

Article

Adaptive Ensemble Clustering Using Meta-Heuristics-Algorithms for Global Navigation Satellite System (GNSS) Line of Sight (LOS)/Non Line of Sight (NLOS) Classification

Gianmarco Baldini ^{1,*}  and Fausto Bonavitacola ²¹ Joint Research Centre, European Commission, 21027 Ispira, Italy² Pikel, Via Ernesto Breda, 176, 20126 Milano, Italy; fausto.bonavitacola@ext.ec.europa.eu

* Correspondence: gianmarco.baldini@ec.europa.eu; Tel.: +39-3342300960

Abstract

Global Navigation Satellites Systems (GNSSs) have become a predominant feature in the digital life of citizens, and they provide positioning services in various applications including pedestrian and vehicular navigation. In urban environments with the presence of buildings and another obstacles, GNSS positioning may be unreliable because of non-line-of-sight (NLOS) conditions, and the classification of observed satellite visibility between LOS and NLOS may improve GNSS receivers to improve their performance to provide the positioning services. In this context, machine learning algorithms using features like signal noise ratio, pseudorange, elevation angle, and others have been applied to this problem both in supervised and unsupervised mode. Because the ground truth information on LOS/NLOS conditions may not always be available, unclustering algorithms have been applied for unsupervised classification, but the classification performance is still limited. This paper proposes an ensemble approach where different clustering algorithms, both historical and recently introduced in the literature, are combined to improve the LOS/NLOS classification accuracy. Even if the ensemble approach manages to achieve a significant improvement, a novel and more sophisticated approach is proposed in this paper, where the contributions of each clustering algorithm are weighted. The optimal values of the weights are estimated using various Meta-Heuristics Algorithms (MHA) on a subset of GNSS data where the ground-truth information is available (i.e., labeled data set). In a subsequent step, the performance of the optimal weighted clustering ensemble is evaluated. The approach is applied to a recent public data set with 57 satellites, where it is shown to outperform the specific clustering approaches by a large margin (more than 7%). The Meta Heuristics Algorithm (MHA)s have similar performance, with the Dynamic Opposition Grey Wolf Optimization (DOLGWO) having a minor advantage against the other MHAs.



Academic Editors: Mohammed Qaraad and David Guinovart

Received: 6 June 2026

Revised: 30 June 2026

Accepted: 4 July 2026

Published: 7 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: GNSS; line of sight; optimization; clustering; machine learning

1. Introduction

Global Navigation Satellite Systems (GNSS) have become a very important function in everyday life, but can be disturbed by the presence of obstacles, which block the path between the receiver and the satellite, creating the condition of Not Line of Sight (NLOS), thus generating pseudorange errors in the range of tens of meters. The knowledge of the Line Of Sight (LOS) or NLOS condition is useful to improve the performance of GNSS

receivers [1], and in recent times Machine Learning (ML) algorithms have been used to classify LOS/NLOS condition using the features extracted from the basic observables and measurements, like signal strength or carrier-to-noise-ratio (C/N0), satellite elevation angle, code pseudorange measurement, etc. These kinds of features were referred to as national marine electronics association (NMEA)-level or receiver independent exchange format (RINEX)-level features as they can be obtained from the commonly used NMEA/RINEX files [1,2].

The application of supervised ML algorithms requires labeled data sets, which may not always be available, and unsupervised approaches may be of more practical use. In particular, a context in which a relatively small labeled data set is available for semi-supervised training, while most of the data to be classified for the LOS/NLOS condition are not labeled, is highly probable. This is the context of this article, which uses only a portion of the public UrbanNav data set [3], which is an open-source data set that contains multiple localization sensors, including LiDAR, stereo camera, inertial navigation system and GNSS. The position ground truth is provided by the NovAtel SPANCPT unit described in https://docs.novatel.com/OEM7/Content/SPAN_CPT7/SPANCPT7_Overview.htm (accessed on 3 July 2026).

Both conventional ML algorithms and deep learning (DL) architectures can be used for the LOS/NLOS classification task in unsupervised mode. On the other hand, the choice of DL has also been used in some cases, but it requires significant computing resources and a large amount of data, which may not be practical in a GNSS context, where the sampling rate is usually between 1 and 10 Hertz (in this study, the GNSS receiver was configured with a sampling rate of 1 Hz). It would require a significant amount of time to create a labeled data set for the application of DL [1,2]. For this reason, this paper focuses only on ML without DL architectures.

One significant challenge in unsupervised classification of GNSS LOS/NLOS condition is the potential lack of accuracy due to various factors, as described in the related work Section 2. In particular, each GNSS satellite may not be fully visible to the GNSS receiver, especially in data collected in urban environments, such as the UrbanNav data set used in this paper. As a consequence, the amount of data to be classified can be limited and with great variability, and the choice of the clustering algorithm is quite important. The problem is that it is known a priori which clustering algorithm is more appropriate for the specific problem.

To mitigate this problem, this paper proposes the application of an ensemble of clustering algorithms to mitigate the false decisions of each algorithm. In the first step, a basic voting scheme is used where a label is chosen on the basis of the hard label information provided by each clustering. In a second phase, a novel and more sophisticated weighing algorithm is adopted, where weights are calculated using labelled information from a limited portion of the data set. The optimal value of the weights is estimated using MHA, where the cost function is the error rate of the ensemble compared to the ground truth information. The application of such MHA optimized weighted ensemble algorithm produces the LOS/NLOS classification result, which are significantly higher than each clustering algorithm and the basic ensemble algorithm.

Scope: We highlight that the main goal of this study is the unsupervised classification of GNSS LOS/NLOS condition using the features provided by the GNSS receiver. The following aspects are considered outside of the scope of this study: (1) The analysis of the specific type of NLOS condition (e.g., multiple reflections), (2) the application of DL to signals at the input of the GNSS receiver, and (3) the application of supervised ML algorithms for the final results of the GNSS LOS/NLOS classification.

The structure of this paper is the following: Section 3 describes the overall methodology of the proposed approach including the Section 3.2, which describes the data set used in this study. Section 4 describes the clustering algorithms used separately to build the ensemble and the optimal weighted ensemble. Section 5 provides a high-level description of the MHAs used in this study. Section 6 provides the results of the approach adopted in the public data set described in Section 3.2. Finally, Section 7 presents the conclusions of the study and describes future developments.

2. Related Work

GNSS LOS/NLOS classification is a problem, which is increasing in importance in recent years especially in urban environments where the NLOS conditions are plentiful. Almost all studies in the literature focus on supervised classifications ML such as [4] where support vector machine (SVM) and Shallow Neural Networks (SNN) are used in the same UrbanNav data set used in this study. A more sophisticated approach based on Large Language Model (LLM) is presented in [5] still for supervised classification, but the use of LLM requires significant computing resources. A tutorial paper on classification is presented in [1], where it is highlighted that elevation angle and receiver correlator output from a single channel of satellite signal are the most popular input features for both supervised and unsupervised algorithms. Then, in [1], the benchmark results for supervised classification are provided. In another recent study [6], the authors propose an approach for NLOS Identification with continuous learning to address the problem that supervised machine learning may not be able to generalize across new GNSS environments. The method preserves performance on previously encountered scenarios while adapting to new environments by employing simple regularization techniques that do not require access to the data from the old scenarios.

All the papers described above focus on providing results for supervised classification, but various researchers have recognized the challenge of having labeled data sets that can be generated by independent sources such as ray tracing, building models, and 3D mapping aided with positioning. As a consequence, the problem of unsupervised classification of GNSS LOS/NLOS has more practical utility, and this is the focus of this paper.

Regarding the research literature on unsupervised classification of clustering GNSS LOS/NLOS, the authors of [7] have used unsupervised machine learning algorithms using Indian NavIC signals in both L5 (1176.45 MHz) and S band (2492.028 MHz) frequencies. The same approach of using data from different frequencies was also adopted in this study but with a larger variety of GNSS constellations (GPS, Galileo, Beidou). In addition, the authors of [7] did not use an ensemble of unsupervised algorithms as in this paper, nor an optimized weighted combination of algorithms.

The authors in [8] propose a multipath detection algorithm, which is based on clustering of K-means that belongs to the class of unsupervised ML algorithms. The algorithm processes measurement sets computed for each satellite as in this study, namely, carrier phase, pseudorange, and carrier-to-noise ratio, creating clusters of consistent measurements with the goal of identifying multi-path conditions, which may lead to LOS/NLOS classification. Compared to [8], this study proposes an analysis based on a richer set of unsupervised ML algorithms, which have also been introduced more recently in the literature compared to the clustering of K-means. In addition, ref. [8] does not propose an ensemble approach as in this study.

The authors of [9] have also used the K-means algorithm to distinguish between LOS and NLOS conditions in an unsupervised manner on a data set collected from the urban area of Nanjing. Even if the results presented in [9] are promising, this study uses a much larger set of clustering algorithms and also proposes an ensemble approach.

The authors of [10] use the K-means clustering algorithm to mitigate multipath effects in GNSS positioning. The approach consists of two steps. In the first step, a least squares equation is constructed using the relationship between the classical satellite elevation angle noise model and the signal-to-noise ratio (SNR) noise model. The least squares residual is used to determine the NLOS and LOS satellites based on the K-means clustering algorithm. The second method mitigates the LOS multipath signal based on the consistency cheque between prediction residuals and prediction covariance of the extended Kalman Filter. Even if in [10] the clustering approach is used for a subsequent goal, the study still uses only one clustering algorithm, which is also rather old, and no other clustering algorithms were adopted as in this study.

A step forward to the application of only the K-means algorithm for this LOS/NLOS classification problem is presented in [11], where both the K-means and fuzzy c-means clustering techniques are used with results that indicate a comparable performance of both algorithms.

Although the approaches proposed in previously reviewed papers demonstrate potential, most of the existing studies rely on a single clustering algorithm, which may limit stability and generalization of the results in practical applications. This is a problem well known in other domains where the combination of different clustering algorithms can be attempted, such as in healthcare in [12] or [13] in the transport sector. For these reasons, this paper proposes the use of an ensemble of clustering algorithms.

The paper more similar to this study, in the context of GNSS, is [14] where the authors have applied a set of unsupervised algorithms to the same problems of distinguishing LOS from NLOS conditions and have also used an ensemble of algorithms to improve classification performance. The results shown in [14] demonstrated (as in this study) that the ensemble approach is superior to the specific clustering algorithms and is more stable as a result. On the other hand, the approach proposed in this study proposes a number of improvements compared to [14]: (1) this study proposes a novel (in this context) weighted ensemble clustering approach based on the use of MHAs such as GWO, PO, and MGO, (2) this study uses a set of clustering algorithms, which have been proposed more recently in the literature such as IDKC and Torque clustering, while [14] relies on algorithms introduced before 2015. In addition, this study uses eight clustering algorithms to build the ensemble while [14] uses only six algorithms. (3) The features used in this study and [14] are similar (C/N_0 , residuals, elevation angle), but this study uses data from two frequencies while [14] uses only the L1 frequency.

MHAs have been extensively used in the transportation sector where GNSS is an important capability for various optimization problems, as in [15] where a real-coded genetic algorithm with fuzzy control was used to optimize system dynamics models in a railway transportation problem. Similarly, real-coded genetic algorithms were used in [16]. This algorithm can be further extended with federated learning as in [17]. In general, optimization and MHA methods are used in the transportation sector in various applications, including cost and operational efficiency, planning and scheduling, logistics and transportation, supply chain and logistics networks, energy and sustainability, and demand and users [18].

The application of MHA algorithms to the GNSS problem is relatively recent, with few studies published in recent years. The study presented in [19] uses various MHAs, including Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) and others, to the Least Squares (LS) method used to improve the accuracy of the GNSS network. In a similar context, the study described in [20] proposes an error checking and correcting strategy based on a least-squares ambiguity decorrelation Adjustment-Grey Wolf optimization

algorithm (LAMBDA-GWO) combined with the Tikhonov regularization method. The final goal of [20] is to improve the accuracy of integer ambiguity for fast satellite positioning.

In another study, the authors have applied Grey Wolf Optimization (GWO) and other algorithms to the selection and reduction of reference points in the determination of the local GNSS/leveling geoid [21]. In another area, still related to GNSS, GWO was used to enhance the array anti-jamming capability by optimizing the power inversion algorithm (PI), which is widely used in GNSS [22].

A recap table of the studies cited above specifically for unsupervised classification of GNSS LOS/NLOS is provided in the following Table 1. We do not report the results of studies that apply supervised learning, nor in the following summary table, nor in the results Section 6, because the accuracy values achieved reported in such studies are not comparable (i.e., they are higher) to the accuracy values obtained in the application of unsupervised algorithms where the knowledge of the labels is not used in the creation of the models.

Table 1. Summary of the related work on unsupervised learning for GNSS LOS/NLOS classification.

Title	Method	Limitations	Ref.
Unsupervised machine learning approach for multipath classification of NavIC signals	Use unsupervised learning algorithms on NavIC data	No ensemble clustering is used. Only the NavIC data is used.	[7]
Multipath detection based on K-means clustering	K-means applied to carrier phase, pseudorange, and carrier-to-noise ratio for different satellites	Only K-means is used and no ensemble is applied.	[8]
Multipath/NLOS detection based on K-means clustering for GNSS/INS tightly coupled system in urban areas	K-means	Only K-means is used and no ensemble is applied.	[9]
Effective multipath mitigation methods for RTK in urban environments.	Applies K-means to phase measurement in L1 K-means	Only K-means is used and no ensemble is applied.	[10]
Comparative analysis of pseudorange multipath mitigation performance using K-means and Fuzzy c-means clustering techniques	It uses K-means and Fuzzy c-means clustering	Only K-means and Fuzzy c-means are used and no ensemble is used	[11]
ISVEC: unsupervised ensemble clustering with iterative semantic voting for GNSS LOS/NLOS identification	Use an ensemble of 6 clustering algorithms.	The clustering algorithms were proposed in research literature before 2015. No weighted ensemble is used	[14]

In summary, none of the above-listed articles have proposed the application of MHAs to optimize a weighted ensemble clustering for the unsupervised classification of the GNSS LOS/NLOS condition as in this paper.

3. Method and Materials

The general methodology is pictorially described in the Figure 1.

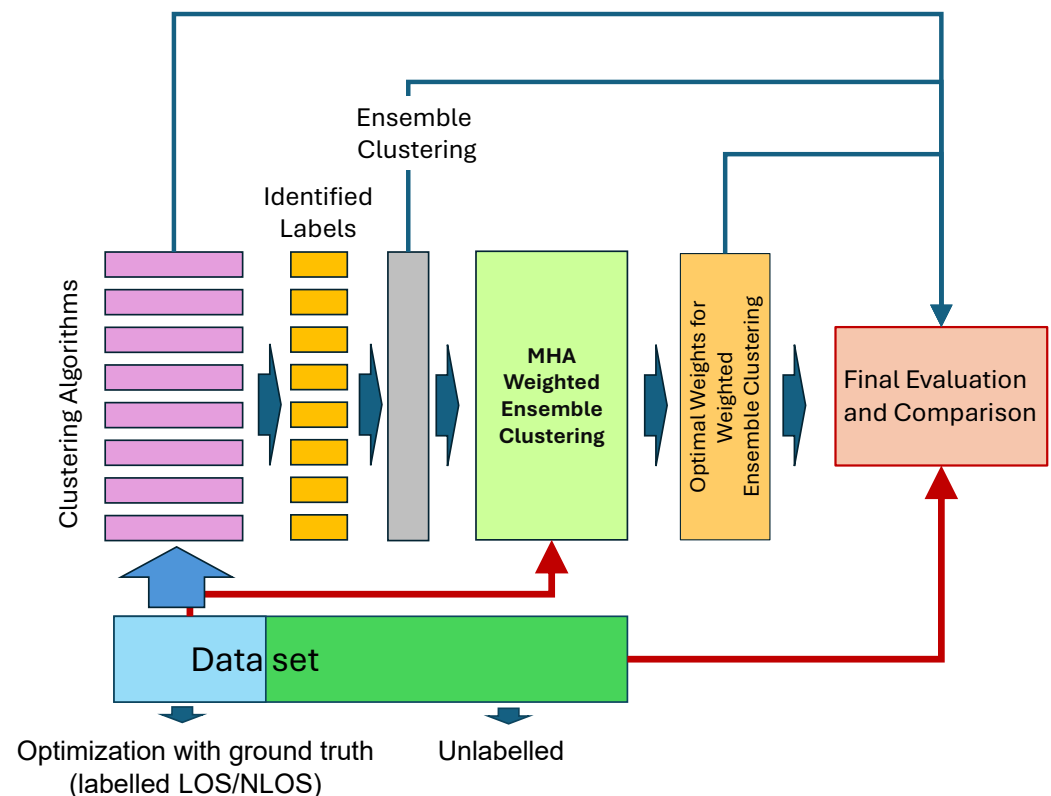


Figure 1. Methodology of the proposed approach.

The methodology is based on a semi-supervised approach. A portion of the data set N_O (which is labeled to indicate the LOS/NLOS condition) is allocated to the optimal definition of the weights used in the adaptive ensemble approach through MHA and for the selection of features using reliefF, which is the supervised feature selection algorithm [23]. This step requires knowledge of the ground truth (i.e., labels) even if clustering algorithms are used. The rest of the data set N_C is used for clustering. In this study, we selected 1/6 of the data set to create N_O by choosing samples from the entire data set in random using the MATLAB run-perm function. The rest of the data set was used to create N_C . Eight clustering algorithms are used to implement the approach. Clustering algorithms are identified and described in Section 4 and have been chosen as a balance between historical (e.g., KMeans++) and recently introduced algorithms (e.g., torque clustering) and for the public availability of the implementation source code with MATLAB. In this study, we only used the label information as the output of the clustering algorithm and not other information (e.g., sample probabilities or distances from each cluster). The reason is that not all the clustering algorithms provide the latter information in an equal way while the output label information is always provided. As shown in Section 6, the performance of each clustering algorithm is not the same. The concept of the proposed approach is to compare and unify the results of the clustering algorithms in a simple ensemble fashion (e.g., the sum of the label values of the clustering algorithms) and in a more sophisticated weighted ensemble fashion where the optimal weight values are defined using MHA against the truth of the ground label. The cost function MHA is designed to minimize the classification error of the weighted sum of the label information clustered in the N_O portion of the data set. In other words, the cost function uses the hard label information provided by the clustering algorithms in a weighted sum where the weights must be optimized.

If the weighted sum of one class (LOS) is larger than the second class (NLOS), the weighted ensemble class is determined to be LOS; otherwise, it is NLOS. The description of the function loss algorithm used by the MHAs in this study is provided below in Algorithm 1, where n is the number of samples in the labeled N_O portion of the data set, $GTLABEL$ is the ground truth label information in N_O , W^k is the weight associated with each of the k (1..8) clustering algorithms, and $Label_i^k$ is the hard label information provided by the k clustering algorithm for sample i . $LossValue$ is the final loss value of the cost function calculated as 1 minus the sum of all correctly identified samples in N_O against the ground truth of the labeled data set. The optimized weights WO^k estimated in the Algorithm 1 are then used to predict the labels for the GNSS LOS/NLOS classification according to the Algorithm 2 where the hard label information predicted by each clustering algorithm is combined in an ensemble fashion.

Algorithm 1 Function Loss used in the MHAs

```

for  $i \leftarrow 1, n$  do
   $WSUM_{LOS} \leftarrow \sum_j^k (W^k \times (Label_i^k == LOS))$ 
   $WSUM_{NLOS} \leftarrow \sum_j^k (W^k \times (Label_i^k == NLOS))$ 
  if  $WSUM_{LOS} \geq WSUM_{NLOS}$  then
     $WLABEL(i) \leftarrow LOS$ 
  else
     $WLABEL(i) \leftarrow NLOS$ 
  end if
end for
 $accweighted \leftarrow \sum_i^n (WLABEL == GTLABEL)$ 
 $LossValue \leftarrow (1 - accweighted)$ 

```

In Algorithm 2, the suffix U identifies the variables applied to the unsupervised clustering portion of the data set N_C . Then, nu is the number of samples in N_C , $WULABEL$ is the final weighted ensemble predicted label information and WU_k is the weight associated with the k cluster algorithm CA^k , which provides the label information to predict HL^k . $accweightedu$ is the final accuracy of the GNSS LOS/NLOS classification.

Algorithm 2 Weighted ensemble clustering

```

for  $k \leftarrow 1, C$  do
   $HL^k$  from  $CA^k$ 
end for
for  $i \leftarrow 1, nu$  do
   $WUSUM_{LOS} \leftarrow \sum_i^k (WU^k \times (HL_i^k == LOS))$ 
   $WUSUM_{NLOS} \leftarrow \sum_i^k (WU^k \times (HL_i^k == NLOS))$ 
end for
for  $i \leftarrow 1, nu$  do
  if  $WUSUM_{LOS} \geq WUSUM_{NLOS}$  then
     $WULABEL(i) \leftarrow LOS$ 
  else
     $WULABEL(i) \leftarrow NLOS$ 
  end if
end for
 $accweightedu \leftarrow \sum_i^n u(WULABEL == GTULABEL)$ 

```

Theoretically, clustering algorithms with better classification performance should be assigned a larger weight than other ones.

In the final evaluation and comparison step, the results of the Algorithm 2 are obtained for all satellites in the UrbanNav data set. As shown in Results Section 6, for each satellite,

different MHAs can obtain the best classification accuracy, even if, in general, the proposed approach is demonstrated to be superior to the basic ensemble algorithm.

The performance metric is the classification accuracy, which is defined as $Accuracy = ((PrL == TL) / sum(TL))$, where PrL is the number of predicted labels and TL is the number of true labels (i.e., LOS/NLOS) present in the data set.

3.1. Computing Platform

The scientific computing environment is based on MATLAB Release 2024b from MathWorks based in Natick, Massachusetts, USA and the ML classifiers implementation is from the Machine Learning toolbox. The computing platform is an Intel I9-990KF Central Processing Unit (CPU) with a clock speed of 3.6 GHz and 32 Gbytes of Random Access Memory (RAM).

3.2. Data Set

The data set used to evaluate the approach proposed in this article was published by the authors of [3,24], where it is called the UrbanNav data set. It was created by driving a car equipped with a GNSS Ublox F9P receiver, and the data were collected in Tsim Sha Tsui (TST), Whampoa (WP), and Mong Kok (MK).

The data set is publicly available at <https://www.kaggle.com/competitions/gnss-classification/data> (accessed on 3 July 2026).

The authors have used only the training portion of the GNSS raw data set as it is the most detailed in terms of features. The data set uses four GNSS constellations with 57 visible satellites: 17 GPS satellites, 11 Galileo satellites, 13 Glonass satellites, and 15 Beidou satellites. The data format acquisition is based on RYNEX 3.02. The ground truth labeling information, which is used for the application of MHA is based on both Skymask and the code pseudorange double difference observable as explained in [24]. The Skymask is defined as the skyplot with building boundary information. If the elevation angle of satellite is lower than that of the building boundary at the same azimuth angle, it is labeled as NLOS, otherwise, it is labeled as LOS. The Skymask method is easy and straightforward, but it has the drawback that it is not accurate for satellites near the building edges. To solve this problem, the authors of the data set have used the pseudorange double difference observable code, which is defined later in this subsection.

The distribution of the samples across the constellations and satellites is shown in the histogram Figure 2, which shows a great variability in the number of samples across the satellites. This is due to the path taken by the vehicle used to collect the GNSS data in the urban environment, which obscures the path between the vehicle and the GNSS satellite. For some satellites, most of the time the path is blocked, which explains the low number of samples for those satellites. The naming convention is that the prefix G means the USA GPS constellation, R the Russian Glonass constellation, E the European Galileo constellation, and C the Chinese Beidou constellation.

To the authors' knowledge, this is the only available data set with fully labeled information for the GNSS LOS/NLOS classification in the urban environment. Future developments will take into account other data sets when available.

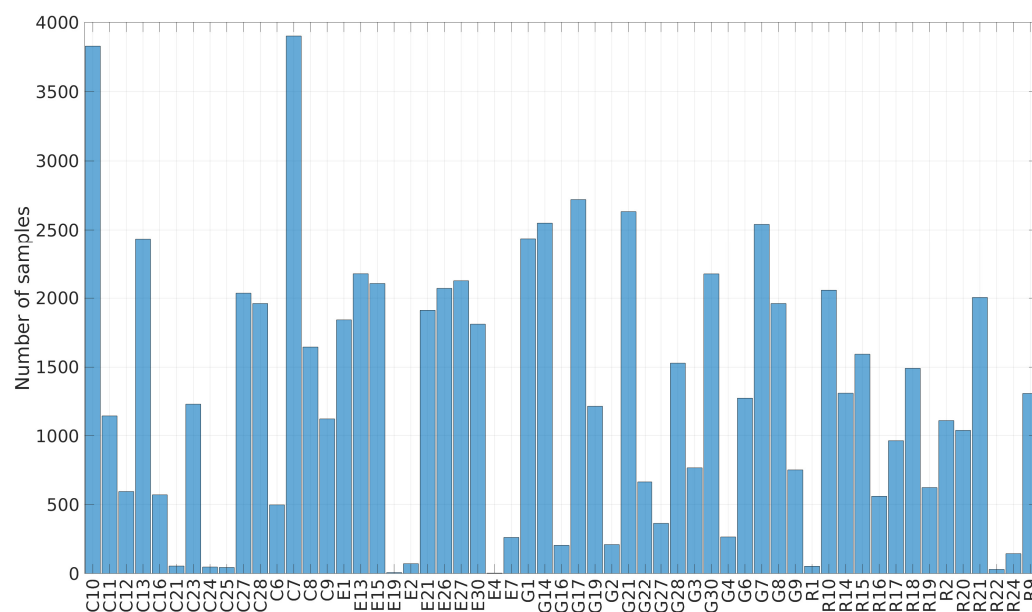


Figure 2. Distribution of samples across the different satellites visible during the data collection phase in the UrbanNav data set [3,24].

A description of the data columns is provided in the following Table 2. The underlying formal definitions with the related formulas are presented in this paper after Table 2.

Table 2. Fields of the data set used in this study.

Name	Definition	Units
$GPS_{Time(s)}$	GPS time	UTC Time seconds
$Satellite_{code}$	Pseudorandom Noise (PRN) code of satellite (associated to a naming scheme)	
$code_{L1}$	code measurement in L1	meters
$phase_{L1}$	phase measurement in L1	whole cycle
$doppler_{L1}$	doppler in L1	Hz
cnr_{L1}	C/N_0 signal strength in L1	dB-Hz
$code_{L2}$	code measurement in L2	meters
$phase_{L2}$	phase measurement in L2	whole cycle
$doppler_{L2}$	doppler in L2	Hz
cnr_{L2}	C/N_0 signal strength in L2	dB-Hz
label	LOS/NLOS label of the satellite, where 1 represents LOS, 0 represents NLOS. This is gained by 3D building model and the ground truth location.	
Pr_{Error}	Pseudorange error (assumed mostly contributed by <i>multipath/NLOS</i>). The value is calculated by double-difference using the closest ground station measurement and the highest elevation angle satellite.	meters
Latitude	Latitude of the ground truth location	degrees
Longitude	Longitude of the ground truth location	degrees
Height	Ellipsoidal height of the ground truth location	meters

Some basic information on how some of the features presented in Table 2 are shown here. We use the notation from [24]. In basic terms, the double difference (*DD*) of carrier phase measurements is one of the standard methods for determining the relative position between two receivers. The code phase in L1 and L2 determines the travel time of the signals. Doppler in L1 and L2 measures the frequency shift in satellite signals caused by the relative motion between satellites and a receiver. The signal strength in L1 and L2 is represented by the Carrier-to-Noise Density ratio (C/N_0) and is measured in dB-Hz.

This metric represents the ratio of the power of the received carrier signal to the spectral density of the power of the noise. The pseudorange of codes between receiver k and satellite j can be modeled as:

$$\rho_k^j(f) = r_k^j + c(\delta t_k - \delta t^j) + Iono_{kj} + Tropo_{kj} + \epsilon_{k,NLOS}^j + e_k^j \quad (1)$$

where r_k^j is the geometric range, $(\delta t_k - \delta t^j)$ are the clock bias of the receiver k and the satellite j , respectively, $Iono_{kj}$ and $Tropo_{kj}$ are the ionospheric and tropospheric errors, respectively, $\epsilon_{k,NLOS}^j$ is the error induced by NLOS and finally e_k^j represents other unmodeled errors such as thermal noise.

The data set was created using a reference receiver in open-sky conditions. This receiver is called A in the rest of this section, whereas the moving receiver mounted on the vehicle is called B. The reference station is supposed to be free of multipath and NLOS effects, thus $\epsilon_{A,NLOS}^1 = 0$, $\epsilon_{A,NLOS}^2 = 0$. To generate the observable *DD*, the *SD* single difference is first created by differentiating ρ_A^j and ρ_B^j .

$$SD_{AB}^1(f) = \Delta r_{AB}^1 + c(\delta t_A - \delta t_B) - \epsilon_{B,NLOS}^1 + \Delta e_{AB}^1 \quad (2)$$

$$SD_{AB}^2(f) = \Delta r_{AB}^2 + c(\delta t_A - \delta t_B) - \epsilon_{B,NLOS}^2 + \Delta e_{AB}^2 \quad (3)$$

where Δr_{AB}^j is $r_A^j - r_B^j$ and Δe_{AB}^j is $e_A^j - e_B^j$.

DD is then obtained by differencing SD_{AB}^1 and SD_{AB}^2 .

$$DD_{AB}^{12}(f) = \nabla \Delta r_{AB}^{12} + \epsilon_{B,NLOS}^2 - \epsilon_{B,NLOS}^1 + \nabla \Delta e_{AB}^{12} \quad (4)$$

where $\nabla \Delta r_{AB}^{12} = \Delta r_{AB}^1 - \Delta r_{AB}^2$ and $\nabla \Delta e_{AB}^{12} = \Delta e_{AB}^1 - \Delta e_{AB}^2$. After the *DD* computation, the clock biases of both the receiver and transmitter are eliminated, as well as the correlated errors such as ionospheric and tropospheric errors, using as a reference the satellite with the highest elevation angle. As a consequence, Equation (4) becomes the following Equation (5):

$$DD_{AB}^{12}(f) = \nabla \Delta r_{AB}^{12} - \epsilon_{B,NLOS}^1 + \nabla \Delta e_{AB}^{12} \quad (5)$$

This is the formula used to calculate the pseudorange error defined in Table 2.

4. Clustering Algorithms

In this study, eight different clustering algorithms were used. Some of them, such as K++ Means and K++ Medoids, were used for historical reasons and a baseline, while some clustering algorithms were used because they were recently proposed in the literature. The driving factors used to select the algorithms were the following:

1. Different algorithms underline algorithmic concepts to support variability in the cluster label selection. The rationale is that the ensemble function (especially in the optimized weighted definition) will exploit the strengths of each clustering algorithm while minimizing the weaknesses.

2. Source code availability with the language used in this study: MATLAB from Mathworks. This is needed to support the reproducibility of the results.
3. A good degree of scalability is achieved taking into account that the GNSS data set can be composed of thousands of samples.
4. A limited set of hyper-parameters in the design of the clustering algorithm.

The list of potential clustering algorithms could easily be extended to many other algorithms, but the driver for the proposed approach was to strike a balance between computational complexity (which should be minimized and grows with the number of adopted algorithms) and clustering performance in terms of accuracy of identified samples (which may increase with a wider set of clustering algorithms). In each of the following subsections, the clustering algorithms used in this study are described including the design concept, the source code availability, the computing complexity and the potential hyper-parameters.

4.1. K++ Means

The K-means algorithm was one of the first clustering algorithms proposed in the literature [25,26] and is used here as a baseline and a comparison against more advanced algorithms. The algorithm follows a relatively simple set of steps which are described below:

1. Place K (the number of assumed clusters) points in the feature space represented by the samples that need to be clustered. These points represent the initial group centroids.
2. Assign each sample to the group that has the closest centroid.
3. When all samples have been assigned, the positions of the K centroids have been recalculated.
4. Repeat steps 2 and 3 until the centroids no longer shift in the feature space or the maximum number of iterations is reached. At this stage, the algorithm should generate a separation of the samples into groups from which the metric to be minimized (e.g., intraclass distance) can be calculated.

The distance used in the algorithm can be of different types: Euclidean distance, city-block distance, and so on. In this study, we have used the city-block distance for its simplicity in computing and because we could not find a significant difference in clustering performance. The maximum number of iterations was set to 100. The *kmeans* MATLAB function from the Statistics and Machine Learning Toolbox was used with MATLAB Release 2024b, with default settings and the number of clusters equal to 2.

The initial position of the seeds is an important element which can foster faster convergence of the algorithm. In this paper, we adopt the k-means++ algorithm [27], which uses a heuristic to find centroid seeds for k-means clustering.

The computing complexity of the k-means++ algorithm is $O(n \times k \times d \times i)$, where n is the number of samples, k is the number of clusters, d is the number of features, and i is the number of iterations to converge.

4.2. K++ Medoids

In the k-means algorithm described in Section 4.1, the center of the subset is the mean of the measurements in the subset (i.e., the centroid). In the k-medoids algorithm, the center of the subset is the median of the measurements in the subset, and it is called a medoid.

The k-medoids algorithm returns medoids that are the actual data points in the data set. This allows you to use the algorithm in situations where the mean of the data does not exist within the data set. This is the main difference between k-medoids and k-means, where the centroids returned by k-means may not be within the data set. Hence, k-medoids are useful for clustering categorical data where a mean is impossible to define or interpret [28]. As for

the K++ Means algorithm, the maximum number of iterations is 100 and the distance is the city-block distance.

The computing complexity of the K++ Medoids algorithm is similar to K++ Means.

The *kmedoids* MATLAB function *kmedoids* from Statistics and Machine Learning Toolbox was used with MATLAB Release 2024b with the default settings and the number of clusters equal to 2.

4.3. Spectral Clustering

Spectral clustering is an algorithm for clustering data points based on a graph-based algorithm, which constructs a graph, finds its Laplacian matrix, and uses this matrix to find k eigenvectors to split the graph k ways, where k is the number of clusters. We used MATLAB *spectral* cluster implementation, which computes the normalized Laplacian random-walk matrix using the method described by Shi-Malik [29].

A key point of spectral clustering is to convert distances to similarity measures using the kernel transformation to produce a similarity matrix on which the Laplacian matrix is calculated. The kernel scale was set to 2^{-5} . The graph is constructed using the nearest neighbors algorithm, which connects two points i and j , when i is the nearest neighbor of j or j is the nearest neighbor of i . This option was chosen because it leads to a denser representation of the similarity matrix. As in K-Means, the distance was city-block.

Spectral clustering typically has a computational time complexity of $O(n^3)$, which is a disadvantage of this algorithm compared to more efficient algorithms.

The *spectralcluster* MATLAB function from Statistics and Machine Learning Toolbox was used with MATLAB Release 2024b with the default settings and the number of clusters equal to 2.

4.4. DBSCAN

The DBSCAN algorithm was introduced by the authors in 1996 in [30] and is a density-based clustering algorithm. The algorithm groups together a given a set of points in some space by distinguishing the points (i.e., samples), which are closely packed (points with many nearby neighbors) and the outlier samples that lie alone in low-density regions (those whose nearest neighbors are too far away). In this study, the *DBSCAN*.function from the MATLAB Machine Learning toolbox was used to implement DBSCAN. DBSCAN has two main parameters: the threshold for a neighborhood search radius ϵ and the minimum number of neighbors Min_{pts} required to identify a core point. In this study, these parameters were set respectively to 2.5 and 8 because they provided the best performance in the N_O portion of the data set. The distance was set to city block distance. The computing complexity of DBSCAN is, in general, $O(n \times \log n)$.

The MATLAB *dbscan* function was used with the parameter $\epsilon = 0.5$ and the minimum number of neighbors required for a core point equal to 10.

4.5. Gaussian Mixture

Gaussian Mixture Models (GMMs) are probabilistic, unsupervised learning models that cluster data by assuming that it originates from a finite mixture of Gaussian distributions with unknown parameters. GMMs are capable of providing soft clustering with probabilistic membership [31]. In this study, we used the Expectation-Maximization (EM) algorithm to estimate cluster parameters in GMM.

The computing complexity of GMM is $O(t \times n \times k \times d^3)$ where t is the number of iterations, n is the number of samples, k is the number of clusters and d is the dimensionality of the data. In this study, the *fitgmdist* MATLAB function was used to implement GMM with the default settings and the number of clusters equal to 2.

4.6. Kernel-Based Clustering via Isolation Distributional Kernel

The Kernel-based Clustering algorithm via Isolation Distributional Kernel (called IDKC in the rest of this paper) was introduced by the authors in [32]. It addresses the limitations of existing kernel-based clustering algorithms that they usually produce weaker clustering outcomes than density-based clustering, and they have high computational cost. The authors contend that these limitations are mainly due to the use of (a) the Expectation-and-Maximization algorithm as an optimization procedure, and (b) a non-adaptive kernel. Then, the authors of [32] propose the first clustering algorithm that employs an adaptive distributional kernel without any optimization, while achieving a similar optimization objective function. The algorithm uses kernel mean embedding (KME) to define a distribution kernel to measure the similarity between two distributions to implement the clustering algorithm, but instead of a conventionally used Gaussian kernel, the authors have adopted the Isolation Kernel because it was proven to perform better than the Gaussian kernel in [33]. The most significant parameter to adjust the algorithm is the ϕ parameter of the Isolation Kernel. The computing complexity of the IDKC is $O(s^2 + n(bk + d)\phi)$, where s is the number of seeds of the cluster (a limited number especially in this case), d , k , b are the numbers of dimensions, clusters and iterations, respectively, and ϕ is the parameter of the Isolation Kernel. The implementation code of the IDKC algorithm written in MATLAB is publicly available at <https://github.com/zhu ye88/IDKC> (accessed on 3 July 2026). The used parameters were $\psi = 100$, $t = 100$, number of clusters equal to 2, $v = 0.9$ and s equal to number of samples for each satellite.

4.7. Torque Clustering

The torque clustering algorithm was recently introduced by the authors in [34]. It is based on the intuitive principle that a cluster should merge with its nearest neighbor with a higher mass unless both clusters have relatively large masses and the distance between them is also substantial. By identifying peaks in mass and distance, the torque clustering algorithm effectively detects and removes incorrect mergers. One of the advantages of the proposed algorithm is that it is entirely parameter-free, allowing it to autonomously recognize various cluster types, determine the optimal number of clusters, and identify noise. The computing complexity of the torque clustering algorithm is nominally $O(n^2)$, but by using a fast approximate nearest neighbor, which avoids the computation of the distance matrix, the time complexity of the torque becomes approximately $O(n \log n)$. The other advantage of the torque clustering algorithm is robustness to noisy samples, which is particularly important in the processing of GNSS noisy data. The MATLAB code for the torque clustering algorithm is publicly available at <https://github.com/JieYangBruce/TorqueClustering> (accessed on 3 July 2026). The Minkowsky distance was used to create the distance matrix.

4.8. Density Peak Clustering Based on Nearest Neighbors

The Density peak clustering based on nearest neighbors (DPC-AKNN) presented in [35] tries to address some limitations of the the density peak clustering (DPC) algorithm: (1) sensitivity to density kernels and involved parameters, and (2) its non-central data allocation method, which may result in error propagation. In the rest of this paper, this algorithm will be simply called AKNN.

The AKNN is based on the idea to use a Gaussian kernel constrained by K-nearest neighbors and to determine the cutoff distance based on local data distribution adaptively. To address the problem of non-central data allocation, the algorithm avoids error propagation by using the K-nearest neighbors of allocated data and unallocated data alternately. Starting from the cluster centers, the algorithm searches the k-nearest neighbors of allocated data points, and include the qualified nearest neighbors into clusters. In a subsequent

step, for each unallocated data point, the algorithms decide its belonging based on the allocated data points in its k-nearest neighbors. Finally, the algorithm does a re-allocation of some allocated data points based on the allocated data points in their k-nearest neighbors, to improve the clustering accuracy further.

The computing complexity of the algorithm is $O(n_c \times n^2)$, where n_c is the number of clusters and n is the number of samples in the data set. The MATLAB code of AKNN used in the study is available on request from the authors of [35].

The used parameters were $k = 3$, $\gamma = 1$ and $\eta = 10$.

5. Meta-Heuristics Algorithms

Several MHA algorithms were used to evaluate the approach in the two chosen data sets. The literature on MHA algorithms is quite vast, as it spans decades of research and hundreds of algorithms. The algorithms MHA were selected on the basis of three main criteria:

1. recent publication in the literature,
2. proven performance for a high dimensional data set (like the ones used in this study)
3. public implementation of the algorithm in MATLAB for reproducibility.

The results of the selection are mainly focused on nature-inspired algorithms based on swarm optimisation concepts, as they are particularly suitable for high dimensional data sets and are usually considered quite efficient in terms of convergence to the optimal solution [36].

To complement the algorithms presented in this section, Section 5.7 discusses other potential MHAs, which can be used for this purpose.

5.1. Grey Wolf Algorithm

The GWO was initially presented by the authors of [37] and is inspired by the hunting behavior of gray wolves (*Canis lupus*), which is based on a leadership hierarchy with four types of gray wolves such as alpha, beta, delta, and omega and different phases of hunting, searching for prey, encircling prey, and attacking prey. The most powerful wolf is alpha, which leads the entire pack in hunting, migration, and feeding. When the alpha wolf is not in the pack, ill, or dead, the strongest wolf from the beta wolf family takes the lead of the pack. The power and domination of delta and omega are less than alpha and beta. This social hierarchical structure is the main inspiration of the GWO algorithm together with the hunting approach structured in different phases, as mentioned above.

The advantages of the GWO algorithms are a good generalization capability for different domains, it has very few parameters, no derivation information is required in the initial search, and it has a special capability to strike the right balance between exploration and exploitation during the search, which leads to a favorable convergence rate to the optimal solution [37].

On the other hand, some limitations of GWO have been identified in the recent research literature, including the probability of falling to a local minimum [38] or a lack of balance between the exploration and exploitation processes [39].

The computational complexity of GWO is $O(N \times T \times D)$ where N is the number of agents (i.e., wolves), T is the number of iterations, and D is the dimension of the specific problem.

The MATLAB implementation of GWO, which was used in this paper, is available in <https://github.com/adajani/gwo> (accessed on 3 July 2026).

5.2. Dynamic Opposition-Based Learning Grey Wolf Optimizer Algorithm

The algorithm DOLGWO is an improved version of the Grey Wolf Optimizer (GWO), which was introduced by the authors of [38]. The enhancement introduced in DOLGWO is to use dynamic opposition-based learning (DOL), which is a new variant of the opposition-based learning (OBL) strategy, to address the issue of premature GWO convergence to local minima. In DOLGWO, this is implemented by learning from the opposite points in an asymmetric and dynamic search space to better explore the search space. The DOLGWO algorithm expands the search space by generating new starting points to update the search population. The key parameters of DOLGWO are a jumping rate, which is used to update the population of the search agents, and a weighting factor w , which is a positive factor that aims to balance the ability to explore and exploit. The main hyper-parameters are the same as in the GWO: the number of search agents N and the maximum number of iterations T as the jump rate and the weighting factor are set to the values recommended in the algorithm. The MATLAB implementation of the DOLGWO algorithm, which was used in this paper, can be retrieved by the authors of [38].

5.3. Parrot Optimization Algorithm

The optimization algorithm Parrot Optimizer (PO) was recently introduced (2024) and is inspired by the behavioral traits of *Pyrrhura Molinae*, a well-liked parrot species [40]. The algorithm is based on four different traits: foraging, staying, communicating, and a fear of strangers. In particular:

1. In foraging behavior, the parrots estimate the approximate location of food primarily by observing the food's location or by considering the owner's position and then fly to the identified location.
2. The staying behavior is when the parrot flies to the owner's body and stays there for a certain period.
3. The communication behavior may be both the action of flying to the flock or communicating to the flock. In the adopted implementation, both behaviors are assumed to occur with equal probability, and the mean position of the current population is employed to symbolize the center of the flock.
4. In fear of strangers behavior, the parrots tend to distance themselves from unfamiliar individuals and seek safety by flying to their owners.

The algorithm PO is based on the interaction of the four behaviors, which are implemented in four distinct formulas to facilitate the search for optimal solutions. The authors of the PO algorithm claim in [40] that the use of all four behaviors together provides an advantage in opposition to traditional MHAs that are based only on separate exploration and exploitation phases because each agent (e.g., the parrot) randomly uses one of these four behaviors during each iteration and significantly improves population diversity. The deviation from the common (in the majority of MHAs) two-phase structure of exploration-exploitation may support the PO algorithm to implement a more effective mitigation of the risk of trapping in local optima, which is a common problem in MHAs.

The computing complexity of the PO algorithm is indicated in [40] as $O(N \times T) + (N \times T \times D + 1) + O(T \times N \times \log(N))$, where N is the number of agents (i.e., hawks), T is the number of iterations, and D is the dimension of the specific problem. The parameters to tune are N and T . The MATLAB implementation of the PO algorithm, which was used in this paper, is available at <https://github.com/junbolian/PO> (accessed on 3 July 2026).

5.4. Moss Growth Algorithm

The Moss Growth Optimization (MGO), introduced in [41], is an algorithm inspired by the moss growth in the natural environment. The MGO draws inspiration from

asexual reproduction, sexual reproduction, and vegetative reproduction of moss, where spore dispersal search and dual propagation search are proposed for exploration and exploitation, respectively.

Taking inspiration from the dispersal mechanism of moss growth spores based on the wind direction, the MGO algorithm defines the evolutionary direction of the search population to partition the population of search agents in an optimal way. In addition, the cryptobiosis mechanism alters the traditional metaheuristic algorithm's approach of directly modifying individuals' solutions, preventing the algorithm from getting trapped in local optima. The computing complexity formula of MGO was not provided in [41], but the computing time results presented in [41] show that it has quite similar computing times to the GWO and HHO (also considered in the study). The MATLAB implementation of the MGO, which was used in this paper, is publicly available at: <https://github.com/ali-asgharheidaricom/The-Moss-Growth-Optimization-MGO-Concepts-and-performance> (accessed on 3 July 2026).

5.5. Puma Optimization Algorithm

The Puma Optimizer (PO) is an optimization algorithm inspired by the intelligence and life of pumas [42]. This algorithm uses a new mechanism to change the phases of exploration and exploitation, which can be automatically adjusted to the nature of the problem. The phase change is inspired by the intelligence of the pumas and their degree of experience. In the first approach, the pumas do not have enough experience or energy, and they try to explore unfamiliar territory to find prey simultaneously. After passing three generations, pumas have an acceptable experience in deciding to change the phase, and in the continuation of the iterations, they choose only one phase for the optimization operation. In the experienced phase, three different functions f_1 , f_2 , and f_3 are used for scoring. The first function f_1 emphasizes the escalation component and leads to the priority of one of the two phases of exploration and exploitation that have been selected and performed better than each other. The second function also emphasizes the resonance component and leads to a phase that performs better than the other prioritized phases. The third function in the selection mechanism emphasizes the diversity component and leads to the fact that the phase that is not selected in many repetitions also has a chance to be selected because the selection of one phase leads to collapsing into the local optimal trap. As in other MHA, there is an exploration and exploitation phase. The computing complexity of the puma algorithm is $O((2T + 1)N^2 \times N \times D)$. T is the maximum number of iterations, D is the dimension of the problem, and N is the total number of the population.

The MATLAB implementation of the Puma optimization algorithm, which was used in this paper, is available to the public at <https://ch.mathworks.com/matlabcentral/fileexchange/157231-puma-optimizer-po> (accessed on 3 July 2026).

5.6. Python Snake Optimization Algorithm

The Python Snake Optimisation Algorithm (PySOA) simulates the Python snake hunting mechanism in nature [43]. These reptiles are not poisonous, but they hunt their prey through ambushes. They can detect their prey using senses such as smell, eyesight, and infrared vision. The hunting mechanism consists of three major phases: searching for prey, scanning for prey, and attacking the prey. The searching-for-prey step contributes to exploration, while attacking prey is dedicated to exploitation, and scanning for attack enhances the balance between the two. The computing complexity of PySOA is $O(T \times N \times D)$. T is the maximum number of iterations, D is the dimension of the problem, and N is the total number of the population. Then, PySOA has a similar asymptotic efficiency as other

lightweight metaheuristics such as GWO, but introduces a more flexible balance between exploration and exploitation without increasing time or memory requirements.

The MATLAB implementation (Rel 2024b) of the Python Snake Optimization Algorithm, which was used in this paper, is public available at <https://it.mathworks.com/matlabcentral/fileexchange/175654-a-novel-bio-inspired-python-snake-optimization-algorithm> (accessed on 3 July 2026).

Additional details on the PySOA algorithm are provided in [44].

5.7. Analysis of Supplementary MHAs for Future Developments

This section discusses other potential MHAs that were not considered in this study but could be adopted in future developments of this study. Various MHAs studies have been published in [45–48]. These surveys identify hundreds of MHAs belonging to different families, including genetic algorithms, nature inspired algorithms, quantum inspired algorithms, and so on. This field of research is constantly evolving with new MHAs appearing in the literature every year on the basis of new concepts or new animal behaviors. In this study, we focus specifically on nature-inspired algorithms due to the problem to be addressed (GNSS LOS/NLOS classification), which is characterized by highly dynamic and noisy conditions due to vehicle movement in the urban environment. As a consequence, we exploit the strength of nature inspired algorithms, which are able to readily adapt to changing variables without requiring comprehensive knowledge of the underlying mathematical system [47]. In addition, nature-inspired algorithms are excellent at avoiding getting stuck in local optima, frequently scanning the entire search space to locate the global best solution [45,46]. This aspect excluded algorithms such as genetic algorithms and their recent variants such as real-coded genetic algorithms [15,16,49] and its variant in a federated learning context [17]. Other MHAs such as Differential Evolution (DE) were also excluded, because their performance is dependent on various parameters (e.g., mutation factor F , crossover rate CR) which must be manually tuned for specific problems [50]. This would significantly expand the analysis due to the large number of satellites to be addressed in the UrbanNav data set and more generally for any GNSS LOS/NLOS classification problem. In this context, PSO is identified as one of the most popular swarm optimization algorithms in the literature [45], but we prefer to adopt more recent variants (PSO was presented more than 20 years ago) such as GWO and its variant DOLGWO. We also preferred to use the specific algorithm (e.g., PO, GWO) rather than hybrid variants to optimize the computing efficiency considering that the data from 57 satellites have to be analyzed.

6. Results

This section provides the results of the approach described in Section 3 with the accuracy metric. The results are related to all the 57 satellites used in the data set described in Section 3.2.

Figure 3 shows the mean classification accuracy across the 57 satellites in the data set for the different clustering algorithms, the basic ensemble and the adaptive weighted sets using the different MHA used in this study. It can be seen from Figure 3 that the MHA-based ensembles clearly outperform the single clustering algorithms and also the basic ensemble. These results show the validity of the proposed approach because the MHAs in the training phase are able to find a combination of weights for the different outcomes of the clustering algorithms to reduce the classification error. Among the MHAs, the DOLGWO performs slightly better than the other MHAs, particularly the GWO from which it is derived. The GWO, PO, and the very recently introduced PySOA do also provide a very good performance. Among the clustering algorithms, DBSCAN has a much better classification accuracy than the other algorithms with spectral clustering in the second

place. The performance of the clustering algorithm depends very much on the structure of the data and on the basis of the 'free lunch' theorem [51], no clustering algorithm or MHA is able to outperform other algorithms on all data sets.

The results presented in Figure 3 using the mean among the results of all satellites are confirmed by the results using the median as presented in Figure 4. As for the mean, the DOLGWO is able to outperform the other MHA adaptive weighted approaches, with GWO, PO, and PySOA as close seconds.

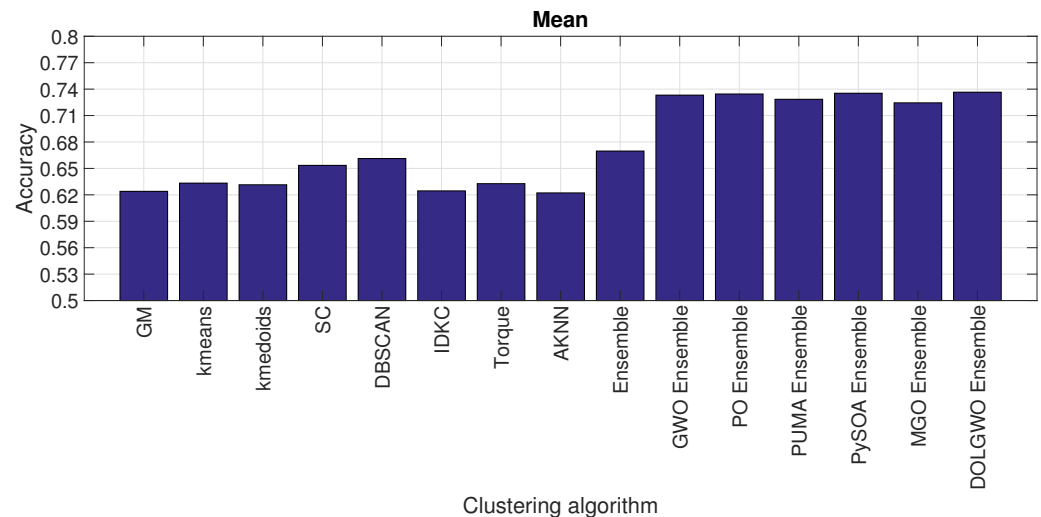


Figure 3. Mean classification accuracy.

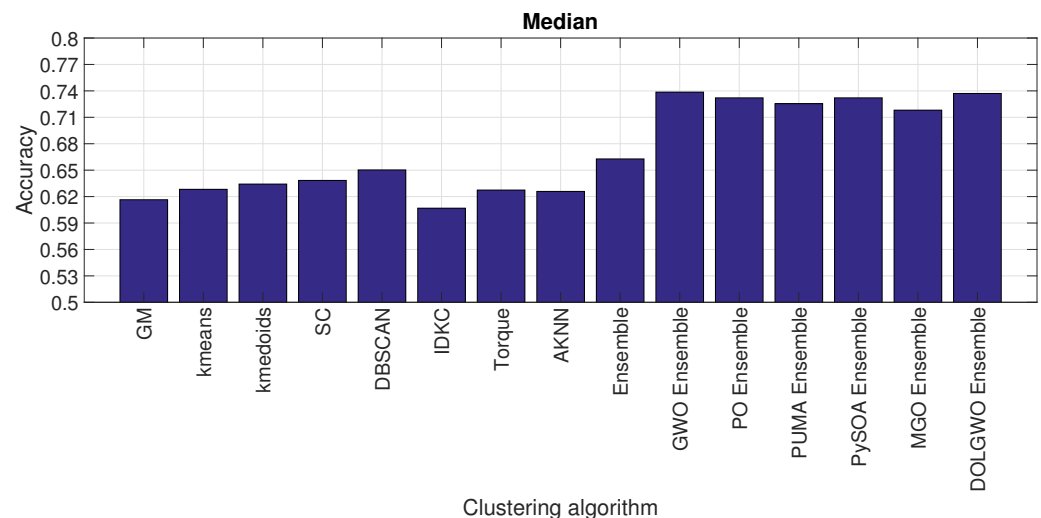


Figure 4. Median classification accuracy.

The results presented graphically in the previous Figures 3 and 4 are also summarized in the following Table 3 together with the values provided by the trimmed mean with the percentages 10% and 20%. The trimmed mean excludes the highest and lowest k data values with $k = N_{sat} \times (\text{percent}/100)$, where $N_{sat} = 57$ is the number of satellites analyzed and the percentage can have values of 10 and 20.

The results presented in Table 3 confirm that the MHA-ensembled approach significantly improves the classification accuracy of the GNSS LOS/NLOS condition compared to the specific clustering algorithms and the ensemble. The bold values represent the highest values for all the methods. It can be seen that the MHA-based ensemble methods are similar with a very minor prevalence of the DOLGWO algorithm.

Table 3. Summary of the accuracies obtained with different approaches using mean, median, and trimmed mean with outlier percentage 10% and 20%. The values highlighted in bold are the best performance metrics.

Algorithm	Mean	Median	Trimmed Mean (10%)	Trimmed Mean (20%)
GM	0.624	0.6164	0.6315	0.6172
K-means	0.6333	0.6283	0.6281	0.6244
K-medoids	0.6315	0.6342	0.6272	0.6265
SC	0.6535	0.6384	0.6491	0.6457
DBSCAN	0.6612	0.6503	0.6566	0.6531
IDKC	0.6245	0.6068	0.6186	0.6154
Torque	0.6327	0.6274	0.6284	0.6252
AKNN	0.6222	0.6259	0.6182	0.6175
Ensemble	0.6697	0.6627	0.6683	0.6672
GWO ensemble	0.7332	0.7386	0.7330	0.7316
PO ensemble	0.7345	0.7321	0.7352	0.7333
PUMA ensemble	0.7285	0.7255	0.7289	0.7288
PySOA ensemble	0.7354	0.7321	0.7356	0.7341
MGO ensemble	0.7245	0.7181	0.7246	0.7243
DOLGWO ensemble	0.7365	0.7371	0.7375	0.7375

The previous results for the mean and median across the satellites are also confirmed by an evaluation of the distribution of the results for the best accuracy across the satellites among the different algorithms. The following figures show such a distribution for each of the considered MHAs. Figure 5 shows the number of occurrences for each clustering algorithm, the ensemble, and the adaptive weighted approach DOLGWO MHA. The results show that most of the occurrences are with the DOLGWO MHA algorithm. Similar results are obtained, respectively, with PO, PUMA, PySOA, and MGO, respectively, in Figures 6–9.

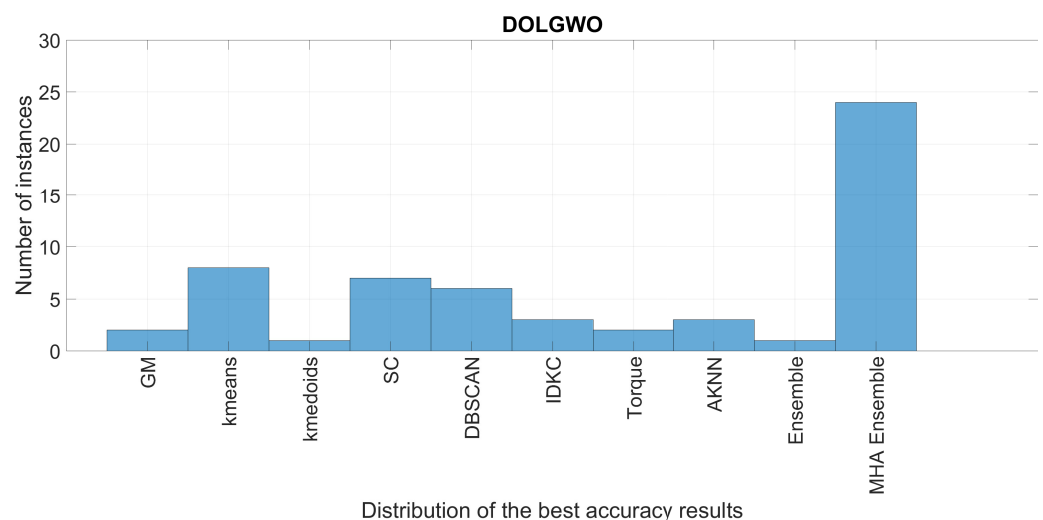


Figure 5. DOLGWO distribution.

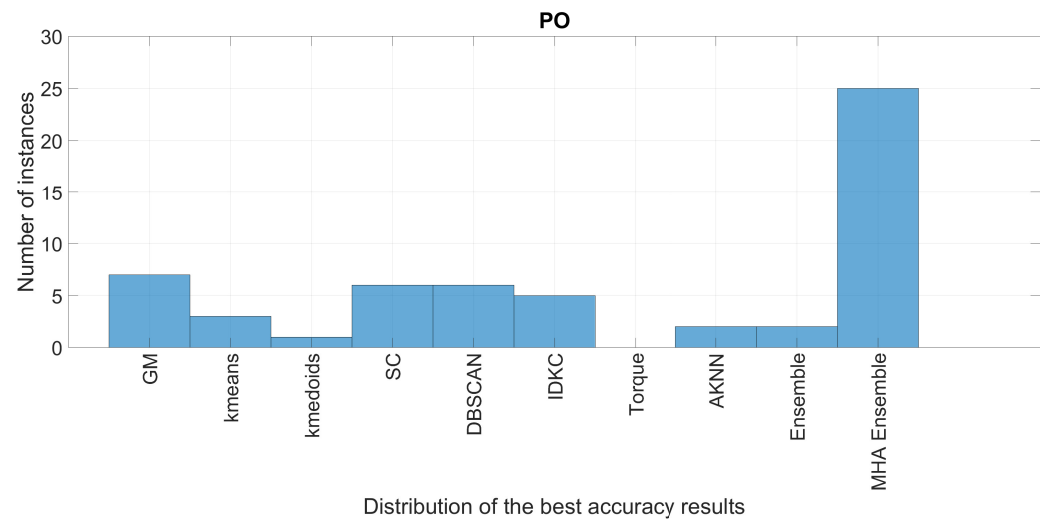


Figure 6. PO distribution.

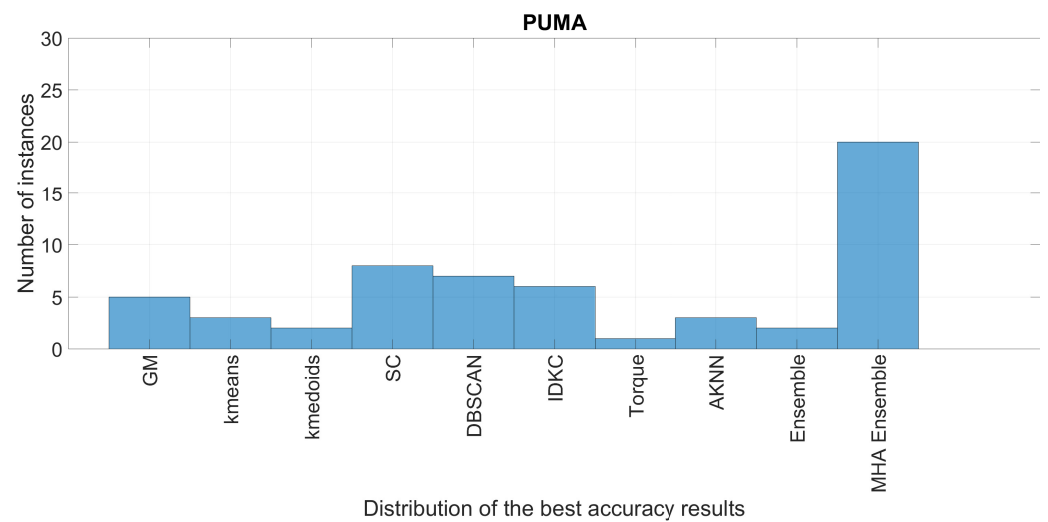


Figure 7. PUMA distribution.

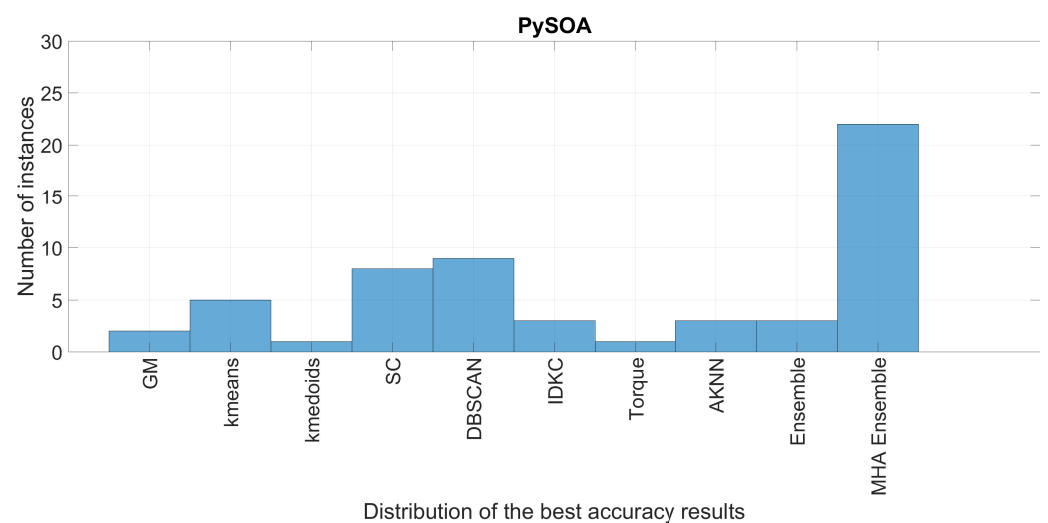


Figure 8. PySOA distribution.

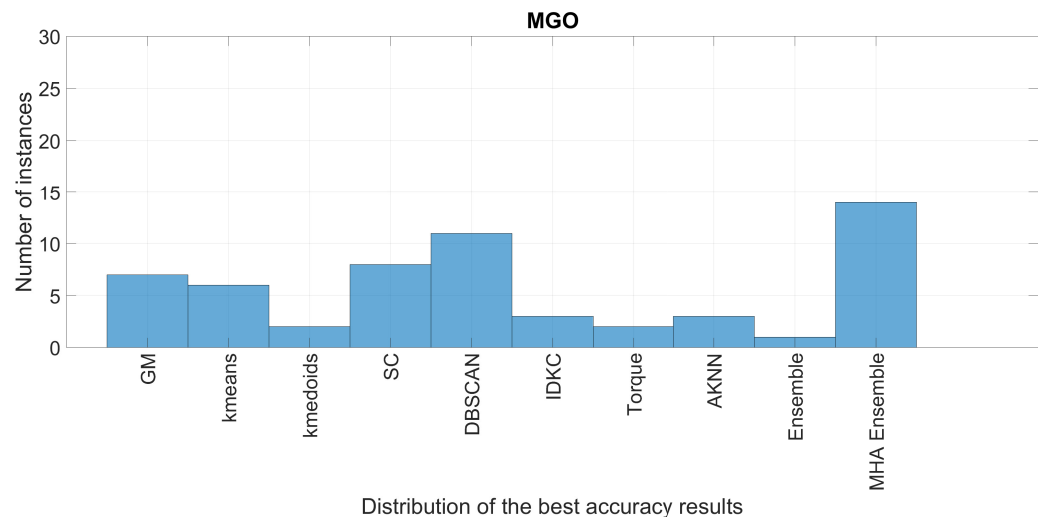


Figure 9. MGO distribution.

The previous figures have shown the performance across all the satellites, but it is also useful to evaluate the performance of the proposed approach across specific satellites. Because the number of satellites considered is significant (i.e., 57), we only show a partial view of the satellites with two figures showing two different sets of satellites for reasons of space. We remind the reader that the prefix G means the USA GPS constellation, R the Russian Glonass constellation, E the European Galileo constellation, and C the Chinese Beidou constellation.

Figure 10 shows the performance of the various clustering algorithms and the ensemble algorithms for specific satellites: G28, R16, E21, C7. In general, the MHA-based ensemble clustering algorithms are able to outperform the specific clustering algorithms, in particular, for satellite C7. Depending on the satellite id, each MHA may slightly outperform the others, even if the DOLGWO algorithm has on average a high accuracy, in particular for a satellite G28. Specific clustering algorithms also have good performance reaching almost the MHA ensemble algorithms for selected satellite ids (e.g., DBSCAN for satellite E21, IDKC for satellite G28 and Torque for satellite C7), but their performance is uneven across the satellites, which is the reason why the mean and median results presented in Figures 3 and 4, respectively, have a lower performance compared to the MHA ensemble approach.

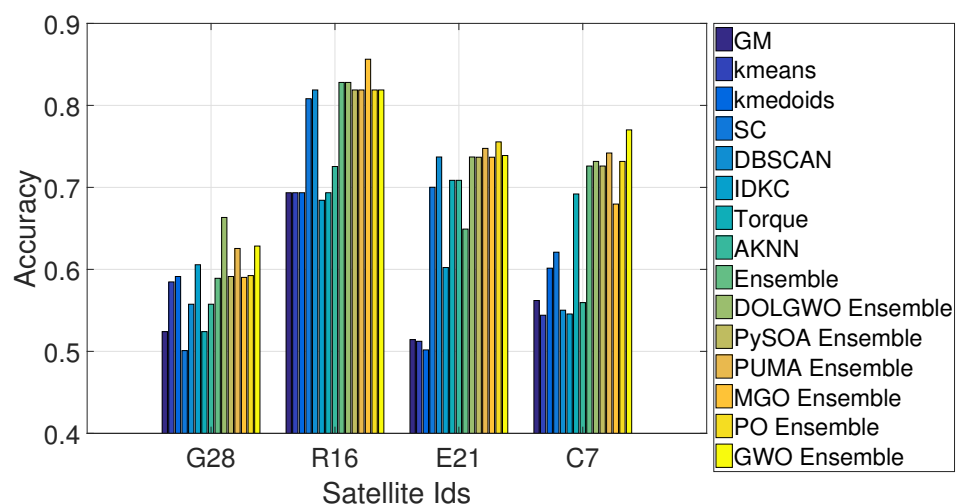


Figure 10. Selected satellites (G28, R16, E21, C7) clustering performance against the ground truth.

Similarly, Figure 11 shows the performance of the various clustering algorithms and the ensemble algorithms for the specific satellites: G21, R10, E13, C27. The results are similar to the previous figures even if it shows the great variability in the optimal clustering algorithm and the optimal MHA. In this second set, the DOLGWO algorithm has optimal performance because it is the best MHA for satellites E13 and C27 and it is second in G21 and R10 after MGO. The good performance of the spectral clustering and DBSCAN algorithms for satellites E13 and C27 is also noted, even if they are inferior in performance to the ensemble approaches.

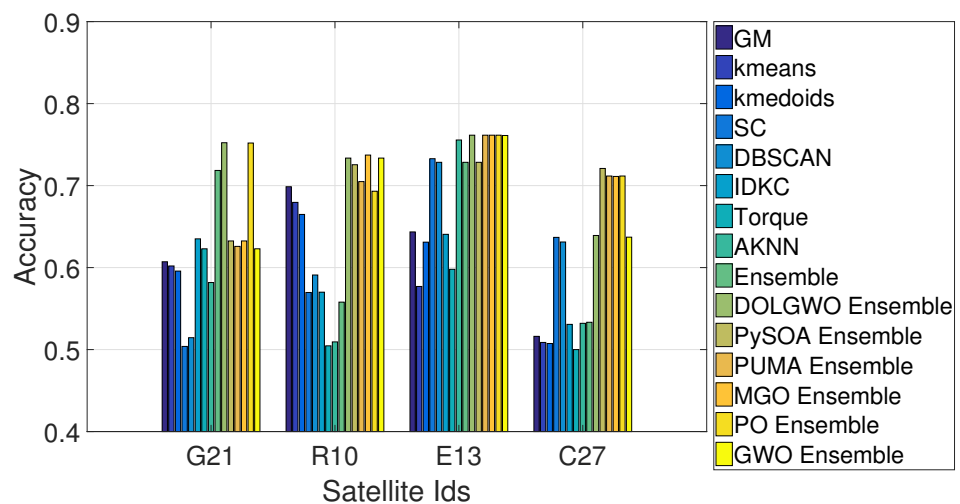


Figure 11. Selected satellites (G21, R10, E13, C27) clustering performance against the ground truth.

To complement the previous figures providing the accuracy, the following Figures 12–15 present the confusion matrices for the GWO ensemble, PUMA ensemble, K-means, and DBSCAN, respectively, for the satellite E26. The confusion matrices confirm the previous results that the MHA ensembles provide a stronger performance than the specific clustering algorithms.

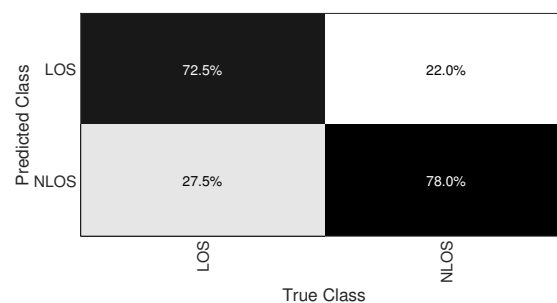


Figure 12. Confusion matrix for the GWO ensemble and satellite E26.

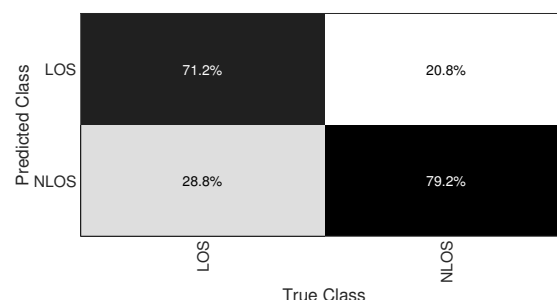


Figure 13. Confusion matrix for the PUMA ensemble and satellite E26.

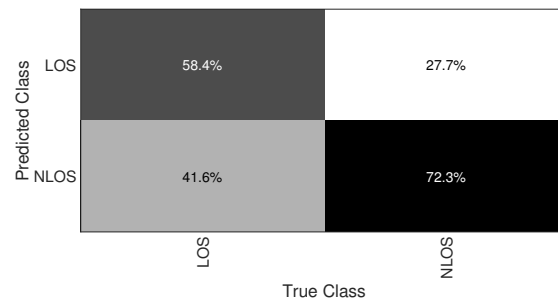


Figure 14. Confusion matrix for K-Means and satellite E26.

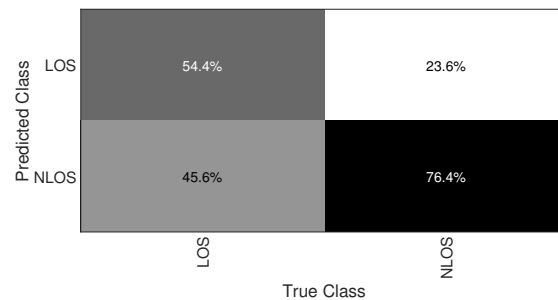


Figure 15. Confusion matrix for DBSCAN and satellite E26.

An important aspect related to the application of MHA is the convergence of the algorithm. In this study, we used the same configuration of number of agents and number of iterations for consistency. Figure 16 shows the convergence trends for selected GNSS satellites in the number of iterations. It can be seen that all the MHA convergence in a relatively small number of iterations to the final values. The results presented in Figure 16 show that the number of iterations and the number of agents were chosen with an appropriate range.

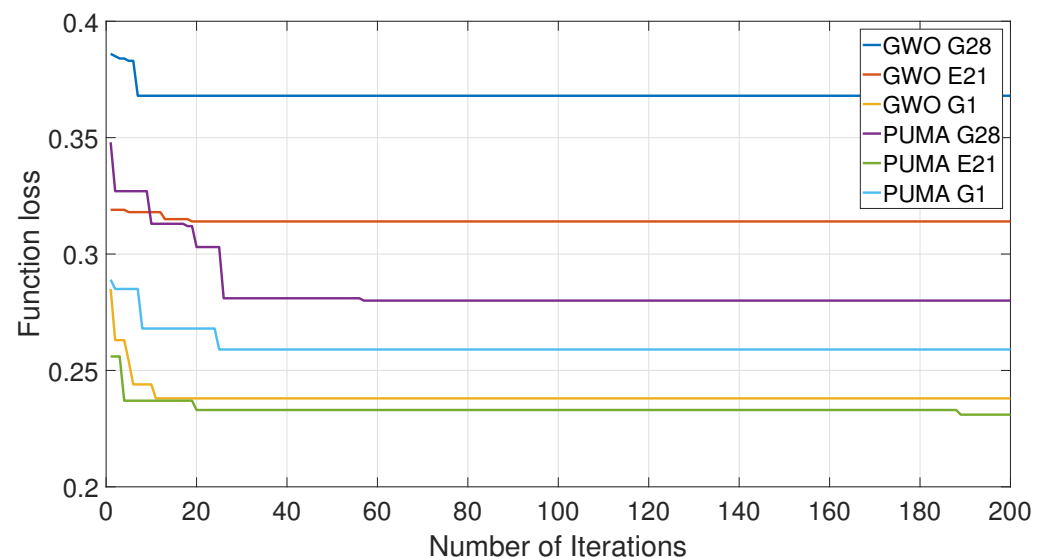


Figure 16. Convergence trends of selected meta-heuristics algorithms for specific satellites.

Finally, in Table 4, we present the computing times of the different steps of the proposed approach for the specific satellite C7, which has the largest number of samples in the study. The computing times are the results of the average of 10 execution times of the specific algorithm on the computing platform described above in Section 3.1.

Table 4. Computing times for the execution in seconds of the different clustering algorithms and the MHAs for satellite C7 on the computing platform described in Section 3.1.

Algorithm	Computing Time (Seconds)
GM	0.09
K-means	0.069
K-medoids	0.206
SC	0.8
DBSCAN	0.13
IDKC	5.4
Torque	2.99
AKNN	1.5
GWO	29.8
DOLGWO	34.8
MGO	20.08
PO	59.81
PUMA	91.12
PySOA	97.42

The computing times shown in Table 4 show that most of the computing time is taken by the MHA compared to the clustering algorithm. This is to be expected, considering that the MHA has to search the large space of the weights to identify the optimal weights. It should be noted that the MGO algorithm is faster in time than the other MHAs. This may be due to a faster convergence. The results also show that the computing time of some algorithms (K-means, K-medoids, GM, and DBSCAN) is much faster than some other clustering algorithms such as IDKC and torque clustering algorithms, but this may be due to the efficient implementation of native MATLAB algorithms (part of the official MATLAB machine learning toolbox), while torque and IDKC are open-source libraries, which may not be optimised. As a consequence, the empirical comparison of the computing times shown in Table 4 may be affected by this aspect. This is not the case for the MHAs, which are based on an open-source implementation.

7. Conclusions and Future Developments

This paper has proposed a semi-supervised learning approach for GNSS LOS/NLOS classification, which is based on an ensemble clustering approach, where the supervised phase is used to tune the weights contributing to the ensemble implementation (adaptive weighting). Once the weights are optimized using the meta-Heuristics Algorithm (e.g., Grey Wolf), they are used in the unsupervised learning phase. The results on a GNSS LOS/NLOS public data set where the ground truth is available (e.g., label information) show that the proposed approach is capable of significantly outperforming the specific clustering approaches by a large margin (more than 7%). Because the approach is applied to the different 57 GNSS satellites and the improvements are coherent across the satellites, the proposed approach shows a significant generalization capability. It is also shown that the different MHA algorithms have slightly different performances in all satellites. As a consequence, there is no MHA, which is superior to the others for all the satellites even if the Dynamic Opposite Learning assisted Grey Wolf Optimization (GWO) algorithm (a recent improvement of the GWO) has good performance across all the satellites, with the Parrot Optimization algorithm a close second.

Future developments could expand the analysis to different data sets for GNSS LOS/NLOS once they are made public. At the time of writing this document, this was the only public data set that satisfied the requirements of the study. Even if the number of clustering algorithms and MHAs adopted in this study is larger, other recent algorithms could also be applied to evaluate their performance. Finally, the approach is quite generic and can be extended to other LOS/NLOS classification problems in the wireless communication domain.

Author Contributions: Conceptualization, G.B.; methodology, G.B. and G.B.; validation, G.B. and F.B.; writing—G.B.; writing—review and editing, G.B. and F.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The paper was internally approved in the EC DG JRC Pubsy system. Ethical review and approval were waived for this study because it is not applicable as the study did not involve humans or animals.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this study is public available at <https://www.kaggle.com/competitions/gnss-classification/data> (accessed on 3 July 2026).

Conflicts of Interest: Author Fausto Bonavitacola was employed by the company Pikel (Italy). The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

C/N0	Carrier-to-Noise ratio
DD	Double Difference
DE	Differential Evolution
DL	Deep Learning
DOLGWO	Dynamic Opposition Grey Wolf Optimization
DPC	Density peak clustering
DPC-AKNN	Density peak clustering based on nearest neighbors
GMM	Gaussian Mixture Models
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GWO	Grey Wolf Optimization
IDKC	Isolation Distributional Kernel
LOS	Line of Sight
MGO	Moss Growth Optimization
MHA	Meta Heuristics Algorithm
ML	Machine Learning
NLOS	Non Line of Sight
PO	Parrot Optimizer
PRN	Pseudo-Random Noise
PSOA	Python Snake Optimization Algorithm
RYNEX	Receiver INdependent EXchange format
UTC	Coordinated Universal Time

References

1. Xu, P.; Zhang, G.; Yang, B.; Hsu, L.T. Machine learning in GNSS multipath/NLOS mitigation: Review and benchmark. *IEEE Aerosp. Electron. Syst. Mag.* **2024**, *39*, 26–44. [\[CrossRef\]](#)
2. Li, L.; Elhajj, M.; Feng, Y.; Ochieng, W.Y. Machine learning based GNSS signal classification and weighting scheme design in the built environment: A comparative experiment. *Satell. Navig.* **2023**, *4*, 12. [\[CrossRef\]](#)
3. Hsu, L.T.; Kubo, N.; Wen, W.; Chen, W.; Liu, Z.; Suzuki, T.; Meguro, J. UrbanNav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021)*; The Institute of Navigation: Manassas, VA, USA, 2021; pp. 226–256.
4. Sowmya, B.S.; Ipsitha, M.; Ammana, S.R.; Sridher, T.; Jyothsna, K.S.; Devanaboyina, V.R. Comparative Analysis of SVM and SNN Models for LOS/NLOS GNSS Signal Classification. In *Proceedings of the 2025 Third International Conference on Emerging Applications of Material Science and Technology (ICEAMST)*; IEEE: New York, NY, USA, 2025; pp. 1–6.
5. Yao, Q.; Yang, J.; Wang, S.; Xiao, Z.; Huang, M. Few-Sample GNSS Signal LOS/NLOS Identification via Physical-Semantic Distillation from Large Language Models. *IEEE Signal Process. Lett.* **2026**, *33*, 2365–2369. [\[CrossRef\]](#)
6. Sun, Y.; Li, S.; Fu, L.; Yin, L.; Deng, Z. NICL: Non-line-of-sight identification in global navigation satellite systems with continual learning. *IEEE Trans. Veh. Technol.* **2024**, *74*, 2480–2490.
7. Shukla, A.K.; Sinha, S.A. Unsupervised machine learning approach for multipath classification of NavIC signals. In *Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022)*; The Institute of Navigation: Manassas, VA, USA, 2022; pp. 2618–2624.
8. Savas, C.; Doyis, F. Multipath Detection based on K-means Clustering. In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*; The Institute of Navigation: Manassas, VA, USA, 2019; pp. 3801–3811.
9. Wang, H.; Pan, S.; Gao, W.; Xia, Y.; Ma, C. Multipath/NLOS detection based on K-means clustering for GNSS/INS tightly coupled system in urban areas. *Micromachines* **2022**, *13*, 1128. [\[PubMed\]](#)
10. Lin, K.q.; Deng, Z.l.; Yin, L. Effective multipath mitigation methods for RTK in urban environments. In *Proceedings of the China Satellite Navigation Conference*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 565–576.
11. Uaratanawong, V.; Satirapod, C. Comparative analysis of pseudorange multipath mitigation performance using K-means and Fuzzy c-means clustering techniques. *J. Appl. Geod.* **2025**, *19*, 713–735. [\[CrossRef\]](#)
12. Burton, R.J.; Cuff, S.M.; Morgan, M.P.; Artemiou, A.; Eberl, M. GeoWaVe: Geometric median clustering with weighted voting for ensemble clustering of cytometry data. *Bioinformatics* **2023**, *39*, btac751. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Zhu, Z.; Xu, M.; Ke, J.; Yang, H.; Chen, X.M. A Bayesian clustering ensemble Gaussian process model for network-wide traffic flow clustering and prediction. *Transp. Res. Part C Emerg. Technol.* **2023**, *148*, 104032. [\[CrossRef\]](#)
14. Huang, H.; Gao, W.; Pan, S.; Gan, Y.; Zhao, Q.; Tao, X. ISVEC: Unsupervised ensemble clustering with iterative semantic voting for GNSS LOS/NLOS identification. *GPS Solut.* **2026**, *30*, 105. [\[CrossRef\]](#)
15. Beklaryan, G.L.; Akopov, A.S.; Khachatryan, N.K. Optimisation of system dynamics models using a real-coded genetic algorithm with fuzzy control. *Cybern. Inf. Technol.* **2019**, *19*, 87–103. [\[CrossRef\]](#)
16. Chen, X.F.; Shi, Z.k. Real-coded genetic algorithm for signal timing optimization of a single intersection. In *Proceedings of the International Conference on Machine Learning and Cybernetics*; IEEE: New York, NY, USA, 2002; Volume 3, pp. 1245–1248.
17. Chaudhary, R.K.; Kumar, R.; Saxena, N. Ameliorating clustered federated learning using real-coded genetic algorithm. *Clust. Comput.* **2025**, *28*, 415. [\[CrossRef\]](#)
18. Bueno-Ferrer, Á.; De Pablo Valenciano, J.; De Burgos Jiménez, J. Unveiling the potential of metaheuristics in transportation: A path towards efficiency, optimization, and intelligent management. *Infrastructures* **2024**, *10*, 4. [\[CrossRef\]](#)
19. Elzein, A.; Atasever, U.H.; Abdalla, A. A metaheuristic perspective on the accuracy and stability of GNSS network adjustment. *Surv. Rev.* **2026**, 1–15. [\[CrossRef\]](#)
20. Gao, G.; Li, L.; Shi, Q.; Xie, P. Integer ambiguity parameter identification for fast satellite positioning and navigation based on LAMBDA-GWO with tikhonov regularization. *Appl. Sci.* **2023**, *13*, 1239. [\[CrossRef\]](#)
21. Tütüncü, K.; Şahman, M.A.; Tuşat, E. A hybrid binary grey wolf optimizer for selection and reduction of reference points with extreme learning machine approach on local GNSS/leveling geoid determination. *Appl. Soft Comput.* **2021**, *108*, 107444. [\[CrossRef\]](#)
22. Chen, Q.; Yang, F.; Liang, S.; Zhai, Y. GNSS Array Adaptive Anti-Jamming Beamforming Based on Whale-Grey Wolf Hybrid Swarm Intelligence Algorithm. In *Proceedings of the 2026 6th International Symposium on Intelligent Robotics and Systems (ISOIRS)*; IEEE: New York, NY, USA, 2026; pp. 1–10.
23. Urbanowicz, R.J.; Meeker, M.; La Cava, W.; Olson, R.S.; Moore, J.H. Relief-based feature selection: Introduction and review. *J. Biomed. Inform.* **2018**, *85*, 189–203. [\[CrossRef\]](#) [\[PubMed\]](#)

24. Xu, B.; Jia, Q.; Luo, Y.; Hsu, L.T. Intelligent GPS L1 LOS/multipath/NLOS classifiers based on correlator-, RINEX-and NMEA-level measurements. *Remote Sens.* **2019**, *11*, 1851. [\[CrossRef\]](#)
25. Forgy, E.W. Cluster analysis of multivariate data: Efficiency versus interpretability of classifications. *Biometrics* **1965**, *21*, 768–769.
26. McQueen, J.B. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*; University of California Press: Oakland, CA, USA, 1967; pp. 281–297.
27. Arthur, D.; Vassilvitskii, S. k-means++: The advantages of careful seeding. *Proc. Soda* **2007**, *7*, 1027–1035.
28. Park, H.S.; Jun, C.H. A simple and fast algorithm for K-medoids clustering. *Expert Syst. Appl.* **2009**, *36*, 3336–3341. [\[CrossRef\]](#)
29. Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 888–905. [\[CrossRef\]](#)
30. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proc. KDD* **1996**, *96*, 226–231.
31. Reynolds, D.A.; Rose, R.C. Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Trans. Speech Audio Process.* **2002**, *3*, 72–83.
32. Zhu, Y.; Ting, K.M. Kernel-based clustering via isolation distributional kernel. *Inf. Syst.* **2023**, *117*, 102212. [\[CrossRef\]](#)
33. Ting, K.M.; Zhu, Y.; Zhou, Z.H. Isolation kernel and its effect on SVM. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; Association for Computing Machinery: New York, NY, USA, 2018; pp. 2329–2337.
34. Yang, J.; Lin, C.T. Autonomous clustering by fast find of mass and distance peaks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *47*, 5336–5349. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Lin, H.; Hou, J.; Yuan, H. Density peak clustering based on nearest neighbors. *Eng. Appl. Artif. Intell.* **2025**, *160*, 111981. [\[CrossRef\]](#)
36. Ma, Z.; Wu, G.; Suganthan, P.N.; Song, A.; Luo, Q. Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms. *Swarm Evol. Comput.* **2023**, *77*, 101248. [\[CrossRef\]](#)
37. Faris, H.; Aljarah, I.; Al-Betar, M.A.; Mirjalili, S. Grey wolf optimizer: A review of recent variants and applications. *Neural Comput. Appl.* **2018**, *30*, 413–435.
38. Wang, Y.; Jin, C.; Li, Q.; Hu, T.; Xu, Y.; Chen, C.; Zhang, Y.; Yang, Z. A dynamic opposite learning-assisted grey wolf optimizer. *Symmetry* **2022**, *14*, 1871. [\[CrossRef\]](#)
39. Mittal, N.; Singh, U.; Sohi, B.S. Modified grey wolf optimizer for global engineering optimization. *Appl. Comput. Intell. Soft Comput.* **2016**, *2016*, 7950348. [\[CrossRef\]](#)
40. Lian, J.; Hui, G.; Ma, L.; Zhu, T.; Wu, X.; Heidari, A.A.; Chen, Y.; Chen, H. Parrot optimizer: Algorithm and applications to medical problems. *Comput. Biol. Med.* **2024**, *172*, 108064. [\[CrossRef\]](#) [\[PubMed\]](#)
41. Zheng, B.; Chen, Y.; Wang, C.; Heidari, A.A.; Liu, L.; Chen, H. The moss growth optimization (MGO): Concepts and performance. *J. Comput. Des. Eng.* **2024**, *11*, 184–221. [\[CrossRef\]](#)
42. Abdollahzadeh, B.; Khodadadi, N.; Barshandeh, S.; Trojovský, P.; Gharehchopogh, F.S.; El-Kenawy, E.S.M.; Abualigah, L.; Mirjalili, S. Puma optimizer (PO): A novel metaheuristic optimization algorithm and its application in machine learning. *Clust. Comput.* **2024**, *27*, 5235–5283. [\[CrossRef\]](#)
43. Mohammed, K.K.; Darwish, A.; Hassanien, A.E. Snake Optimization of Multiclass SVM for Efficient Diagnosis of Heart Disease Risk Prediction. In *Artificial Intelligence for Environmental Sustainability and Green Initiatives*; Springer: Berlin/Heidelberg, Germany, 2024; pp. 169–182.
44. Diab, M.S.; Darwish, M. M.; Oliva, D.; Hosny, K. M. PySOA: A novel bio-inspired python snake optimization algorithm. *Clust. Comput.* **2026**, *29*, 274. [\[CrossRef\]](#)
45. Hussain, K.; Mohd Salleh, M.N.; Cheng, S.; Shi, Y. Metaheuristic research: A comprehensive survey. *Artif. Intell. Rev.* **2019**, *52*, 2191–2233.
46. Khanduja, N.; Bhushan, B. Recent advances and application of metaheuristic algorithms: A survey (2014–2020). In *Metaheuristic and Evolutionary Computation: Algorithms and Applications*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 207–228.
47. Mandal, P.K. A review of classical methods and Nature-Inspired Algorithms (NIAs) for optimization problems. *Results Control Optim.* **2023**, *13*, 100315. [\[CrossRef\]](#)
48. Tomar, V.; Bansal, M.; Singh, P. Metaheuristic algorithms for optimization: A brief review. *Eng. Proc.* **2024**, *59*, 238. [\[CrossRef\]](#)
49. Akopov, A.S.; Beklaryan, L.A.; Thakur, M. Improvement of maneuverability within a multiagent fuzzy transportation system with the use of parallel biobjective real-coded genetic algorithm. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 12648–12664. [\[CrossRef\]](#)
50. Ahmad, M.F.; Isa, N.A.M.; Lim, W.H.; Ang, K.M. Differential evolution: A recent review based on state-of-the-art works. *Alex. Eng. J.* **2022**, *61*, 3831–3872.
51. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **2002**, *1*, 67–82.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.