Master's thesis
Master's Programme in Computer Science

Geospatial DBSCAN Hyperparameter Optimization with a Novel Genetic Algorithm Method

Jussi Steenari

January 9, 2023

Supervisor(s):	Dr. Lucy Ellen Lwakatare
	Professor Jukka K. Nurminen

Examiner(s):	Dr. Lucy Ellen Lwakatare
	Professor Jukka K. Nurminen

UNIVERSITY OF HELSINKI
FACULTY OF SCIENCE

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Koulutusohjelma — Utbildningsprogram — Degree programme Master's Programme in Computer Science

Tekijä — Författare — Author

Jussi Steenari

Työn nimi — Arbetets titel — Title

Geospatial DBSCAN Hyperparameter Optimization with a Novel Genetic Algorithm Method

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year January 9, 2023	Sivumäärä — Sidantal — Number of pages 62

Tiivistelmä — Referat — Abstract

Ship traffic is a major source of global greenhouse gas emissions, and the pressure on the maritime industry to lower its carbon footprint is constantly growing. One easy way for ships to lower their emissions would be to lower their sailing speed. The global ship traffic has for ages followed a practice called "sail fast, then wait", which means that ships try to reach their destination in the fastest possible time regardless and then wait at an anchorage near the harbor for a mooring place to become available. This method is easy to execute logistically, but it does not optimize the sailing speeds to take into account the emissions. An alternative tactic would be to calculate traffic patterns at the destination and use this information to plan the voyage so that the time at anchorage is minimized. This would allow ships to sail at lower speeds without compromising the total length of the journey.

To create a model to schedule arrivals at ports, traffic patterns need to be formed on how ships interact with port infrastructure. However, port infrastructure is not widely available in an easy-to-use form. This makes it difficult to develop models that are capable of predicting traffic patterns. However, ship voyage information is readily available from commercial Automatic Information System (AIS) data. In this thesis, I present a novel implementation, which extracts information on the port infrastructure from AIS data using the DBSCAN clustering algorithm.

In addition to clustering the AIS data, the implementation presented in this thesis uses a novel optimization method to search for optimal hyperparameters for the DBSCAN algorithm. The optimization process evaluates possible solutions using cluster validity indices (CVI), which are metrics that represent the goodness of clustering. A comparison with different CVIs is done to narrow down the most effective way to cluster AIS data to find information on port infrastructure.

ACM Computing Classification System (CCS): (1) Computing methodologies → Machine learning → Machine learning approaches→ Bio-inspired approaches → Genetic algorithms

(2) Computing methodologies → Machine learning Learning paradigms → Unsupervised learning → Cluster analysis

(3) Applied computing → Operations research → Transportation

Avainsanat — Nyckelord — Keywords

DBSCAN, clustering, maritime industry, optimization, genetic algorithm, cluster validation indices

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

The shipping industry is one of the biggest global industries. It is estimated that about 80-90 % of the world's goods are transported via the international shipping industry [41]. Shipping is also a major factor in global greenhouse gas emissions, and the shipping industry recently has taken steps to make the industry more environmentally friendly [58]. However, one main contributor to ships' emissions is their non-optimal logistics while at sea [1]. Ships have traditionally followed the "sail fast, then wait" approach when sailing, which means that they try to sail to their destination as fast as possible and then wait near the harbor area for a berth to become available. By optimizing the speed of the ship to minimize emissions, their total greenhouse gas emissions could be reduced by as much as 15% if the sailing speeds are lowered by one knot on average [58]. Applied to the global fleet, this could end up saving 60 million tons of CO_2 per year, which is the equivalent of a medium-sized country. Lowering the ship's speed can also decrease the wear on the ship's hull, thus decreasing the need for maintenance. However, reducing the ships' speeds would increase the traveling time and crew costs. In a highly competitive market such as maritime logistics, it is clear that not many actors would voluntarily make this sacrifice. On the other hand, maritime actors could be more willing to embrace this change if the overall length of the traverse could be executed in the same duration by cutting the waiting time at anchorages.

One way to decrease the waiting time in anchorages is to have an up-to-date overview of the traffic congestion of the port so that ships can arrive at their destination when there is a mooring place available. Acquiring traffic congestion information would require knowing the port layout, i.e. where the piers and wharves are located in the harbor. However, getting this information is easier said than done since harbors are independent entities and oftentimes reluctant to share their data. In some cases, the data on harbor infrastructure might not be easily digestible, and language barriers and bureaucracies can significantly slow the acquisition of harbor infrastructure data.

On the other hand, ship location information can be easily acquired through commercial means. The ship location information is created by the automatic identification system (AIS), which transmits messages containing operational details of the ship's journey and characteristics. The original intent of this messaging system was to provide ships with a tool for collision avoidance. The primary use remains the same, but since then, data generated by this system has seen many use cases, such as analyzing ship movements or estimating maritime emissions [13, 59]. This thesis introduces a novel optimization implementation that aims to find the layout of port infrastructure from AIS messages. This implementation uses the AIS data to detect wharves in ports using clustering. A wharf is a structure or natural formation on the shore so vessels may be moored alongside to handle port operations such as unloading or loading cargo. A wharf consists of multiple berths, while a berth is defined as a place where a single ship can moor at a given time. The difference between a wharf and a berth is demonstrated in Figure 1.1.

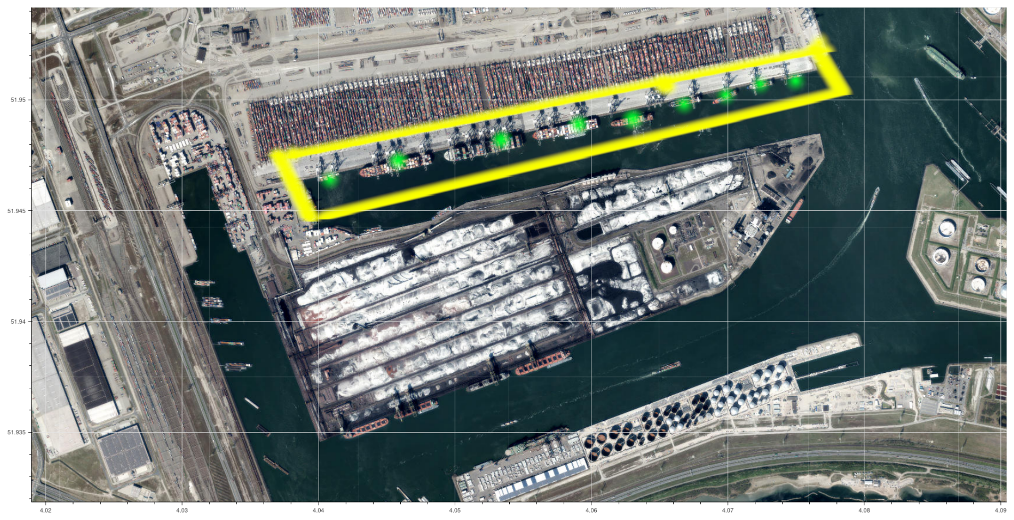

Figure 1.1: Port nomenclature demonstrated: berth locations are marked in green while the wharf is outlined in yellow. The harbor consists of all facilities at the shore, i.e. the wharves and all equipment directly related to them.

Detecting the wharves in the port can create a more detailed model of port congestion. This can, for example, allow an analysis of port statistics such as average ship stay per terminal area, what kind of ships the port can service, and how long ships have to stay at anchorage before a berth becomes available. Having an up-to-date overview of the port infrastructure would allow a more in-depth analysis of the ship traffic patterns to be created using AIS data. This would allow ships to plan their journey so that there will be a berth available in the port upon their arrival, or at least so that the time at anchorage is minimized. This just-in-time arrival would allow the ships to lower their speeds at sea without compromising the total length of the journey.

The implementation uses a clustering algorithm to find areas of interest derived from AIS data. Literature suggests that when clustering spatial data with lots of noise points, which AIS messages are known to contain, the best performance can be achieved with a density-based clustering approach [28, 22]. This thesis evaluates how the DBSCAN clustering algorithm can cluster AIS points to find the wharves.

The DBSCAN clustering algorithm has demonstrated its effectiveness in clustering spatial data, and it has also been used in use cases where the primary data source is AIS messages [18, 2, 32, 17]. One known difficulty of clustering is that it does not have an inherent way to validate the results [3]. While there are methods that evaluate clustering results against some predetermined correct ground truth, getting this ground truth could be an issue. Suppose, for example; port authorities are unwilling to provide their terminals' exact coordinates. In that case, the validation has to be based on simple statistics, visual confirmation, and domain knowledge.

Another difficulty in clustering is finding the optimal hyperparameters [16]. A hyperparameter is a parameter whose value is used to control the learning process in machine learning. Contrary to parameters, which are derived during the training process, hyperparameters are given to the machine learning process by the user before execution. The correct choice of hyperparameters is vital in finding an optimal clustering solution, but selecting these is usually the result of trial and error or derived from a deep understanding of the data. While there have been some attempts at using cross-validation techniques in clustering problems, the results have not been consistent and the method appears to work best with simple clustering algorithms such as k-means [17].

Cluster validity indices (CVI) are a tool developed to evaluate clustering with just innate characteristics of the clustering [53]. These validity indices, in theory, can provide an objective overview of the goodness of clustering. However, using these indices is not that simple, and the research on them has had some inconsistent results [19]. In this thesis, the novel optimization implementation uses CVIs to find the most optimal DBSCAN hyperparameters for AIS data sets collected from different harbor areas. The implementation is tested to see if suitable hyperparameters are detected that can generate clusters that represent the wharves from AIS data using the DBSCAN clustering algorithm.

This thesis presents and tests the novel optimization implementation to detect suitable hyperparameters for the DBSCAN algorithm. This experiment aims to identify areas that represent wharves in harbors. The implementation uses a genetic algorithm to find the most suitable hyperparameters that optimize various CVIs and other metrics. The performance of different approaches in defining the fitness function of the optimizer is compared to each other by simple statistics and visual inspection. In the next chapter, the background of the problem will be discussed in more detail.

2. Background

2.1 AIS data

One of the most widespread data formats in maritime is the automatic identification system (AIS) messages [2]. The use of AIS technology as a standard for maritime identification originated from the International Convention for the Safety of Life at Sea (SOLAS) held in 1974 [24]. This technology was initially designed to prevent collisions of seaborne vehicles [22]. Since then, this technology has spawned many applications in other fields. Protocols regarding AIS technology are regulated by the International Maritime Organization (IMO). The basic premise of the AIS messaging system is that ships transmit their locations and status at different time intervals [24]. Ships larger than 500 gross tonnages are required to transmit AIS messages at all times they are operational while ships larger than 300 gross tonnages are required to transmit AIS messages when operating in international waters. Furthermore, all passenger ships are required to transmit AIS messages regardless of size. Leisure crafts, fishing boats, and warships are not required to carry AIS equipment, but often these kinds of ships transmit AIS messages voluntarily. Additionally, some port administrations might require all ships to carry AIS equipment while within their jurisdiction [9].

AIS transponders are divided into two categories: AIS-A and AIS-B transponders [24]. AIS-A equipment is required to have full maritime functionality outlined by IMO. All ships that are required to carry AIS equipment must carry AIS-A capable equipment. AIS-B is a lightweight version of AIS-A that can be used on vessels that do not require carrying AIS equipment, such as pleasure craft or fishing boats. The experiment in this thesis uses only AIS-A-type messages. The data fields transmitted by class A transponders are depicted in Table 2.1

The AIS messages are transmitted as dynamic and static messages [22]. Dynamic messages carry information about the current location of the ship as well as anything related to primary ship operations, such as speed and course. Static messages contain information on the ship metadata such as ship name and dimensions. Static messages are transmitted at 6-minute intervals, while dynamic messages are transmitted at different intervals based on their operations, ranging from 3 minutes when moored or at

Dynamic	**Static**	**Voyage related**
Timestamp	MMSI	Ship's draft
MMSI	Call sign and name	Hazardous cargo (type)
Position coordinates	IMO number	Destination and ETA
Speed	Length and Beam	Route plan
Course/Heading	Type of ship	
Rate of turn	Location of AIS transmitter antenna	
Navigational status		

Table 2.1: Dynamic, static, and voyage-related data fields sent by the type-A-AIS transmitter. [24]

anchor to 3 seconds when sailing at speeds over 23 knots. This means that the amount of data generated by AIS is quite large. On average, a single ship sending an AIS message once every minute would generate over half a million rows of data in a single year. Data collected in large or heavily trafficked areas can easily grow to terabytes.

AIS data is collected by terrestrial stations often operated by local authorities or companies specialized in AIS data. AIS data is a sought-after commodity and many companies capitalize on this. Companies such as Marine Traffic, ORBCOMM, or Spire provide access to their AIS databases for a fee [59]. These companies provide either historical data by FTP transfer or live data through their API services [52]. Open-source platforms to share AIS data, such as AISHub, Sailwx, and Google APRS, provide free AIS data for the consumer. However, there are no guarantees for the data quality in these services and most of them only provide live data. Also, some services such as AISHub have caveats that to access the data, the user must provide their own data stream to the service. Lastly, AIS data can also be collected with consumer-grade equipment from nearby areas. This can provide an unfiltered view of a specific area, but the equipment must be installed near a highly trafficked area to get meaningful data.

2.1.1 AIS data use in applications and research

AIS data was originally designed to just prevent sea collisions, but the overall volume and easy access of this data have stemmed many new applications [59]. The AIS technology was originally designed for regional use and the limits of its transmit range made a wide-scale collection of AIS data difficult at first. However, the introduction of satellite-based AIS made the large-scale collection of AIS data much more feasible, which has sparked interest in researching this domain.

The most widely used application of AIS data is still in navigation [52]. The AIS system has displaced the need for manual lookout and surveillance both at sea and in port, and it is widely utilized in major port areas for navigational safety. AIS data can be used to monitor anomalous behavior in ships such as ships entering restricted areas or traveling with unusual speeds or patterns. This can be used to detect, for example, unlicensed fishing [59].

One of the most widely used implementations of AIS data since its inception is to use it to monitor ship movements. Initially, the monitoring focused on ship-to-ship communication, but since then the focus has switched to more centralized platforms, with more fine-grained risk assessment [59]. In addition to potentially hazardous situations, AIS data can be used to monitor potentially illegal activities and other anomalies [29].

Another navigation application is the route prediction of ships [52]. Route prediction tries to build a model that estimates the ship's future position. Ship movement has specific characteristics when compared to land or air vehicles. Firstly changing of movement for ships takes a much longer time compared to other vehicles i.e. ships take a long time to turn, stop or reverse. Secondly, the movement of ships occurs largely in vast two-dimensional space compared to restricted two-dimensional space for land vehicles or three-dimensional space for aircraft.

While it is difficult to get any insights from raw AIS data, data mining methods have been extensively used to get a better overall picture of the data. One of the most used methods is to use AIS messages to construct trajectories and do the analysis based on these[46]. Clustering methods such as k-means or DBSCAN are often used to extract trajectories. These trajectories can be used for further research, such as trajectory prediction and anomaly detection [59].

AIS data can also be used in analyzing much larger entities than the shipping industry. One prominent study by Adland et al. [2] used AIS data to get an overview of global trade volumes. They concluded that AIS data can be used to get a rough estimate of global trade that is in line with official numbers.

AIS data can also be used to optimize ship voyages by combining it with other data sets. Linking weather data and ship trajectory data can be used to find an optimal route to save either time or fuel. AIS data has also been used to estimate the amount of greenhouse gases emitted in various areas, which can be a great asset when estimating the maritime industry's effect on climate change [58].

2.1.2 AIS data quality issues

While AIS data is available in large quantities, its usage is hindered by the overall poor quality of the data [13]. The lack of rules and regulations on data transmission results in a lack of real consequences for transmitting poor quality or even false data. While international treaties outline that ships should send AIS messages while operational, there are no real consequences if the ship stops transmitting messages. Furthermore, there is no enforcement to verify that the data being sent is correct, other than that ships do not send messages under falsified identity [22].

Since static AIS messages are filled in by the crew, the messages are more susceptible to errors compared to dynamic messages. Some estimates depict that as much as 80% of AIS data is incomplete in some way, and most of this is due to static messages [22]. The most common missing attributes are the ship dimensions and their vessel types. In addition, several columns frequently have incorrect information in them. For example, the estimated time of arrival can be inputted as the departure time, or the destination might be misspelled or deliberately entered incorrectly. The amount of data errors in the static messages makes this data source unreliable, and most AIS data research is focused on using dynamic messages.

Although dynamic messages in general are of better quality than static messages, they still have some issues that need to be solved [30]. Often AIS datasets are missing some messages, which would indicate that either ship are turning off their transmitters or the signals are lost somehow, the latter being the more probable cause. There are some cases where ships can turn off their transponder, such as under threat of piracy, but it is generally expected that they continuously transmit while they are operational [30]. Messages can be also lost due to external factors such as weather, coast geometry, and geomagnetism. There is also some indication that in busy waterways, the terrestrial stations can get overloaded and thus drop some messages [13].

2.2 Port logistics and intelligence

The efficient logistic of ships in port areas is key in making the world economy function. A broad definition of a port is that it is a transfer point between water and land, including onshore facilities and logistical infrastructure. This is in contrast with the concept of the harbor, which is a sheltered body of water where boats or ships may moor or anchor. More than 80% of international cargo by volume is transported via sea, which makes the function of ports important in global trade [6]. The exact number of ports in the world is not known, estimates range from 2000 to 30000 ports or terminal facilities depending on the definition. Ports have distinct assets and services available for ships, and there does not exist a universally agreed-upon definition for these [6].

Ports are complex entities that have various actors and operators with different goals. Ports generate trade and are usually major employers in the areas that they reside in. Furthermore, ports play a vital role in a country's trade efficiency as well as sustainability. This makes ports a vital strategic asset as well. Port can be owned by the state, the municipal government, or even private companies [6]. Also, a new form of port ownership called landlord port has emerged and has gained prominence. In this kind of port ownership, the state or the city owns the land that is then leased out to private entities. The multitude of actors operating in a given port makes acquiring a complete picture of the port terminals tedious, if not impossible. This makes the process of acquiring this information via alternative means such as the method depicted in this thesis more appealing.

Ships usually arrive in ports using the "sail fast, then wait" approach [50]. This means that after a ship leaves its origin port, it sails the fastest route toward the destination without regard for the local conditions and then waits for a berth to become available while waiting at anchor near the destination port. This method is easy to implement, but it won't necessarily contribute to any significant time savings. On the other hand, the current method of sailing at maximum speed contributes much more to the carbon emissions that the ship produces. It is estimated that by having a just-in-time approach to voyage planning, ships could reach their destination at the same time while reducing their carbon emissions by around 15%. In addition to the reduction of carbon emissions, lowering the operational speed of ships has additional benefits such as decreasing ship hull degradation, reducing underwater noise pollution, and lowering the chance of the ship colliding with whales [47].

Port intelligence focuses on the correct use of allocation of resources for ports. The goal of port intelligence is to quickly and proactively provide quality information on port logistics. The key component of port intelligence is to identify the port infrastructure so models of the ship traffic can be formed. Most of the research on port intelligence has been on trying to estimate the traffic congestion at ports. Chen et al. used AIS messages to generate an overview of container ship traffic in the ports of Hong Kong and Singapore [10]. They used manually labeled berthing areas and calculated visits to them using AIS data. While their estimations of traffic deviated about 2% from officially published numbers, they demonstrated that port traffic can be estimated from publicly available AIS data.

Another port traffic monitoring approach was suggested by Rhodes et al. [44] that also relied on predefined ground truths for the berth areas. Their research used a modified version of the Fuzzy ARTMAP neural network classifier to detect abnormal behavior, such as unusual trajectories or excess speeds of ships, in port areas. Similarly, in a study done by Rajabi et al. [41] regarding smart port architecture, the wharves were predefined by hand.

A study by Millefiori et al. [35] identified wharves from AIS data gathered from areas near the port of Shanghai using an unsupervised statistical approach called kernel density estimation. The authors proposed a methodology that uses AIS data to define the exact seaport location and its operational boundaries. The proposed method was applied to a dataset of more than 57 million AIS messages and they were able to parse areas from the port area and estimate traffic congestion based solely on AIS data. Other port-based AIS studies include analyzing ship traffic near predetermined choke points in the Singapore port area [60]. A similar study was done on ships maneuvering from Busan New Port in Korea [32]. AIS data is also used to gain insights into specific port operations, such as ship refueling also known as bunkering [18].

The work presented in this thesis is a precursor to implementing the just-in-time approach to voyage planning. By detecting the wharves in a given port, their statistics can be analyzed and used to determine the traffic patterns of said port. This in turn allows ships to sail at a much slower speed, thus lowering their greenhouse gas emissions, while still arriving at the target port in the allotted time. The novel optimization implementation presented in this thesis uses a similar clustering approach as Millefiori et al. [35]. However, instead of kernel density estimation, this implementation uses DBSCAN to cluster the data points. Compared to kernel density estimation, DBSCAN is much more resistant to noise. Furthermore, the optimization approach applied in this implementation allows the method to detect the best hyperparameters for each port area with little human intervention.

2.3 Hyperparameter tuning in clustering

The process of selecting the best hyperparameters for clustering algorithms has mostly relied on expert knowledge and trial-and-error approaches, but some steps have been made for more systematic approaches [26]. However, compared to hyperparameter tuning in supervised machine learning, this topic has gotten much less attention in the unsupervised machine learning field. Moreover, interest in this topic seems to be relatively new, since most of the published research is only a couple of years old.

Karami et al. [26] used a differential evolution approach to determine the best minimum points and epsilon hyperparameters for the DBSCAN clustering algorithm. They used a simple fitness function based on external cluster validation, meaning they knew the ground truth when doing the evaluation. The evaluation was done using the Davies-Bouldin and Dunn indices as well as an entropy metric. Using this approach they were able to find epsilon and minimum points parameters that captured most of the predetermined labels.

There have been some frameworks designed to tackle the hyperparameter tuning problem. Firstly Fan et al. [16] devised a framework that tuned the hyperparameters for clusters in the Stochastic Blockmodel. They used a cross-validation approach to find the optimal number of clusters in a given clustering, which in addition to the hyperparameters tuning, also performed a model selection process. A similar framework based on cross-validation was devised by Ditton et al. [12]. Their approach used internal cluster validity indices (CVIs) in addition to meta-criteria, such as the stability of clusters in different samplings, as a way to evaluate different hyperparameters. Compared to previous methods, their evaluation was not based on ground truth labels and instead relied on domain-specific evaluation, which was manually inputted during the hyperparameter tuning process.

A framework called AutoClust was developed by Poulakis et al. [40]. Their optimization used CVIs to select the most suitable clustering algorithm as well as the optimal hyperparameters. They compared 24 different data sets and used their method to select the best clustering multiple CVIs. The CVIs were introduced to a regression model, which used various CVIs to predict the Adjusted Rand Index, which is an external cluster validity index that compares the clustering to known ground truth. While showing promising results, they concluded that the selection of the optimization goal is not straightforward when dealing with clustering problems and that even though CVIs provide a tool to evaluate clustering, the selection of the right CVI for a given clustering problem is crucial.

A similar approach was tested by Falahiazar et al. [15] that instead used a multi-optimization implementation of a genetic algorithm to optimize the hyperparameters of DBSCAN. Compared to the approach suggested by Karami et al., this implementation tried to determine the best hyperparameters without any outside ground truths to verify the results. Their proposed solution was based on a multiobjective genetic algorithm, which evaluates optimization based on multiple fitness criteria. They used silhouette and Dunn indices as well as their own Outlier index as a fitness function for the genetic algorithm. This approach proved to be computationally expensive and they were required to make some further adjustments to the code, such as only optimizing two metrics at a time. To evaluate the results they used the Jaccard, Rand, Minkowski, Precision, Recall, and F1 indices, which meant that they had access to the ground truth of the cluster labels. While their approach was able to find hyperparameters that performed relatively well in regard to the above-mentioned indices, they also stressed that finding the right hyperparameters requires expert knowledge in most cases.

In this thesis, the hyperparameter tuning process is done similarly as in the work of Falahiazar et al. [15] However instead of using multi-optimization based on several CVIs, the generic algorithm optimizes only one CVI at a time. The aim of the optimization is to find clusters in port areas that correspond to the wharves of the port. In the next chapter, the methods used in the experiment will be discussed in more detail. The section explains the DBSCAN algorithm, which is used as the clustering algorithm when generating the clusters. Moreover, the CVIs used as the fitness function of the optimizer are discussed in more detail. Finally, the genetic algorithm that is used as the optimizer in finding the optimal hyperparameters is explained in a more thorough manner.

3. Methods

3.1 Clustering methods

Clustering is an unsupervised learning method that groups data points to groups or clusters based on some similarity metric [5]. Clustering methods have been successfully used in tasks such as pattern recognition, image segmentation, document retrieval, data mining, and geographical information system analysis. Clustering methods are generally classified into three categories: partitional, hierarchical, and density-based methods.

Partitional methods divide the data into clusters based on some partitioning metric. These methods are sometimes referred to as centroid-based methods and they work by assigning data points into k-clusters by using iterative processes. Partitional methods are generally thought to be the simplest clustering methods and can perform well if the structure of the data is quite simple [20]. However, these methods rely on knowing the approximate number of clusters expected to be in the data to perform well. Additionally, these methods are not deterministic, different starting points for the same data set could produce radically different clusters.

Hierarchical clustering methods work by initially classifying each data point as a separate cluster and iterating over the data merging close clusters together based on some metric, and iterating the process until clusters are merged together. Hierarchical clustering methods produce a dendrogram that shows the order in which the data points are merged as well as the perceived distance for these merges. Compared to partitional methods, hierarchical methods have a more dynamic method of defining the number of clusters in the data. The visual analysis of the dendrogram can give a good overview of how many clusters are in the data. The downside of these methods is that compared to partitional methods, they can be quite computationally expensive, so these methods perform poorly with large data sets. Furthermore, these methods are quite sensitive to outliers so they don't work that well with noisy data.

Density-based clustering methods work by identifying distinct clusters in the data based on density distribution. Objects in dense areas are considered to be part of a cluster while data points in sparse areas are considered to be noise. What is considered a dense area is usually defined by parameters given to the algorithm. These methods try to find clusters at different levels of granularity as well as to detect noisy data points. Density-based methods can detect clusters of any shape and size. Density-based clustering methods are best suited for noisy data sets where the number of clusters is not previously known. These methods can deal well with noisy data and can identify clusters of various sizes. In this experiment, the data is known to contain noisy points as well as clusters of various sizes, so density-based methods are best suited for finding insights into the data.

3.1.1 DBSCAN

The novel implementation uses DBSCAN as the choice of the clustering algorithm. Short for density-based spatial clustering of applications with noise, DBSCAN is a density-based clustering algorithm developed by Ester et al. in 1996 [14]. DBSCAN was the first of the density-based clustering algorithms and the need for such an algorithm arose from the fact that previous clustering algorithms did not detect clusters with arbitrary shapes very effectively. Moreover, DBSCAN requires much less prior domain knowledge when compared to conventional clustering algorithms, for example, the number of expected clusters is not needed for DBSCAN. Despite its relatively old age, it is widely used in various applications today and new variations are constantly suggested [28].

The algorithm works by finding areas that have sufficient density of points [14]. The two key parameters for DBSCAN are the epsilon parameter and the minimum points parameter. The pseudocode of the DBSCAN algorithm is given in Algorithm 1. The process works by scanning the neighborhood of each point given a distance parameter epsilon and detecting if there is a minimum number of neighboring points within the given distance. If the point has enough neighboring points inside the epsilon radius, it is classified as a core point. If the point does not have enough points in the epsilon neighborhood, but it is part of the neighborhood of a core point, it is classified as a border point. If neither of these cases applies, the point is classified as a noise point. The process works iteratively for each point in the data set. If two forming clusters become density-reachable, i.e. they become connected through a series of core points, they are merged together.

The DBSCAN algorithm can support a variety of distance metrics to detect the points within the given epsilon radius. The most used distance metric is the Euclidean distance, which is calculated with the Cartesian coordinates using the Pythagorean theorem. The Euclidean distance is simple to compute and provides accurate measurements in any dimension, given that the space is evenly distributed. However, when

Algorithm 1 DBSCAN algorithm

```
procedure DBSCAN(pointdata, Min_points, $\epsilon$)
    for point $\in Pointdata$ do
        $visited[point] = True$
        $N \leftarrow GetNeighbors(point, \epsilon)$
        if $|N| < Min\_points$ then
            $point[label] = -1$                          ▷ -1 is the noise label
        else
            $C \leftarrow point$
        end if
        for $point' \in N$ do
            $N \leftarrow N \backslash point'$
            if $visited[point'] == False$ then
                $visited[point'] == True$
                $N' \leftarrow GetNeighbors(point', \epsilon)$
            end if
            if $|N'| > Min\_points$ then
                $N \leftarrow N \cup N'$
            end if
            if $point'[label] == None$ then
                $C \leftarrow C \cup point'$
            end if
        end for
    end for
end procedure
```

dealing with surfaces that are not even, such as the case of coordinates on a sphere, the Haversine distance is more accurate. Another distance metric for DBSCAN that has gained popularity recently is the Mahalanobis distance metric, which measures the distance between a point and a distribution [43]. The Mahalanobis distance metric gives higher priority to points that are in dense areas than to solitary points and thus encourages clustering. However, the default version of the DBSCAN does not support this distance metric and the algorithm needs to be modified to accommodate it.

The advantage of using DBSCAN compared to partition-based clustering approaches is that the clustering algorithm does not need to know the number of clusters beforehand [25]. Also, DBSCAN can find arbitrarily shaped clusters and even clusters that are inside other clusters. The algorithm can detect outliers which allows it to work with noisy data. DBSCAN can also benefit from tree structures to speed up the regional queries, having a significant speedup process on the execution of the algorithm. However, DBSCAN is not entirely deterministic, with the ordering of the data affecting the result [25]. This means that a given point can be classified into two clusters, with the latter cluster membership giving the label. This can essentially create clusters that are smaller than the minimum points parameter. Moreover, DBSCAN struggles to create clusters with large differences in densities, since the epsilon parameter is fixed to a certain value. Overall DBSCAN is really sensitive to hyperparameter tuning, and even a small change in the hyperparameters can greatly affect the number of clusters found [27]. One of the goals of the implementation presented in this thesis is to find a way to automatically detect suitable hyperparameters for the DBSCAN algorithm that find suitable clusters.

Variants

One of the most widely known variants of the DBSCAN algorithm is the hierarchical version of DBSCAN or HDBSCAN [8]. This variant extends the DBSCAN algorithm by converting it into a hierarchical clustering algorithm. This method was created to allow the DBSCAN algorithm to take into account different densities when creating the clusters. It also needs just the minimum points parameter to function, since the epsilon value is derived from the hierarchical ordering. It uses a technique to extract a flat clustering based on the stability of clusters. Compared to other hierarchical methods, such as single-linkage clustering, HDBSCAN can work with many kinds of data types. Also, HDBSCAN utilizes a local density cut threshold, that helps in finding the most significant clusters.

HDBSCAN works similarly to DBSCAN, finding neighborhoods where at least a minimum number of points are within epsilon distance from each other and iterating over the found results, and producing clusters based on these findings [8]. This allows HDBSCAN to detect arbitrarily shaped clusters. In addition, HDBSCAN creates a hierarchy of the data points where the probability of a point being in a cluster is taken into account. This hierarchy can be visualized as a dendrogram, where the data points are linked to nearby data points based on distance. The clusters can then be extracted by cutting off the dendrogram at a set height.

DBSCAN algorithm has been modified and extended to either improve the performance, accuracy, or scope of the algorithm [28]. One of the most used of these extensions is GRIDBSCAN, which partitions the data into grids, which allows to detection of optimal parameters for each subarea of the data. This improves the accuracy of the clustering but increases the computational complexity. A performance-optimized DBSCAN called FDBSCAN was proposed by Zhou et al. [61] in 2000 that improves the neighborhood search query by pre-sorting the data set. A spatial-temporal variant of the DBSCAN algorithm was introduced by Birant et al. in 2007, which optimized the algorithm when spatial-temporal characteristics were present [7].

Custom distance matrices for DBSCAN algorithm

One way to modify the DBSCAN algorithm is to redefine the way it calculates the distances between points. Most DBSCAN implementations allow the user to define a custom distance matrix for the algorithm and use it to calculate the core points. In this thesis, a custom distance metric is introduced that takes into account the way the bow of the ship is facing. A more detailed description of this custom distance metric is given in Chapter 4.

3.2 Cluster validity indices

Cluster validity indices (CVIs) can be used to evaluate whether a given clustering method produces valid results [33]. Optimal clusters should be far apart from each other but dense. In other words, each point should have good connectivity with points within its cluster but poor connectivity to points outside said cluster. The quality of clusters is affected greatly by the choice of the clustering algorithm, the quality of the data as well as the choice of hyperparameters for the clustering algorithm.

Evaluating unsupervised machine learning such as clustering is more challenging than evaluating supervised methods such as classifiers[53]. In supervised learning real labels are available and can be used to compute statistics such as accuracy or precision. In clustering, we usually do not have such labels and we can only base our estimates on the data and the partition in question.

CVIs are divided into three categories, internal criteria, external criteria, and relative criteria [21]. Internal criteria are extracted only from features inherent to the data set. These criteria can be features such as distances between clusters or the density of clusters. Metrics that focus on internal criteria include the silhouette score, Dunn index, Davies-Bouldin index, and the Calinski-Harabasz index [33].

Internal CVIs usually evaluate two aspects of the clustering: compactness and separation [33]. Compactness measures how close the data points in a cluster are, and separation measures the distance between different clusters. Compactness is usually measured by the variance, where lower variance indicates a better grouping of data points. Alternatively, the compactness can be derived from a distance metric, such as maximum or average pairwise distance, and maximum or average center-based distance. Separation is mostly derived from a distance metric, whether it be the distance between cluster centers or cluster boundaries. Most of the CVIs work best on globular (i.e. spherical or globe-shaped) clusters [37]. In the case of density-based clustering, such as with DBSCAN, the shape of clusters can be much more complex. Due to this shortfall, there has been some research done to create metrics that work on non-globular clusters [37].

In contrast, external criteria are derived from comparing the clustering to some outside reference. Usually, these are comparing two clusterings to each other, where the other is known to contain the ground truth. The most used external criteria metrics are the Jaccard coefficient, Rand index, and Fowlkes-Mallows index [21].

Relative criteria metrics are either external or internal criteria metrics, where the score can be ranked according to their quality [37]. For example, according to this premise, a clustering of points with a silhouette score of 0.9 should be a better clustering than one with a silhouette score of 0.8. However, as demonstrated later in this thesis, this is not always this straightforward. Still, relative criteria are important when trying to find the optimal clustering algorithm and hyperparameters for a given data set [54]. There also exists clustering algorithms that use these metrics as a guideline to implement a better clustering [28, 43].

Different data validation metrics work on different kinds of data [33]. A common strategy is to use external evaluation metrics to compare the clustering to "ground-truth" labels that reflect an optimal clustering solution [53]. However, these labels are often not available and when they are available they reflect only one grouping of the data. When "ground-truth" labels are not available, internal validation metrics are a good alternative. However, these metrics also have certain limitations that will be discussed in the following sections.

The choice of the validation metric should reflect what is expected of the clustering. Some metrics can work with noisy data, while others can work on data that has uneven distribution. Usually, the choice of metric is a trade-off of some kind, since none of the metrics can give good scores to all well-formed clusterings. There are also suggestions in the research literature of aggregate metrics that combine several validation metrics to one score [55]. However the effectiveness of these approaches is not that well studied, and most cluster validation techniques calculate several metrics individually and find some sort of compromise to select the best clustering. The CVIs used in the implementation presented in this thesis are internal and relative metrics: the silhouette, Calinski-Harabasz, Davies-Bouldin, and the DBCV index.

3.2.1 Silhouette index

The silhouette index or the silhouette coefficient is an internal criterion metric, that measures how distinct the clusters are from each other. The silhouette index measures cluster compactness based on the pairwise distances between all points in the cluster and separation based on pairwise distances between all points in the cluster and all points in the closest cluster.

The index score is a value in the range from -1 to 1, where 1 means the clusters are distinct and dense, whereas a score of -1 indicates that the clusters are mixed together and sparse. The silhouette index is first calculated for each point in the clustering and then the silhouette index for the whole clustering is calculated by taking the mean of these values. The silhouette score for one point is calculated with formula

$$s = \frac{b-a}{max(a,b)} \tag{3.1}$$

where a is the average distance from the sample to other points in the same cluster and b is the average distance from the sample to points in the next nearest cluster. The silhouette index for the whole data set is calculated by the formula

$$S = \frac{1}{n}\sum_{i=1}^{n}\frac{b-a}{max(a,b)} \tag{3.2}$$

The advantage of using the silhouette index metric is that since the value is bounded between values -1 and 1, the interpretation of the results is straightforward [33]. Furthermore, the metric itself tends to give good scores to clusters that are dense and well-separated, which is in most cases the goal when using clustering algorithms. However, using the silhouette index is most accurate when dealing with convex clusters. Thus using this method on concave clusters would give a bad score even though the clusters would be formed correctly. This makes this metric somewhat problematic when evaluating clusters formed by density-based methods such as DBSCAN. Another drawback of using this metric is its high computational cost of $O(n^2)$, making it unfeasible when evaluating large clusterings. However, a simplified version is available that measures the distance between points and cluster centroids [53].

3.2.2 Calinski-Harabasz Index

Calinski-Harabasz index, also known as the variance ratio criterion, is a cluster evaluation metric based on the degree of dispersion between and within clusters [33]. It defines compactness based on the distance between cluster points and cluster centroids. It defines separation as the distance between the cluster centroid and the data centroid.

The index score is a value between 0 and infinity and larger values indicate better clustering. However, if every cluster has a variance of zero i.e. all points are the same within the cluster, the Calinski-Harabasz score defaults to one. For a set of data E of size n, with k cluster labels, the Caliski-Harabasz index is calculated with the following formula:

$$s = \frac{tr(B_k)}{tr(W_k)} * \frac{n-k}{k-1} \tag{3.3}$$

where $tr(B_k)$ is defined as the trace of the between-group dispersion matrix and $tr(W_k)$ as the trace of the within-cluster dispersion matrix. They are defined as followed:

$$\begin{gathered} W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T \\ and \\ B_k = \sum_{q=1}^{k} n_q (c_q * c_E)(c_q - c_E)^T \end{gathered} \tag{3.4}$$

where C_q is the set of points in a cluster q, c_q the center of said cluster, c_E the center of E and n_q the number of points in cluster q.

The Calinski-Harabascz index is relatively fast to compute compared to the silhouette index, with a computational complexity of $O(nlog(n))$. Compared to the silhouette index, the Calinski-Harabascz performs better when clusters are close together, i.e. forming subclusters. However, noisy data can adversely affect the Calinski-Harabascz index, making it less than ideal when dealing with data with lots of outliers [33].

3.2.3 Davies-Bouldin index

Davies-Bouldin index is a CVI that compares the distance between clusters to the size of the clusters [11]. A lower score indicates better clustering. The Davies-Bouldin index defines compactness as the distance of points in the cluster to its centroid and separation based on distances between centroids. This method has been used extensively in the k-means clustering approach to validate that the number of clusters is the optimal [33]. The index is calculated as follows:

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left(\frac{I(c_i) + I(c_j)}{I(c_i, c_{,j})} \right) \tag{3.5}$$

, where $I(c_i)$ represents the average of the distances between the objects belonging to the cluster C_i and its center. $I(c_i, c_j)$ represents the distance between the centers of the two clusters C_i and C_j. The index score is a value between 0 and infinity and a lower score indicates better clustering.

The Davies-Bouldin index is simpler to calculate when compared to the silhouette index. However, the drawback of the Davies-Bouldin index is that its distance metric only works in euclidean space, making it inaccurate when dealing with real-world spatial data. Also, like the silhouette index and Calinski-Harabaszc index, the Davies-Bouldin index gives better scores for convex clusters compared to other shapes. The Davies-Bouldin index has a time complexity of $O(nlog(n))$.

3.2.4 DBCV

Most of the CVIs work poorly to evaluate concave clusters or have a lot of noise in the data, which is the usual justification for using density-based methods. To get a better estimator for clusters created by density-based methods Moulavi et al. suggested a CVI, density-based cluster validation (DBCV), that would accurately evaluate whether a given density-based clustering is satisfactory [37]. This approach utilizes the density-contour trees to compute the least dense region inside a cluster and the densest region between the clusters. These values are used to measure how the clusters are connected within and between each other. This method has a clear advantage compared to previous metrics meant to measure density-based clusterings. These metrics either failed to take into account the arbitrary-shaped clusters or these methods required some preliminary information about the data sets to work properly. Similarly to the silhouette index, DBCV has a value between -1 and 1 and should be maximized.

In DBCV a mutual reachability distance graph is constructed for each cluster [37]. In this graph, the vertices represent cluster points and the edge weights represent the reachability distances. A minimum spanning tree is created for each of these graphs. From the internal edge weights of this tree, the density sparseness of clusters can be defined. The density separation of two clusters can be defined as the minimum reachability distance between internal nodes of the minimum spanning tree.

While the DBCV metric overcomes some issues related to ranking arbitrarily shaped clusters, upon closer inspection the implementation provided by the authors is not capable of handling large data sets, but since then faster implementations have emerged. The authors didn't include any information on the computational complexity of this approach in their paper, but other studies evaluate it to be $O(n^2)$, which is similar to the silhouette index [56].

3.3 Problems with cluster validation indices

Even though CVIs provide a tool to validate clustering results, justification for their use is oftentimes not that well defined in research [19]. The problem is that clustering is an unsupervised machine learning method, thus having an objective and generalizable function to evaluate the results is hard or even impossible. Often times the results of CVI match expert knowledge poorly [19]. Furthermore, some data sets can have more than one equally valid clustering, which further hampers the search for the optimal clustering [11]. Also, radically different clusterings can have CVI scores that are very close together, which makes evaluating the best solution difficult.

There is a trend to design new CVIs to overcome deficiencies of previous metrics, which has led to an overabundance of available CVIs[19]. Thus selecting the right metric can be overwhelming. There also is a lack of good clustering data sets with predefined labels that are meant to test the effectiveness of CVIs [3]. However, Gagolewski et al. [19] tested various clustering metrics on a small set of benchmark data sets that had predefined correct labels. What they found was that while CVIs can be a useful tool to validate that the clustering produces the right amount of clusters, they are less effective in checking whether the data set is partitioned optimally. This means that validation metrics are better at validating the results of simple methods such as k-means compared to more advanced methods such as density-based clustering methods.

3.4 Hyperparameter selection for clustering algorithms

The choice of clustering algorithm can have an enormous effect on the quality of the clusters detected. Furthermore, the chosen algorithm can produce different results depending on applied hyperparameters. Also, since clustering methods usually lack ground truth, hyperparameter tuning methods applicable to supervised machine learning problems, such as cross-validation, can not be utilized in finding the best hyperparameters. While there is quite a lot of research done on the correct choice of clustering algorithm and researchers tend to justify their choice of clustering algorithm pretty decently most of the time, not that much emphasis is devoted to the choice of hyperparameters for the clustering [48]. In most cases, the hyperparameters are either chosen based on prior knowledge or by trial and error.

The quality of clustering is often times measured by validity metrics or by visual inspection [48]. Some researchers have developed extended versions of existing clustering algorithms that automatically tune the algorithm to use optimal parameters based on some metric [26, 15]. However, CVIs don't always tell the whole picture. As mentioned above, validation metrics tend to work best on globular data, where the clusters are far apart from each other. This does not always reflect real-world data.

When finding the best hyperparameters for the DBSCAN algorithm, the literature would suggest that most effort should be put into finding the optimal epsilon value [28]. For minimum points hyperparameter, a good rule of thumb is to have the value be 2 times the number of dimensions of the data to be clustered. [45] So in our case, where the number of dimensions is 2, the optimal value for minimum points would be close to 4. However, with large data sets, data with lots of duplicates, and data sets with lots of noise, a larger value for minimum points could be appropriate [45]. Since AIS data is known to be noisy and potentially contain duplicate points, we can not make these assumptions about the optimal minimum points hyperparameter in the optimization process.

3.5 Preprocessing

The data sets were preprocessed to make them less noisy and more suitable for clustering. The first steps of preprocessing include the removal of all duplicate AIS messages and the removal of all AIS messages where the heading is null or over 360 degrees. A single ship should transmit only one message at a given timestamp, so all timestamp/MMSI duplicates can be removed as erroneous. The duplicate rows were removed so that the first instance of the duplicate was kept and all other rows dropped. The rows with erroneous heading values were removed because the custom metric outlined in Section 4 relies on calculating the median heading of each mooring event, and including these could mess up the distance calculation.

The second step of the preprocessing involved calculating the groups for each individual mooring event. This was done by first sorting the data using the MMSI and timestamp columns, then by checking whether the MMSI or the navigational status changes compared to the previous row. This information was then converted to sequential numbering using the cumulative sum function.

The training data was generated by taking a group by operation of the data sets where the group by key was the mooring visit number. The data was filtered to include only rows with a navigational status of moored. The coordinates were transformed to Universal Transverse Mercator (UTM) format, which allows calculating distances between points using the Euclidean formula [38]. Then median longitudes and latitudes were calculated for each group so that each mooring visit was represented by just one set of coordinates. The thought process for this approach was that without doing this mooring events would have various amounts of AIS messages based on how long they stayed at the wharf. So, for example, a ship staying in an area for 10 days would have ten times the weight compared to a ship staying for one day. An additional benefit of this approach is that by doing this it essentially labels as noise ships from the data set that have their AIS transponders transmitting moored status during the journey. In this case, this approach would produce coordinates in the middle of the sea, which density-based clustering algorithms would most likely classify as noise, thus eliminating AIS data points that are mislabeled quite effectively from the clustering.

In addition to the median coordinates, the training data also included the ship type, length, width and mean heading of the data point. The ship type, length, and width attributes were derived by taking the median of the values in the group. These values are included in the AIS transponder during installation and it is not expected that a given ship changes these values while operational [24]. However, since the MMSI that is used to identify ships is not unique in the sense that it can be reissued for a new ship, it is possible these values are not static. The likely hood for this is relatively low, since changing of MMSI is a relatively rare occurrence and in this case, both the old ship and the new ship have to be in the same geographical area in the same time frame.

The mean heading was calculated with the Yamartino method. Since the heading value is circular, the regular median calculation does not produce correct results. The formula for the Yamartino method is depicted in equation 3.5

$$s_\alpha = \frac{1}{n}\sum_{n}^{i=1} sin(\theta_i), \tag{3.6}$$

$$c_\alpha = \frac{1}{n}\sum_{n}^{i=1} cos(\theta_i), \tag{3.7}$$

$$\theta_\alpha = arctan(c_\alpha, s_\alpha) \tag{3.8}$$

The data sets contain duplicate points for AIS messages where the coordinates were exactly the same. The accuracy of the coordinate measurements is six decimals, meaning an accuracy of 10 cm. While it is unlikely that two different ships produce exact same coordinates while moored due to the different locations of the AIS transmitter, it is feasible that a single ship moors to the same berth multiple times with the given accuracy. The clustering process itself can work with duplicate data, but some of the clustering metrics fail if a cluster has only duplicate values. The density-based cluster validation index fails if there is no deviation between points in a cluster since this creates a division by zero situation.

This situation can be prevented by either removing all the duplicate values from the data or introducing a small amount of noise to the data sets to ensure that there are no duplicate points. The removal of duplicate points can cause the optimization algorithm to select a too-small minimum points parameter while adding noise to the data can have an effect on the selection of the epsilon parameter. However, the all-or-nothing approach of removing duplicates from the data would affect the results more profoundly than introducing little Gaussian noise to the data. Furthermore, the noise could be added to the data after the clustering, thus not affecting the clustering process. Adding a Gaussian noise with 0 mean and 0.1 standard deviation changes the DBCV index by about 0.04 percent, which most likely will not affect the ranking order of potential solutions. So when testing for the DBCV score, a Gaussian noise of 0 mean and 0.1 standard deviations was added to the data set to ensure that the index score is calculated.

Lastly, each of the data sets was filtered by a geospatial bounding box to include only points that are within a certain distance from the port. This step was vital in ensuring that the clustering concentrates only on the port area. While clustering is possible to do with lots of outlier points, initial results indicate that having lots of points outside the port area affects the validation metrics i.e. the fitness function of the optimization in adverse ways. These bounding boxes were created manually by taking coordinates from Google Maps services.

3.6 Genetic algorithm

Genetic algorithms are a group of computational models that are modeled after natural selection [57]. The purpose of these algorithms is to find the optimal solution to a given problem in a process that mimics evolution. The original concept of the genetic algorithm was proposed by John Holland in 1973, but since then multiple variations of this approach have emerged [57, 23]. The term itself has also broadened since its conception and nowadays any population-based optimization model is referred to as a genetic algorithm, even though these might not be based on Holland's work. Compared to other optimization methods, genetic algorithms have a better chance of finding the global optimum of a function if multiple local optimums are present [4].

The basic concept of the genetic algorithm requires a problem encoding and an evaluation (or fitness) function [57]. Problem encoding is presenting the problem with given parameters, also known as genes, in a way that produces a result that can be evaluated. The evaluation function takes the result of the problem encoding and gives it a numerical value that can be compared with other permutations of the genes given to the problem encoding. The evaluation metric can be either the result of the problem encoding or a separate function that evaluates the results of the encoding. Most use cases of genetic algorithms try to minimize or maximize non-linear problems, since finding the optimal solution to linear problems is much easier and can be done with simpler tools such as binary search.

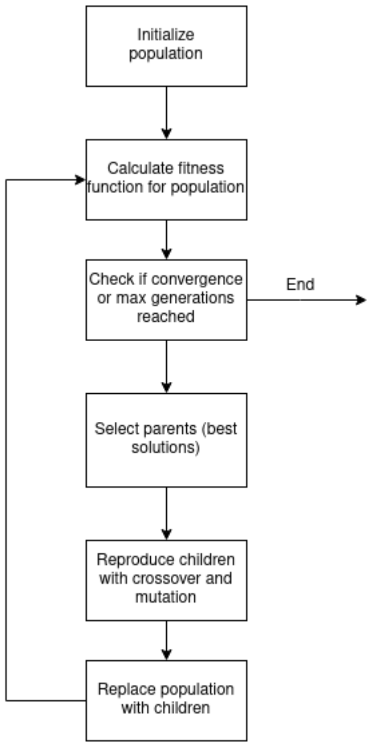
Initialize population
Calculate fitness function for population
Check if convergence or max generations reached
End
Select parents (best solutions)
Reproduce children with crossover and mutation
Replace population with children

Figure 3.1: An overview of the process of genetic algorithm

The genetic algorithm works on a population consisting of some solutions where the population size is the number of solutions. Each individual solution has a gene, which is represented as a set of parameters. The genes are built from the chromosomes. Each chromosome has a set of values, which are either continuous or discrete values. For example, if a genetic algorithm would have chromosomes {1,2} and {3,4}, the possible permutations for the genes would be (1,3), (1,4), (2,3), and (2,4). Usually, a genetic algorithm does not test all the available permutations of genes, but instead, it selects the best parameters based on the performance of previous populations. Since the genetic algorithm solves optimization problems by introducing randomness to the selection process, it is likely to find the global optimum even in situations where multiple local optimums exist. Given enough generations and a good mutation rate, the genetic algorithm is very likely to converge on the global optimum. [4]

Selection of the best solution is based on their fitness function value [57]. This is used to generate a so-called mating pool where the individual with a higher fitness value has a higher probability of being selected for the next iteration. This eliminates bad solutions from the gene pool. The individual solutions in the mating pool are called parents. Two parents will produce two offspring for the next iteration. The genes of the offspring can differ from the parents' genes based on a mutation chance. The process of selecting the best solutions for breeding is repeated until a predetermined number of generations has been reached, or when the evaluation metric converges to a minimum value.

Multiple strategies can be implemented on how the genetic algorithm chooses the genes to breed the next generation [4]. The most used selection methods are roulette wheel selection, rank selection, steady state selection, and tournament selection. In the roulette wheel selection, each individual is given a chance of selection for the breeding proportional to their fitness, the better the fitness is, the higher chance for that individual to be chosen for breeding. Then choosing the next generation is like spinning a roulette wheel, where the size of the slots depends on the probability of the genes. Rank selection is mostly used in cases where the population has very close fitness values. Rank selection works such as roulette wheel selection, but the probability is derived from the rank of the solution instead of the value of the fitness function. In the steady-state selection method, only the worst-performing genes are removed from the gene pool. In the tournament selection method, the genes are made to compete against each other and the winner of each tournament is selected for the next crossover.

The genetic algorithms are classified as weak methods since it makes few assumptions on the underlying problem they are trying to solve [57]. This makes the genetic algorithm approach a general optimization tool that can be used for a variety of problem cases. However, this also means that the genetic algorithm might not be the best tool for an optimization problem in some cases if specialized optimization methods are available for the given problem.

4. Experiment

In this chapter, the experimental setup is explained. In the experiment, different CVIs and other validation metrics were tested to see how they perform with various AIS data sets. The data sets used in this experiment are described in more detail in section 4.1. The experiment was done by optimizing the minimum points and epsilon hyperparameters using a genetic algorithm that used the DBSCAN algorithm to cluster the data points and evaluated the clusterings by using a validation metric as the fitness function. The purpose of this optimization was to find clusters that align well with the wharves of a given port. The DBSCAN was executed using both the Euclidean distance metric as well as the custom distance metric that is introduced in Section 4.2. This custom distance metric evaluated the data points based on the distance of the points as well as whether the heading of the points was towards the same direction.

The metrics tested in the experiment with the Euclidean DBSCAN were the silhouette index, Calinski-Harabasz index, Davies-Bouldin index, and DBCV index. These were chosen, since they, apart from DBCV, were also used by Ditton et al. [12] in their cross-validation optimization framework. Furthermore, all of these metrics are available from python libraries and are implemented in a way that optimizes run time [39].

DBCV and silhouette index scores can be calculated using custom distance metrics, allowing them to be used in the custom metric implementation. In addition to these metrics, the optimization was also done using the custom metric in addition to taking the shape of the cluster into consideration. In this case, the shape was measured with the length/width ratio of the cluster, with the length taken from the longer side of the cluster and the width from the shorter. The fitness function for a given clustering was then calculated by taking a weighted sum of all these values from the clusters and multiplying this value with either the silhouette index score or the DBCV score of the whole clustering. The weighted sum was calculated using Formula 4. In the equation $L(c_n)$ and $W(c_n)$ are the length and width of a given cluster, $|c_n|$ is the number of points in the given cluster, and $|c_{total}|$ is the total number of points in the data set.

$$\sum_{n=0}^{n_c} (L_{c_n}/W_{c_n}) * (|c_n|/|c_{total}|) \tag{4.1}$$

In addition to these metrics, the optimization was done using just the length/weight ratio as the fitness function. This was to test whether the inclusion of the validation metrics in the fitness function is beneficial, or could the optimization be done without costly validity index calculations.

4.1 Data sets

The experiment was done on AIS data collected from seven different harbor areas. The data was collected from the port of Brest, France; Ponta de Madeira, Brazil; Hedland, Australia; Dampier, Australia; Long Beach, USA; Antwerpen, Belgium, and Rotterdam, Netherlands. The used data sets are depicted in Table 4.1. These port areas were chosen to reflect different characteristics found in port areas. Brest is a general-purpose port that has relatively low traffic with both tanker and cargo traffic and is also a major ship maintenance hub for large ships. Ponta de Madeira, Hedland, and Dampier ports are major coal ports that have very uniform traffic and simple pier infrastructure. Long Beach harbor is a major container cargo port, which has a more complex infrastructure but relatively uniform traffic. Antwerpen and Rotterdam harbor areas are general-purpose ports that host a very complex riverside infrastructure with a maze-like layout where ships are moored very close to each other.

The Brest data set is open source and available to anyone in a Zenodo repository [42]. The other data sets are owned by NAPA Ltd. and they are not available online. An aggregate of these data sets was formed under the supervision of a NAPA employee according to the steps outlined in section 3.5. The raw data for these data sets are acquired through commercial means and thus they can not be shared online.

	Number of data points	**Timeframe**	**Shiptypes**
Brest	253	183 days	Cargo, Tanker
Ponta de Madeira	187	180 days	Cargo, Tanker, Other
Hedland	1181	181 days	Cargo, Tanker, Other
Dampier	729	181 days	Cargo, Tanker, Other
Long Beach	1588	181 days	Cargo, Tanker, Other
Antwerpen	4510	181 days	Cargo
Rotterdam	7314	181 days	Cargo

Table 4.1: Datasets used in the experiment

4.2 Custom distance metric for DBSCAN

The experiment used a custom distance metric, that takes into account the way the bow of the ship is facing. The heading value is available from the AIS data and since this value is automatically gathered from the GPS data, it is usually reliable if the transmitter is installed correctly. The custom distance matrix used in this experiment was calculated by first calculating the Euclidean distances between points and then inserting a penalty score to the distances if the headings of the data points have an angular distance of over 15 degrees. The penalty score was set up as a value greater than the epsilon to ensure that data points with deviating angular differences are not considered to be in the same neighborhood. Since ships can usually moor into a given pier in two directions, opposite directions need to be treated as equals, i.e. the angular difference between 90 and 270 is calculated as 0 degrees. The equation used to calculate the angular distance between two data points is depicted in Equation 4.2.

$$min(|(\alpha_1 - \alpha_2 + 180) \mod 360 - 180|, |(\alpha_1 - \alpha_2) \mod 360 - 180)|) \tag{4.2}$$

The use of the custom distance matrix allows the DBSCAN algorithm to distinguish different berths from each other even when they are geographically close to each other. An example of how the distance matrix takes into account the penalties is given in Figure 4.1 and Table 4.2.

4.3 Genetic algorithm implementation

This experiment utilized a genetic algorithm as the optimizer function. This approach was chosen since the genetic algorithm is considered to be a fast and effective way

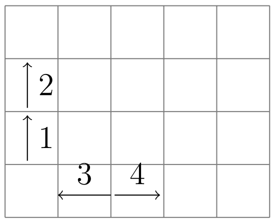

Figure 4.1: Example data of 4 data points with headings

	Point 1	**Point 2**	**Point 3**	**Point 4**
Point 1	0	1	$\sqrt{2}+\mathcal{E}$	$\sqrt{3}+\mathcal{E}$
Point 2	1	0	$\sqrt{3}+\mathcal{E}$	$2\sqrt{2}+\mathcal{E}$
Point 3	$\sqrt{2}+\mathcal{E}$	$\sqrt{3}+\mathcal{E}$	0	1
Point 4	$\sqrt{3}+\mathcal{E}$	$2\sqrt{2}+\mathcal{E}$	1	0

Table 4.2: Distance matrix of points in Figure 4.1. The $\mathcal{E}$ value denotes a large penalty score.

to deal with optimization problems with lots of local optimums [4]. In search for the hyperparameters for the DBSCAN algorithm, it is very likely that a local optimum is encountered since DBSCAN relies on two parameters. Furthermore, two different sets of parameters can produce the exact same clustering, further impeding the search for a global optimum. This made the genetic algorithm an optimal tool for solving this problem.

The genetic algorithm works by choosing potential solutions from genes formed from chromosomes. In this experiment, the chromosomes consisted of different values of the minimum samples and epsilon hyperparameter in a given interval. The chromosome for minimum samples consisted of values between 5 and 100 and the chromosome of epsilon contained values between 25 and 800. Based on combinations of the values in the chromosomes, the total number of genes in the gene pool was 73625.

The fitness function of the genetic algorithm is depicted in Figure 4.2. The process of each optimization step began by calculating the DBSCAN clustering using the given genes. If the custom distance matrix was used, the DBSCAN algorithm used it as the training data, otherwise, it used the data points and calculated the distances during execution. An overview of this process is depicted in Figure 4.3. For non-shape implementation, the algorithm calculated the CVI based on the clustering and then returned the score or the negation of the score in the case of the Davies-Bouldin index. This is because the closer the Davies-Bouldin index is to zero, the better the score [11]. For the shape implementation, after the clustering was calculated, polygons were created from the dataset, and length/weight ratios for each cluster polygon were

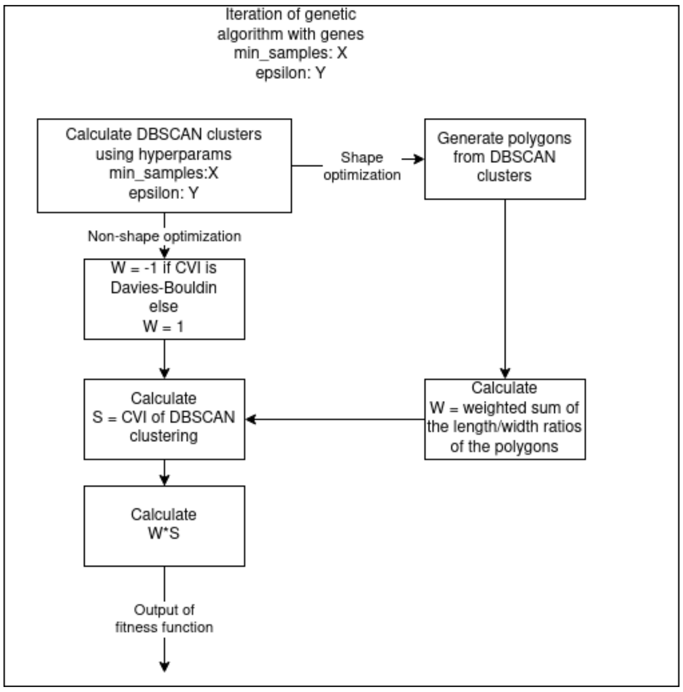
Iteration of genetic algorithm with genes min_samples: X epsilon: Y
Calculate DBSCAN clusters using hyperparams min_samples:X epsilon: Y
Shape optimization
Generate polygons from DBSCAN clusters
Non-shape optimization
W = -1 if CVI is Davies-Bouldin else W = 1
Calculate S = CVI of DBSCAN clustering
Calculate W = weighted sum of the length/width ratios of the polygons
Calculate W*S
Output of fitness function

Figure 4.2: Fitness function calculation for the genetic algorithm

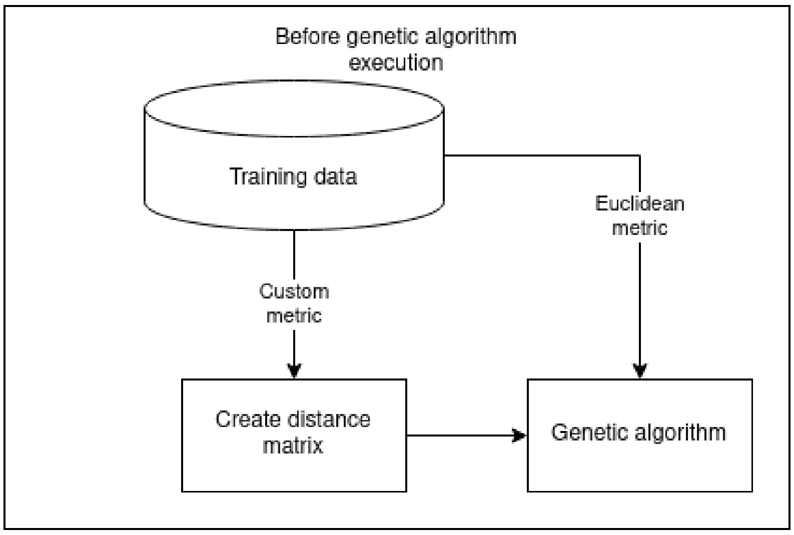

Figure 4.3: Data preparation for genetic algorithm

calculated. Then the weighted sum of these ratios was calculated so that clusters with more points were given more weight. After this, the CVI was calculated from the clustering and the weighted sum was multiplied by the CVI score.

The genetic algorithm optimization was done by selecting the potential solutions using the tournament selection method. The tournament selection works by selecting individuals randomly, making them compete with each other, and picking the fittest of them for breeding, thus ensuring optimal solutions. The tournament selection method is effective in eliminating the weak candidates from the gene pool since these genes are easily discarded when they perform poorly against stronger candidates [36]. This ensures that the solution was likely to converge on the global optimum.

The stop criteria of the genetic algorithm was defined with a saturation threshold of 20. This meant that if the optimization did not improve the solution within 20 generations, the function determined it to have converged to the optimal value. This step was used to mainly save time on the execution.

4.4 Software libraries

The experiment was done on the python programming language, version `3.8.10`. Most of the code was run in the jupyter lab version `3.1.1` environment. The data was stored in data frames using `Pandas` version `1.3.0` for normal data frames or `GeoPandas` version `0.10.2` for GIS data frames. Data from R was parsed to python using the `pyreadr` version `0.4.4` package. `NumPy` version `1.21.4` package was used for array processing and mathematical functions. Clustering was done using the OPTICS and DBSCAN methods available in `scikit-learn` version `1.0.2` and HDBSCAN method available at `HDBSCAN` version `0.8.28`. The metric functions were also provided by the scikit-learn package. The genetical algorithm was implemented via the `PyGad` version `2.17.0` package. Polygon representations of points were created using the Shapely version `1.8.0` package and the visualization of the data was done with `ST_VISIONS` package [51].

4.5 Evaluation metrics

The lack of objective tools to evaluate clusterings makes the results challenging to interpret objectively. Mainly the lack of ground-truth data makes interpreting whether the clusters are the correct size and shape and located in the right area impossible to parse from the clusters alone. While there exist some good rule-of-thumb approaches based on simple statistics to evaluate the goodness of clusters, these are not universal truths [21]. In this thesis, the evaluation of clusters produced by the DBSCAN algorithm using the hyperparameters gained from the optimization process was done using these statistics as well as human-based visual inspection of the clusters.

4.5.1 Computational validation

The goal of the computational validation was to evaluate whether the optimized hyperparameters produce good clusterings. While evaluating whether the clustering of points is optimal is rather difficult, there are a couple of good expectations that would indicate a bad clustering produced by the DBSCAN algorithm. Firstly, the number of noise points should not exceed 30% of the data. However, if there is no noise in the data there could be signs of overfitting. A larger amount of noise would implicate that the epsilon parameter is too small. On the other hand, a too-large epsilon can cause most of the data to be situated in only a few clusters. Secondly, data points should not be clustered into a single cluster. Generally speaking, if 20% to 50% of the data is clustered into a single cluster this would indicate that the epsilon parameter is too large. [45]

The computational evaluation of the clusterings focused on evaluating the results with the above-mentioned constraints in mind. The analysis focused on the number of outliers in the clustering, the total number of clusters, and the size of the largest cluster. In addition, the evaluation also analyzed the CVI score metrics of the optimization process themselves in an objective way. Since most relative CVIs have a limited interval of what the value can be, i.e. [-1, 1] for the silhouette score, it is relatively easy to see if the clustering is good according to the metric.

4.5.2 Human based validation

The clusters were also evaluated visually by inspecting them when they are visualized over a map. This was done by first generating polygons of the points in each cluster and then overlaying these polygons into the visualization tool. The evaluation was done by inspecting the size and shape of the clusters as well as whether the formed clusters intercepted major land areas. The size attribute should be large enough to hold multiple individual berths in the cluster. Ideally, the shape attribute should be rather oblong, so that the cluster only encompasses a single pier. Furthermore, the clusters should not encompass any major land areas and should be confined to water.

5. Results

The next section presents the results of the experiments. The first section contains the results of the optimization of the hyperparameters of the DBSCAN algorithm for both data sets. It also has a visual examination of the formed clusters according to the steps outlined in section 4.5.2.

5.1 Statistical analysis of optimization results

The genetic optimizer as described in Section 4.3 was executed using the data sets described in Section 4.1. The results of the experiment are shown in Table 5.1 Each row represents the results of the optimization for each cluster validation method for each data set. The min samples and epsilon values represent the optimal hyperparameters found in the optimization process and the score value is the result of the fitness function generated by these hyperparameters.

Analyzing the results just based on the CVI score metrics gives some indication of what metrics are best for the optimization task. Given that the silhouette index score is a value between -1 and 1, the general goodness of the index score was easy to estimate. The silhouette index score for the small datasets performed well and these values can generally be thought of as being indicative of good clusterings [21]. The Hedland data set performed especially well, with a silhouette index score of 0.94. The Rotterdam and Antwerpen data sets performed mediocre with a score of about 0.5, which is indicative that some of the clusters in these clusterings are not separated from each other or that the clusters are not dense enough.

	Brest	**Ponta de Madeira**	**Hedland**	**Dampier**	**Long Beach**	**Antwerpen**	**Rotterdam**
Silhouette score	min_samples: 57.0 eps: 746.0 score: 0.808	min_samples: 8.0 eps: 87.0 score: 0.870	min_samples: 7.0 eps: 74.0 score: 0.948	min_samples: 14.0 eps: 384.0 score: 0.880	min_samples: 6.0 eps: 86.0 score: 0.792	min_samples: 94.0 eps: 749.0 score: 0.515	min_samples: 11.0 eps: 603.0 score: 0.542
Calinski-Harabasz score	min_samples: 63.0 eps: 565.0 score: 1035	min_samples: 15.0 eps: 433.0 score: 695	min_samples: 17.0 eps: 212.0 score: 16040	min_samples: 19.0 eps: 595.0 score: 4146	min_samples: 40.0 eps: 654.0 score: 1639	min_samples: 13.0 eps: 785.0 score: 5497	min_samples: 16.0 eps: 788.0 score: 11757
Davies-Bouldin score	min_samples: 30.0 eps: 608.0 score: 0.289	min_samples: 23.0 eps: 561.0 score: 0.312	min_samples: 27.0 eps: 162.0 score: 0.206	min_samples: 68.0 eps: 67.0 score: 0.460	min_samples: 7.0 eps: 379.0 score: 1.07	min_samples: 51.0 eps: 21.0 score: 0.729	min_samples: 27.0 eps: 232.0 score: 1.189
DBCV score	min_samples: 19.0 eps: 377.0 score: 0.878	min_samples: 5.0 eps: 48.0 score: 0.898	min_samples: 11.0 eps: 53.0 score: 0.963	min_samples: 9.0 eps: 66.0 score: 0.851	min_samples: 5.0 eps: 31.0 score: 0.821	min_samples: 8.0 eps: 78.0 score: 0.498	min_samples: 7.0 eps: 29.0 score: 0.554
Custom metric +silhouette index score	min_samples: 12.0 eps: 696.0 score: 0.822	min_samples: 5.0 eps: 313.0 score: 0.950	min_samples: 6.0 eps: 178.0 score: 0.950	min_samples: 10.0 eps: 367.0 score: 0.892	min_samples: 5.0 eps: 122.0 score: 0.823	min_samples: 16.0 eps: 692.0 score: 0.546	min_samples: 5.0 eps: 326.0 score: 0.608
Custom metric + DBCV	min_samples: 5.0 eps: 796.0 score: 0.917	min_samples: 9.0 eps: 290.0 score: 0.924	min_samples: 9.0 eps: 74.0 score: 0.963	min_samples: 8.0 eps: 65.0 score: 0.892	min_samples: 9.0 eps: 51.0 score: 0.799	min_samples: 37.0 eps: 339.0 score: 0.600	min_samples: 24.0 eps: 346.0 score: 0.673
Custom metric + silhouette index score and shape	min_samples: 14.0 eps: 86.0 score: 7.008	min_samples: 6.0 eps: 509.0 score: 9.563	min_samples: 13.0 eps: 379.0 score: 12.022	min_samples: 8.0 eps: 304.0 score: 6.220	min_samples: 11.0 eps: 337.0 score: 9.569	min_samples: 7.0 eps: 109.0 score: 5.337	min_samples: 5.0 eps: 82.0 score: 5.039
Custom metric + DBCV and shape	min_samples: 10.0 eps: 86.0 score: 8.317	min_samples: 10.0 eps: 537.0 score: 7.666	min_samples: 22.0 eps: 341.0 score: 8.209	min_samples: 5.0 eps: 335.0 score: 6.246	min_samples: 6.0 eps: 341.0 score: 9.337	min_samples: 7.0 eps: 101.0 score: 9.517	min_samples: 7.0 eps: 82.0 score: 6.187
Shape	min_samples: 26.0 eps: 125.0 score: 14.724	min_samples: 12.0 eps: 497.0 score: 10.429	min_samples: 63.0 eps: 384.0 score: 15.737	min_samples: 88.0 eps: 324.0 score: 10.183	min_samples: 23.0 eps: 430.0 score: 14.083	min_samples: 7.0 eps: 93.0 score: 16.820	min_samples: 9.0 eps: 79.0 score: 10.956

Table 5.1: Results of optimizing DBSCAN algorithm with a genetic algorithm using various validation metrics as fitness function

For the Calinski-Harabasz index score, the evaluation was much harder since the score does not have an upper limit. Generally speaking, the score should be maximized as the higher value of the index means the clusters are dense and well separated. However, this score seems to be somewhat contradictory to the silhouette index score. While the best Calinski-Harabasz score was generated by the Hedland data set, which also generated the best silhouette score, Rotterdam and Antwerpen data sets also generated large Calinski-Harabasz values even though they performed poorly with the silhouette score. Conversely, Ponta de Madeira data set, which performed well with the silhouette index score had the lowest Calinski-Harabasz index score. This would indicate that these two metrics might not correlate with each other.

The Davies-Bouldin optimization produced results that were in line with the silhouette score. The Davies-Bouldin is easier to evaluate compared to the Calinksi-Harabasz score since while it does not have an upper limit to the values, the lowest value is 0. Since the score is meant to be minimized, scores closest to zero are considered optimal. The smallest value was generated by the Hedland data set, which also performed well with other metrics. Also, other data sets that performed well with silhouette scores produced good Davies-Bouldin scores. The only exception for this is the Long Beach data set, which produced a good silhouette score, but the second worse Davies-Bouldin score.

The DBCV metric also performed similarly to the silhouette score. Both the silhouette and DBCV scores have a range of -1 to 1, with 1 indicating optimal clustering, which makes comparing these two metrics straightforward. Again, the large data sets (i.e. Antwerpen and Rotterdam) produced mediocre results (≈ 0.5), while the rest produced good scores (> 0.8). When optimizing the data sets with either the silhouette index or DBCV index using the custom metric, all data sets produced better scores compared to those produced by the Euclidean metric. This result would indicate that the custom metric is overall beneficial to the optimization process.

Analyzing the custom metric and the shape implementation of the optimization is hard to do by just looking at the scores. Comparing the results with each other is even more difficult than when using just the metrics. Each value is just the best result that can be found for a given data set and comparing the values between data sets might not be beneficial. Furthermore, since the length/width ratio is taken from a clustering as a weighted sum and the use of validation score as a scalar, comparing these values using statistical methods is too complex. Thus a more hands-on approach is needed to analyze the results. In the next chapter, the clusters that are created with the optimal hyperparameters for each metric/data set pair are evaluated visually.

In addition to the CVI score metrics, the ratio of outliers compared to points in clusters was also of interest. The percentage of outliers per data set for each cluster

	Brest	**Ponta de Madeira**	**Hedland**	**Dampier**	**Long Beach**	**Antwerpen**	**Rotterdam**
Silhouette score	13.8%	11.7%	1.6%	1.9%	6.0%	11.8%	2.1%
Calinski-Harabasz score	13.8%	0.0%	1.3%	1.2%	5.2%	0.8%	1.8%
Davies-Bouldin score	13.8%	0.0%	1.6%	90.2%	1.5%	97.6%	12.1%
DBCV score	4.0%	7.5%	2.1%	11.9%	9.3%	14.9%	26.9%
Custom metric + silhouette score	10.3%	0.0%	0.4%	3.4%	3.8%	5.4%	4.4%
Custom metric + DBCV	5.5%	0.0%	1.7%	9.2%	13.0%	19.4%	13.5%
Custom metric + silhouette score and shape	25.3%	0.0%	0.1%	3.5%	3.2%	10.6%	11.0%
Custom metric + DBCV and shape	20.9%	0.0%	1.3%	3.4%	1.8%	11.9%	14.0%
Shape	50.6%	0.0%	0.1%	60.8%	12.6%	12.6%	16.9%

Table 5.2: Percentage of outliers DBSCAN produces for each method

	Brest	**Ponta de Madeira**	**Hedland**	**Dampier**	**Long Beach**	**Antwerpen**	**Rotterdam**
Silhouette score	86.2%	15.0%	8.0%	28.7%	6.4%	31.9%	31.1%
Calinski-Harabasz score	86.2%	44.9%	9.7%	28.7%	30.7%	42.4%	43.8%
Davies-Bouldin score	86.2%	95.2%	9.7%	10.4%	15.1%	1.3%	17.9%
DBCV score	86.166%	15.0%	8.0%	11.9%	6.4%	13.0%	3.4%
Custom metric + silhouette score	56.5%	27.3%	8.0%	26.9%	8.7%	36.0%	12.3%
Custom metric + DBCV	56.5%	27.3%	8.0%	11.5%	6.4%	21.8%	12.3%
Custom metric + silhouette score and shape	25.7%	27.3%	16.5%	26.9%	8.8%	12.9%	5.4%
Custom metric + DBCV and shape	25.7%	27.3%	16.5%	26.9%	8.8%	12.9%	5.4%
Shape	23.715%	27.3%	16.5%	26.9%	18.8%	12.9%	5.1%

Table 5.3: Percentage of data points in the largest cluster

	Brest	Ponta de Madeira	Hedland	Dampier	Long Beach	Antwerpen	Rotterdam
Silhouette score	1	11	21	10	72	6	16
Calinski-Harabasz score	1	3	17	7	5	4	12
Davies-Bouldin score	1	1	17	1	22	2	38
DBCV score	2	12	21	15	91	103	193
Custom metric + silhouette score	4	8	19	13	63	27	90
Custom metric + DBCV	5	7	21	18	70	21	44
Custom metric + silhouette score and shape	5	5	8	13	32	94	155
Custom metric + DBCV and shape	6	5	10	14	34	95	130
Shape	2	5	8	2	19	102	120

Table 5.4: Number of clusters

validation method is depicted in Table 5.2. The outliers are points that have not been assigned any cluster membership with the DBSCAN algorithm. It should be noted that the data sets most likely contained data points that should be classified as outliers, but this percentage should be relatively low since the preprocessing step is used to filter out unusual AIS messages.

While it is estimated that about 80% of AIS messages have some data errors, most of the time these are in the static messages that are inputted by the human crew[22]. The dynamic messages are less prone to errors especially and these errors are usually related to lost messages [31]. However, a study done by Harati et al. showed that the navigational status column in dynamic messages has as much as 30% incorrect values [22]. The study didn't specify which status codes were most often incorrectly reported. The detection of incorrect mooring status was done by detecting messages that reported significant speed and the mooring status. Since the data preprocessing step removes messages that are reporting moored status and speeds larger than one knot, it is most likely that the training data has less than 30% of outlier data. Furthermore, the data sets are filtered by a bounding box around the harbor area, further reducing the number of outliers. However, some of the points classified as outliers could be in fact valid points, which were not able to be added to clusters based on the hyperparameters.

The Davies-Bouldin index performed surprisingly badly in some datasets on this metric. The clustering of the Antwerpen data set produced almost 98% outliers while the Dampier data set produced 90% outliers. These are obviously non-ideal solutions for the optimization process and since clusterings for Antwerpen and Dampier produced good Davies-Bouldin scores (0.72 and 0.46 respectively), this would suggest that the Davies-Bouldin index is not universally suited for the optimization task. In most data sets the Calinski-Harabasz index produced clusterings that had the lowest amount of outliers. This is most likely due to the Calinski-Harabasz index favoring large clusters, which might not be reflective of the ground truth. So while analyzing the number of outliers can allow us to gain some insights into how the metrics perform, a more thorough visual inspection is needed to evaluate how good the overall process is.

Another metric to analyze is the percentage of data points that the largest cluster holds. A good rule of thumb for most density-based clustering is that a single cluster should not hold more than 50% of the data points [45]. The percentage of data points in the largest cluster for each optimization metric is depicted in Table 5.3, and the number of clusters per clustering is depicted in 5.4. These values also count the outlier points to the total size of the data set. Based on these results, the performance of the unmodified validation metrics is inconsistent across several data sets. For example, while the Hedland data set performs well with all validation metrics, some data sets produce results that indicate that the clustering is non-ideal. Some clusterings produced results where over 50% of values are in a single cluster, indicating that the epsilon value is too large. In some cases, these kinds of clusterings produced a single cluster. In these cases, the clusterings do not reflect what we want out of the harbor area, since it is expected that it can be divided into several subareas. In these cases, the high percentage results from the low number of clusters, but some optimizations resulted in skewed distributions of data points despite a relatively high number of clusters. For example, Calinski-Harabasz index score optimization for the Rotterdam data set produced 12 clusters, but the largest cluster contained almost 44% of the data.

Out of the metrics using custom distance calculations, the silhouette index score and DBCV without shape constraints performed well across all data sets but Brest. The validation metrics with custom metrics and shapes performed well on all data sets. In the next section, we will visually evaluate the clusters created by these optimization methods.

Correlation between different cluster validation indices

The results obtained in the section 5.1 would indicate that the CVIs correlate with each other to some degree. For example, optimizations are done using Davies-Bouldin or Calinski-Harabasz metrics as fitness functions discovered very similar hyperparameters. Based on these observations, the Pearson correlation coefficient was calculated for multiple clusterings derived from the Brest data set.

The Pearson correlation coefficient was calculated with all clusterings of the Brest data with hyperparameters minimum points in a range between 5 and 100 and epsilon in a range between 10 and 800. This produced a total of 75050 clusterings to evaluate. Some of these clusterings had to be omitted from the analysis since they either produced a single cluster or no clusters at all, which are cases not handled by CVIs. This filtering narrowed the number of clusterings to 61408. Since the Davies-Bouldin index score differs from the other metrics in that a smaller value indicates better clustering, a negation is done on the values to make rankings between this index compatible with different metrics. The different CVIs produce the following correlations outlined in Table 5.5.

	Davies-Bouldin	**DBCV**	**Calinski-Harabasz**	**Silhouette**
Davies-Bouldin	1.0	-0.19	0.62	0.66
DBCV	-0.19	1.0	-0.41	0.15
Calinski-Harabasz	0.62	-0.41	1.0	0.62
Silhouette	0.66	0.15	0.62	1.0

Table 5.5: Pearsson correlation between the clustering metrics

Based on these results Calinski-Harabasz index, silhouette index, and Davies-Bouldin index correlate with each other, while DBCV does not seem to correlate with either the silhouette index or the Davies-Bouldin index. Calinski-Harabasz and DBCV seem to have a significant negative correlation. However, since this is a negative correlation and both of these metrics aim to maximize their values, this would indicate that these metrics prefer different clusterings.

Similarly to the correlation of the values of the scores, the order of these scores, i.e. if the same data set has the best score in all metrics, can be analyzed using the Kruskal-Wallis one-way analysis of variance [34]. This statistic test is a non-parametric method for testing whether samples originate from the same distribution. The test is used to determine if populations share the same order of values. In this case, it is used to detect if the scores for each data set are in the same order with each metric. The null hypothesis of the test is that the order of values is the same for each group. By analyzing all metrics with the Kruskal-Wallis test, the result is a test statistic of 19.02, which corresponds to a p-value of 0.0019. This means we can reject the null hypothesis and conclude that the ordering is not the same. However, if we test only the silhouette and DBCV index score with both Euclidean and custom metrics, the Kruskal-Wallis test produces a test statistic of 1.24, corresponding to a p-value of 0.74. In this case, we can not reject the null hypothesis.

5.2 Visual inspection of the clusters

In this section, the clusterings that performed best based on numerical analysis are evaluated visually. These clusters are mapped to satellite images of the port area and analyzed visually. The main focus of this analysis is to evaluate how well-aligned the clusters are with regard to the geographical shapes, such as berths and shorelines. Also, clusters are evaluated on how well they reflect the principle of just encompassing a single berthing area i.e. the clusters don't go over land or combine berths across the water.

Visualizing the clusters created by optimizing just the shape of the cluster reveals interesting insights. While some data sets produced similar results in all measures, Brest and Dampier data sets produced many outliers when optimizing by using only the shape. This is most likely due to the optimization process finding clustering with few optimal clusters and ignoring the rest of the data. For example, the Brest data set in Figure 5.1 shows two prominent clusters while leaving out most of the data points. The width/length ratio for both of these clusters is large, which makes the optimizer pick this solution. However the silhouette score for this clustering is only about 0.4, and if this value is used as a scalar for the length/width ratio, the value decreases substantially, and the optimizer can discard this solution as sub-optimal. It could be argued that the CVIs in this case act as a sort of fail-safe that ensures the quality of the clusters.

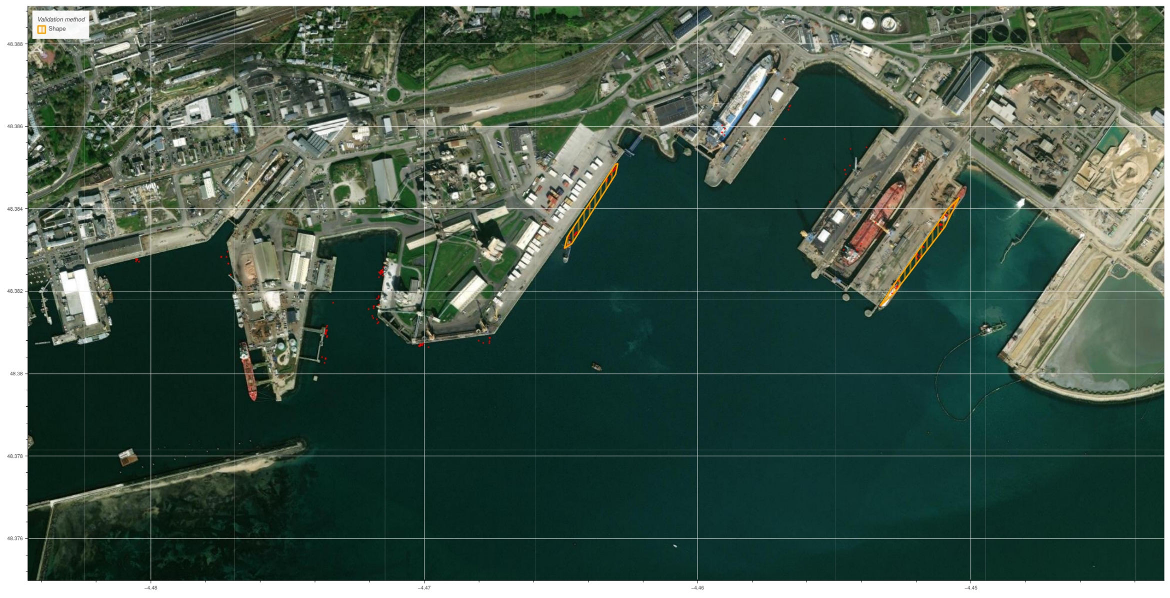

Figure 5.1: Clusters in Brest harbor area using only the shape in optimization

When analyzing the clusters created by optimizing with the length/width ratio and the CVI score, most of the data sets produced fewer clusters compared to just

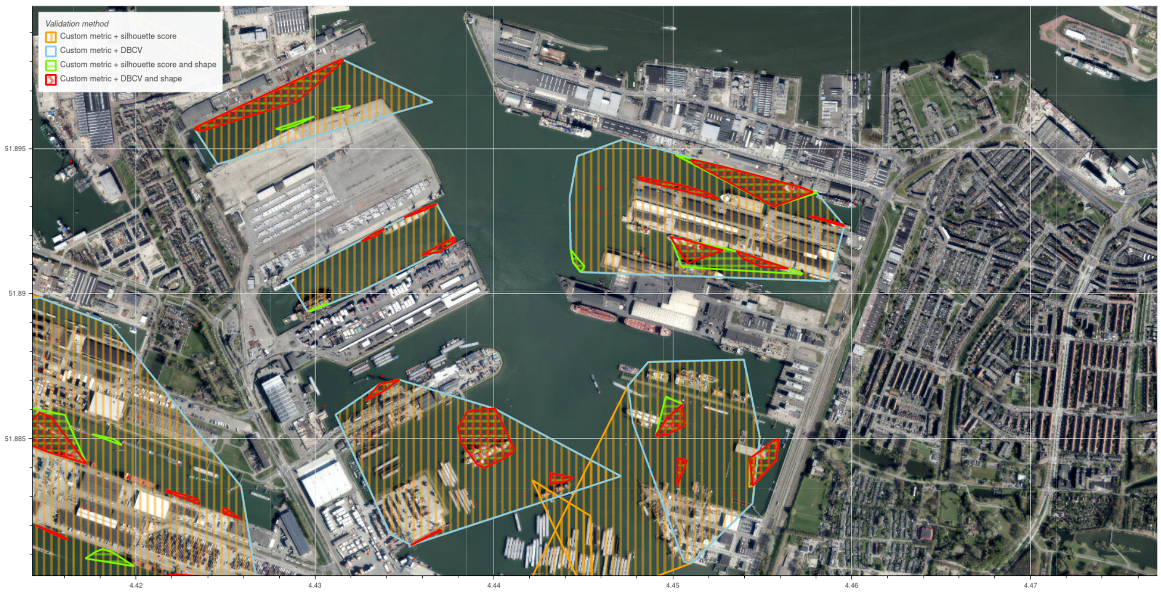
Validation method
Custom metric + silhouette score
Custom metric + DBCV
Custom metric + silhouette score and shape
Custom metric + DBCV and shape
51.895
51.89
51.885
4.42
4.43
4.44
4.45
4.46
4.47

Figure 5.2: Clusters in Rotterdam harbor area

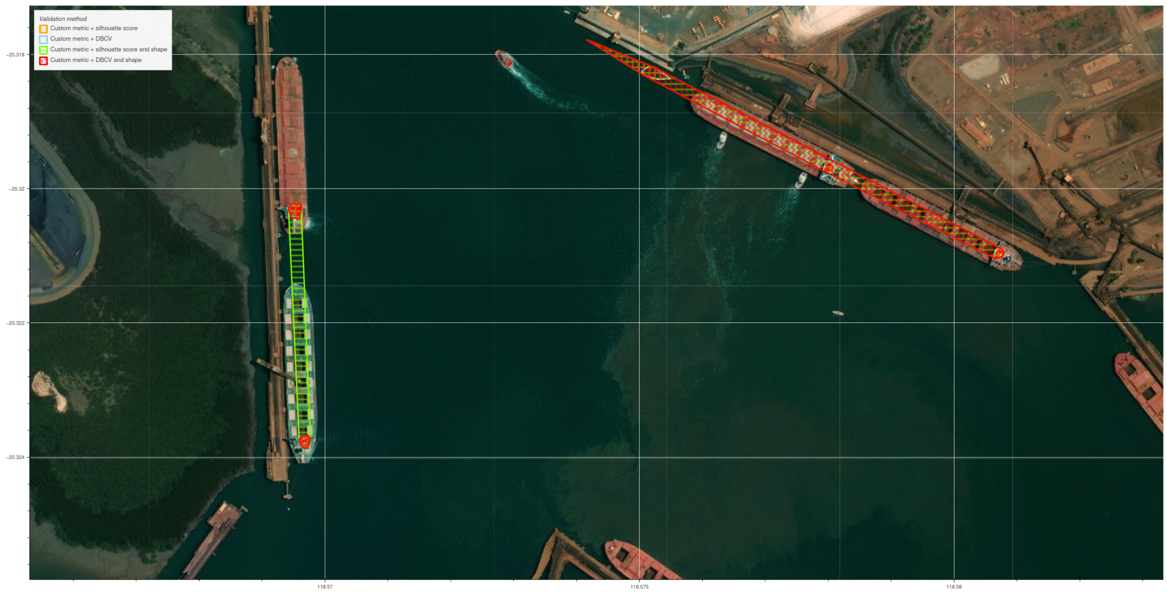
Validation method
Custom metric + silhouette score
Custom metric + DBCV
Custom metric + silhouette score and shape
Custom metric + DBCV and shape
-20.318
-20.32
-20.322
-20.324
118.57
118.575
118.58

Figure 5.3: Clusters in Hedland area

using the CVI score. This is most likely due to the shape implementation favoring long clusters, and thus combining two clusters whenever possible. The exception for this were the Antwerpen and Rotterdam data sets, which produced fewer clusters when the shape optimization was not utilized. This is most likely due to these port areas having the berth areas really close together, thus increasing the likelihood that two separate berths are clustered into a single cluster. Figure 5.2 illustrates some clusters found from the Rotterdam data set. From this picture, we can see that the metrics without shape constraints created larger clusters compared to the shape variant. The nonshape

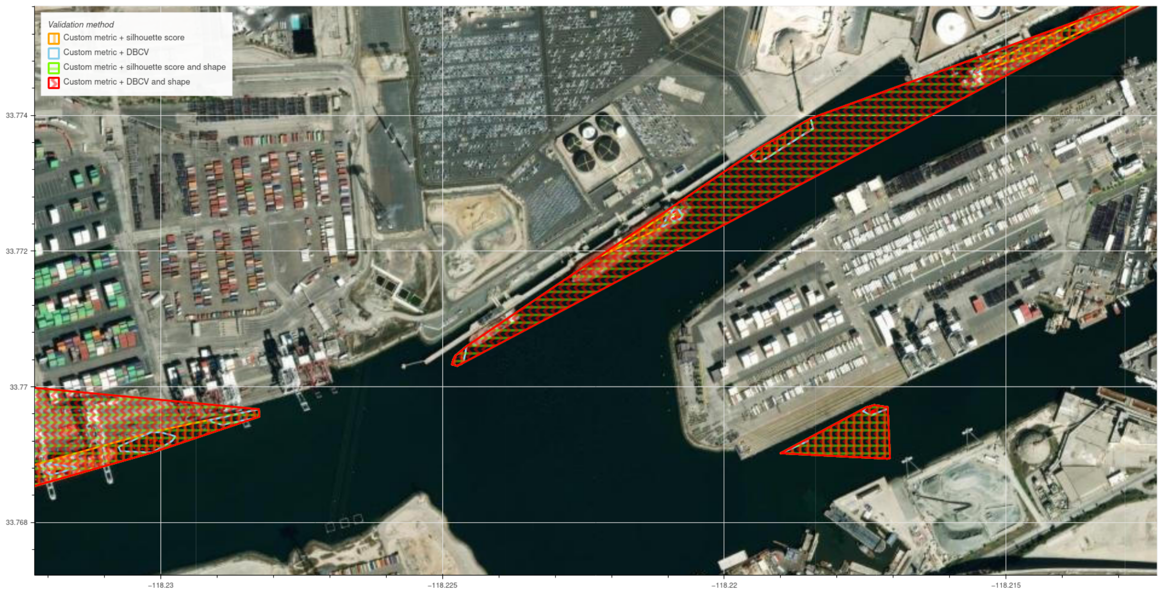
Validation method
Custom metric + silhouette score
Custom metric + DBCV
Custom metric + silhouette score and shape
Custom metric + DBCV and shape
33.774
33.772
33.77
33.768
−118.23
−118.225
−118.22
−118.215

Figure 5.4: Clusters in Long Beach area

variants created clusters that overlapped with each other most of the time. Similarly, the shape variants discovered similar clusters, even though the DBCV implementation divided these clusters in two in some cases and the silhouette index implementation found clusters in areas where the DBCV implementation did not. The silhouette index implementation tended to find longer clusters in other data sets as well as illustrated in Figure 5.3. The nonshape implementations went over both water and land when creating the clusters, while the shape implementations are constrained to one side of a pier. However, compared to the nonshape implementation, the shape implementations miss some points that could be considered valid wharves.

In data sets other than Antwerpen and Rotterdam, the shape implementations produced fewer clusters compared to the nonshape implementation. This is illustrated in Figure 5.4 with clusters generated from the Long Beach data set. In this image, the nonshape implementation produced clusters that are confined within the clusters created by the shape implementation. This is most likely due to the validation metrics favoring rather dense clusters. Similarly, the difference between the shape implementation using the silhouette index score and the DBCV index score can be attributed to DBCV favoring more dense clusters. Since the length/width ratio is calculated in a similar fashion in both cases, this means that wharves characterized by multiple berths are better identified using the silhouette index score shape implementation.

The shape implementations were capable of finding clusters that align with the wharves of the harbor much better than just using the custom distance metric. While there still are some clusters, where two parallel wharves are clustered together, most of the clusters were aligned with the wharves perfectly. In these cases the correct wharves were still included in the clusters, so the end user could extract the desired ground truth from these by manually dividing the clusters. Overall the shape implementation appeared to perform better in almost every situation, whether it be a dense and multipurpose port area like Rotterdam or a more specialized port like Hedland.

6. Discussion

The results of the optimization process presented interesting insights. The results would suggest that using cluster validation indices to evaluate the goodness of geospatial clustering might be a feasible way to find suitable hyperparameters. The CVIs were created to address the ambiguity related to evaluating clustering results, but interpreting the results is not always straightforward. The correct choice of the CVI and its implementation appears to be the key. While optimizing Davies-Bouldin or Calinski-Harabasz index seems to perform relatively well on some data sets, in some data sets, the optimization converges to hyperparameters that indicate bad clustering. For example, using Davies-Bouldin as the fitness function finds hyperparameters for Antwerpen that cluster the data in a way that leaves 97% of data as outliers. The Davies-Bouldin index score is relatively decent in this clustering, which questions whether this CVI is suitable for evaluating geospatial clustering. Similarly, Calinksi-Harabasz optimization converges to hyperparameters that cluster most points into a single cluster in some cases, which is also indicative of less-than-optimal clustering. The value of the Calinki-Harabasz index score appears to increase as the clusters get bigger, which raises the question of whether this CVI is good at evaluating geospatial clustering. Literature would suggest the Calinski-Harabasz index and Davies-Bouldin works best at evaluating partitional clustering [19].

On the other hand, silhouette and DBCV scores appeared to perform relatively well in all data sets except Brest. This bad performance is most likely due to the data in the Brest area being divided into two separate harbor areas, where the other area dominates with the most data points. Most likely extra filtering steps would have been needed for the Brest area. Overall DBCV favors small dense clusters compared to the silhouette index and thus tends to create the most clusters in all data sets. The performance of the silhouette index is less predictable. For example, optimizing the silhouette index score results in lots of small clusters in the Long Beach data set while only resulting in a few large clusters in the Antwerpen data set; even optimizing DBCV results in lots of clusters in both data sets. While the silhouette index and DBCV measure slightly different aspects of the clustering, comparing them to each other can give an overall view of the goodness of the approach, given that they share the range of values from -1 to 1. In all data sets except Antwerpen, the DBCV score was higher than the silhouette score, suggesting that DBCV might be better at evaluating geospatial clustering. This observation was also shared by the original authors of the DBCV score [37].

Using a custom distance metric improves the clustering in all data sets. This is the case in statistical analysis and visual inspection of the cluster. In the statistical analysis, the score improved in all cases, and the key indicators of good clusterings, such as the number of outliers, were more consistent. Adding the shape constraint to the fitness function improves the optimization even further. By all metrics, the implementation using custom distance metric DBSCAN and shape constraints produce the most desirable clusterings.

The shape constraint itself appears to be able to detect optimal hyperparameters in some cases. Still, in some cases, the shape constraint without the CVI seems to converge to a less-than-optimal solution. This was demonstrated to be due to the optimization process in some cases discovering clusterings where only a couple of clusters had a high length/width ratio. These incorrect trivial solutions were missing from the implementations that used the CVIs along the shape constraints. Thus it can be argued that CVIs can be used as a fail-safe that ensures that the optimization process discards these solutions. It appears that using CVIs together with other fitness function goals has a synergistic effect on the goodness of the found clusters.

The evaluation of the clusters was done using statistics that the literature indicates are characteristic of good clusterings [45, 3]. While these are good indicators of whether the clustering is done incorrectly, these don't apply in all cases. For example, it is feasible that the optimal clustering contains all the data points in a single cluster if the data is one dense glob. In this thesis, the statistical analysis was supported by the visual inspection of the clusters and data points to ensure that the data was not unusually skewed.

Furthermore, the usability of CVIs is debated in the research community [19]. It is argued that the CVI reflects expert knowledge poorly in some cases. Using the CVIs in tandem with some other restraint, like the cluster's shape, appears to have some benefits that exceed what just CVIs can achieve. While this requires some domain knowledge, this knowledge is much easier to comprehend than what is necessary to understand the inner workings of CVIs.

A genetic algorithm is a good tool for finding the optimal clustering hyperparameters. Finding suitable hyperparameters for the clustering has many local optima, which makes finding the best solution time-consuming. Given that the number of permutations can be huge, testing them all to find the optimal solution requires substantial calculations. For example, in this experiment, the tested values for the epsilon parameter were integers between 10 and 800, and the sampled values for the min points parameter were integers between 5 and 100. If all combinations of obtained hyperparameters were to be tested, this would result in a total of 75050 evaluations. Since the DBSCAN clustering algorithm runs in $O(nlog(n))$ time and the CVI metrics perform in time complexity as high as $O(n^2)$, executing all these permutations with new data acquisitionally if the data set to be optimized is large.

Even though the genetic algorithm approach performs well, optimizing huge data sets is still unfeasible without further optimization. For example, the Rotterdam data set contains 7314 data points, and the optimization process takes tens of minutes. If the data sets grow into the tens of thousands, running this optimization process becomes unfeasible with local hardware. To ensure that the optimization runs in a reasonable time for larger data sets, migration to a high-performance computing environment might be needed. Also, the custom metric calculation needs to be re-examined if the data size grows too large since the distance calculation issues a $n \times n$ matrix, where n is the number of data points and the quadratic space requirement might cause memory issues.

One of the bottlenecks when trying to evaluate this method was the difficulty in acquiring data. While AIS data is readily available from commercial sources, acquiring free AIS data is challenging. In this experiment, most of the data were obtained through NAPA Ltd., with access to Spire servers. The commercial nature of the data meant that it could not be shared in its raw form and modifications to the data had to be made before it could be shared. This work had to be done manually with a NAPA Ltd. employee, which substantially slowed new data acquisition. This means the method is not yet tested to determine if it works universally, and the focus has been on the seven port areas. Also, the temporal aspect of the data was not taken into account. With more abundant AIS data, it would be interesting to test how the optimal parameters change from year to year, e.g., would data taken from the Rotterdam area from 2020 have the same hyperparameters as data taken from the same location in 2022?

Optimization based on all the unmodified clustering except DBCV appears to have generated hyperparameters that created the same clustering for the Brest data set. This is even though the found hyperparameters were not identical. This is a known issue with clustering hyperparameters since the same clustering can be achieved with multiple hyperparameters. The optimization finds only a single combination of hyperparameters that achieve the best result. For example, suppose the optimal clustering can be achieved with value for min points within the range of 10 and 15 and epsilon within the range of 30 and 40 meters. In that case, the optimization converges to a single combination of these, e.g., min points of 12 and epsilon of 35. The selection of the optimal hyperparameters within the optimal range is random. Whether this range should be discovered or the values minimized or maximized is debatable. On the one hand, this would allow for more accurate results on the optimization. Still, on the other hand, this process would require substantial computational resources, especially when there are multiple hyperparameters to optimize.

Finally, during the initial optimization implementation, I noticed that the validation indices performed poorly without proper data preprocessing. For example, if a data set contains AIS points from multiple harbors, the optimization only found clusters covering the entire harbor area. Similarly, by removing points that have erroneous values, the optimizer's performance increases. While the DBSCAN algorithm is somewhat resistant to noise, the CVIs, except for DBCV, are very susceptible to noisy data [14, 3, 37]. So when dealing with data prone to errors, such as AIS messages, it is vital to ensure that the data is preprocessed efficiently.

6.1 Further work

The objective of this work has been to develop a method that can discover wharves in harbors. Detection of wharves is needed to gain information on port infrastructure, which is essential when developing port intelligence models. The work presented in this thesis will be a precursor to several port intelligence-related tasks. Firstly, the clusters gained with the optimizing approach will be analyzed further by combining them with AIS data. A similar approach was used in my previous work [49]; however, in that case, the hyperparameters were configured manually for the DBSCAN algorithm. Secondly, the found clusters are used to detect trajectories between anchorage areas and wharves. The goal is to develop a logistic regression model that can predict what wharf a ship is most likely going to end up visiting based on which anchorage area it enters. This would also require the identification of anchorage clusters, which can most likely be done using a modified version of the optimization approach presented in this thesis. The detection of anchorage clusters would most likely not work with the given custom distance metric since ships are not expected to have a static heading when at anchor. Also, the shape constraint would probably not produce the desired outcome. While exploring AIS data reveals that individual anchorage spots create circular or crescent shapes, there is no defining shape for larger anchorage areas.

Further work also includes implementing the optimizer for different kinds of use cases. Since the optimizer only tries to maximize the fitness function value, it could be easily reconfigured to find hyperparameters for different data sets or clustering algorithms. It can also be configured to test multiple clustering algorithms to see which performs better with a given task. This would require extensive research on the CVIs to determine what kind of data each metric is best at evaluating. This topic would fit right into the larger scope of unsupervised AutoML. While the initial setup of the optimization process requires some domain knowledge, once the optimizer is set up, users unfamiliar with the inner workings of clustering methods could use these methods to introduce cluster analysis to their work.

Lastly, one way to improve the optimization implementation would be to use multiobjective optimization. While the implementation presented in this thesis used multiobjective optimization when using the shape and CVI as the fitness function, a more complex multiobjective optimization could ensure better clustering. Methods such as the Pareto method or scalarization could give a more systematic approach to multiobjective optimization [4].

7. Conclusions

This thesis has two main contributions. Firstly, it presented a novel optimization tool to find the best hyperparameters for the DBSCAN algorithm applicable to geospatial data. Secondly, it explored the role of CVIs in cluster validation and their limitations. The novel implementation found hyperparameters for the algorithm using real-world data that produced clusters that reflected the requirements.

Using CVIs as the fitness function for the genetic algorithm seemed to be able to find good clusterings, but the correct choice of the CVI appeared to be the key. The Calinski-Harabasz and the Davies-Bouldin indexes performed relatively poorly when evaluating geospatial clusterings. However, the silhouette index and DBCV appear to be good at evaluating geospatial clustering in this use case, especially if a custom distance metric is implemented. Also, modifying the genetic algorithm to optimize specific characteristics, like in this case, the length/width ratio can lead to even more relevant clustering if the method is implemented effectively. If the optimizable character does not directly correlate with the quality of the clustering, CVIs can be used as a quality assurance tool to filter out trivial solutions.

CVIs are a tool that has been used more and more recently, but the research on the subject is limited [19]. Most research in this area focuses on developing new CVIs to combat the shortfalls of previous CVIs. Still, there appears to be a lack of research evaluating previous CVIs. While some CVIs, such as DBCV, have defined distinct use cases where the CVI excels, others are more generic. This makes the choice of the correct CVI to evaluate the clustering difficult, and most research defaults to the silhouette index as the go-to CVI [19].

The proposed approach was applied to find the wharves of a given port using AIS messages. The genetic algorithm implementations that used the fitness function that evaluated the shapes were able to detect DBSCAN hyperparameters that produced clusters that corresponded to the expected features pretty closely. In some data sets, it could be argued that the produced clusters reflect the ground truth almost perfectly, such as in the case of Ponta de Madeira. At the same time, there appears to be some issues still in some of the clusters produced in this manner, such as a cluster containing two wharves; overall, the clusters could be used to reflect the ground truth with small modifications.

While this thesis focused on clustering two-dimensional geospatial data, the methods introduced in this thesis will most likely work on other data types. The choice of CVI used in the genetic algorithm's fitness function depends on the data set used, which is an issue that needs further research. However, once the correct CVI is found for the task, the genetic algorithm can be easily modified to account for this. Furthermore, this method is not limited to DBSCAN, and other clustering methods can be easily tested to find the most suitable hyperparameters for them.

However, while these results show promise, most likely, this method is not a panacea to handle all possible scenarios. The noisiness of the data combined with the ambiguous requirement for the desired end product means that a solution that always produces correct results is unlikely. Most likely, the clusters created by this method will require some human oversight before the clusters can be used in further work. Also, it is still unknown how well this optimization process works with other types of data and clustering algorithms. Furthermore, the use of CVIs for the fitness function of the optimizer still has some unknown aspects, and further research needs to be done to identify which CVI to use with different data types. Overall, finding suitable hyperparameters for clustering appears to be a challenging task to automate. In most cases, the final decision on the hyperparameters has to be made using a human expert. However, further exploring the methods presented in this thesis could lead to interesting insights into how clustering processes can be at least partially automated.

7.1 Acknowledgements

This thesis was done as part of the VesselAI project (Horizon 2020 Framework Programme of the European Union under grant agreement No 957237). I would also like to thank Jaakko Talonen from NAPA Ltd. for helping me create a large part of the training data used in this thesis.

Bibliography

[1] R. Adland and H. Jia. Dynamic speed choice in bulk shipping. *Maritime Economics & Logistics*, 20(2):253–266, 2018.

[2] R. Adland, H. Jia, and S. P. Strandenes. Are ais-based trade volume estimates reliable? the case of crude oil exports. *Maritime Policy & Management*, 44(5):657–665, 2017.

[3] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. An extensive comparative study of cluster validity indices. *Pattern recognition*, 46(1):243–256, 2013.

[4] P. Bajpai and M. Kumar. Genetic algorithm–an approach to solve global optimization problems. *Indian Journal of computer science and engineering*, 1(3):199–206, 2010.

[5] P. Bhattacharjee and P. Mitra. A survey of density based clustering algorithms. *Frontiers of Computer Science*, 15(1):1–27, 2021.

[6] K. Bichou. *Port operations, planning and logistics.* CRC Press, 2014.

[7] D. Birant and A. Kut. St-dbscan: An algorithm for clustering spatial-temporal data. *Data & knowledge engineering*, 60(1):208–221, 2007.

[8] R. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[9] J. Carson-Jackson. Satellite ais–developing technology or existing capability? *The Journal of Navigation*, 65(2):303–321, 2012.

[10] L. Chen, D. Zhang, X. Ma, L. Wang, S. Li, Z. Wu, and G. Pan. Container port performance measurement and comparison leveraging ship gps traces and maritime open data. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1227–1242, 2015.

[11] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, 2:224–227, 1979.

[12] E. Ditton, A. Swinbourne, T. Myers, and M. Scovell. Applying semi-automated hyperparameter tuning for clustering algorithms. *arXiv preprint arXiv:2108.11053*, 2021.

[13] T. Emmens, C. Amrit, A. Abdi, and M. Ghosh. The promises and perils of automatic identification system data. *Expert Systems with Applications*, 178:114975, 2021.

[14] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[15] Z. Falahiazar, A. Bagheri, and M. Reshadi. Determining the parameters of dbscan automatically using the multi-objective genetic algorithm. *J. Inf. Sci. Eng.*, 37(1):157–183, 2021.

[16] X. Fan, Y. Yue, P. Sarkar, and Y. R. Wang. On hyperparameter tuning in general clustering problemsm. In *International Conference on Machine Learning*, pages 2996–3007. PMLR, 2020.

[17] W. Fu and P. O. Perry. Estimating the number of clusters using cross-validation. *Journal of Computational and Graphical Statistics*, 29(1):162–173, 2020.

[18] G. Fuentes. Generating bunkering statistics from ais data: A machine learning approach. *Transportation Research Part E: Logistics and Transportation Review*, 155:102495, 2021.

[19] M. Gagolewski, M. Bartoszuk, and A. Cena. Are cluster validity measures (in) valid? *Information Sciences*, 581:620–636, 2021.

[20] R. Gelbard, O. Goldman, and I. Spiegler. Investigating diversity of clustering methods: An empirical comparison. *Data & Knowledge Engineering*, 63(1):155–166, 2007.

[21] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, 17(2):107–145, 2001.

[22] A. Harati-Mokhtari, A. Wall, P. Brooks, and J. Wang. Automatic identification system (ais): a human factors approach. *Journal of Navigation*, 60(3):373–389, 2007.

[23] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM journal on computing*, 2(2):88–105, 1973.

[24] I. M. O. (IMO). Revised guidelines for the onboard operational use of shipborne automatic identification systems (ais). https://wwwcdn.imo.org/localresources/en/OurWork/Safety/Documents/AIS/Resolution%20A.1106(29).pdf, 2015. Accessed: 2022-09-25.

[25] H. K. Kanagala and V. J. R. Krishnaiah. A comparative study of k-means, dbscan and optics. In *2016 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6. IEEE, 2016.

[26] A. Karami and R. Johansson. Choosing dbscan parameters automatically using differential evolution. *International Journal of Computer Applications*, 91(7):1–11, 2014.

[27] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[28] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE, 2014.

[29] K. Kowalska and L. Peel. Maritime anomaly detection using gaussian process active learning. In *2012 15th International Conference on Information Fusion*, pages 1164–1171. IEEE, 2012.

[30] P. Last, C. Bahlke, M. Hering-Bertram, and L. Linsen. Comprehensive analysis of automatic identification system (ais) data in regard to vessel movement prediction. *The Journal of Navigation*, 67(5):791–809, 2014.

[31] P. Last, M. Hering-Bertram, and L. Linsen. How automatic identification system (ais) antenna setup affects ais signal quality. *Ocean Engineering*, 100:83–89, 2015.

[32] H.-T. Lee, J.-S. Lee, H. Yang, and I.-S. Cho. An ais data-driven approach to analyze the pattern of ship trajectories in ports using the dbscan algorithm. *Applied Sciences*, 11(2):799, 2021.

[33] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. Understanding of internal clustering validation measures. In *2010 IEEE international conference on data mining*, pages 911–916. IEEE, 2010.

[34] P. E. McKight and J. Najab. Kruskal-wallis test. *The corsini encyclopedia of psychology*, pages 1–1, 2010.

[35] L. M. Millefiori, D. Zissis, L. Cazzanti, and G. Arcieri. A distributed approach to estimating sea port operational regions from lots of ais data. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1627–1632. IEEE, 2016.

[36] B. L. Miller, D. E. Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[37] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander. Density-based clustering validation. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 839–847. SIAM, 2014.

[38] J. A. O'Keefe. The universal transverse mercator grid and projection. *The Professional Geographer*, 4(5):19–24, 1952.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[40] Y. Poulakis, C. Doulkeridis, and D. Kyriazis. Autoclust: A framework for automated clustering based on cluster validity indices. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1220–1225. IEEE, 2020.

[41] A. Rajabi, A. K. Saryazdi, A. Belfkih, and C. Duvallet. Towards smart port: an application of ais data. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1414–1421. IEEE, 2018.

[42] C. Ray, R. Dréo, E. Camossi, A.-L. Jousselme, and C. Iphar. Heterogeneous integrated dataset for maritime intelligence, surveillance, and reconnaissance. *Data in brief*, 25:104141, 2019.

[43] Y. Ren, X. Liu, and W. Liu. Dbcamm: A novel density based clustering algorithm via using the mahalanobis metric. *Applied Soft Computing*, 12(5):1542–1554, 2012.

[44] B. J. Rhodes, N. A. Bomberger, M. Seibert, and A. M. Waxman. Maritime situation monitoring and awareness using learning mechanisms. In *MILCOM 2005-2005 IEEE Military Communications Conference*, pages 646–652. IEEE, 2005.

[45] E. Schubert, S. Hess, and K. Morik. The relationship of dbscan to matrix factorization and spectral clustering. In *LWDA*, 2018.

[46] P. Sheng and J. Yin. Extracting shipping route patterns by trajectory clustering model based on automatic identification system data. *Sustainability*, 10(7):2327, 2018.

[47] G. K. Silber, J. Slutsky, and S. Bettridge. Hydrodynamics of a ship/whale collision. *Journal of Experimental Marine Biology and Ecology*, 391(1-2):10–19, 2010.

[48] A. Starczewski, P. Goetzen, and M. J. Er. A new method for automatic determining of the dbscan parameters. *Journal of Artificial Intelligence and Soft Computing Research*, 10, 2020.

[49] J. Steenari, L. E. Lwakatare, J. K. Nurminen, J. Talonen, and T. Manderbacka. Mining port operation information from ais data. In *Hamburg International Conference of Logistics (HICL) 2022*, pages 657–678. epubli, 2022.

[50] I. Sung, H. Zografakis, and P. Nielsen. Multi-lateral ocean voyage optimization for cargo vessels as a decarbonization method. *Transportation Research Part D: Transport and Environment*, 110:103407, 2022.

[51] A. Tritsarolis, C. Doulkeridis, N. Pelekis, and Y. Theodoridis. St_visions: a python library for interactive visualization of spatio-temporal data. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*, pages 244–247. IEEE, 2021.

[52] E. Tu, G. Zhang, L. Rachmawati, E. Rajabally, and G.-B. Huang. Exploiting ais data for intelligent maritime navigation: A comprehensive survey from data to methodology. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1559–1582, 2017.

[53] T. Van Craenendonck and H. Blockeel. Using internal validity measures to compare clustering algorithms. *Benelearn 2015 Poster presentations (online)*, pages 1–8, 2015.

[54] L. Vendramin, R. J. Campello, and E. R. Hruschka. Relative clustering validity criteria: A comparative overview. *Statistical analysis and data mining: the ASA data science journal*, 3(4):209–235, 2010.

[55] X. Wang and Y. Xu. An improved index for clustering validation based on silhouette index and calinski-harabasz index. In *IOP Conference Series: Materials Science and Engineering*, volume 569, page 052024. IOP Publishing, 2019.

[56] M. Wegmann, D. Zipperling, J. Hillenbrand, and J. Fleischer. A review of systematic selection of clustering algorithms and their evaluation. *arXiv preprint arXiv:2106.12792*, 2021.

[57] D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[58] M. Winther, J. H. Christensen, M. S. Plejdrup, E. S. Ravn, Ó. F. Eriksson, and H. O. Kristensen. Emission inventories for ships in the arctic based on satellite sampled ais data. *Atmospheric Environment*, 91:1–14, 2014.

[59] D. Yang, L. Wu, S. Wang, H. Jia, and K. X. Li. How big data enriches maritime research–a critical review of automatic identification system (ais) data applications. *Transport Reviews*, 39(6):755–773, 2019.

[60] L. Zhang, Q. Meng, and T. F. Fwa. Big ais data based spatial-temporal analyses of ship traffic in singapore port waters. *Transportation Research Part E: Logistics and Transportation Review*, 129:287–304, 2019.

[61] S. Zhou, A. Zhou, W. Jin, Y. Fan, and W. Qian. Fdbscan: a fast dbscan algorithm. *Ruan Jian Xue Bao*, 11(6):735–744, 2000.