UNIVERSITY OF PIRAEUS
Department of Digital Systems

Comparative Analysis of Trajectory Similarity Techniques for Vessels in Real Time: A Case Study on Maritime Traffic Monitoring

Papageorgopoulos Nikos

ME2031

A thesis submitted in partial fulfillment for the
Postgraduate degree

in the
Information Systems & Services
Area of Study: Big Data and Analytics

Supervising Professor: Christos Doulkeridis

Athens 2023

The paths we take in life are infinite, and the trajectories we follow are as unique as the stars in the sky

Megan Mayhew Bergman

Abstract

The objective of this thesis paper is to compare the performance of trajectory similarity techniques for vessels in real-time. The study presents a comprehensive review of multiple trajectory similarity techniques and identifies the most widely used methods. The methods selected for comparison include Lock-Step Euclidean distance, Dynamic Time Warping and Longest Common Subsequence.

The study begins with a comprehensive examination of existing trajectory similarity techniques and their applications in the maritime domain. Following that, a fresh dataset comprised of various vessel trajectories is assembled in order to evaluate the performance and attributes of the chosen approaches. The evaluation is primarily concerned with computational efficiency.

In addition to the primary focus on comparing the performance and properties of trajectory similarity techniques for real-time vessel tracking, this thesis also encompasses a thorough analysis of the tools and supplementary techniques employed in the used algorithm. A distributed processing system is employed to compute the evaluations using Spark, notably Spark Streaming for real-time data. In terms of partitioning strategies, uniform grid partitioning has been applied.

Overall, a dataset of vessel trajectories is collected from a real time monitoring system of AIS, and the selected techniques are applied to the dataset. The results show that LSED performs better than the other two methods in terms of accuracy and computational efficiency. In general, the findings of this study contribute to the advancement of vessel trajectory analysis and provide guidance for selecting appropriate techniques for real-time vessel monitoring.

Περίληψη

Στόχος της παρούσας διατριβής είναι η σύγκριση των επιδόσεων των τεχνικών ομοιότητας τροχιάς για πλοία σε πραγματικό χρόνο. Η μελέτη παρουσιάζει μια ολοκληρωμένη ανασκόπηση των πολλαπλών τεχνικών ομοιότητας τροχιάς και προσδιορίζει τις πιο ευρέως χρησιμοποιούμενες μεθόδους. Οι μέθοδοι που επιλέχθηκαν για σύγκριση περιλαμβάνουν την Lock-Step Euclidean distance, την Dynamic Time Warping και την Longest Common Subsequence.

Η μελέτη ξεκινά με μια ολοκληρωμένη εξέταση των υφιστάμενων τεχνικών ομοιότητας τροχιάς και των εφαρμογών τους στον θαλάσσιο τομέα. Στη συνέχεια, συγκροτείται ένα νέο σύνολο δεδομένων που αποτελείται από διάφορες τροχιές πλοίων, προκειμένου να αξιολογηθούν οι επιδόσεις και τα χαρακτηριστικά των επιλεγμένων προσεγγίσεων. Η αξιολόγηση αφορά κυρίως την υπολογιστική απόδοση.

Εκτός από βασικό στόχο στη σύγκριση των επιδόσεων και των ιδιοτήτων των τεχνικών ομοιότητας τροχιάς για τον εντοπισμό πλοίων σε πραγματικό χρόνο, η παρούσα διατριβή περιλαμβάνει επίσης μια διεξοδική ανάλυση των εργαλείων και των συμπληρωματικών τεχνικών που χρησιμοποιούνται στον αλγόριθμο που χρησιμοποιείται. Για τον υπολογισμό των αξιολογήσεων χρησιμοποιείται ένα κατανεμημένο σύστημα επεξεργασίας με τη χρήση του Spark, ιδίως του Spark Streaming για δεδομένα πραγματικού χρόνου. Όσον αφορά τις στρατηγικές κατάτμησης, έχει εφαρμοστεί ομοιόμορφη κατάτμηση πλέγματος.

Συνολικά, συλλέγεται ένα σύνολο δεδομένων με τροχιές πλοίων από ένα σύστημα παρακολούθησης AIS σε πραγματικό χρόνο και οι επιλεγμένες τεχνικές εφαρμόζονται στο σύνολο δεδομένων. Τα αποτελέσματα δείχνουν ότι η LSED αποδίδει καλύτερα από τις άλλες δύο μεθόδους όσον αφορά την ακρίβεια και την υπολογιστική αποδοτικότητα. Σε γενικές γραμμές, τα ευρήματα της παρούσας μελέτης συμβάλλουν στην πρόοδο της ανάλυσης τροχιών πλοίων και παρέχουν καθοδήγηση για την επιλογή κατάλληλων τεχνικών για την παρακολούθηση πλοίων σε πραγματικό χρόνο.

Acknowledgements

I would like to express my sincere gratitude to the following individuals and organizations who have contributed to the completion of this thesis:

I want to start by expressing my gratitude to my supervisor, Christos Doulkeridis, for his invaluable advice and assistance during my graduate studies. My research and writing have benefited greatly from his knowledge and encouragement.

I also want to thank my Ubitech coworkers and my unit for their understanding and support while I tried to balance my work and thesis responsibilities. Their support, adaptability, and willingness to work together allowed me to balance the demands of both worlds.

For their unwavering love and support, especially during the trying times of my research and writing, I am grateful to my family and friends. My perseverance has been sustained by their support and understanding throughout this journey.

Thank you all for your contributions and support.

Contents

List of Figures

List of Notations & Abbreviations

AIS	**A**utomatic **I**dentification **S**ystem
ETL	**E**xtract, **T**ransform, **L**oad
NCA	**N**orwegian **C**oastal **A**dministration
DTW	**D**ynamic **T**ime **W**arping
LSED	**L**ock-**S**tep **E**uclidean **D**istance
LCSS	**L**ongest **C**ommon **S**ub**s**equence
MBR	**M**inimum **B**ounding **R**ectangle
RDD	**R**esilient **D**istributed **D**atasets
DStreams	**D**iscretized **Streams**

A, B	Trajectories **A** and **B**
n, m	Total Number of points for trajectories **A** and **B** respectively
a, b	Points from trajectories **A** and **B** respectively
t	time

To my parents, who have instilled in me the value of education and hard work. Thank you for providing me with opportunities and for always believing in me. This thesis is a testament to your love and dedication.

Chapter 1

Introduction

The shipping industry is an essential part of the economy of the entire world since it is in charge of the transportation of commodities and other items across the oceans of the world, while the economic development of different countries is significantly dependent on the movement of a containerized cargo [15]. According to a recent study, conducted by Lloyd's Marine Intelligence Unit, maritime transportation accounts for $\approx 75\%$ of the total volume and $\approx 60\%$ of the total value of global trade [24]. Because of the growing size and complexity of vessels, it is becoming increasingly vital to optimize vessel routing in order to guarantee that operations will be carried out in an effective and risk-free manner. The utilization of Automatic Identification Systems (AIS) and data in real time has brought about a revolutionary change in the manner in which vessel routing is carried out[2].

The purpose of this thesis is to examine vessel routing by making use of AIS and real time data, with a particular emphasis on trajectories and the use of waypoints to detect similarities between the itineraries taken by different vessels. Waypoints are predetermined positions that a vessel is required to travel through in order to reach its destination. They are also frequently utilized to guarantee safe navigation and to avoid potentially dangerous regions.

In this work, we will investigate how vessel trajectories may be examined by making use of real time data in order to determine the extent to which different vessel paths are similar, regarding the similarity metrics. The results of this research will assist in determining the most effective routing techniques and provide valuable insight for decision making in the shipping business. The utilization of AIS data will also be investigated as a potential method for the collection of real time information on the locations, speeds, and courses of individual vessels.

The case study for this thesis will require examining vessel trajectories in a certain geographical area, and specifically the minimum bounding rectangle (MBR) which is the smallest rectangular area that can fully enclose a set of geographic features, such as points, lines or polygons. The

MBR is defined by the minimum and maximum coordinates in the x and y dimensions of the geographic dataset. In this regard, this study performs a comparison of the pathways taken by different vessels, and identifying common waypoints utilized in their routes. The followed methodology will comprise the following steps:

1. Define the MBR based on a waypoints dataset and proceed with relevant space partitioning accordingly
2. Grouping together the real time data taken from AIS, in order to create trajectories of vessels
3. Calculation of Trajectory Similarity of each vessel route with the given waypoints data
4. Compare performance
5. Create evaluation for performance and characteristics for each trajectory similarity measure

All of these steps will be realisen in order for this work to be completed. Moreover, the findings of this research will assist to optimize vessel traffic in the region by providing insights into the success of various routing techniques and providing information about those tactics.

1.1 Motivation

Optimizing vessel routing has become more crucial as vessels' sizes and complexity increase in order to ensure effective and secure operations. The collection of vital data like vessel positions, speed, and course has revolutionized vessel routing thanks to the use of AIS and real time data.

With a particular emphasis on trajectories and the use of waypoints, the goal of this thesis is to investigate the potential of vessel routing using AIS and real time data. The optimization of vessel traffic in particular geographic areas can help to increase efficiency and safety. This is done by analyzing vessel trajectories and identifying common waypoints.

Moreover, this work aims to contribute to the growing body of knowledge on vessel routing by providing insight on how real time data and AIS are used in the shipping industry to guide decision making. real time vessel trajectory analysis can assist in determining the best routing options and guiding vessel traffic management in particular areas.

Overall, the need to improve vessel routing in the maritime sector and the potential of AIS and real time data to support decision making and increase effectiveness and safety in vessel traffic management serve as the driving forces behind this thesis.

1.2 State Of The Art

In the field of trajectory similarity for vessels, there are several key aspects, including data sources, similarity measures, clustering algorithms, and anomaly detection methods. This section will describe the current knowledge about the subject through the analysis of similar or related published work.

The primary source of data for analyzing vessel trajectories is the Automatic Identification System (AIS). AIS data provides a wealth of information, including vessel positions, speeds, and headings, which can be utilized to analyze vessel movement patterns, predict future routes, and detect abnormal behavior. The preprocessing of this data involves data cleaning, filtering, interpolation, and handling uncertainty and noise[10].

When comparing vessel trajectories, various similarity measures have been developed and applied, such as Dynamic Time Warping (DTW), Longest Common Subsequence (LCSS), Edit Distance on Real sequences (EDR), Frechet Distance, and Hausdorff Distance. These measures consider different aspects of trajectory similarity, such as the order of points, the shape of the trajectory, and the ability to handle varying speeds and sampling rates. Researchers have also explored adapting these similarity measures for specific maritime applications by incorporating additional information, such as vessel types or contextual attributes[34].

Clustering algorithms, such as DBSCAN, k-means, and hierarchical clustering, have been employed to group similar trajectories together based on their spatial and temporal characteristics. These techniques enable the identification of common patterns and anomalies in vessel movements, which can be used to inform maritime safety, security, and efficiency efforts. Recent advancements in machine learning and deep learning have also been applied to trajectory clustering, including autoencoders for learning trajectory representations and graph based approaches for analyzing the relationships between trajectories[33].

Anomaly detection and pattern recognition are critical components of vessel trajectory analysis. A variety of supervised and unsupervised methods have been proposed for detecting abnormal behavior in maritime traffic, taking into account both the spatial and temporal aspects of vessel movement. By integrating contextual information, such as weather, ports, and maritime regulations, these methods can provide more accurate and comprehensive detection of anomalies[31].

In summary, the state of the art in vessel trajectory similarity includes a wide range of techniques and methodologies, with ongoing research intended to enhance the accuracy, efficiency, and applicability of these methods to real world maritime scenarios. Future work in the field can advance our understanding of vessel movement patterns and contribute to the development of more effective and intelligent maritime systems by building on this foundation.

1.3 Problem Definition

The objective of this thesis is to explore the use of trajectory similarity analysis for vessel routing in real time by making use of past data. The study of vessel trajectories will be used to determine which routing methods are the most successful in order to increase efficiency and safety in the shipping sector. This will be accomplished by identifying the most effective routing strategies.

The necessity to optimize vessel routing in the face of increasing vessel size and complexity has given rise to the challenge that we are currently experiencing. The adoption of AIS and real time data has completely transformed the process of vessel routing by making it possible to gather crucial data such as the positions, speed, and courses of individual vessels. Despite this, it is necessary to do a thorough analysis of this data in order to determine which trajectory similarity schemes are the most successful.

The method we have been proposing in this work, is to do a trajectory similarity analysis in order to compare the itineraries taken by different vessels and locate the similarity metric the routes share. This study can assist to optimize vessel traffic in certain geographic regions by assessing real time data in addition to historical data. As a result, both efficiency and safety will be improved.

Chapter 2

Background

2.1 Spatial data

Geospatial information, sometimes known as spatial data or geographic data, involves data that represents objects, phenomena, or features with specific Earth surface locations. In fields like environmental management, urban planning, transportation, agriculture, and disaster management, this data is crucial. Geospatial data can be expressed in different forms, such as vector, raster, and attribute data[10].

1. Vector-based data represents spatial data by using points, lines, and polygons. These geometric entities are defined by a set of coordinates, typically latitude and longitude. Points represent discrete locations such as cities or buildings. Lines represent linear features such as roads or rivers, whereas polygons represent areas such as country boundaries or land parcels. Geographic Information Systems (GIS) frequently use vector-based data for analysis, modeling, and visualization.

2. Grid-based data, also known as raster data, represents spatial data as a matrix or grid of cells, with each cell containing a unique value corresponding to a specific Earth surface location. A cell's value can represent various attributes such as elevation, temperature, or land cover type. For satellite imagery, digital elevation models, and remote sensing applications, grid-based data is commonly used.

3. Non-spatial data, also known as attribute data, is information associated with spatial features that provides context or additional information about the features. This information can be stored in tables and linked to the corresponding spatial data through the use of unique identifiers. A city's attribute data, for example, could include population, area, and GDP.

GIS software (e.g., ArcGIS, QGIS), spatial databases (e.g., PostGIS, Oracle Spatial), and web mapping platforms can all store and manage spatial data (e.g., Google Maps, OpenStreetMap)[4]. These applications enable users to create, edit, analyze, and visualize spatial data in a variety of formats[9].

Some key concepts related to spatial data include:

1. Coordinate systems: A coordinate system defines how spatial data is represented in terms of location and measurement. There are various coordinate systems, such as the geographic coordinate system (latitude and longitude) and projected coordinate systems (e.g., Universal Transverse Mercator).
2. Spatial reference frameworks: A spatial reference framework (SRF) is a combination of a coordinate system and a datum (a reference surface for Earth's shape). SRF ensures that spatial data from different sources can be accurately integrated and analyzed.
3. Scale and resolution: Scale refers to the ratio between the size of a feature on the Earth's surface and its representation on a map or image. Resolution, on the other hand, is the level of detail at which spatial data can be represented. Higher resolution data has finer detail but may require more storage and processing power.
4. Spatial examination: Spatial examination is the process of studying and modeling patterns, relationships, and trends within spatial data. This can involve various techniques such as buffering, overlay, interpolation, and network analysis, depending on the problem being addressed.
5. Spatial data quality: The quality of spatial data is essential for accurate analysis and decision making. Quality can be affected by factors such as positional accuracy, attribute accuracy, and completeness of the data.

In conclusion, geospatial data is an important part of many applications because it helps us understand how geographical features and events are related. Using GIS tools to manage and analyze it lets people in many fields make better decisions and solve problems more effectively[9].

2.2 Trajectory

Trajectories, also known as routes, are a type of spatial data that represents the movement of objects or people through space and time. Vehicles, animals, people, and even natural phenomena such as storms and wildfires can be considered[23]. Trajectories and routes are useful

for analyzing and comprehending a wide range of spatiotemporal patterns, behaviors, and relationships in fields such as transportation planning, wildlife ecology, human mobility, and emergency response[38].

A combination of spatial and temporal data can be used to represent and analyze trajectory and route data:

- The spatial component of a trajectory or route is made up of a series of spatial points that define the path of the object. Each point on the trajectory corresponds to a specific location on the Earth's surface (e.g., latitude and longitude). These points can be connected in vector data to form a polyline or a line feature, which represents the object's movement from one location to another.
- The timestamps associated with each spatial point are referred to as the temporal component of a trajectory or route. Timestamps indicate when an object was at a specific location, allowing for analysis of movement speed, duration, and temporal patterns. Temporal data can be stored as an attribute of spatial points or in a separate table linked to spatial data.

Working with trajectories and routes as spatial data requires a variety of techniques and methods:

1. Data collection: Trajectory data can be collected using a variety of methods, including GPS tracking devices, mobile phone data, vehicle sensors, and social media check-ins. These methods produce raw spatial and temporal data that can be cleaned, processed, and formatted for further analysis.
2. Data preprocessing: To address issues such as noise, gaps, or errors in the data, trajectory data frequently requires preprocessing. To improve the quality and usability of the data, techniques such as filtering, interpolation, and smoothing can be used. Trajectories can be segmented and clustered based on specific criteria such as distance, time, or speed. This aids in identifying and analyzing various phases of movement or events within the trajectory. Clustering techniques can be used to group similar trajectories or routes, allowing common patterns, trends, or anomalies to be identified[14].
3. Spatiotemporal analysis: Examining the relationships between the spatial and temporal components of data is common when analyzing trajectories and routes. Calculating distances, speeds, and durations, as well as identifying points of interest along the trajectory, are all examples of this. Various spatiotemporal analyses can be performed using time series analysis, spatial statistics, and geographic information systems (GIS).

4. Visualization: Trajectories and routes can aid in the discovery of patterns, trends, and relationships in data. Static maps, animated maps, space-time cubes, and interactive web maps are all common visualization techniques. GIS software, programming languages (e.g., R, Python), or web mapping platforms can be used to create these visualizations (e.g., Leaflet, Mapbox).

5. Applications: Trajectories and routes have a wide range of applications in a variety of fields. They can be used in transportation planning to analyze traffic patterns, optimize public transportation routes, and identify accident hotspots. They can aid in the study of animal migration, habitat use, and the impact of human activities on animal movement in wildlife ecology. They can help in emergency response by predicting the spread of wildfires, tracking storms, and planning evacuation routes.

In conclusion, trajectories and routes as spatial data provide important insights into the movement of objects or individuals through space and time. Researchers and practitioners can better understand and address various spatiotemporal patterns, behaviors, and relationships in a variety of fields by collecting, preprocessing, analyzing, and visualizing this data[38].

2.3 Trajectory similarity

Trajectory similarity for vessels can be determined using various algorithms and techniques. In maritime applications, such as monitoring ship movements, trajectory similarity is crucial for detecting abnormal behavior, estimating travel times, and evaluating route efficiency. Here are a few popular techniques to assess trajectory similarity for vessels[38]:

- Dynamic Time Warping (DTW)
- Longest Common Subsequence (LCSS)
- Edit Distance on Real sequences (EDR)
- Fréchet Distance
- Trajectory Clustering[33]

When evaluating vessel trajectory similarity, it is critical to consider factors such as data accuracy and frequency, as well as the specific application requirements. Depending on the context and the desired level of similarity between trajectories, different methods may be more appropriate[31].

2.3.1 Similarity measures

Several different distance metrics, such as Lock-Step Euclidean distance, Dynamic Time Warping, and Longest Common Subsequence, can be utilized to determine how similar two trajectories are to one another. These metrics quantify the degree to which two trajectories are comparable by comparing the distance or dissimilarity between the corresponding locations in space and time of the two sets of trajectories[34].

The application and the features of the trajectories being compared are two factors that influence the decision on which similarity metric to use. For instance, Euclidean distance may be suitable for trajectories that consist of straight lines, but Dynamic Time Warping may be more ideal for trajectories that differ in terms of their lengths or speeds, while both methods are used to calculate distance. Some of the most important methods for measuring the similarity of trajectories are presented below:

Lock-step Euclidean distance (LSED) counts every pair of related points along both trajectories, calculating the total distance between them. Lock-step Euclidean distance in the continuous case necessitates that the lengths of two trajectories match. In the discrete case, lock-step Euclidean distance demands that two trajectories have the same number of points or that we can extrapolate over the length of the trajectories. Formally, we may interpret the trajectory data as points in the Euclidean space $\mathbb{R}^{2n}$ and calculate their Euclidean distance if $n = m$ [34].

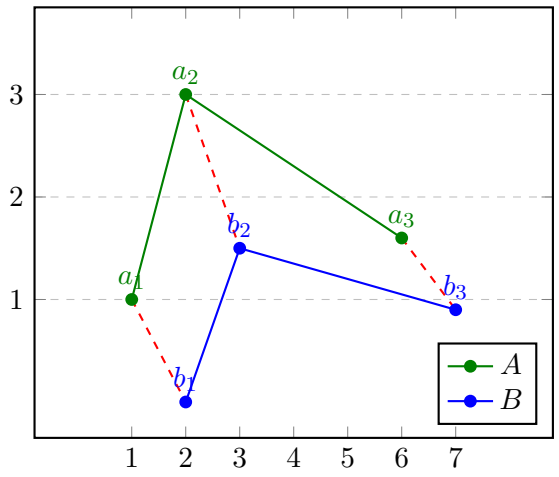
The lock-step Euclidean distance of A and B is defined as:

$$Eu(A,B) = \sqrt{\sum_{i=1}^{n} dist_2^2(a_i, b_i)}$$

The average distance between measurements is a common variant:

$$Eu'(A,B) = \frac{1}{n}\sqrt{\sum_{i=1}^{n} dist_2^2(a_i, b_i)}$$

The maximum distance may also be used in instead of the average distance.



$SQRT(dist(a_1, b_1)^2 + dist(a_2, b_2)^2 + dist(a_3, b_3)^2)$, n = m = 3

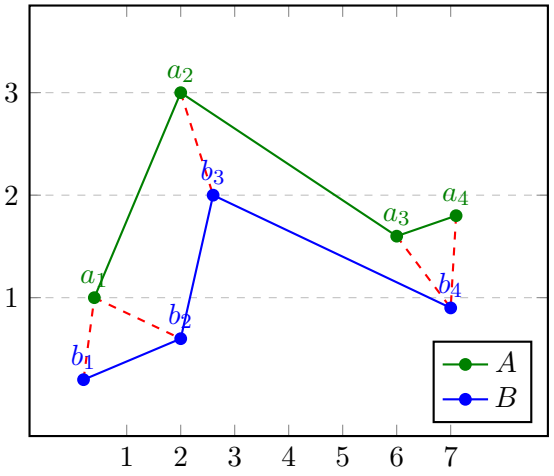
FIGURE 2.1. Demonstration of LSED.

When the two trajectories are in sync with one another in terms of time, the concept above becomes most clear. That is, if $t_i^a = t_i^b$ for all $1 \leq i \leq n = m$, then the distance between the trajectories at respective times is then measured by LSED.

Dynamic time warping (DTW) is a classic example of dynamic programming. It was first implemented for the purpose of speech detection. Time series data have been effectively analyzed using DTW. Successively, it became one of the most widespread approaches for evaluating trajectory similarity. The DTW is defined as[12]:

$$DTW(A,B) = \begin{cases} 0 & , if\ m = n = 0 \\ \infty & , if\ m = 0\ or\ n = 0 \\ dist_2^2(a_1, b_1) + min(DTW(A_{|2,n|}, B_{|2,m|}), & , otherwise \\ DTW(A, B_{|2,m|}), DTW(A_{|2,n|}, B)) & \end{cases}$$

DTW determines the best match between the elements of two sequences by building a matrix with all possible combinations of two elements in the sequences as entries and the distance between them. The total distance between two sequences is the sum of the elements in the matrix representing the shortest contiguous path. Because it discovers at least one match for all items and then aggregates the distance values, DTW is susceptible to noise. For example, if a trajectory A contains a stop that is very distant from all of B's stops, even though all of A and B's other stops are close, the distance will be dominated by the distant stop[8].



$dist(a_1, b_1)^2 + dist(a_1, b_2)^2 +$
$dist(a_2, b_3)^2 + dist(a_3, b_4)^2 +$
$dist(a_4, b_4)^2$, n = m = 4

FIGURE 2.2. Demonstration of DTW.

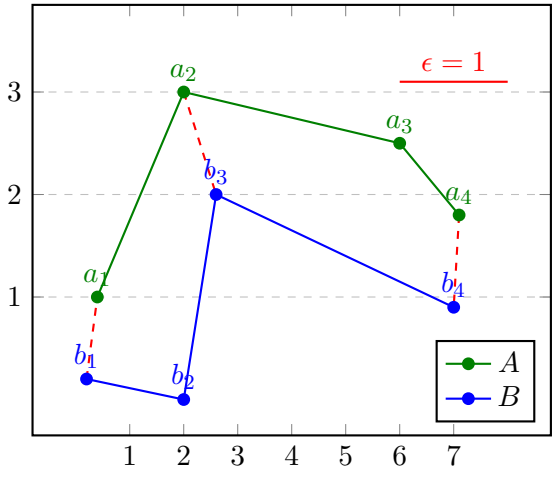
What distinguishes a similarity measure such as DTW is how the cost of a path is determined, as the cost of a path indicates how well two trajectories are aligned along that path. The cost of a path is equal to the sum of the squared distances between all aligned point pairs. Similar to other distance metrics based on squared distance, this metric can provide tolerance for outliers[21].

Longest Common Subsequence (LCSS) measure attempts to quantify the similarity between two trajectories by calculating the length of the longest point sequence they have in

common[13]. The length of the longest common subsequence between A and B is defined as:

$$LCSS(A,B) = \begin{cases} 0 & , if\ A\ or\ B\ is\ empty \\ 1 + LCSS(A_{[1,n-1]}, B_{[1,m-1]}) & , if\ dist_{\infty}(a_n, b_m) < \epsilon \\ & \quad and\ |n-m| \leq \delta \\ max(LCSS(A_{[1,n-1]}, B), LCSS(A, B_{[1,m-1]})) & , otherwise \end{cases}$$

The definition uses two parameters, δ and ϵ. ϵ is a matching threshold distance between two matching points in order to align the trajectories in time (i.e. two points closer than ϵ are considered to match). Notably, δ is not unique to LCSS and could be added to any of the other measures[35].



No. of matched pts. = 3 , n = m = 4

FIGURE 2.3. Demonstration of LCSS.

2.4 Real Time Processing

Real time data processing is the execution of data in a short amount of time, yielding an immediate outcome. It is a rapid data processing solution that integrates data capture, data processing, and data exports. Big Data real time processing aims to create a system that processes data at a near instant rate, which necessitates a constant flow of data intake and output in order to maintain real time insights.

Big data processing in real time is made possible by a variety of technologies, methods, and infrastructure elements, including:

1. **Stream processing** which is a technique for processing data as it arrives in real time, without the need for storage. Stream processing platforms such as Apache Flink, Apache Storm, Apache Samza and Apache Spark Streaming allow for real time data analysis and transformation[25]. They frequently support windowing and event time processing, which allow users to process data at specific time intervals or based on event occurrences. They are also designed to handle large scale data processing tasks with low latency[29].

2. **Data Ingestion**, where the data are generated by a variety of sources, including social media, IoT devices, log files, and sensors. Real time data ingestion entails quickly and efficiently capturing, filtering, and preprocessing this data. For real time data ingestion, popular tools include Apache Kafka, Amazon Kinesis, and Google Cloud Pub/Sub[22].

3. **Data storage** for real time processing necessitates storage systems capable of handling high velocity data while also providing low latency access to stored data. Traditional databases are unsuitable for this purpose. For real time big data storage, NoSQL databases like Apache Cassandra, Amazon DynamoDB, and Google Cloud Datastore, or distributed file systems like Apache Hadoop HDFS, are frequently used[20].

4. Using **Machine Learning algorithms** and **Analytics** techniques to derive insights and make predictions is common in real time big data processing. TensorFlow, PyTorch, and scikit-learn machine learning frameworks, as well as specialized tools like Apache Mahout and H2O, can be integrated with real time data processing systems to enable real time analytics.

To efficiently process large volumes of data with low latency, stream processing is frequently combined with distributed processing systems. These systems process data record by record as it arrives, enabling real time or near real time analysis. Distributed processing systems, on the other hand, divide processing tasks among multiple nodes or machines in order to increase throughput and fault tolerance[27].

When a streaming system receives input data streams, it may use a technique called micro-batching to divide the data into small batches. Micro-batching provides a balance between the low latency of stream processing and the efficiency of batch processing. Each batch contains a small set of records, which are processed together in parallel by the distributed processing system as shown in the Fig3.3. This enables the system to achieve high performance while maintaining low latency, allowing for near real time processing of the data.

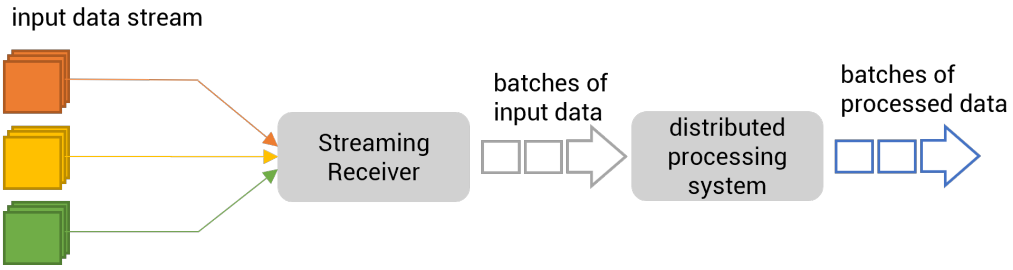


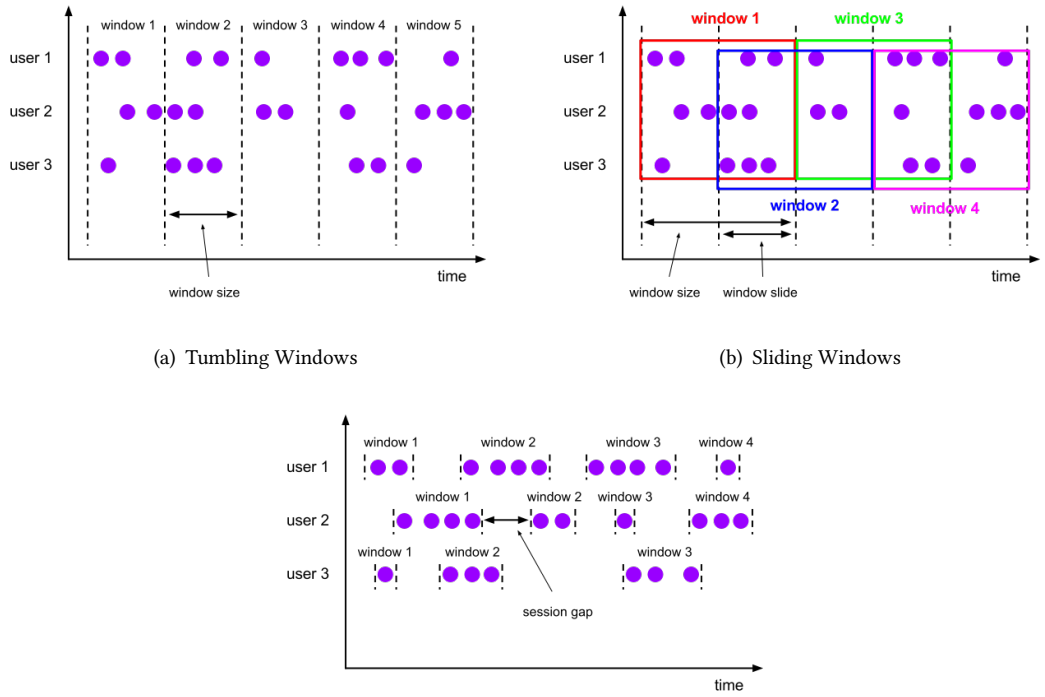
FIGURE 2.4. Data Stream Processing.

For instance, Apache Flink, a popular stream processing framework, supports both record by record processing and micro-batching, allowing users to select the appropriate strategy based on their use case and performance requirements. Similarly, Apache Spark Streaming uses micro-batching to handle data streams in near real time by breaking them into small batches and processing them using Spark's distributed processing capabilities.

However, efficiently handling unbounded data streams requires additional techniques that can break down the continuous flow of data into smaller, manageable units. Windowing is one such technique, enabling users to perform various operations on finite subsets of data, making real time analytics feasible[5].

Windowing divides unbounded data streams into smaller chunks called windows based on specific criteria in stream processing systems. Among the most common types of windows used in stream processing systems are:

- **Tumbling windows:** Divide the data stream into non overlapping, fixed size time intervals, which are ideal for computing aggregates or summaries over consistent time ranges.
- **Sliding windows:** Divide the data stream into fixed size time intervals with overlap between consecutive windows, which is useful for tracking moving averages or tracking trends over time with fine grained updates.
- **Session windows:** These are ideal for analyzing user interactions or session based activities because they group together events that are closely related in time based on a specified gap duration.



(a) Tumbling Windows

(b) Sliding Windows

(c) Session Windows

FIGURE 2.5. Windows.

We can efficiently manage and analyze unbounded data streams by using stream processing systems and techniques that break them down into smaller, manageable chunks. This enables real time or near real time data operations such as aggregation, filtering, and transformation, as well as the extraction of valuable insights. A crucial role in implementing and scaling the aforementioned across different use cases are the **real time Processing Frameworks**.

2.4.1 Apache Flink

Apache Flink is an open source data processing solution for streaming and batch data. Flink's architecture is built on top of a distributed dataflow engine that can handle both batch and stream processing workloads. It provides APIs for processing data in real time streams as well as for processing batch data. Flink also is based on the idea that many different types of data processing applications, such as real time analytics, continuous data pipelines, batch data processing, and iterative algorithms (machine learning, graph analysis), may be represented and run as pipelined fault tolerant dataflows [19]. Flink's capability to handle stateful stream processing is one of its key features. Flink is appropriate for complex event processing, real time analytics, and other applications that call for stateful processing because it can maintain state across multiple streams. Apache Flink was created by the data Artisans team in Berlin

and became an Apache project in 2014. Savepoints and checkpoints are advanced features of Flink that allow for application state management and recovery. It also has a strong ecosystem with connectors for a variety of data sources and sinks, such as Apache Kafka, Apache Cassandra, and Amazon S3. Flink can be run on-premises, in the cloud, or on a variety of cluster management systems such as Apache Mesos and Kubernetes[11].

2.4.2 Apache Storm

Apache Storm provides a simple programming framework that allows developers to create applications that can process data streams in real time. It is based on a topology based architecture, with a topology being a directed acyclic graph (DAG) made up of spouts and bolts. Spouts are in charge of ingesting data into the system, while bolts are in charge of data processing, transformation, and storage. Storm's data is represented as tuples, which are immutable and ordered lists of elements.Nathan Marz and the team at BackType (later acquired by Twitter) initially developed Apache Storm. It was open sourced in 2011 and became a top level Apache project in 2014. Storm is intended to integrate easily with other data storage and processing systems such as Apache Hadoop, Apache HBase, and Apache Cassandra. It also supports multi language programming, allowing developers to create topologies in various programming languages[36].

2.4.3 Apache Samza

Apache Samza is a distributed stream processing framework designed for real-time processing of large scale data streams. It was created by LinkedIn and became an Apache project in 2013[30]. Apache Samza uses Apache Kafka for messaging and Apache Hadoop YARN for cluster resource management. Samza offers fault tolerant stream processing, stateful stream processing, and flexible deployment options[6].

2.4.4 Apache Spark

Spark is an other open-source distributed computing system that provides an interface for programming entire clusters. It also has implicit data parallelism and fault tolerance. It can be used for batch processing, interactive queries, machine learning. It is designed to be a fast and general-purpose engine for large-scale data processing. Although performs better than Hadoop, lacks a distributed storage system of its own[28].

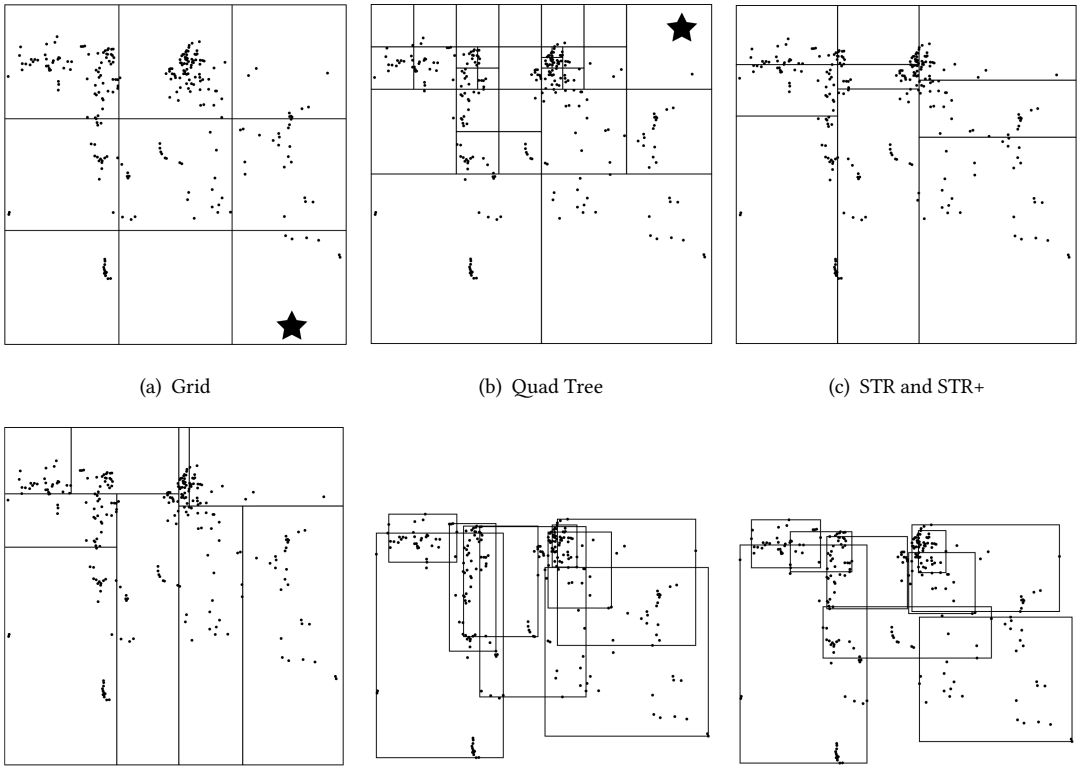
A library build on top of Spark that is called Apache Spark Streaming, allows the developer to process real time data streams using the same programming model as batch processing. With

support for well-known data sources, it offers a high-level API for ingesting and processing data in real time[25].

2.5 Partitioning

When it comes to distributed storage and parallel computing for large amounts of data, data partitioning plays an important and powerful role. Big data can be partitioned into relatively tiny and independent blocks, which is a fundamental and powerful method for enhancing the efficiency of data storage and management systems. This can be accomplished by using data partitioning. In addition to this, the concept of data partitioning known as "divide and conquer" can help increase data processing and computing. For instance, if the data partitioning was done correctly, it should only be necessary to scan a few partitions rather than the full dataset [37].

Partition techniques can be divided to space partitioning (Grid, Quad tree), data partitioning (STR, STR+, K-d tree) and space filling curve (Z-curve, Hilbert curve). These techniques can also be grouped, according to boundary object handling, into replication-based techniques and distribution-based techniques [16].



(a) Grid (b) Quad Tree (c) STR and STR+

(d) K-d Tree (e) Z-curve (f) Hilbert Curve

FIGURE 2.6. Partitioning Techniques.

2.5.1 Space Partitioning Techniques

Uniform Grid This approach divides the input MBR into a uniform grid of $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil$ grid cells and uses the replication method to handle boundary objects, thus it does not need a random sample.

Quad Tree In this technique, where k is the sample size and n the cells, all sample points are inserted into a quadtree with a node capacity of $\lfloor k/n \rfloor$. All leaf node borders are used as cell boundaries. To allocate records to cells, we employ the replication mechanism [17].

2.5.2 Data Partitioning Techniques

STR The capacity of each node is set to $\lfloor k/n \rfloor$ in this technique. The random sample is then bulk loaded into an R-tree using the STR algorithm [26]. The cell borders are the MBRs of leaf nodes. The distribution method, which distributes a record r to the cell with the most overlap, is used to handle boundary objects.

STR+ This technique is similar to the STR technique, but it handles boundary objects using the replication method.

K-d tree This technique uses the K-d tree [7] partitioning method to partition the space into n cells accordingly. It divides the input MBR into n cells by starting with it as a single cell and partitioning it n times. The replication mechanism assigns records to cells.

Z-curve This method sorts the sample points according to their position on the Z-curve [32] and divides the curve into n segments, each comprising about $\lfloor k/n \rfloor$ points. It employs the distribution approach to assign a record r to a single cell by mapping the MBR's center to one of the n splits.

Hilbert curve This approach [18] is identical to the Z-curve method, except that it employs a Hilbert space filling curve with superior spatial features.

Chapter 3

Notation & Real-time Trajectory Similarity

In this work, trajectory similarity measures are studied while being applied to a real time processing application. The goals of this research are to investigate the performance of each measure and compare the characteristics of these measures when applied to the specific application. In order to accomplish what was outlined as the use case, streaming big data along with historical data were utilized.

3.1 Datasets Description

The Norwegian Coastal Administration provided the data used for our study, which encompasses the entirety of Norway (NCA).The data consist of:

- One set of reference routes
- A collection of AIS messages regarding vessel movements in the Norwegian region

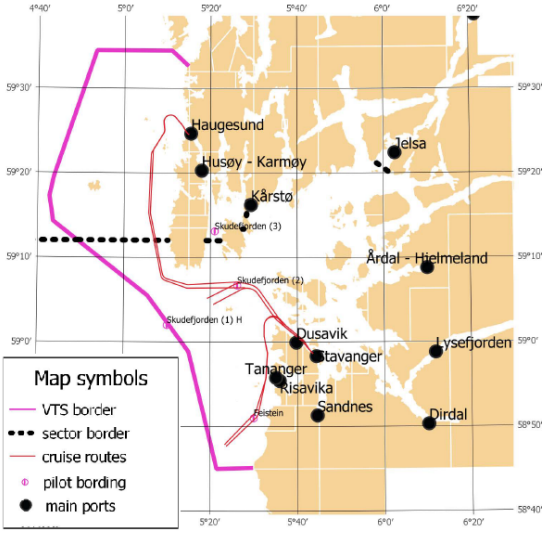
3.1.1 Routes dataset

The routes dataset is formed by the reference routes for Navigation, that are provided by the Norwegian Coastal Administration (NCA), which are recommendations from NCA to support voyage planning based on their best practices[1]. The reference routes for vessel navigation are predefined routes that assist mariners in safely navigating waterways by avoiding obstacles and taking the most efficient routes between ports or other locations. International organizations, such as the International Maritime Organization (IMO), create and maintain them in

order to provide guidance and support to mariners who may be unfamiliar with the waters they are navigating[3]. The information that these may include is:

- Tracks or routes that are recommended based on factors such as weather, currents, or the presence of navigational hazards
- Waypoints along the recommended route that serve as navigational reference points.
- Information on potential navigational hazards along the route, such as shallow areas, wrecks, or other impediments.
- Information about areas that require extra caution or attention, such as narrow channels or high-traffic areas.
- Local regulations or requirements that may affect navigation, such as speed limits or reporting requirements, are listed.
- Other pertinent data, such as the location of ports, anchorages, or other facilities.

For the purposes of this research, the ETL procedure was used in the set of reference routes. As it was already stated before in section 2.2, a trajectory (route), is made up of a series of spatial points that define the path of the object. These spatial points are called *Waypoints* and can be found in each route file that has been downloaded from the NCA. In the Fig 3.1 the trajectories are displayed, and the number of waypoints in each one can be seen.



Route Name	Waypoints
Cruise Haugesund - Skudefjorden Inbound	11
Cruise Stavanger - Skudefjorden Outbound	8
Cruise Stavanger - Feistein Inbound	11
Cruise Haugesund - Skudefjorden Outbound	11
Cruise Stavanger - Skudefjorden Inbound	8
Cruise Stavanger - Feistein Outbound	11

FIGURE 3.1. Reference Routes.

Each file that has been downloaded from the NCA contains a route in RTZ format. The RTZ format, which stands for "Routeing Guide and Zone," is a digital format used to electronically

store and transmit reference route information. It contains the same information as the printed Routeing Guide, but in a machine-readable format that navigation software and other digital tools can easily access and use[3]. The RTZ files use XML-based syntax to store waypoints, making them simple to parse with programming tools. Finally, a final dataset containing all of the waypoints was created, keeping only the necessary information.

3.1.2 AIS Data

Data from the Automatic Identification System (AIS) is a real time positioning information system that is mainly used in maritime navigation. It allows ships to share and receive information about their identity, position, course, and speed with other nearby vessels and maritime authorities, improving situational awareness, safety, and security in congested sea lanes.

AIS works by transmitting data packets between vessels and shore-based stations over VHF radio frequencies. These packets contain critical information such as the ship's Maritime Mobile Service Identity (MMSI), GPS coordinates, heading, speed over ground, and navigational status. The ship's speed and course changes cause AIS data to be updated at regular intervals ranging from a few seconds to a few minutes[2].

Historical data from AIS messages in the Norwegian region for 499 unique MMSIs (vessels) were obtained from January to June of 2021. The AIS messages arrived by monthly files. For the purposes of this study, it was assumed that these data would be sent as a stream into the real time processing system in order for the examinations to take place. More information of how the system receives the AIS messeges can be found in Appendix A.

3.2 Procedure Outline

Based on the problem's description (1.3), the primary objective is to examine various trajectory similarity measures in real time vessel routing compared to the predefined historical trajectories described in section 3.1.1. Regarding the methodology, a number of steps must be carried out.

The first step was to retrieve both datasets, as described in section 3.1, that had been used in the system as reference routes and AIS messages in order to proceed with data partitioning. It is worth noting that the distance to each waypoint (of the route) is computed for each AIS message within its allocated partition. Figure 3.2 shows that the first two AIS messages received are close to route B, and as more messages are received, the distance for this vessel becomes closer to route A.

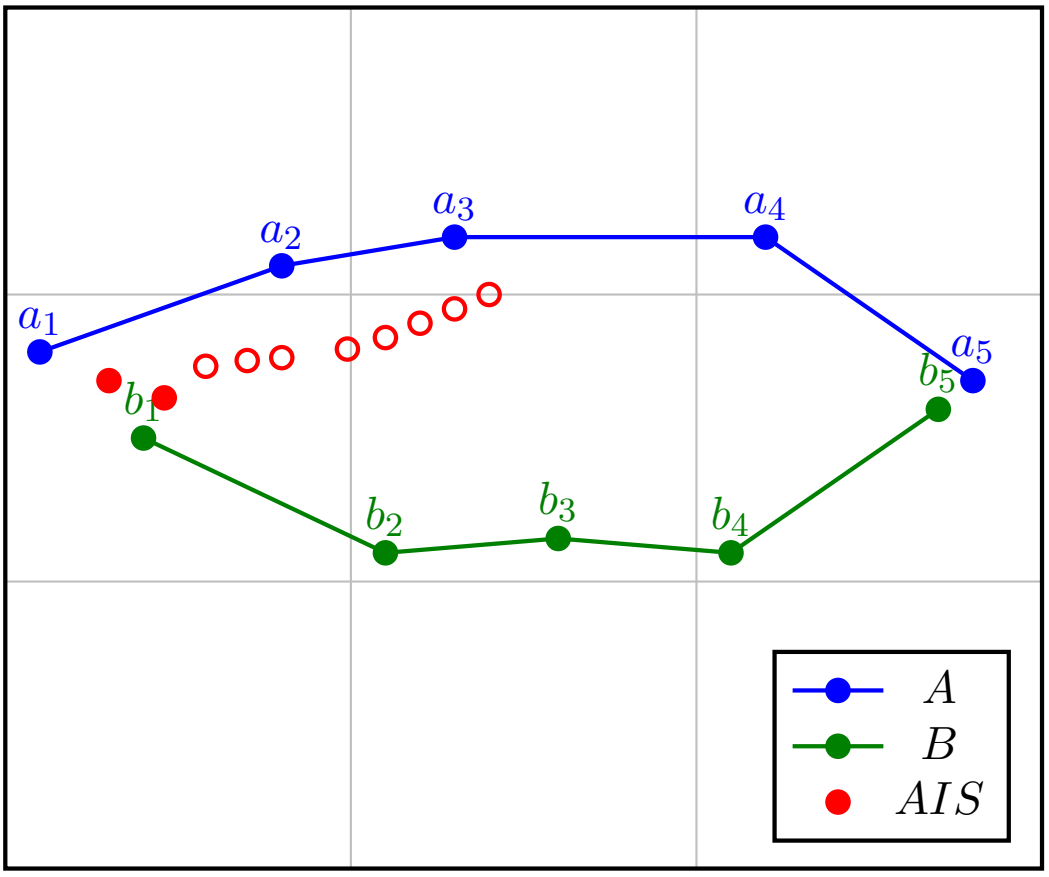


FIGURE 3.2. Demonstration of distance computation for each AIS message and waypoint.

After computing the distance, the system assigns a score (probability) to each route based on the similarity measure it employs each time. By comparing the scores and evaluating the output, a vessel's trajectory may match a route or more based on its similarity. Finally, the goal is to examine the processing time and results for the three similarity measures (LSED, DTW, LCSS) we are evaluating for this type of problem.

3.3 Batch Processing API

In order to perform efficient processing of large datasets across a cluster of machines, it has been used in conjunction with Apache Spark, and in particular, Spark streaming, in order to meet the requirements of the thesis. The Resilient Distributed Datasets (RDDs) and the Discretized Streams (DStreams) are the two core abstractions that have been used [28].

RDDs are immutable collections of objects that are distributed across a system and can be processed in parallel. RDDs are fault-tolerant, which means that they are able to recover from the failure of individual nodes without losing any data. Users have the option of either loading data from external storage systems such as HDFS, HBase, or Amazon S3 in order to create

RDDs, or transforming already existing RDDs. RDDs can be cached in memory so that they can be accessed more quickly and reused more frequently. RDDs allow users to perform a wide variety of operations, including transformations like map, filter, and reduce, as well as actions like count, collect, and save.

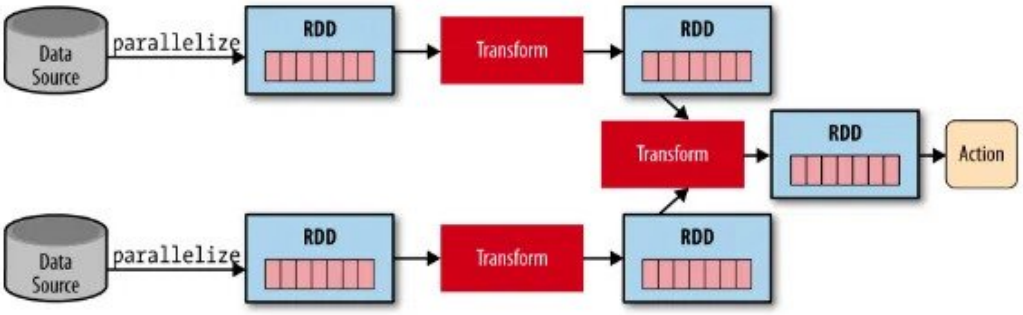


FIGURE 3.3. Resilient Distributed Datasets (RDDs) pipeline.

DStreams are a high-level abstraction built on top of Spark's RDDs that make it possible to process live data streams. These abstractions were developed by Apache Spark. DStreams are a representation of a continuous stream of data that is segmented into discretely small groups of data. Users are granted the ability to apply RDD operations such as map, reduce, and filter to the stream because each batch is handled as if it were an RDD. Spark Streaming is able to be used to perform processing on DStreams after the streams have been created by connecting to various sources such as Kafka, Flume, and HDFS. Spark Streaming is an extension of the core Spark API that enables users to process live data streams using the same API that is used for batch processing[25].

In conclusion, RDDs and DStreams are two essential abstractions for this work that enable the process of the waypoint datasets and real time AIS data in a timely and effective manner, respectively. They offer a method to represent immutable, distributed collections of objects that are fault-tolerant and can be processed in parallel, in order provide efficient results.

3.4 Partitioning Implementation

Given that the development makes use of distributed processing, it is necessary to devise a method that is more efficient for computing the vast amount of data. In order to improve the effectiveness of the system, the strategy of Uniform Grid Partitioning was put into action.

In this work, a grid partitioning is required for the historical data of the waypoints before we can move forward with the calculations of the similarities.

As it is already stated in the section 2.5.1, dividing the input MBR into a grid of $n \times n$ cells and then replicating the waypoints from each cell into the cells that are adjacent to it in order to handle boundary objects. For instance on Fig 3.4 every point has to be assigned to the neighboring cells. For instance the points in cell 11 getting replicated on the 5, 6, 7, 10, 12, 15, 16 and 17. Below is the algorithm of this method.

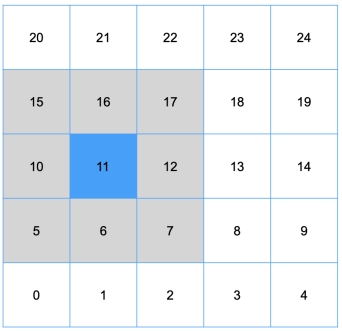


FIGURE 3.4. Grid Partitioning.

Algorithm 1 Uniform Grid Partitioning

Require: number of partitions n

$P \leftarrow \emptyset$
$l \leftarrow \sqrt{n}$
$h \leftarrow \frac{\text{height of } \mathbf{MBR}}{l}$
$w \leftarrow \frac{\text{width of } \mathbf{MBR}}{l}$
for $i \leftarrow 0\ l-1$ **do**
    for $j \leftarrow 0\ l-1$ **do**
        $x_{min} \leftarrow i \times w$
        $y_{min} \leftarrow j \times h$
        $x_{max} \leftarrow (i+1) \times w$
        $y_{max} \leftarrow (j+1) \times h$
        $P \leftarrow P \cup \{\text{partition}(x_{min}, y_{min}, x_{max}, y_{max})\}$
    end for
end for
return P

3.5 Trajectory Similarity Implementation

Although the development's earlier sections were significant, this one is the most noticeable. Here is how the similarity measures are implemented throughout the application. Before anything else, the "trajectories" that refer to the collections of points for real time and historical data (after the grid has been partitioned) need to be defined. In order to continue with the efficient calculations of distance that it has already defined, the streaming data of AIS need to be allocated on cells as well.

The methodology begins with a cross-join of the two datasets in order to proceed with formation of the routes and the trajectories of the waypoints and the AIS messages accordingly and filtering them for the relevant partitions. On the newly created Dstream a a *custom partitioner*

applies, with a given input of the number of the partitions. The recently created Dstream contains the points that have been filtered and assigned to each partition or cell in the appropriate manner. The points have to be transformed into trajectories on a new Dstream, and in order to accomplish that transformation, the points have to be grouped by their respective IDs. One more cross-join operation is carried out for the created trajectories of the distributed waypoints and the vessel live routes before moving on to the next step of the distance calculation. It is noticeable that for every transformation, the *custom partitioner* is getting used, so there will not be any data shuffling among the partitions. Below is the described methodology.

Algorithm 2 Data Merge and Trajectory Formation

Ensure: w waypoints data RDD, d AIS data Dstream

1: **for each** $rdd \in d$ **do**
2: $rdd \times d$
3: $j \leftarrow d_i, w_i$
4: **end for**
5: **for each** $t_{id} \in j$ **do** ▷ t_{id} trajectory id
6: $d_t \leftarrow dG(t_{id})$
7: $w_t \leftarrow wG(t_{id})$
8: **end for**
9: **for** $i \leftarrow 0$ **do**
10: $w_t \times d_t$
11: ...
12: $trajectorySimilarity()$
13: **end for**

3.5.1 Lock-Step Euclidean Distance

As we have already stated, calculating the distance between every related pair of points along each trajectory is required to solve this problem. This must be done for each pair of trajectories. We then calculate the length of the each trajectory, which will determine the number of points to compare. For each point i from 1 to $length(A)$ we calculate the Euclidean distance between the corresponding points in trajectory A and B.

Algorithm 3 Lock-Step Euclidean Distance

Require: Two trajectories A and B of lengths m and n, respectively, where $m \leq n$

Ensure: The Euclidean distance between A and B

$d \leftarrow 0$
for $i \leftarrow 1$ to m **do**
    $d \leftarrow d + (a_i - b_i)^2$
end for
for $i \leftarrow m + 1$ to n **do**
    $d \leftarrow d + b_i^2$
end for
return $\sqrt{d}$

3.5.2 Dynamic Time Warping

In addition, we have previously mentioned DTW finds an optimal alignment between two time series by warping them in the time dimension. First, we define the two sequences of points as trajectories, where each point is a tuple of (x, y) coordinates. Let's call the two trajectories A and B. We create a 2D matrix $DTW[][]$ with dimensions $(length(A) + 1) \times (length(B) + 1)$. We initialize the first row and first column to infinity and the $DTW[0][0]$ element to zero. We iterate through the matrix starting from element $DTW[1][1]$ and compute the cumulative distance between each pair of points in trajectories A and B. At each point in the matrix, we take the minimum of the values of the three neighboring cells of the matrix(above, left, and diagonal) and add the distance between the corresponding points in A and B. The DTW distance between the two trajectories is the value in the last cell of the matrix $DTW[n][m]$, where n, m, the total number of points for trajectories A and B .

Algorithm 4 Dynamic Time Warping

Require: Two trajectories A and B of lengths n and m, respectively
Ensure: The DTW distance between A and B
Initialize $D_{0,0} \leftarrow 0$, $D_{i,0} \leftarrow \infty$, $D_{0,j} \leftarrow \infty$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$
for $i \leftarrow 1$ to m **do**
    for $j \leftarrow 1$ to n **do**
        $cost \leftarrow |A_i - B_j|$
        $D_{i,j} \leftarrow cost + \min\{D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}\}$
    end for
end for
return $D_{n,m}$

3.5.3 Longest Common Subsequence

As previously noted, the goal, as the name suggests, is to find the longest subsequence that is shared by both trajectories. In order to implement the two trajectories, we first define them as lists of points, where each point is a tuple of (x, y) coordinates, similar to how we previously implemented the other measures. We refer to the two trajectories as A and B. We create a 2D matrix $LCSS[][]$ with dimensions $(length(A) + 1) \times (length(B) + 1)$. We initialize the first row and first column to zero. We compare every pair of points in trajectories A and B as we iteratively go through the matrix beginning with element $LCSS[1][1]$. We take the maximum of the three neighboring cells' values at each point in the matrix (above, left, and diagonal) and add 1 if the corresponding points in A and B are the same, or 0 if they are not. The length of the LCSS between the two trajectories is the value in the last cell of the matrix $LCSS[length(A)][length(B)]$

Algorithm 5 Longest Common Subsequence

Require: Two trajectories A and B of lengths n and m, respectively
Ensure: The length of the LCSS between A and B

Initialize a matrix C of size $(n + 1) \times (m + 1)$ to all zeros
for $i \leftarrow 1$ to n **do**
    for $j \leftarrow 1$ to m **do**
        if $A_i = B_j$ **then**
            $C_{i,j} \leftarrow C_{i-1,j-1} + 1$
        else
            $C_{i,j} \leftarrow \max\{C_{i-1,j}, C_{i,j-1}\}$
        end if
    end for
end for
return $C_{n,m}$

Chapter 4

Experimentation & Validation

In this chapter, we evaluate a comparative analysis of the LSED, DTW, and LCSS trajectory similarity measures. And provide a performance evaluation as well as an examination of their characteristics in relation to the issue of real time trajectory similarity in maritime.

4.1 Experimentation Setup

The datasets that were utilized for the experiments were obtained from:

- Waypoints
- AIS Data

To begin, in order to facilitate the execution of our tests, a cluster consisting of three virtual machines (VMs) has been created. The cluster is made up of two workers and a master virtual machine (VM) that also performs the duties of a worker.

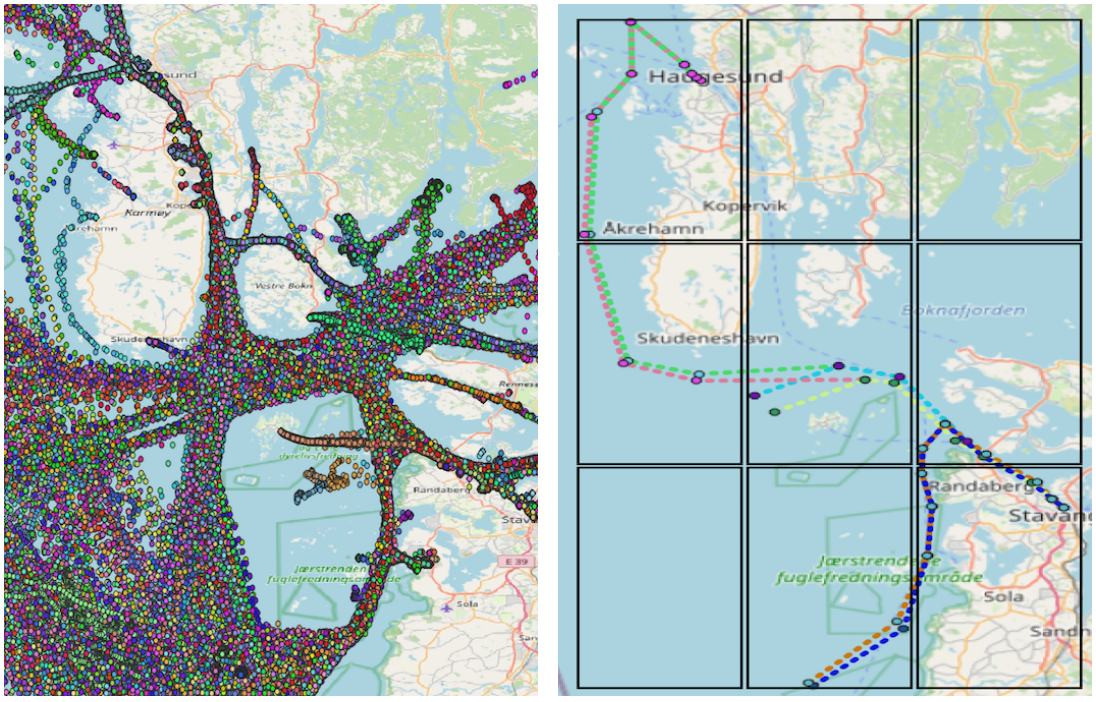
VM	CPU	RAM
Master	4	4 GB
Worker	6	6 GB
Worker	6	6 GB

In the cluster, the applications have been run using the *spark-submit*, bellow is an example of the command:

```
$ spark-submit --deploy-mode cluster “
--class com.similaritymeasure.LockStepEuclideanDistance “
--executor-memory 8G “
--total-executor-cores 16 “
--master spark://83.212.80.23:6066 “
--files waypoints.csv “
```

```
 --conf "spark.driver.extraJavaOptions=-Dlog4j.configuration=
↪ log4j-spark.properties" "
 --conf "spark.executor.extraJavaOptions=-Dlog4j.configuration
↪ =log4j-spark.properties" "
 main.jar waypoints.csv
```

Because we have already established that grid partitioning is necessary for the experiment, the number of cells/partitions that we have devided the space is nine. Bellow is a map representation the 2 datasets and the grid partitioning (3×3) of space:



(a) AIS data (b) Waypoints

FIGURE 4.1. Dataset representations.

4.2 Results

In this section, using grid partitioning, we examine the performance of the trajectory similarity measures in Spark streaming. It is important to point out that Spark obtains AIS data at a frequency that tends 100 records per second through the use of a customized receiver.

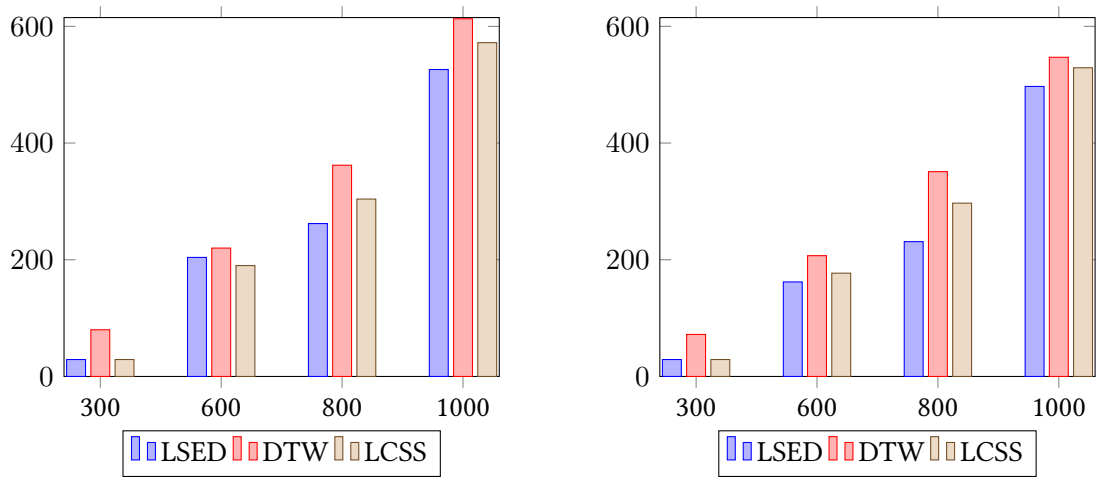
In order to investigate how long it takes the aforementioned algorithms to dispatch and process data, we run twenty four different setups: one for each of the measures LSED, DTW, and LCSS, as well as batch intervals for the streaming window, using grid partitioning for nine and sixteen

cell partitions. The results of the execution time for each batch is shown in table 4.1. And the results for the *total* execution time is shown in the Fig 4.2.

9 No of Partitions												
Similarity Measure	**LCSS**				**DTW**				**LCSS**			
Records/ Batch	**300**	**600**	**800**	**1000**	**300**	**600**	**800**	**1000**	**300**	**600**	**800**	**1000**
1	5	28	14	3	8	17	12	19	6	10	9	17
2	4	41	50	96	14	33	72	118	5	40	61	101
3	5	27	47	107	15	38	68	116	5	38	60	116
4	4	36	52	110	16	41	71	119	4	22	62	108
5	6	47	48	102	14	39	69	120	5	34	57	114
6	5	25	51	108	13	52	70	121	4	46	55	116

16 No of Partitions												
1	5	17	15	28	6	16	27	17	5	9	11	23
2	5	32	47	86	9	30	62	108	4	28	58	101
3	6	34	46	95	15	37	65	99	5	36	60	98
4	4	26	45	97	14	40	61	107	4	32	62	101
5	4	30	38	93	15	41	67	105	6	33	55	99
6	5	23	40	98	13	43	69	111	5	39	51	107

TABLE 4.1: Execution time (in seconds) per Batch of Records



(a) 9 No of Partitions

(b) 16 No of Partitions

FIGURE 4.2. Total Execution time in seconds.

In this particular application, it is abundantly clear that the LCSS and DTW measures both incur significantly higher costs in terms of performance when compared to the LSED measure. According to the experiment results, the performance degrades as the number of records in each batch increases. Furthermore, we can see that the speed improves marginally when the algorithm is executed on 16 partitions. Because the technique made use of Spark's distributed processing capabilities, the computation time for more partitions was lowered as expected.

In terms of trajectory similarity measurements, DTW underperformed the others because it required more actions to be performed in order to be calculated.

However it is highly noted that even if the performance the algorithmic measures provide differs, it has come to our attention that the outcomes of this application may turn out differently due to the unique aspects of each measure. As we have already mentioned [34] LSED is simple and effective, and it generally calls for trajectories to be of the same length. DTW and LCSS are able to handle trajectories of varying lengths, but they are computationally expensive and there is a possibility that they will not work well when the trajectories are related in a nonlinear way. In conclusion the choice of similarity measure is dependent on the specific application as well as the characteristics of the trajectories that are being compared; however, it is crystal clear that the lock-step Euclidean distance performs better with the grid partition technique when it comes to the specific application, and datasets in question.

Chapter 5

Conclusions & Future Work

This work has investigated that vessel routing problem by making use of AIS and real time data, with a particular emphasis being put on trajectories and the use of waypoints to detect similarities between the itineraries taken by different vessels. In particular, a comparison of the LSED, DTW, and LCSS was carried out based on their computing performance on this specific problem and some of their characteristics.

Finally, several conclusions are highlighted, based on the results of our experiments. Firstly, LSED performs better than the other techniques that were tested in an application where the length of the trajectories is dependent to the starting and finishing positions, but there is a significant difference between them on the abundance of points. The performance of trajectory similarity methods is dependent on a number of variables, including the length and complexity of the trajectory, the sampling rate of the data, and the distance metric that is used. It is important to consider the trade-off between accuracy and computational efficiency in real time vessel tracking applications. This is because some techniques may be too computationally intensive to be applied in real time scenarios. In conclusion, the selection of a technique for determining trajectory similarity is dependant on the particular application requirements as well as the characteristics of the vessel trajectory data.

Overall, the comparative analysis of trajectory similarity techniques for vessels in real time emphasizes the significance of selecting the most suitable technique based on the specific application requirements and characteristics of the vessel trajectory data. It also emphasizes the need for continued research and development in this area to enhance the precision and effectiveness of trajectory similarity techniques for real time vessel tracking applications.

5.1 Future Work

Based on the findings of the comparative analysis of trajectory similarity techniques for vessels in real time, there are several avenues for future enhancements in this area. To begin, it is possible to investigate the possibility of integrating multiple trajectory similarity techniques into one another in order to capitalize on the benefits offered by a variety of approaches and enhance accuracy overall. Some other methods that have not been used in this work and it is noticable to be studied are Edit distance and Fréchet distance. In addition, it is important to mention that a good way to improve performance is to integrate a different partitioning technique from those described in Chapter 2 and perform a comparison. This is worth mentioning because it is important to note that this is a good way to improve performance.

Overall, future work in this area can focus on improving the accuracy, efficiency, and applicability of trajectory similarity techniques for vessel tracking applications in real time scenarios. This can be achieved through further evaluation, integration of multiple techniques, real time implementation and partitioning methods.

Appendix A

Custom Receiver

This is a Custom Receiver class in charge of receiving AIS messages as a stream to the real-time processing system.

```
public class AisReceiver extends Receiver¡String¿ -

    private  String host;
    private  int port;

    public AisReceiver(String host, int port) -
        super(StorageLevel.MEMORYANDDISK2());
        this.host = host;
        this.port = port;

    public AisReceiver() -
        super(StorageLevel.MEMORYANDDISK2());


    @Override
    public void onStart() -
        new Thread() -
            @Override
            public void run() -
                receive();

        .start();


    @Override
    public void onStop() -
```

```
 private void receive() -
     try -
         File[] filesInDirectory = new File(StaticVars.dataAIS
↪ ).listFiles();
         Arrays.sort(filesInDirectory);
         BufferedReader reader = null;
         while (!isStopped()) -
             for (File f : filesInDirectory) -
                 String filePath = f.getAbsolutePath();
                 String fileExtenstion = filePath.substring(
↪ filePath.lastIndexOf(".") + 1, filePath.length());
                 if ("csv".equals(fileExtenstion)) -
                     reader = new BufferedReader(new
↪ FileReader(filePath));
                     String inputLine;
                     List¡String¿ buffer = new ArrayList¡¿();
                     while ((inputLine = reader.readLine()) !=
↪  null && !isStopped()) -
                         buffer.add(inputLine);
                         if (buffer.size() ¿= StaticVars.
↪ recordsBuffer) -
                             store(buffer.iterator());
                             buffer.clear();
                         Thread.sleep(1000);


                     if (!buffer.isEmpty()) -
                         store(buffer.iterator());
                         buffer.clear();




         reader.close();
         restart("Trying to connect again");
       catch (Exception e) -
         restart("Error receiving data", e);
```

Bibliography

[1] [n.d.]. Digital route service for navigation - Kystverket. https://www.routeinfo.no/cruiseroutes.

[2] [n.d.]. Free AIS vessel tracking | AIS data exchange | JSON/XML ship positions. https://www.aishub.net/.

[3] [n.d.]. Ships' routeing. https://www.imo.org/en/OurWork/Safety/Pages/ShipsRouteing.aspx.

[4] [n.d.]. What is GIS? | Geographic Information System Mapping Technology. https://www.esri.com/en-us/what-is-gis/overview.

[5] 2015. Windows. https://nightlies.apache.org/flink/flink-docs-release-1.15/docs/dev/datastream/operators/windows/

[6] 2023. Samza. https://samza.apache.org/.

[7] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (sep 1975), 509–517. https://doi.org/10.1145/361002.361007

[8] Donald J. Berndt and James Clifford. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (Seattle, WA) *(AAAIWS'94)*. AAAI Press, 359–370.

[9] Roger Bivand, Edzer Pebesma, and Virgilio Gómez Rubio. 2013. *Applied Spatial Data Analysis With R.* https://doi.org/10.1007/978-1-4614-7618-4

[10] Peter A. Burrough and Rachael A. McDonnell. 1998. *Principles of Geographical Information Systems.*

[11] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache Flink™: Stream and Batch Processing in a Single Engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 36, 4 (2015), 28–38.

[12] Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data* (Baltimore, Maryland) *(SIGMOD '05)*. Association for Computing Machinery, New York, NY, USA, 491–502. https://doi.org/10.1145/1066157.1066213

[13] Jiafeng Ding, Junhua Fang, Zonglei Zhang, Pengpeng Zhao, Jiajie Xu, and Lei Zhao. 2019. Real-Time Trajectory Similarity Processing Using Longest Common Subsequence. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)* (2019). https://doi.org/10.1109/hpcc/smartcity/dss.2019.00194

[14] Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. 2008. Towards a Taxonomy of Movement Patterns. *Information Visualization* 7 (08 2008), 240–252. https://doi.org/10.1057/palgrave.ivs.9500182

[15] Maxim A Dulebenets, Junayed Pasha, Olumide F Abioye, and Masoud Kavoosi. 2021. Vessel scheduling in liner shipping: A critical literature review and future research needs. *Flexible Services and Manufacturing Journal* 33 (2021), 43–106.

[16] Ahmed Eldawy, Louai Alarabi, and Mohamed F. Mokbel. 2015. Spatial Partitioning Techniques in SpatialHadoop. *Proc. VLDB Endow.* 8, 12 (aug 2015), 1602–1605. https://doi.org/10.14778/2824032.2824057

[17] R. A. Finkel and J. L. Bentley. 1974. Quad trees a data structure for retrieval on composite keys - Acta Informatica.

[18] David Hilbert. 1891. Ueber die stetige Abbildung einer Line auf ein Flächenstück - Mathematische Annalen.

[19] Fabian Hueske and Vasiliki Kalavri. 2019. *Stream Processing with Apache Flink.* O'Reilly Media.

[20] Kandrouch Ibtissame, Redouani Yassine, and Chaoui Habiba. 2017. Real time processing technologies in big data: Comparative study. In *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI).* 256–262. https://doi.org/10.1109/ICPCSI.2017.8392202

[21] E. Keogh and C. A. Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems* 7, 3 (3 2005), 358–386.

[22] Jay Kreps. 2014. *I Heart Logs: Event Data, Stream Processing, and Data Integration.*

[23] Patrick Laube and Stephan Imfeld. 2002. Analyzing Relative Motion within Groups ofTrackable Moving Point Objects. 132–144. https://doi.org/10.1007/3-540-45799-210

[24] Chung-Yee Lee and Dong-Ping Song. 2017. Ocean container transport in global supply chains: Overview and research opportunities. *Transportation Research Part B: Methodological* 95 (2017), 442–474.

[25] Gyewon Lee, Jeongyoon Eo, Jangho Seo, Taegeon Um, and Byung-Gon Chun. 2018. High-Performance Stateful Stream Processing on Solid-State Drives. In *Proceedings of the 9th Asia-Pacific Workshop on Systems* (Jeju Island, Republic of Korea) *(APSys '18)*. Association for Computing Machinery, New York, NY, USA, Article 9, 7 pages. https://doi.org/10.1145/3265723.3265739

[26] S.T. Leutenegger, M.A. Lopez, and J. Edgington. 1997. STR: a simple and efficient algorithm for R-tree packing. In *Proceedings 13th International Conference on Data Engineering*. 497–506. https://doi.org/10.1109/ICDE.1997.582015

[27] Xiufeng Liu, Nadeem Iftikhar, and Xike Xie. 2014. Survey of real-time processing systems for big data. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/2628194.2628251

[28] Gerard Maas and Francois Garillot. 2019. *Stream Processing with Apache Spark*. O'Reilly Media.

[29] Nathan Marz and James Warren. 2015. *Big Data:Principles and Best Practices of Scalable Realtime Data Systems*.

[30] Shadi A. Noghabi, Kartik Paramasivam, Yi Pan, Navina Ramesh, Jon Bringhurst, Indranil Gupta, and Roy H. Campbell. 2017. Samza: Stateful Scalable Stream Processing at LinkedIn. *Proc. VLDB Endow.* 10, 12 (aug 2017), 1634–1645. https://doi.org/10.14778/3137765.3137770

[31] Giuliana Pallotta, Michele Vespe, and Karna Bryan. 2013. Vessel Pattern Knowledge Discovery from AIS Data: A Framework for Anomaly Detection and Route Prediction. *Entropy* 15, 6 (2013), 2218–2245. https://doi.org/10.3390/e15062218

[32] Frank Ramsak, Volker Markl, Robert Fenk, Martin Zirkel, Klaus Elhardt, Rudolf Bayer, and Tu Mnchen. 2000. Integrating the UB-Tree into a Database System Kernel. (12 2000).

[33] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2019. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal* 29, 1 (oct 18 2019), 3–32.

[34] Y. Tao, A. Both, R. I. Silveira, K. Buchin, S. Sijben, R. S. Purves, P. Laube, D. Peng, K. Toohey, and M. Duckham. 2021. A comparative analysis of trajectory similarity measures. *GIScience Remote Sensing* 58, 5 (jun 23 2021), 643–669.

[35] M. Vlachos, G. Kollios, and D. Gunopulos. 2002. Discovering similar multidimensional trajectories. In *Proceedings 18th International Conference on Data Engineering*. 673–684. https://doi.org/10.1109/ICDE.2002.994784

[36] Wenjie Yang, Xingang Liu, Lan Zhang, and Laurence Yang. 2013. Big Data Real-Time Processing Based on Storm. 1784–1787. https://doi.org/10.1109/TrustCom.2013.247

[37] Xiaochuang Yao, Mohamed F. Mokbel, Louai Alarabi, Ahmed Eldawy, Jianyu Yang, Wenju Yun, Lin Li, Sijing Ye, and Dehai Zhu. 2017. Spatial coding-based approach for partitioning big spatial data in Hadoop. *Computers & Geosciences* 106 (2017), 60–67. https://doi.org/10.1016/j.cageo.2017.05.014

[38] Y. Zheng. 2015. Trajectory Data Mining. *ACM Transactions on Intelligent Systems and Technology* 6, 3 (may 12 2015), 1–41.