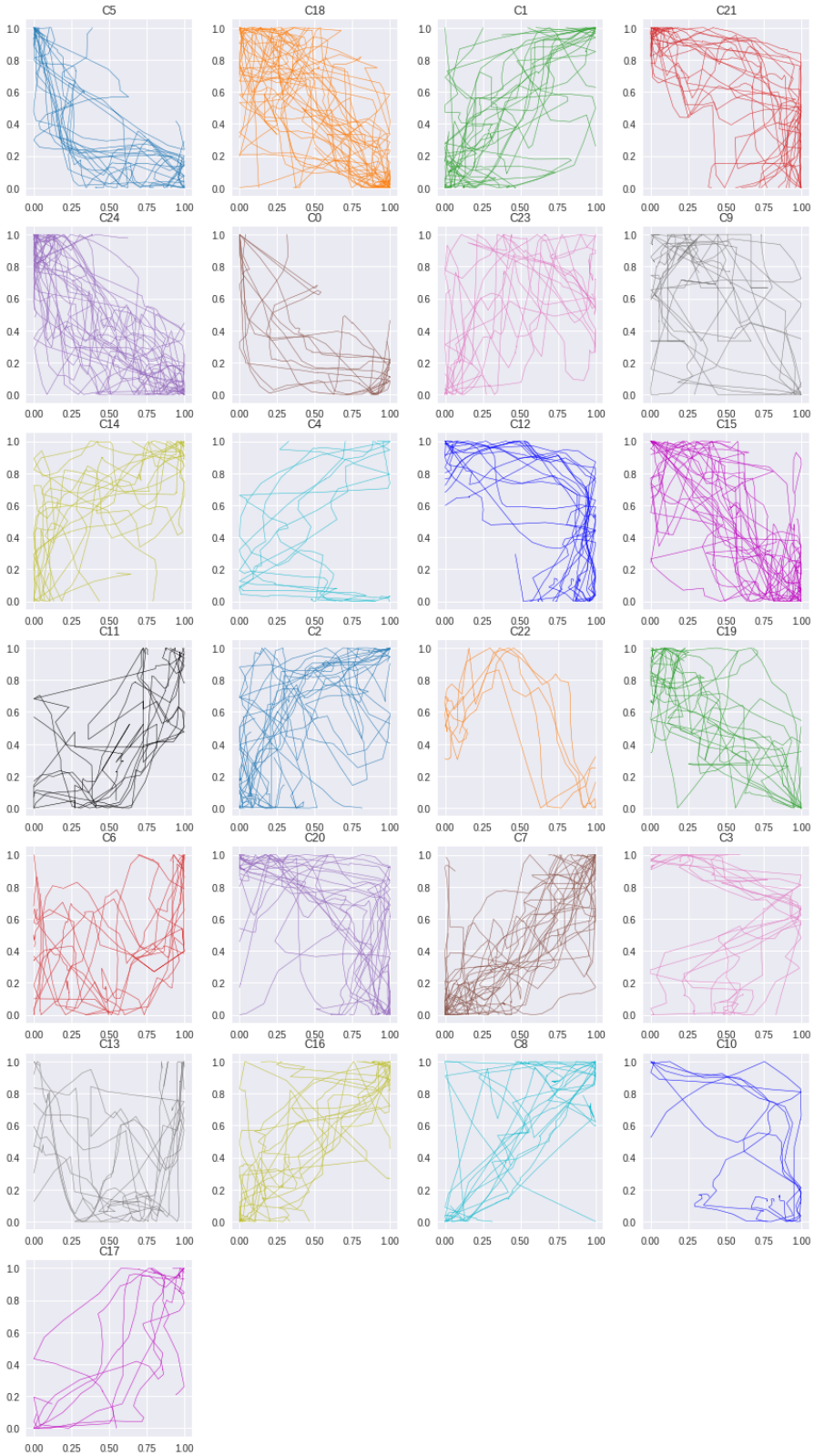


Figure B.7: Affinity Propagation: Move-Split-Merge, cost= 1

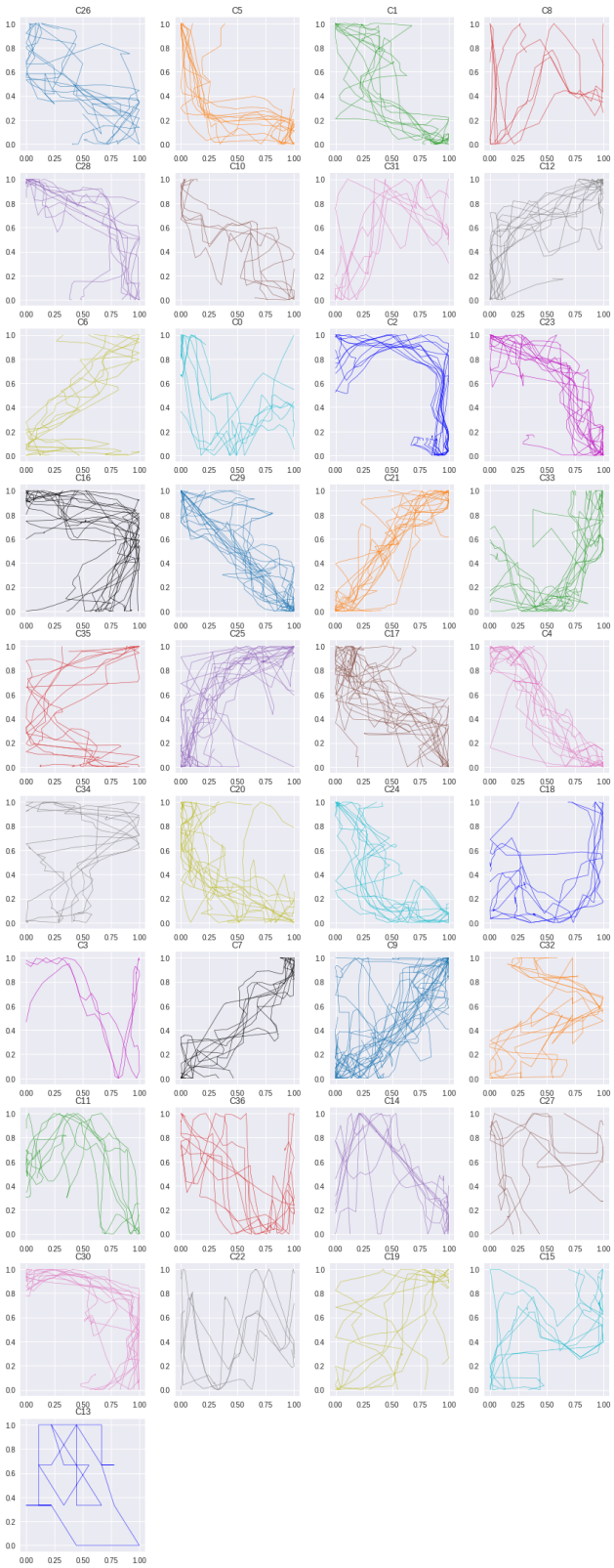


Figure B.8: Affinity Propagation:Move-Split-Merge, cost= 0.1

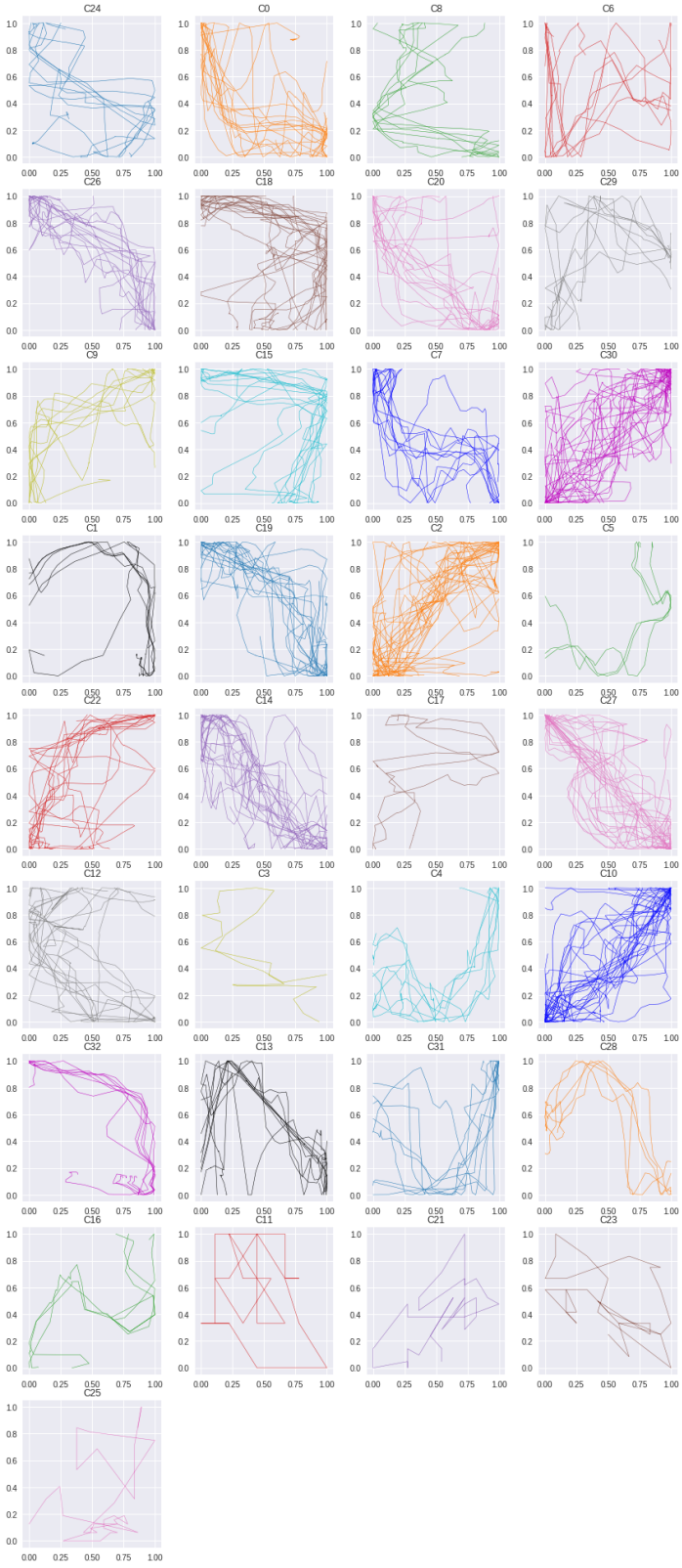


Figure B.9: Affinity Propagation:Move-Split-Merge, cost= 0.01

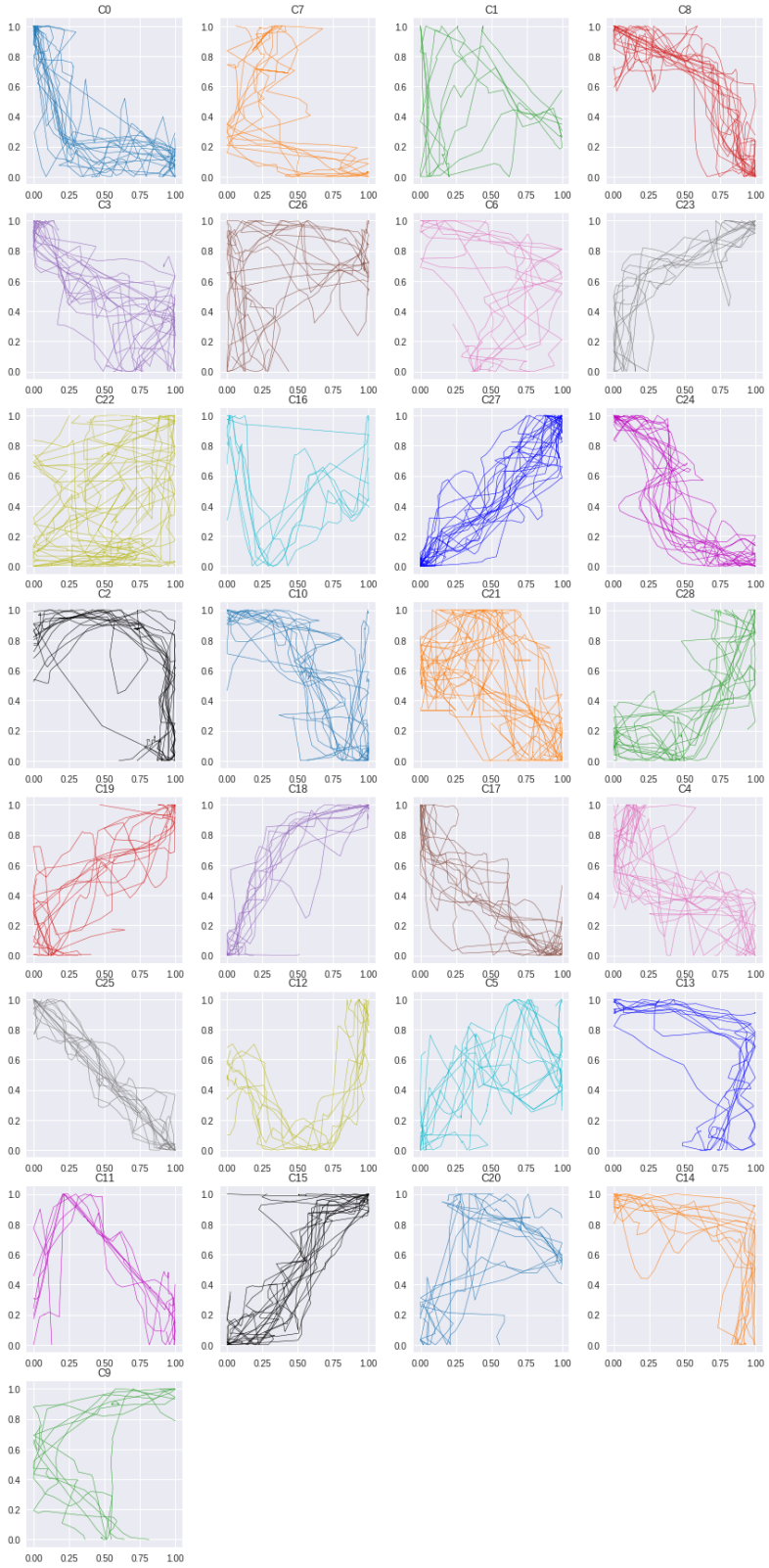


Figure B.10: Affinity Propagation: Symmetrized Segment-Path Distance

Appendix C

Agglomerative Hierarchical Clusters

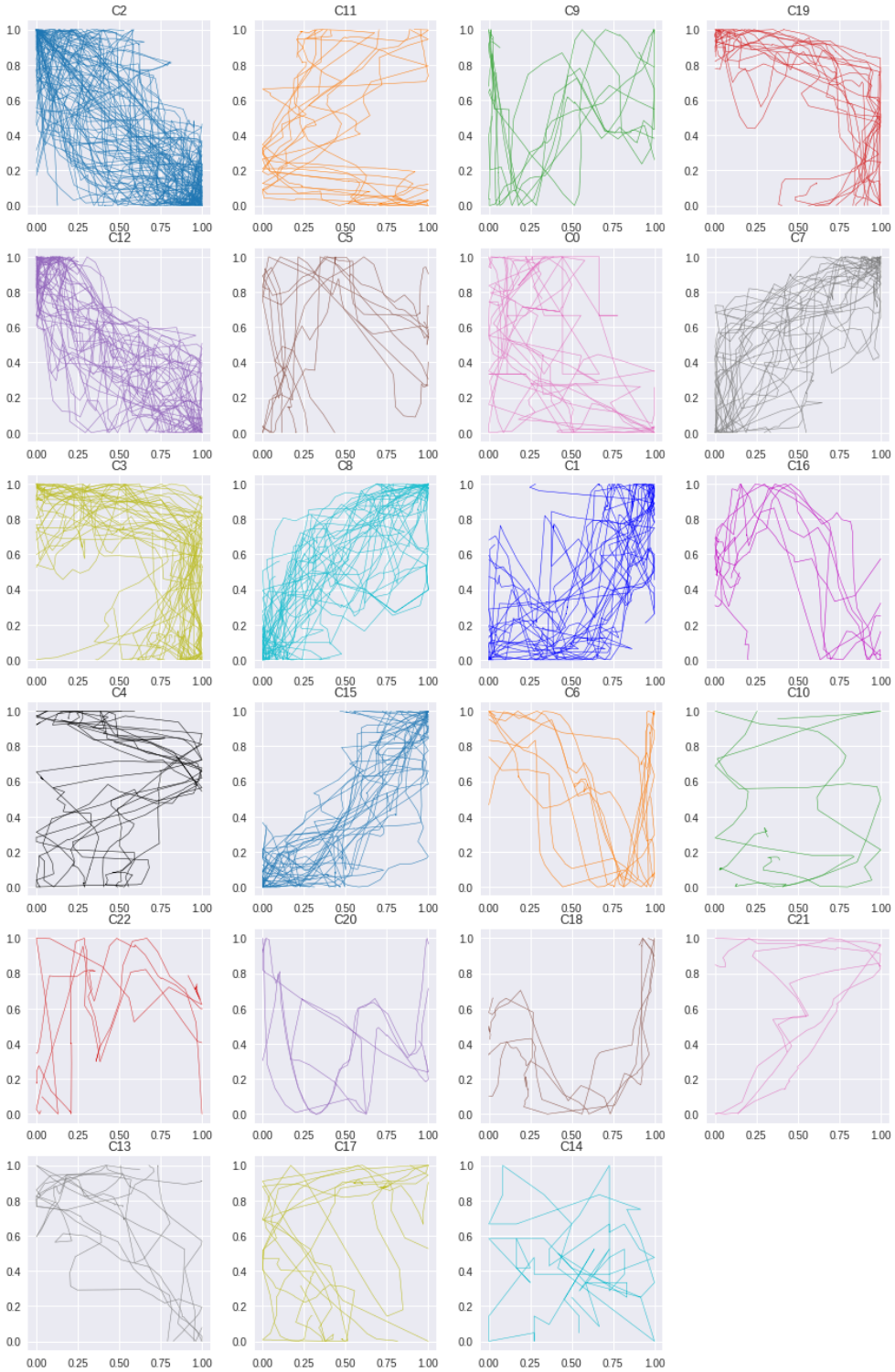


Figure C.1: Hierarchical Clusters: Dynamic Time Warping

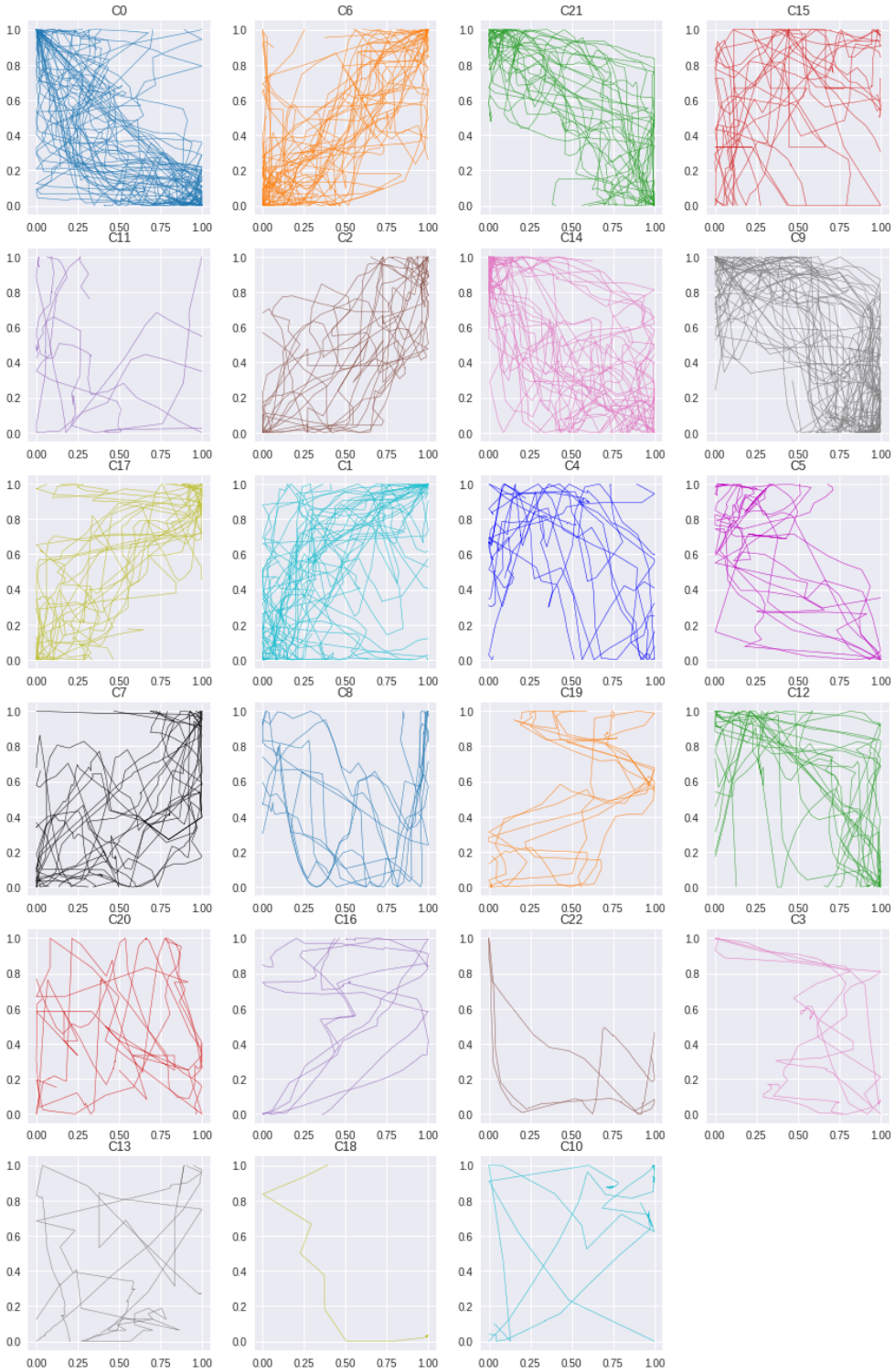


Figure C.2: Hierarchical Clusters: Euclidean Distance

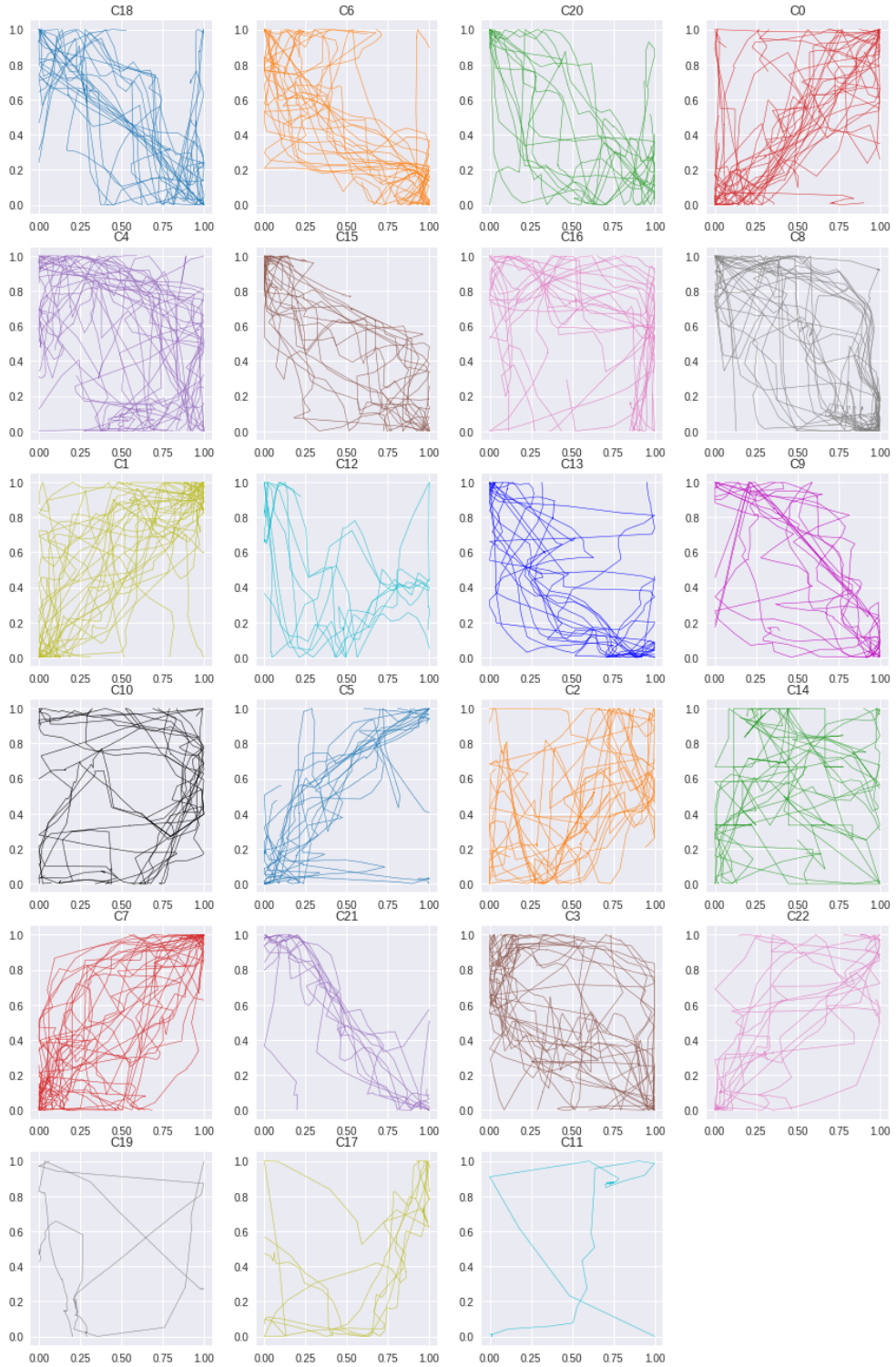


Figure C.3: Hierarchical Clusters: Edit Distance on Real Sequence, $\epsilon = 0.101$

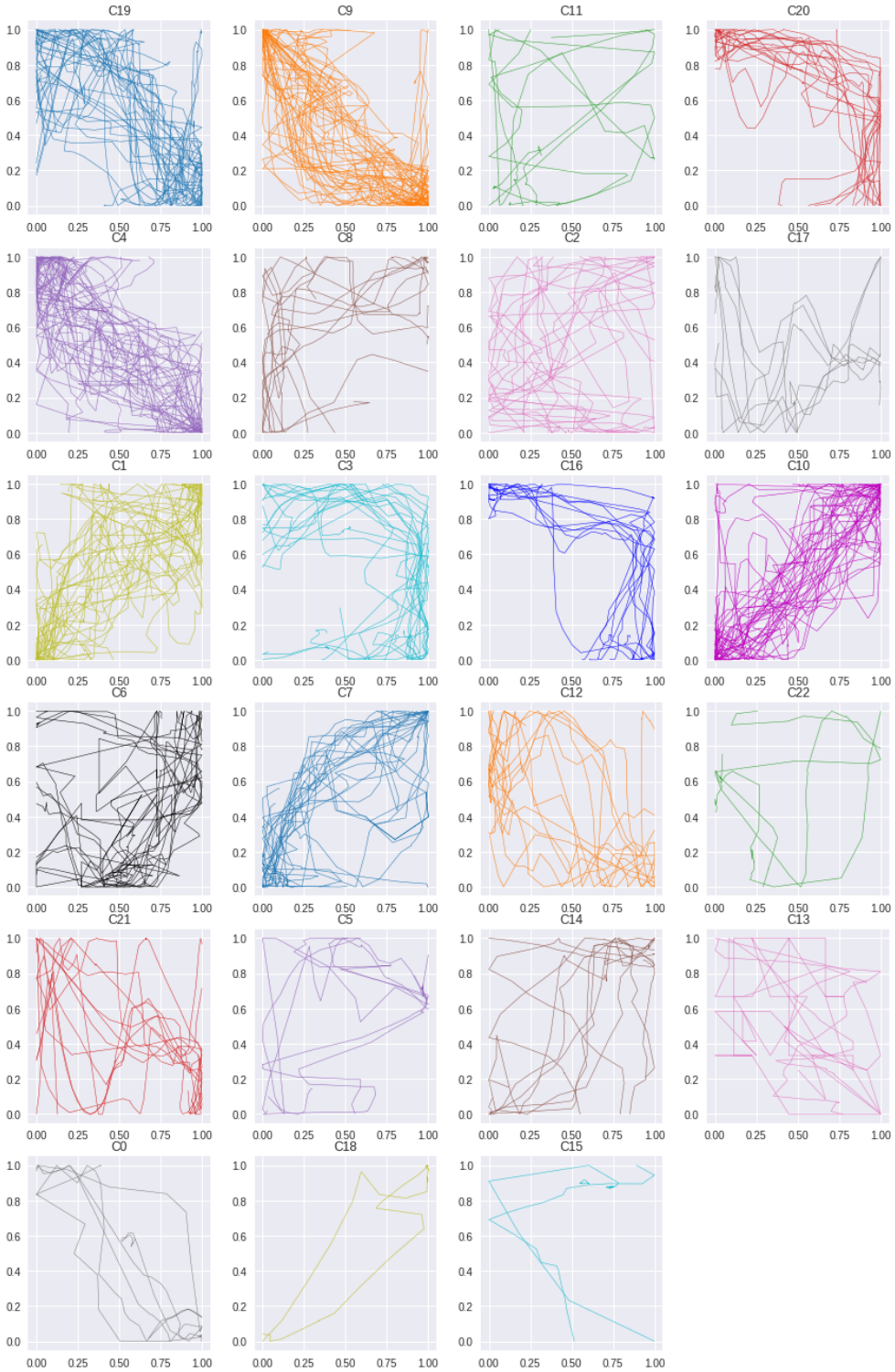


Figure C.4: Hierarchical Clusters: Edit Distance on Real Sequence, $\epsilon = 0.203$

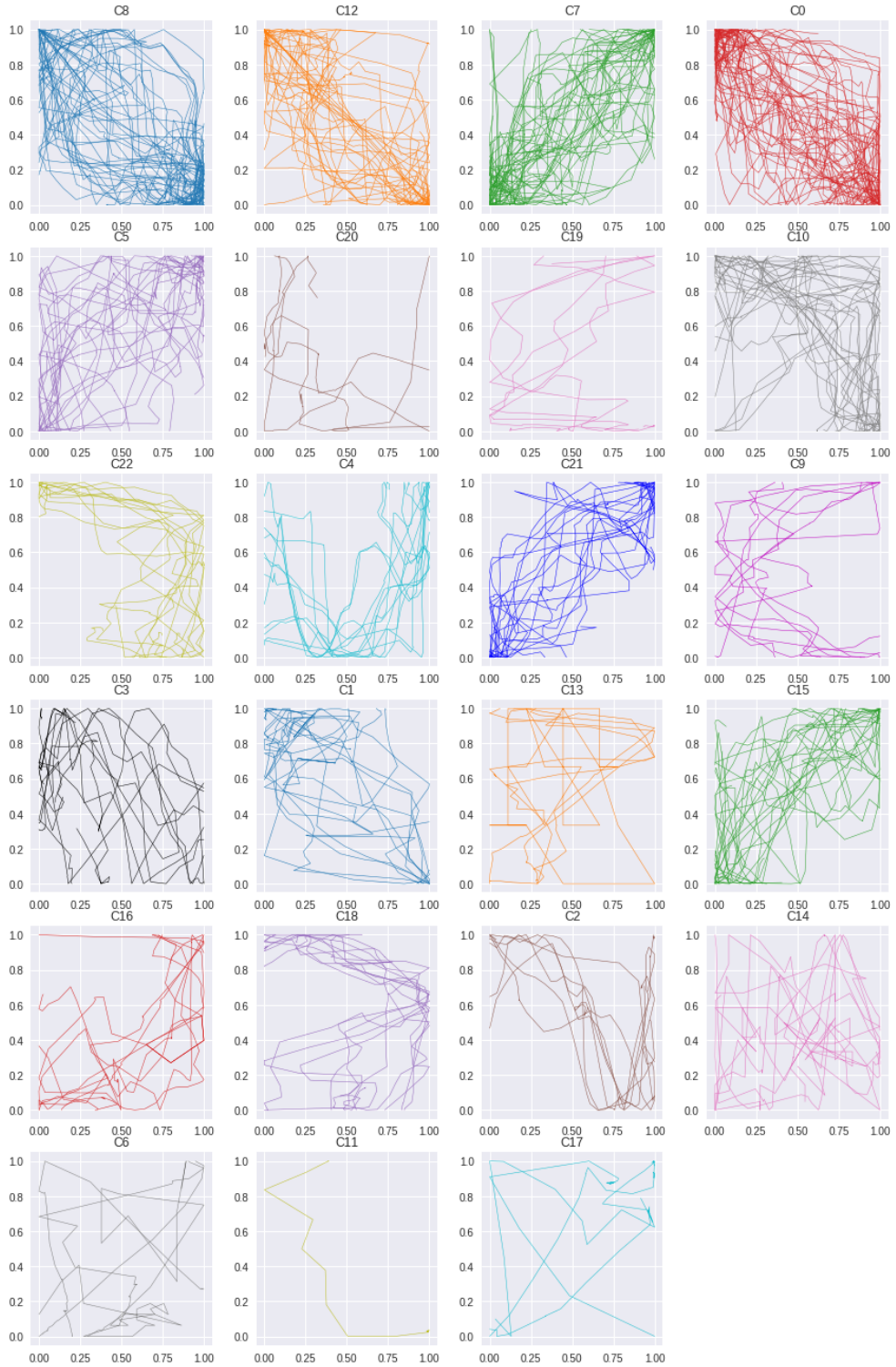


Figure C.5: Hierarchical Clusters: Edit Distance with Real Penalty

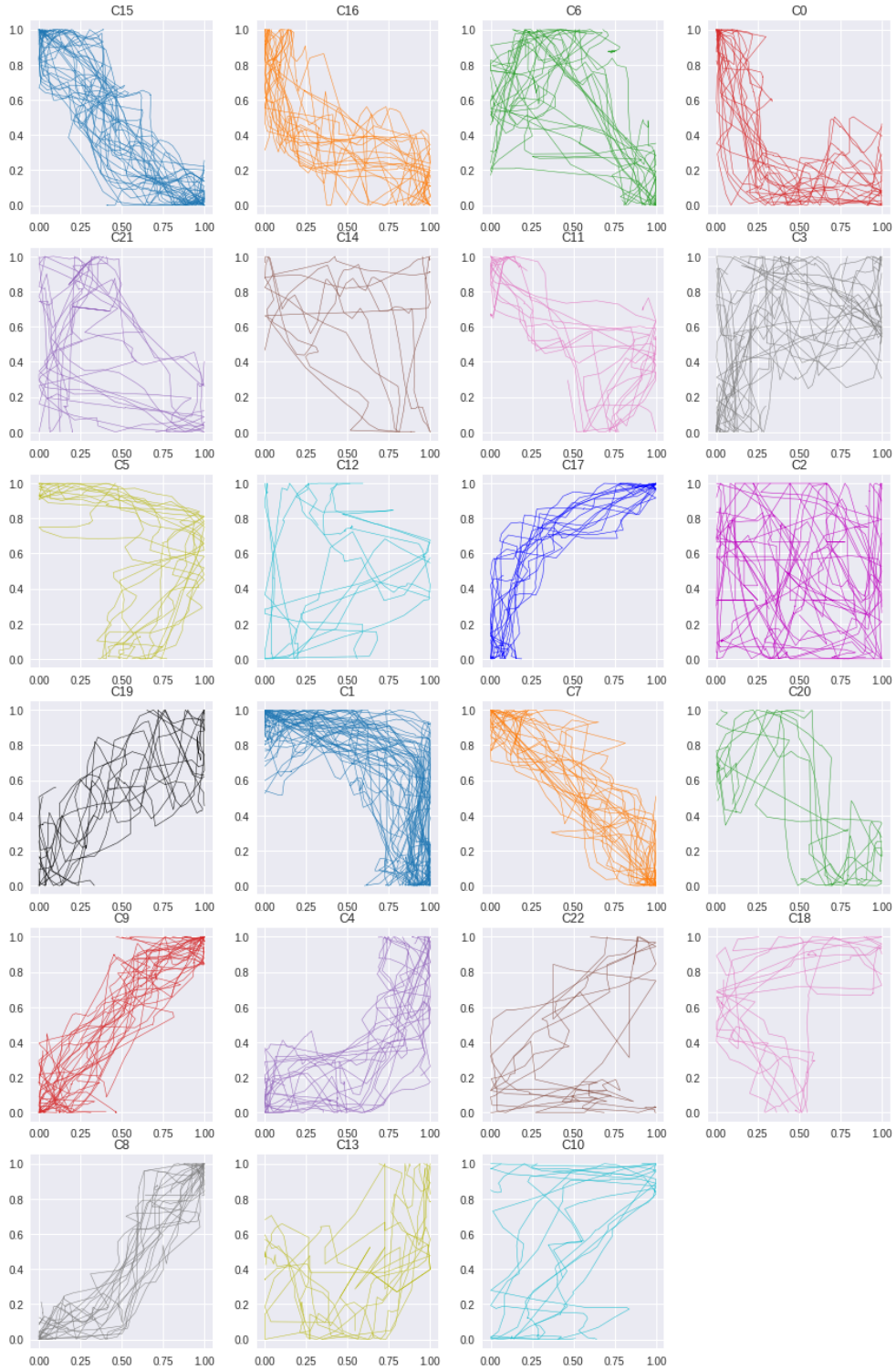


Figure C.6: Hierarchical Clusters: Hausdorff Distance

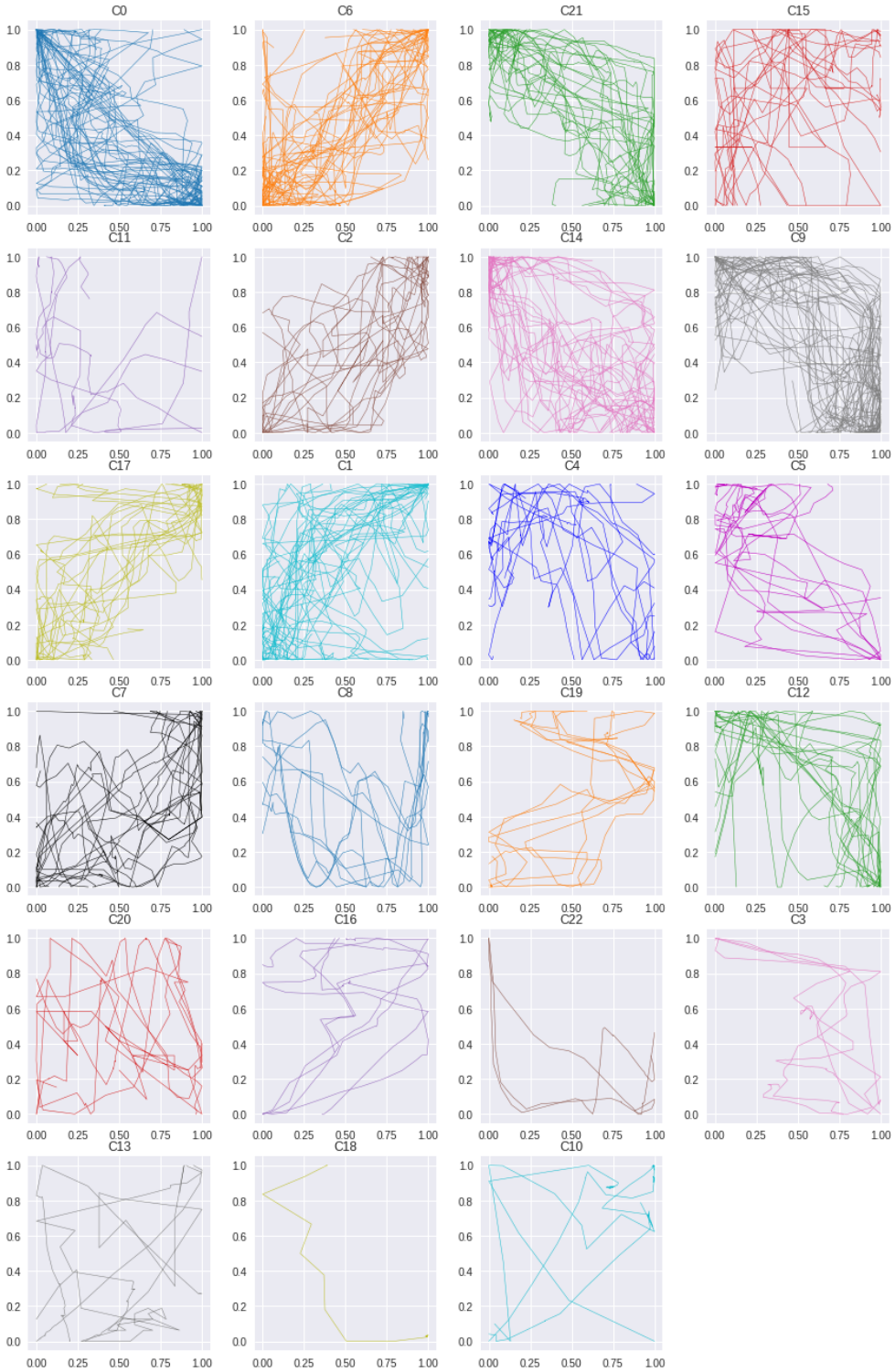


Figure C.7: Hierarchical Clusters: Move-Split-Merge, cost= 1

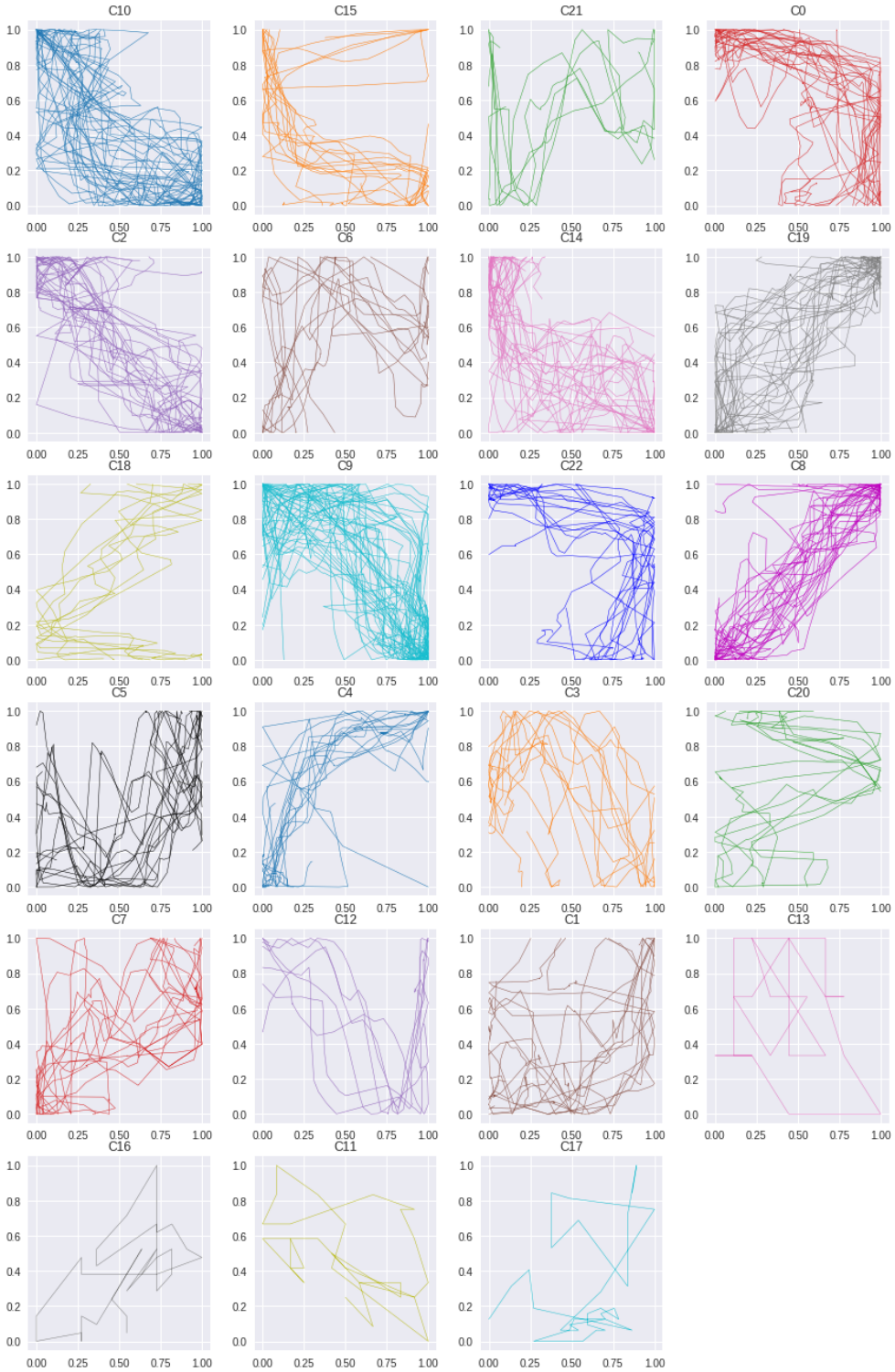


Figure C.8: Hierarchical Clusters:Move-Split-Merge, cost= 0.1

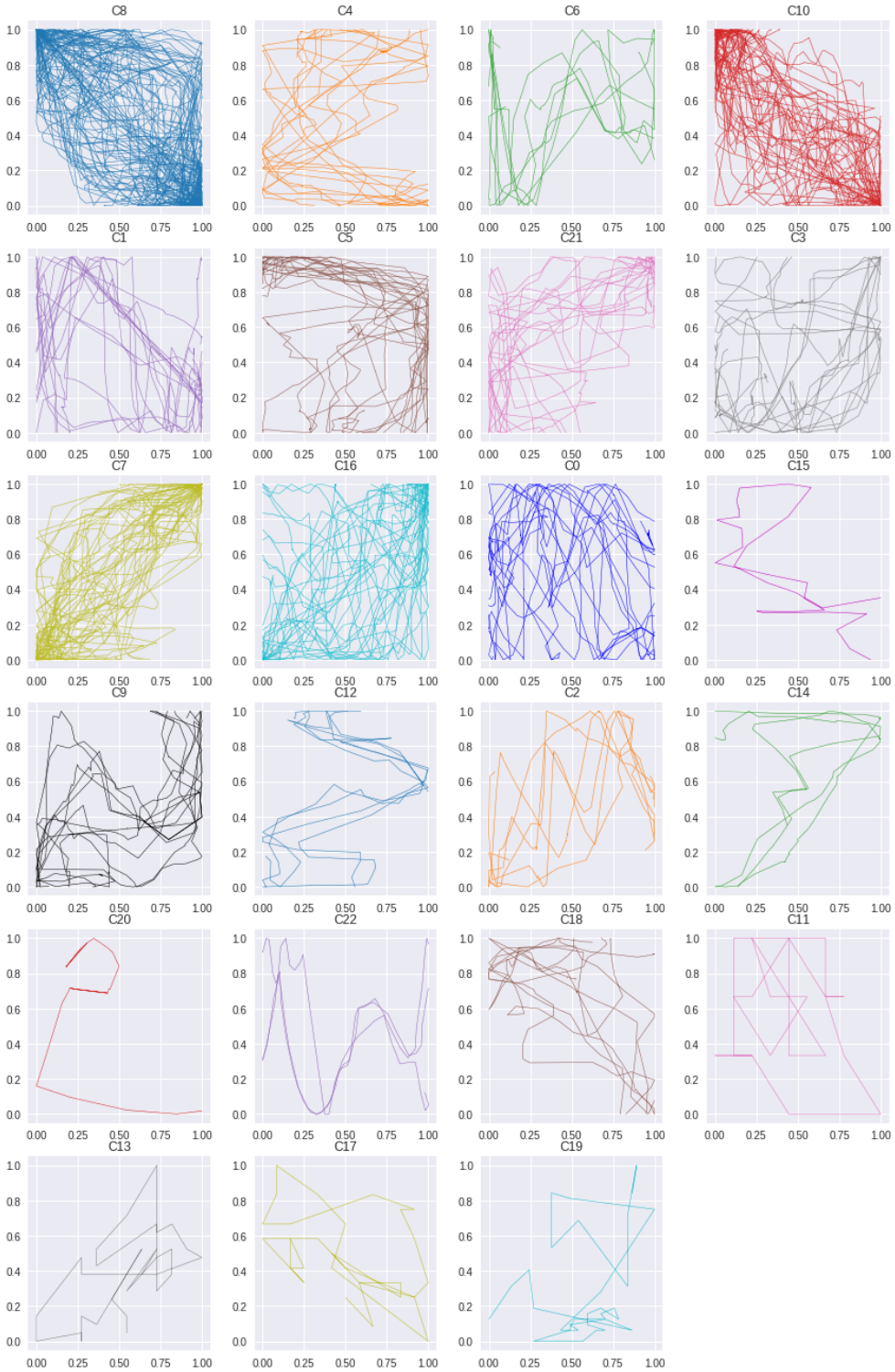


Figure C.9: Hierarchical Clusters:Move-Split-Merge, cost= 0.01

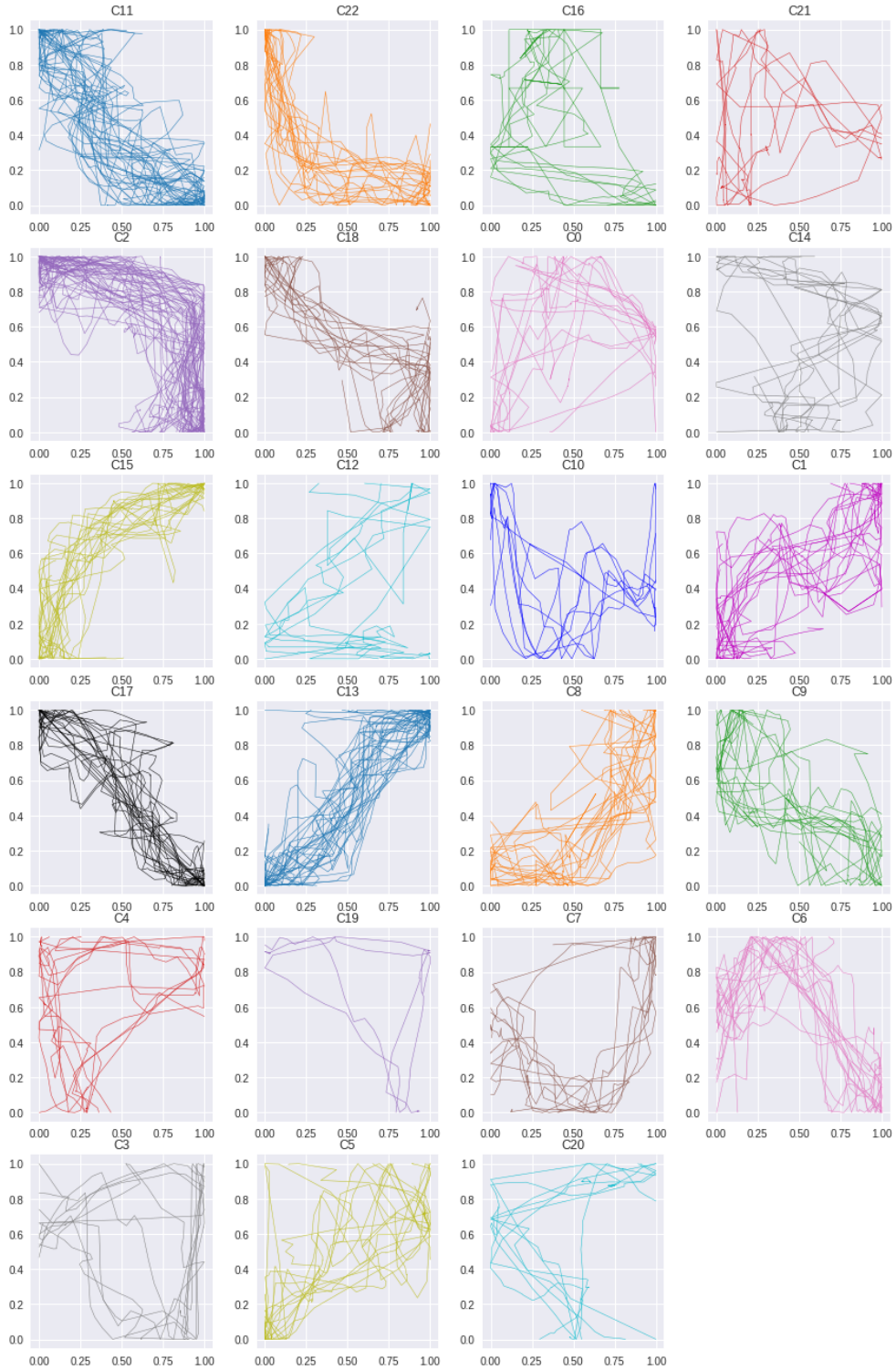


Figure C.10: Hierarchical Clusters: Symmetrized Segment-Path Distance

Katrine Lie Holm

Surveying Similarly Distance for Trajectory Data

NTNU
Norwegian University of
Science and Technology