Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Improved partitioning technique for density cube-based spatio-temporal clustering method

Devi Fitrianah [a,*], Hisyam Fahmi [b], Achmad Nizar Hidayanto [c], Aniati Murni Arymurthy [c]

[a] Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta-Indonesia
[b] Faculty of Science and Technology, UIN Maulana Malik Ibrahim, Indonesia
[c] Faculty of Computer Science, Universitas Indonesia, Indonesia

ARTICLE INFO

Article history:
Received 6 April 2022
Revised 26 July 2022
Accepted 5 August 2022
Available online xxxx

Keywords:
Clustering
Spatio-temporal clustering
Density-cube spatio-temporal clustering
Partitioning technique
Imstagrid

ABSTRACT

This work proposes a novel partitioning technique on the density-cube-based data model for the Spatio-temporal clustering method. This work further adapts this clustering approach to Spatio-temporal data. We have compared the IMSTAGRID-the proposed algorithm to the ST-DBSCAN, AGRID+, and ST-AGRID algorithms and have found that the IMSTAGRID algorithm improves the data partitioning technique and the interval expansion technique and is able to achieve uniformity in the spatial and temporal dimensional values. Three types of Spatio-temporal data sets have been used in this experiment: a storm data set and two synthetic data sets – synthetic data set 1 and synthetic data set 2. Both the storm data set and synthetic data set 2 were comparable in terms of the scattering of the data points, while synthetic data set 1 contained clustered data. The performance of the IMSTAGRID clustering method was measured via a silhouette analysis, and its results surpassed the other algorithms investigated; the silhouette index for synthetic data set 2 was 0.970, and 0.993 using synthetic data set data set 1. The IMSTAGRID algorithm also outperformed the baseline algorithms (ST-DBSCAN, AGRID+, and ST-AGRID) in labeling accuracy for the storm data set, yielding results of 82.68%, 38.36%, 76.13%, and 78.66%, respectively.

1. Introduction

DBSCAN (density-based spatial clustering of applications with noise) is a clustering method that is commonly used to handle large and dense volumes of spatial data even in the presence of noise. DBSCAN can also identify data clusters of various shapes, including linear, oval, and concave clusters (Ester, Sander, Kriegel, & Xu, 1998). The DBSCAN algorithm requires that the eps value for the eps-neighborhood of each data point in a cluster is larger than zero and that the eps-neighborhood must contain a minimum number of points (MinPts), which means that the density in an eps-neighborhood must be larger than a specified threshold.

Many algorithms have been inspired by DBSCAN and have sought to improve its efficiency and effectiveness. In recent studies, The DBSCAN algorithm is still utilized to apply the clustering task, and the results were satisfying (Wu, Shi, & Mamoulis, 2018; Shi & Pun-cheng, 2019; Deng et al., 2022; Latifi-Pakdehi & Daneshpour, 2021; Zhu et al., 2021).

One of the methods used to improve efficiency is the use of a grid-based approach. Grid-based algorithms speed up distance calculations in the formation of clusters by transforming the data into representative grids/boxes which occupy a particular minimum area (Cook et al., 2022). Each representative grid must have a minimum number of points to satisfy a specified threshold (MinPts), which speeds up the cluster formation process (Sun et al., 2005).

The processing methods of density- and grid-based clustering algorithms do not depend on the size of the database (Wang et al., 2019); they depend on the number of transformed dimensions for the gridded data (Huang & Bian, 2009; Sun et al., 2005). The number of transformed dimensions is heavily affected by the distance selection method used in the formation of the gridded area (eps); thus, eps selection is essential in determining the efficiency of an algorithm (Hu et al., 2021). A smaller eps distance will reduce efficiency, while a larger eps distance will reduce clustering accuracy. The minimum number of data points, MinPts, in a grid is also essential. This value determines whether or not a grid is a core

* Corresponding author.
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2022.08.006
1319-1578/

Please cite this article as: D. Fitrianah, H. Fahmi, A.N. Hidayanto et al., Improved partitioning technique for density cube-based spatio-temporal clustering method, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2022.08.006

grid (Huang & Bian, 2009). Eps and MinPts are determined at the beginning of a clustering process.

All algorithms derived from the DBSCAN algorithm include a distance calculation process to generate neighborhood statistics for all points; therefore, neighboring query efficiency is also essential (Sander, Ester, Kriegel, & Xu, 1998). Neighborhood calculations can be approached in many ways. DBSCAN uses an indexing method, R* Tree (Ester et al., 1998), to improve data query efficiency. Other methods such as R + Tree, Quadtrees, and X Tree (Birant & Kut, 2007) may also be used to increase spatial access efficiency.

Many existing density- and grid-based clustering algorithms are exclusively applied to spatial data (Tork, 2012). Some adjustments are needed to enable such algorithms to perform spatial and temporal data clustering (Wang et al., 2019). In density-based Spatio-temporal data clustering, two distances must be considered (Birant & Kut, 2007): eps 1 and eps 2; where eps 1 is the distance for the spatial data, and eps 2 is the corresponding distance for the non-spatial data. Subsequently, the spatial and temporal neighbor selection process is implemented by comparing non-spatial attributes to form both spatial and temporal clusters.

The AGRID + algorithm is another clustering algorithm, similar to DBSCAN, based on data and grid density applicable to multidimensional data (Zhao, Cao, Zhang, & Zhang, 2011). This algorithm has been proven to identify clusters with reasonable accuracy and relatively fast computation speeds. Since the algorithm can process n-dimensional data, it can be utilized for Spatio-temporal clustering, and with modifications, it can be made applicable to Spatio-temporal data sets. Algorithms based on density and grid approaches are suitable for spatial data (Huang & Bian, 2009). However, there are some obstacles to their implementation for Spatio-temporal data. Generally, Spatio-temporal data have three dimensions; two spatial and one temporal dimension. Spatio-temporal data also have specific characteristics: there are latitude and longitude coordinates in the spatial dimension, represented by either positive or negative real values, and a temporal dimension that includes data with time units (for example, daily, weekly, or monthly).

ST-AGRID (Fitrianah, Hidayanto, Fahmi, Lumban Gaol, & Arymurthy, 2015) is an adaptation of the AGRID + algorithm that modifies the partitioning stage, the distance threshold stage, and the density complexity calculation stage. These directly affect the transformation of multidimensional data from n-dimensions to three dimensions: two spatial dimensions (longitude and latitude) and a temporal dimension. However, the distribution of the spatial and temporal intervals is suboptimal when the ST-AGRID algorithm is used. The partitioning technique of the ST-AGRID algorithm only tackles the issue of spatial dimensions (latitude and longitude). The algorithm cannot equalize intervals for the temporal dimension, which can be an aggregate of daily, monthly, or even yearly data. Therefore, some cells may be imbalanced (not cube-shaped) and cannot be included in the distance threshold radius in the determination of neighbors (neighboring points) and neighborhoods (neighboring cells) for L_spatial.

To address this weakness, the authors have improved a *partitioning* technique by equally distributing the data in both the spatial and temporal dimensions to achieve a cubed cell as a dimensional data unit. The issues solved in the proposed algorithm improvement are as follows:

1. The algorithm improves the development of a precise data *partitioning* method and results in representative spatial and temporal sub-units usable in subsequent processes.
2. The algorithm enhances the creation of a density-and cube-based Spatio-temporal clustering algorithm that can process data efficiently and generate accurate results.

1.1. Related works

Spatio-temporal data mining is often used to analyze data based on remote sensing data and geographic information system applications (Roddick & Lees, 2001; Roddick & Spiliopoulou, 1999). Such datasets are typically large and heavily dependent on spatial and temporal scale data and contain multidimensional interactions, such as season and weather patterns, that can be used to explore cause and effect relationships (Roddick & Lees, 2001). Spatio-temporal clustering is one approach used in Spatio-temporal data mining to analyze object data without including class labels (Georgoulas et al., 2013). The algorithm makes it suitable for extracting information from vast repositories with unknown data labels (Kantardzic, 2003; Shi & Pun-cheng, 2019). The Spatio-temporal clustering method is extensively used, and examples can be found in the medicine, security, environment, biology, pathology, health, and fishery fields (Sahu & Mardia, 2005; Lorena et al., 2014; Anbaroglu, Heydecker, & Cheng, 2014; Liu et al., 2018). Spatio-temporal clustering methods can be implemented in one clustering stage or two clustering stages: beginning with spatial information and followed by temporal information, or vice versa. Each clustering approach produces slightly different results since each approach emphasizes a different dimension – either the spatial or the temporal dimension (Abraham & Roddick, 1998; Yao et al., 2018).

A DBSCAN-based Spatio-temporal clustering implementation has been developed into an algorithm known as ST-DBSCAN (Birant & Kut, 2007). This algorithm can find spatial, non-spatial, and temporal clusters. ST-DBSCAN uses four parameters: Eps1, Eps 2, MinPts, and $\Delta\in$. The first three parameters are consistent with parameters in the original DBSCAN algorithm. In contrast, the fourth parameter, $\Delta\in$, is used to prevent clusters from unifying due to slight differences in the non-spatial values of neighboring locations. An additional distance metric is included in ST-DBSCAN to determine clusters' radius, εt, and neighbors in the temporal dimension. These two parameters are determined using a k-dist graph and are then used as inputs for ST-DBSCAN.

The algorithm inspired by DBSCAN in Spatio-temporal clustering is the trajectory identification (Chen, Ji, & Wang, 2014). The algorithm works by considering the time sequential characteristics along a trajectory. State continuity within a single stop and temporal disjuncture among stops are two of the premises proposed as a theoretical basis for regulating the trajectory point selection in clustering.

Other Spatio-temporal clustering algorithms have been proposed. The first is ST-GRID, a DBSCAN-based algorithm that has been used to analyze the order of an earthquake (Wang, Wang, & Li, 2006). ST-GRID partitions both the spatial and temporal dimensions into cells. The second algorithm that examines Spatio-temporal clustering is ST-AGRID (Fitrianah et al., 2015), which has been used to investigate the clustering of fish catch data.

1.2. Proposed density cube-based spatio-temporal clustering

This section explains the proposed clustering algorithm method. It includes a discussion of the data structure, adaptations made to the baseline algorithm, the partitioning stage, the distance threshold, and the density compensation calculations.

1.2.1. The spatio-temporal data structure

The proposed algorithm is density- and cube-based. The cube structure is necessary because three dimensions are represented in the data: east longitude, south latitude, and time. The cube structure will be discussed further in later sections, but we begin by describing the Spatio-temporal data, as seen in Fig. 1 below.

In general, a Spatio-temporal data model can be represented using a notation where Z indicates the time dimension, X indicates latitude, and Y indicates longitude in the spatial dimension. For each combination, there are non-spatial data. This data model can be seen in Fig. 3. The figure above illustrates the Spatio-temporal data model, which consists of location data, represented by X and Y lines, and time data indicated by cells. Since the time dimension is aggregated, the data structure is abstractly a cube. A1, ... A_n data cubes and B1, ... B_n cubes for every cluster formed spatially and temporally. Each cube consists of data points with their representative non-spatial data, represented by A11, A12, ... A_{nm}. Fig. 2 below shows the image of a cube consisting of other data points.

For all data points (A11, A12, ... A_{nm}), the composite data structure consists of both spatial and temporal data and non-spatial attributes. It can be concluded that the data structure of the new proposed algorithm consists of at least a spatial and temporal dimension. The spatial dimension has either positive or negative real values (longitude and latitude coordinates). A temporal dimension consists of a time unit with positive integer data (date, month, and year) and other data attributes. The structure of the data points represented in the cube used in the clustering algorithm is shown in Fig. 3 below.

In the above figure, dd is the date attribute, mm is the month attribute, and yy is the year attribute. All three attributes are temporal data. The spatial attributes are longitude (lon) and latitude (lat). Finally, atr1 to atr-n are data attributes from related representative points.

1.2.2. Improvement of the partitioning technique

- Improvement Stage Analysis In ST-AGRID Algorithm

In order to determine the number of cells in the partitioning stage of the ST-AGRID algorithm, the cell interval must be calculated. The cell interval value (L) is obtained by dividing the range of each dimension (upper bound – lower bound) by the number of cells, m. This approach emphasizes the similarity of cell numbers of the spatial dimensions (longitude and latitude), while the temporal dimension can be determined by using daily, weekly, or monthly temporal units. After each (spatial and temporal) cell is formed, each data object is inserted into a cell that matches its Spatio-temporal coordinate—the results in L_spatial and L_temporal intervals. The partitioning stage algorithm used in the partitioning approach for the ST-AGRID algorithm is shown below in Fig. 4.

The partitioning approach in the ST-AGRID algorithm above will result in an interval value from each cell that will vary for each dimension (the X and Y spatial and temporal dimensions). This result in unequal data intervals on each dimension (spatial and temporal). Therefore, the cell formed in such a process cannot be a perfect cube. Suppose different data intervals (L) are used. In that case, some parts of the data within each cell may not be included in the distance threshold radius that determines neighbors (neighboring points) and neighborhoods (neighboring cells) for L_spatial (shown in Fig. 5 as GAP). Fig. 5 shows data that have been partitioned using a non-uniform data interval for the spatial and temporal dimensions.

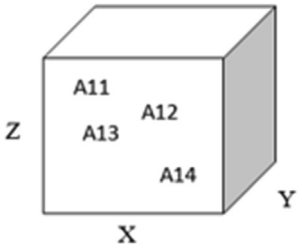


Fig. 2. The illustration of the Cube data space model.

- Proposed Improvements to the Partitioning Technique

Unequal partitioning of multidimensional data causes gaps; therefore, a partitioning technique capable of producing cube-shaped dimensional data is proposed. The proposed approach is to determine a uniform interval (L) for the spatial and temporal dimensions. In Fig. 5 above, which illustrates partitioning, the interval partitioning is based on the number of cells, m. The data ranges in the spatial dimension (longitude and latitude) are different; this is also true for the temporal dimension. If a typical m value is used, each dimension will have a different interval value. For example, if the data range for the X data is 10°, while the data range for Y data is 5°, and the temporal dimension consists of 1000 days, if the value of m = 5, then each dimension has a different interval (L). The data in the X dimension will have a cell range with an interval of 2. The data in the Y dimension will have a cell range with an interval of 1, as shown in Fig. 5. Similarly, in the temporal dimension, the 1000 days will be divided by 5, thus resulting in each cell having an interval of 200 days.

Based on the explanation above, problems may arise if there is a gap in the data in one or more dimensions when determining r (distance threshold), which is the radius that determines neighbors (neighboring points) and neighborhoods (neighboring cells). The value of r is L/2. From the illustration shown in Fig. 5, we know the L_spatial value is obtained from the average intervals in

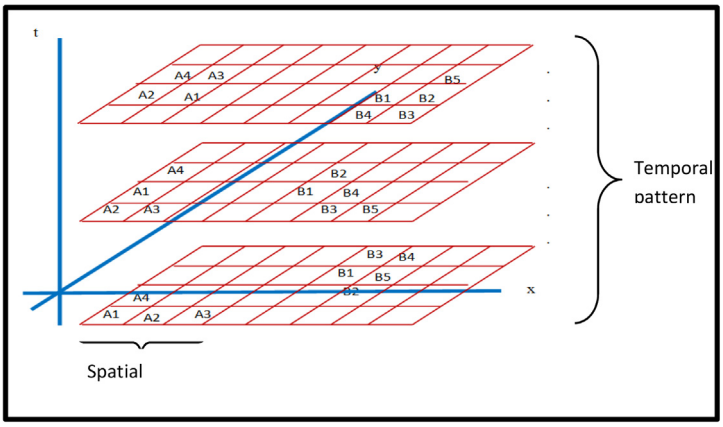


Fig. 1. Spatio-Temporal Data Illustration.

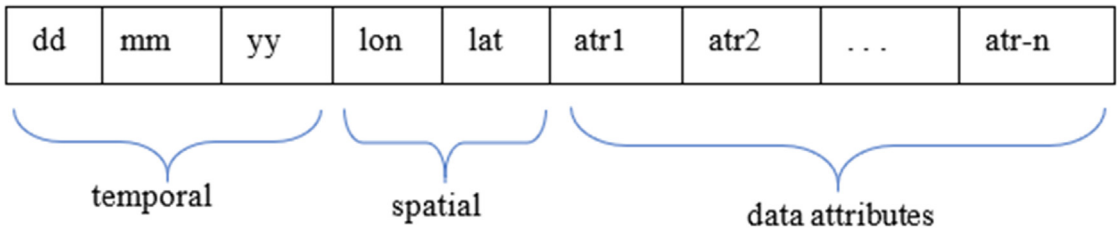


Fig. 3. Spatio-temporal data structure.

```
/* Pseudo code for partitioning ST-AGRID*/
for i=1 to number of Objects
        cell_ID = spatial coordinate of object Oi / length of interval;
     cell_time_ID = aggregate of temporal/s /* days, weeks, or,
         months */
     if the cell_ID and cell_time_ID not yet in the cells
        add cell_ID and cell_time_ID to cells;
     else
        add object to the cell with corresponding cell_ID and
            cell_time_ID
    end if
end for
```

Fig. 4. Dimensional data partitioning algorithm based on *m* value.

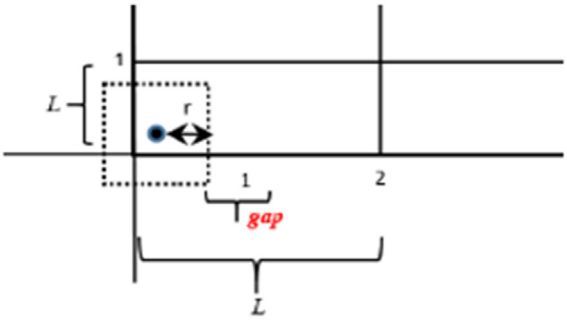


Fig. 5. Partitioning illustration based on numbers of intervals (*m*).

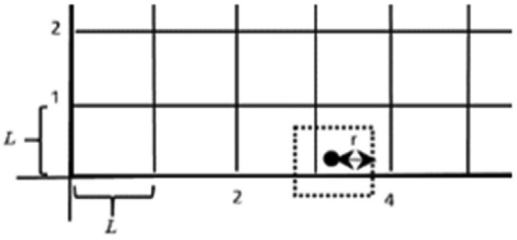


Fig. 6. Partitioning illustration based on interval value (L).

the X and Y dimensions ((2 + 2)/2). Therefore, the value of L_spatial is 1.5. The r value is 1.5/2 = 0.75. The value of r in the Y dimensional data appears adequate. However, in the X dimension, since L = 2, the neighborhood radius (r) value should be 1, and the required value is 0.75. It causes some parts of the data to be excluded from the radius (r). The effect of exclusion of parts of the X dimension will result in disjointed clusters. Additionally, this can also cause misclassification of points.

It is, therefore, necessary to maintain the same interval value for the spatial dimensions (X and Y), as seen in Fig. 6 below, where the spatial dimensions are shown to have the same value. The neighbors and neighborhoods can be determined using the spatial neighboring radius if the same L_spatial value ((1 + 1)/2) = 1, r = ½ = 0.5 is used.

Another problem also arises for L_temporal because of differences between the spatial and temporal dimensions. Fig. 7 illustrates that L_spatial is already a cube while L_temporal is not because the measurement units differ from the spatial dimension. L_temporal follows the desired aggregate unit. If the aggregate unit is based on a daily temporal analysis, its value is 1. Additionally, if the unit is made based on a weekly temporal analysis, the aggregate value is 7. Consequently, if the unit is made based on a monthly temporal analysis, the aggregate value is 30. This problem prevents the dimensional data cell from forming a perfect cube.

Standardizing spatial and temporal dimension values using stretching or interval expansion is proposed to address the above-mentioned problem. This approach creates the same dimensional value for spatial and temporal dimensions and achieves an overall cube shape.

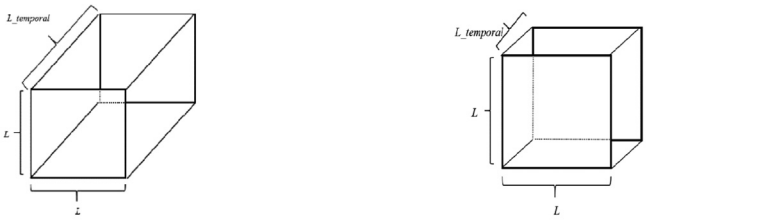
The L_temporal calculation within the partitioning process determines the entry point for each cell by first counting the total number of temporal intervals. It is achieved by using Equation (1). The L_spatial calculation (in Equation 2) utilizes a count of the total number of spatial intervals; the cell coordinate values are calculated using Equation (3) after the interval values for the spatial and temporal dimensions are determined.

$$M\_temp = temporal\ range / temporal\ aggregate$$

$$M\_spa = \text{dim}ension\ range\ /\ L\_spatial$$

$$Coordinate\ value = Coordinate\ /\ L\_spatial\ *\ temporal\ aggregate$$

Fig. 8 shows the pseudocode of the proposed *partitioning* technique for further clarification. The expansion results re shown for the Cell_ID coordinate which has been adjusted to its temporal dimension.



a. Large Temporal Aggregate b. Small Temporal Aggregate

Fig. 7. L_spatial and L_temporal illustrations.

```
/* Pseudo code for partitioning imstagrid*/

M_temp = (max(temp) - min (temp))/agregat temporal
M_spa1 = (max(lat) - min(lat))/L_spatial
M_spa2 = (max(lon) - min(lon))/L_spatial

FOR i=1 to number of Objects
             cell_ID = (spatial coordinate of object Oi / length of
          interval) * aggregat temporal;

     IF the cell_ID not yet in the cells
          add cell_ID and cell_time_ID to cells;
     ELSE
          add object to the cell with corresponding cell_ID
    ENDIF
ENDFOR
```

Fig. 8. Data dimension partitioning alogrithm based on *L* value and expansion.

1.2.3. Threshold distance (r) calculation stage

Next, the distance threshold (r) is adapted, as previously described for the ST-AGRID algorithm. The distance threshold value or r is very influential in determining neighbors (neighboring points in each cell) and neighborhoods (adjacent neighboring cells). The AGRID + algorithm has only one distance threshold value because the data in the n dimensions are the same. Therefore, in that application, only one distance threshold is needed to define proximity between points. Implementing a single distance threshold value for the new proposed algorithm is undesirable because the intervals found in the temporal dimension differ from the intervals in the spatial dimension, which contain the longitude and latitude coordinates.

Based on the characteristics of the Spatio-temporal data, this study proposes a different distance threshold for each dimension: spatial and temporal. This distance threshold approach in the spatial dimension is consistent with the approach in the AGRID + algorithm: r < L/2. Fig. 9 below shows the algorithm used to determine the distance thresholds in spatial and temporal dimensions.

1.2.4. Density compensation calculation stage

The density compensation calculation is related to the determination of the threshold (density threshold) that determines whether a group can be considered a cluster. Therefore, the density compensation calculation is essential in this clustering process. This calculation depends on the number of dimensions. It happens because the compensation of each cell is based on the ratio of the cell volume and the volume in neighboring cells. The volume of all neighboring cell cubes is calculated based on the total number of dimensions, as shown in Eq. (4).

$$C_{densities}(Oi) = densities(Oi) \times \frac{volume\ of\ Oi\ cube}{volume\ of\ all\ Oi\ neighboring\ cubes} \quad (4)$$

In Eq. (4), the parameter densities (Oi) is obtained by dividing the number of data points in a cube (Oi) by the volume of that cube using the formula. Some modifications are required because there

```
/* Pseudo code for computing r*/
r1 = (minimal L)/2;
r2 = (maximal L)/2;
r_spatial = λ*r1+(1-λ)*r2-ε; /* 0 ≤ λ ≤ 1; ε = small number
r_temporal = 1; /* range temporal is 1 aggregate time */
```

Fig. 9. Distance Threshold calculation algorithm.

are only three dimensions. For instance, if a point α in a cube, Cα, is in the top left corner, then only the cube in the top left corner will be included in clustering, as shown in Fig. 10.

Fig. 11 illustrates the calculation of the volume of all *Oi* neighboring cubes, based on order of proximity with the *i-th neighbor*.

Based on Fig. 11, there are four areas (neighbors) where the volume must be determined. The above is the cube where the points are located, considered the same area, and are the neighbor's boundary. To calculate the volume of all Oi neighboring cubes, Equation (5) is used:

$$vol_{neighbor} = k_0(r+a)^2 r_{temp} + k_1(r+a)(r-a)(r_{temp}) + k_2(r+a)(r-a)(r_{temp}) + k_3(r-a)^2(r_{temp}) \quad (5)$$

Therefore, the formula to determine the density compensation is given by Eq. (6).

$$C\_densities(O_i) = densities(O_i) \times \frac{(2r)^{d'}}{k_0(r+a)^2 r_{temp} + k_{1,2}\left((r+a)(r-a)(r_{temp})\right)^2 + k_3(r-a)^2(r_{temp})} \quad (6)$$

The following is the algorithm used to calculate the density compensation (Fig. 12).

2. Results and experimental analysis

Now that the three adaptations have been explained in the previous sections, the next step is to describe experiments that were conducted using the proposed partitioning technique, adapted *distance thresholds,* and density compensation calculation procedure.

2.1. Experimental data sets

In this study we utilized a storm data set, and two synthetic data sets. Each data set is as follows:

- Storm data set

The storm data set contained spatio-temporal data related to the tracking of storms based on wind speed. The data can be downloaded from www.unisys.com. Table 1 explains the data structure of the storm data used. The storm category status is determined from the Saffir-Simpson Scale.

Fig. 13 below shows a plot of the storm data used in this study. These storm data spanned 15 years; from 2000 to 2014. The data were collected in the South Pacific Sea (coordinates: 140° BT – 190° BT and 10° LS – 50° LS).

- Synthetic Data Sets

Synthetic data set 1 consisted of 600 Spatio-temporal data points, randomly generated using a Gaussian distribution centered at each cluster. There were five clusters, each with a predetermined centroid: Cluster 1 consisted of 100 data points with a centroid located at 0, 0, 7; Cluster 2 consisted of 120 data points with a centroid located at 5, 1, 6. Cluster 3 consisted of 200 data points with a centroid located at 0, 5, 20, while Cluster 4 consisted of 70 data points with a centroid located at 5, 5, 20, and finally, Cluster 5 consisted of 110 data points with a centroid located at 2.5, 2.5, 10. The first two attributes are spatial coordinates, and the last attribute is the temporal coordinate. Synthetic data set 2 consisted of 5000 Spatio-temporal data points, raised using random distribution. Synthetic data set 1 contained grouped data; therefore, the

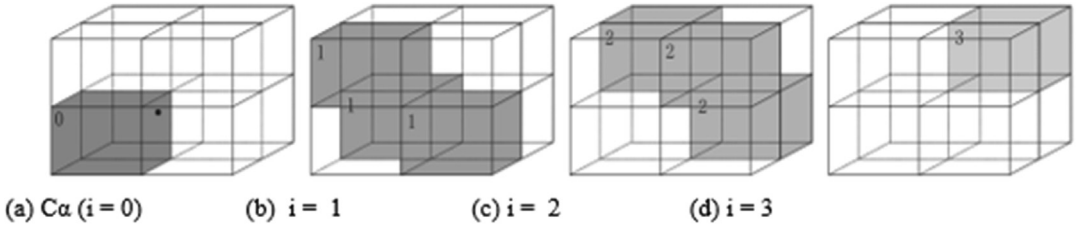


Fig. 10. i-th neighbor order of an α point of a Cα cube.

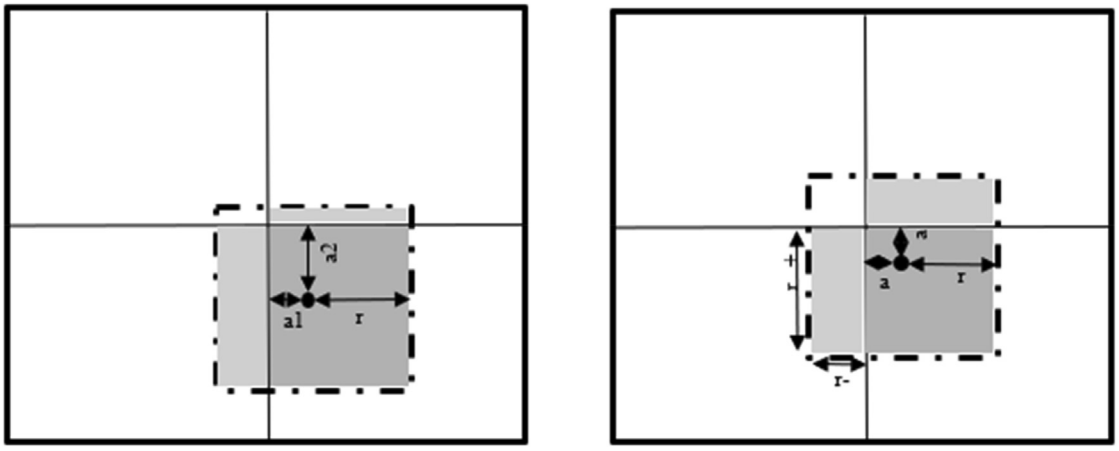


Fig. 11. (a) shows α point which neighborhood of α is the area inside the dotted lines (b) estimated volume of the areas in the neighborhood and neighbor of α with the assumption that there is a diagonal area.

```
/* Density compensating */
FOR all object Oi
ratio = (2r)^d' / Σ_{j=0}^{d} kj(r+a)^{d'-j}(r-a)^k
Compensating density (Oi) = density X ratio;
ENDFOR
```

Fig. 12. Density compensation calculation algorithm.

Table 1
Storm data catalog.

No.	Attribute Type	Description	Data Sample
1.	Date	Storm occurance date	01
2.	Month	Storm occurance month	12
3.	Year	Storm occurance year	2000
4.	Lat	Latitude coordinate	26,5° LS
5.	Lon	Longitude coordinate	113,4°BT
6.	Wind	Wind speed	45 Knots
7.	Status	Storm name	Tropical Storm, Cyclone-1

data already formed distinctively separated clusters, while synthetic data set 2 contained scattered data. Plots of synthetic data sets 1 and 2 can be seen in Fig. 14(a) and (b).

2.2. Experiment results and discussion

In the proposed partitioniong technique experiment, the intervals for the spatial dimension (the X and Y dimensions) were equalized in the partitioning stage. Dimensional range standardizing or stretching was then applied to equalize the temporal dimensions. This change was made so that the interval ranges of both the spatial and temporal dimensions could be equalized, enabling the formation of cube-shaped dimensional data. Moreover, the density compensation calculation was modified as previously discussed. This combination of improved procedures will be identified as the IMSTAGRID algorithm in the remainder of this paper.

The IMSTAGRID Spatio-temporal *clustering* algorithm was implemented on the three previously described data sets: the storm data set, synthetic data set 1, and synthetic data set 2. In the first experiment, the L interval parameter within the algorithm was tuned. The L values used were 0.25, 0.5, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 3. This tuning was applied to investigate the silhouette index's sensitivity to the cube's size. The theta parameter was tuned from the second experiment to determine the density threshold – DT. It has been reported that lowering the theta value will increase the DT value, resulting in the formation of smaller clusters, thus increasing the chance that data points may be misidentified as noise (Zhao et al., 2011). Lower DT values will result in larger clusters, reducing noise.

The silhouette index results of each experiment for each data set are shown in Fig. 15. The storm data set and synthetic data set 2 were scattered data sets; the lower L values result in better silhouette index values. Conversely, synthetic data set 1 is grouped data, and the lower L values result in lower silhouette index values.

Table 2 below shows the silhouette index values for each experimental result on all data sets investigated, in addition to further details.

Table 2 above shows that the highest silhouette index value is achieved for the lowest L value in the storm data set. In other words, smaller cubes result in better silhouette index values.

Conversely, in synthetic data set 1, a higher L value (or larger cube) results in a better silhouette index value. The results for synthetic data set 2 are comparable to those for the storm data set: smaller cubes result in better silhouette index values. Based on these results, a smaller cube is best for scattered data because IMSTAGRID will perform similarly to a density-based clustering algorithm. On the other hand, if the data have already been grouped, it is better to use a larger cube since IMSTAGRID will perform similarly to a grid-based clustering method. To conclude, using lower L values quickly and compactly forms clusters, which is better for scattered data while using higher L values will achieve more optimal cluster results for grouped data. Table 3 provides more detailed results on each Spatio-temporal data set.

Table 3 shows the result of the IMSTAGRID algorithm implementation for the data sets explored in this study. Based on the table, a suitable silhouette index value can be achieved from lower L values on scattered data. In contrast, higher L values produce better silhouette index values for grouped data. Table 4 shows the experimental accuracy achieved for the storm data set with (DS) and without (TS) stretching.

Table 4 shows that the IMSTAGRID clustering algorithm with the application of a stretching technique produces more accurate

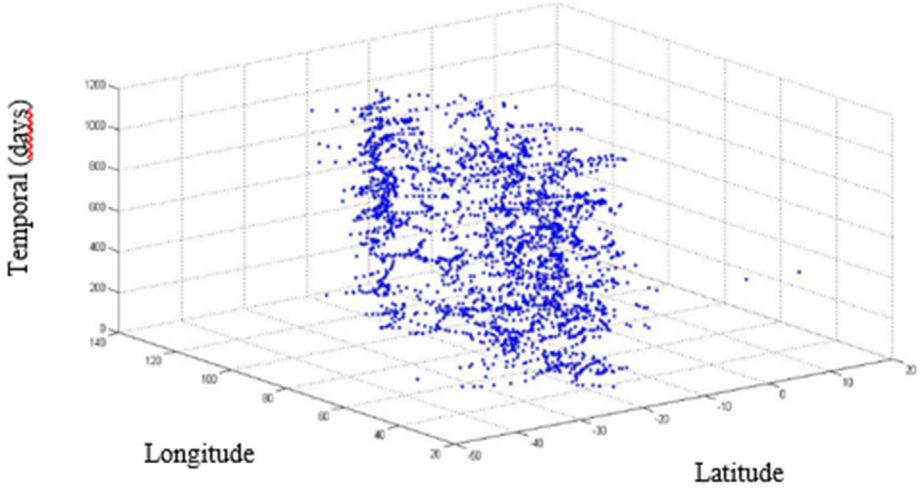
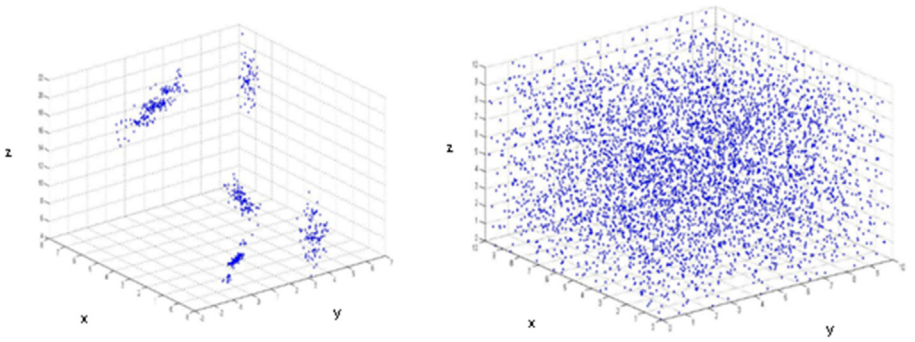


Fig. 13. Tropical storm data plot in South Indian Ocean.



a. Grouped synthetic data plot b. Scattered synthetic data plot

Fig. 14. Synthetic data plots.

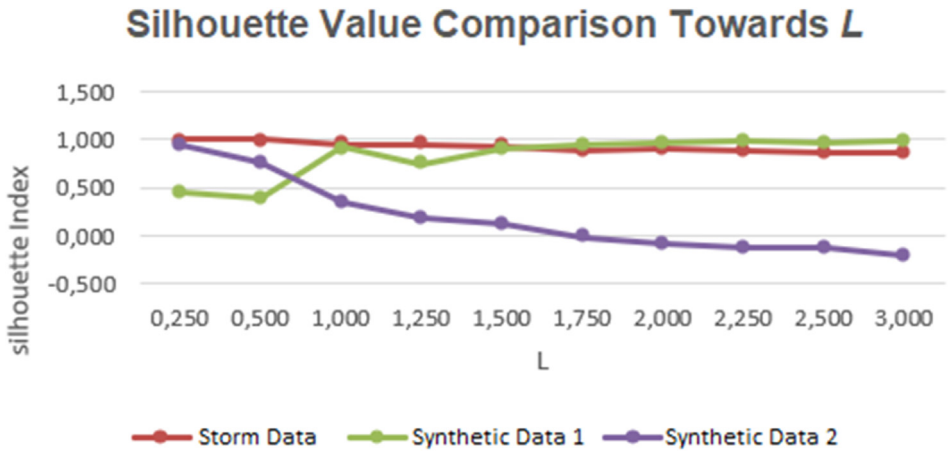


Fig. 15. Silhouette value comparison towards L on each data set type.

Table 2
Results of IMSTAGRID implementations on three types of data (daily data).

	Storm data set		Synthetic data set 1		Synthetic data set 2	
L	Silhouette index	Time (seconds)	Silhouette index	Time (seconds)	Silhouette index	Time (seconds)
0.25	0.999	1.621	0.454	0.717	0.938	7.983
0.5	0.999	1.668	0.382	1.296	0.763	10.187
1	0.956	1.830	0.923	1.971	0.343	22.033
1.25	0.953	1.872	0.747	2.082	0.180	29.634
1.5	0.924	1.868	0.911	2.077	0.120	35.364
1.75	0.881	1.895	0.946	2.098	−0.012	44.746
2	0.899	1.917	0.970	1.938	−0.083	58.683
2.25	0.886	1.932	0.993	2.194	−0.135	81.640
2.5	0.871	1.968	0.972	2.316	−0.135	82.074
3	0.867	1.940	0.980	2.629	−0.199	121.789

Table 3
Results of IMSTAGRID best implementations based on different temporal data set types.

	Daily data			Weekly Data			Monthly data		
	L	Sil. Index	Time (seconds)	L	Sil. Index	Time (seconds)	L	Sil. Index	Time (seconds)
Storm data set	0.25	0.999	1.621	0.25	0.999	1.648	0.25	0.987	1.639
Synthetic data 1	2.25	0.993	2.194	2.25	0.987	2.412	3	0.975	4.106
Synthetic data 2	0.25	0.9382	7.983	0.25	0.905	20.360	0.25	0.710	10.106

Table 4
Comparison of IMSTAGRID performance with data stretching and without data stretching.

	Accuracy (%)	Silhouette Index
With stretching	83.29	0.59
Without stretching	81.74	0.33

results when compared with preceding the application of the stretching technique: 83.29 % compared with 81.74 %. The average silhouette index value with stretching is better (0.59) than without stretching (0.33). Therefore, the use of stretching or spatial and temporal dimension standardization succeeded in improving the accuracy of the IMSTAGRID algorithm.

Visual inspection of the cluster results from synthetic data set 1 shows that the data are relatively accurately clustered. In synthetic data set 1, the L value was set to 3 with the theta parameter = 1 and the temporal aggregate = 4. The IMSTAGRID Spatio-temporal clustering algorithm identified clusters per the original data. However, some clusters were less than indicated in the original dataset because some points were misidentified as noise. Fig. 16 below shows the clustering results.

Table 5 provides an overview of the performance of the IMSTAGRID algorithm in comparison with the baseline algorithms.

ST-DBSCAN is an adequate Spatio-temporal clustering algorithm for identifying spatial or temporal clusters. This algorithm uses a density-based approach in the formation of clusters. Each data point is processed individually. It increases computation time, mainly if the algorithm is applied to Spatio-temporal data based on real-world phenomena. AGRID + is a density- and grid-based algorithm. If fewer intervals are used, its performance is comparable to a density-based algorithm. If the number of intervals increases, its performance is comparable to a grid-based algorithm. Although AGRID + is designed to operate on data with multiple dimensions, it cannot be used directly to process Spatio-temporal data. Adaptations are needed in order to use this algorithm on Spatio-temporal data. The performance of each algorithm was compared using the synthetic data sets: synthetic data set 2 (scattered data type) and synthetic data set 1 (grouped data). The results of the experiment using synthetic data set 2 can be seen in Table 6, and the results of the experiment using synthetic data set 1 can be seen in Table 7.

For consistency, *L* = 1 and the *distance threshold* was set to 0.5 for the AGRID+, ST-AGRID, and IMSTAGRID algorithms. The Eps value was set to 0.5 for the ST-DBSCAN algorithm.

Table 6
Algorithms comparison on synthetic data set 2.

Algorithms	Total clusters	Silhouette index	Time (seconds)
ST-DBSCAN	158	−0.934	3416.941
AGRID+	588	0.787	34.424
ST-AGRID	107	0.939	11.425
IMSTAGRID	31	**0.970**	11.272

Table 7
Algorithms comparison with aggregate data set 1 synthetic 1.

Algorithms	Total cluster	Silhouette index	Time (Seconds)
ST-DBSCAN	11	0.367	1.494
AGRID+	26	0.245	2.0142
ST-AGRID	7	0.921	2.567
IMSTAGRID	20	**0.993**	2.194

Table 8
Clustering method performance accuracy comparison.

Methods	Accuracy (%)
IMSTAGRID	82.68
ST-AGRID	78.66
AGRID+	76.13
ST-DBSCAN	38.36

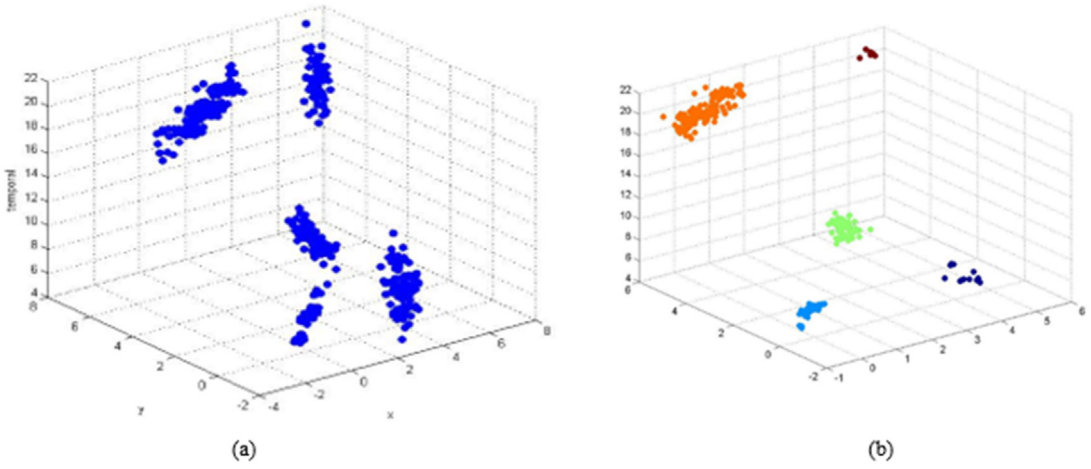


Fig. 16. (a) synthetic data set 1 plot (b) clustering results with L = 3.

Table 5
Differences of four algorithms along with their techniques.

Algorithms	Density based	Grid based	i-th order	*Density compensation*	Spatio-temporal *clustering*
ST-DBSCAN	✓				✓
AGRID+	✓	✓	✓	✓	
ST-AGRID	✓	✓	✓	✓	✓
IMSTAGRID	✓	✓	✓	✓	✓

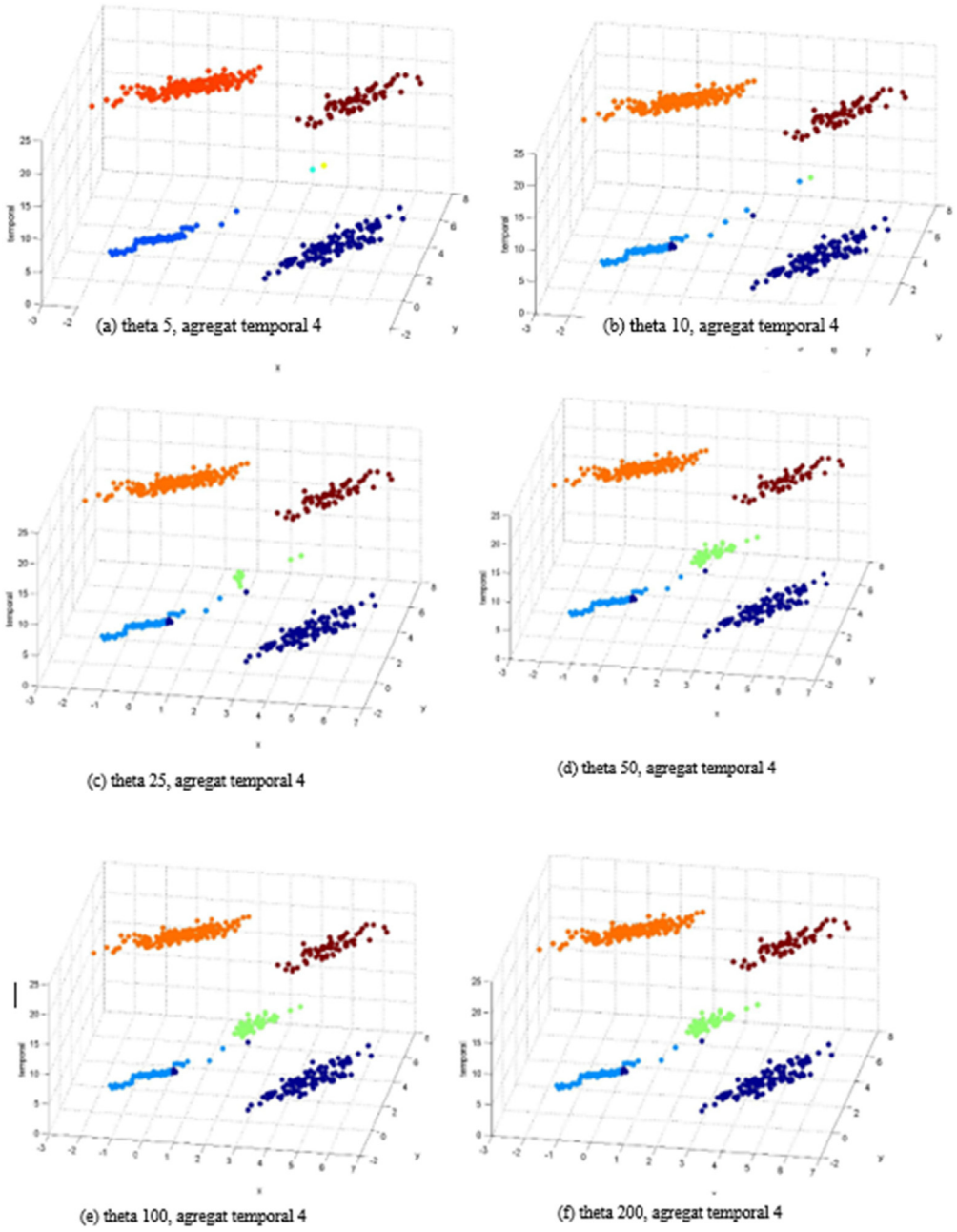


Fig. 17. Clustering results from different theta values.

In comparisons using aggregated synthetic data set 1, the *distance threshold* = 1.125 with L = 2.25 for the AGRID+, ST-AGRID, and IMSTAGRID algorithms. The eps value was set to 1.125 for the ST-DBSCAN algorithm.

The clustering accuracy of each algorithm was tested using the storm data set because it had predefined classes. There were seven classes in the storm data set: CYCLONE 1, CYCLONE 2, CYCLONE 3, CYCLONE 4, CYCLONE 5, TROPICAL DEPRESSION, and TROPICAL STORM. A KNN approach was utilized in order to improve accuracy. The data set was divided into training and test data in 70 % – 30 %. A 10-fold cross validation is utilized. Based on the confusion matrix result, the test data and predicted results were compared with the actual data. The comparison of the results can be seen in Table 8.

Based on the performance accuracy results above, the IMSTAGRID algorithm achieved the highest accuracy. The IMSTAGRID algorithm successfully classified data points according to the existing classes within the storm data set.

After analyzing the influence of L interval values on clustering and the determination of the performance accuracy, further experiments were conducted to investigate the importance of the value of theta to clustering. This experiment used several theta values with a temporal aggregate of 4. Synthetic data set 1 was used, which was known to have five data groups of varying densities. The clustering results can be seen in Fig. 17.

Fig. 17 shows that changing theta affects the number of clusters formed. Higher theta values result in a lower density threshold (DT). A lower DT will produce lower noise, while a higher DT will produce more noise. It causes the outermost points to be considered noise and omitted. Data points that were omitted because they were misidentified as noise can be observed in the lower theta

values; this is shown in Fig. 17(a). Groups that had varied center densities did not converge during clustering.

The data group in the middle of the density is very diverse. It means that during clustering process, this group is still unable to converge. With the increase in theta value, a small DT will result in less noise being removed.

Based on the results that have been achieved in several experiments, the proposed algorithm accommodates Spatio-temporal data structure, either the scatter type data or the grouped data. The proposed algorithm can provide better results in partitioning data to produce representative spatial and temporal sub-units usable in subsequent processes. This algorithm also showed superiority in classifying predicted data over the other algorithms. Yet the proposed algorithm still needs a lot of improvement, including determining the interval value optimally and automatically according to the volume of data (without pre-determination at the beginning).

3. Conclusions

The IMSTAGRID density- and cube-based Spatio-temporal *clustering* algorithm has been successfully developed by adapting the partitioning process, the *distance threshold* calculation, and the *density compensation* calculation during *clustering*. The dimension partition that produces the best result is based on an equal interval (L) and the addition of an expansion technique on the spatial and temporal dimensions. Based on the data sets, lower L values produce better *silhouette index* values for scattered data. Conversely, higher L values produce better *silhouette index* values for grouped data. From the experiment results, for the storm data set, the IMSTAGRID outperformed the ST-DBSCAN, AGRID+, and ST-AGRID algorithms in accuracy; the accuracy percentages were 82.68 %, 38.36 %, 76.13 %, 78.66 %, respectively. We also learn that the changes in θ (theta) or parameter tuning can be used to find clusters of varied densities.

4. Further study

In future work, an adjustable interval value (L) should be optimally and automatically determined for the spatial and temporal dimensions based on the spread of the data. It would enable the partitioning of the data cubes considering the volume of data. Furthermore, an optimal determination of θ (theta), which affects the algorithm's ability to discover clusters of varied densities, is required to improve the current iterative approach.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abraham, T., Roddick, J.F., 1998. Opportunities for knowledge discovery in spatio-temporal information systems. AJIS 5 (2).

Anbaroglu, B., Heydecker, B., Cheng, T., 2014. Spatio-temporal clustering for non-recurrent traffic congestion detection on urban road networks. Transp. Res. Part C 48, 47–65. https://doi.org/10.1016/j.trc.2014.08.002.

Birant, D., Kut, A., 2007. ST-DBSCAN: an algorithm for clustering spatial-temporal data. Data Knowl. Eng. 60 (1), 208–221. https://doi.org/10.1016/j.datak.2006.01.013.

Chen, W., Ji, M.H., Wang, J.M., 2014. T-DBSCAN : a spatiotemporal density clustering for gps trajectory segmentation. Int. J. Online Biomed. Eng. 10 (6), 19–24.

Cook, E., Saleem, M.B., Weng, Y., Abate, S., Kelly-Pitou, K., Grainger, B., 2022. Density-based clustering algorithm for associating transformers with smart meters via GPS-AMI data. Int. J. Electr. Power Energy Syst. 142, (PB). https://doi.org/10.1016/j.ijepes.2022.108291 108291.

Deng, X., Tang, G., Wang, Q., 2022. A novel fast classification filtering algorithm for LiDAR point clouds based on small grid density clustering. Geod. Geodyn. 13 (1), 38–49. https://doi.org/10.1016/j.geog.2021.10.002.

Ester, M., Sander, J., Kriegel, H.-P., Xu, X., 1998. A desity-based algorithm for discovering clusters. Data Min. Knowl. Disc. 2 (2), 169–194. https://doi.org/10.1023/A:1009745219419.

Fitrianah, D., Hidayanto, A.N., Fahmi, H., Lumban Gaol, J., Arymurthy, A.M., 2015. ST-AGRID: a spatio temporal grid density based clustering and its application for determining the potential fishing zones. Int. J. Softw. Eng. Its Appl. 9 (1). https://doi.org/10.14257/ijseia.2015.9.1.02.

Georgoulas, G., Konstantaras, A., Katsifarakis, E., Stylios, C.D., Maravelakis, E., Vachtsevanos, G.J., 2013. "Seismic-mass" density-based algorithm for spatio-temporal clustering. Expert Syst. Appl. 40 (10), 4183–4189.

Hu, L., Liu, H., Zhang, J., Liu, A., 2021. KR-DBSCAN: A density-based clustering algorithm based on reverse nearest neighbor and influence space. Expert Syst. Appl. 186, (August). https://doi.org/10.1016/j.eswa.2021.115763 115763.

Huang, M., & Bian, F. 2009. A Grid and Density Based Fast Spatial Clustering Algorithm. *2009 International Conference on Artificial Intelligence and Computational Intelligence*, 260–263. https://doi.org/10.1109/AICI.2009.228.

Kantardzic, M., 2003. Data Mining-Concepts, Models, Methods. and Algorithms.

Latifi-Pakdehi, A., Daneshpour, N., 2021. DBHC: a DBSCAN-based hierarchical clustering algorithm. Data Knowl. Eng. 135, (April). https://doi.org/10.1016/j.datak.2021.101922 101922.

Liu, X., Wan, C., Xiong, N.N., Liu, D., Liao, G., Deng, S., 2018. What happened then and there: top-k spatio-temporal keyword query. Inf. Sci. 453, 281–301.

Lorena, S., Zarman, W., & Hamidah, I. 2014. Analisis Dan Penerapan Algoritma C4.5 Dalam Data Mining Untuk Memprediksi Masa Studi Mahasiswa Berdasarkan Data Nilai Akademik. *Prosiding Seminar Nasional Aplikasi Sains Dan Teknologi (SNAST)*, (November), 263–272. https://doi.org/10.5829/idosi.weasj.2015.6.2.22162.

Roddick, J.F., & Lees, B. G. 2001. Paradigms for Spatial and Spatio-Temporal Data Mining. *Research Monographs in Geographic Information Systems*, (July), 1–14.

Roddick, J.F., Spiliopoulou, M., 1999. A bibliography of temporal, spatial and spatio-temporal data mining research. ACM SIGKDD Explor. Newslett. 1 (1), 34–38. https://doi.org/10.1145/846170.846173.

Sahu, S., Mardia, K., 2005. Recent trends in modeling spatio-temporal data. In: Proceedings of the Special Meeting on Statistics, pp. 1–15.

Sander, J., Ester, M., Kriegel, H.P., Xu, X., 1998. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. Data Min. Knowl. Disc. 2 (2), 169–194. https://doi.org/10.1023/A:1009745219419.

Shi, Z., Pun-cheng, L.S.C., 2019. Spatiotemporal data clustering : a survey of methods. Int. J. Geo-Inform. 8 (112). https://doi.org/10.3390/ijgi8030112.

Sun, Z., Zhao, Z., Wang, H., Ma, M., Zhang, L., Shu, Y., 2005. A fast clustering algorithm based on grid and density. Can. Conf. Electr. Comput. Eng. 2005, 2276–2279. https://doi.org/10.1109/CCECE.2005.1557443.

Tork, H. F. 2012. Spatio-temporal clustering methods classification. *Doctoral Symposium on Informatics Engineering (DSIE'2012), Porto, Portugal; 01/2012*.

Wang, M., Wang, A., Li, A., 2006. Mining spatial temporal clusters from geo-databases. Adv. Data Min. Appl. 2nd, 263–270.

Wang, R., Wang, Z., Osumanu, A., Zhang, G., Li, B., Lu, Y., 2019. Grid density overlapping hierarchical algorithm for clustering of carbonate reservoir rock types: a case from Mishrif Formation of West Qurna-1 oilfield, Iraq. J. Petrol. Sci. Eng. 182,. https://doi.org/10.1016/j.petrol.2019.106209 106209.

Wu, D., Shi, J., Mamoulis, N., 2018. Density-based place clustering using geo-social network data. IEEE Trans. Knowl. Data Eng. 30 (5), 838–851. https://doi.org/10.1109/TKDE.2017.2782256.

Yao, X., Zhu, D., Gao, Y., Wu, L., Zhang, P., Liu, Y.u., 2018. A stepwise spatio-temporal flow clustering method for discovering mobility trends. IEEE Access 6, 44666–44675. https://doi.org/10.1109/ACCESS.2018.2864662.

Zhao, Y., Cao, J., Zhang, C., Zhang, S., 2011. Enhancing grid-density based clustering for high dimensional data. J. Syst. Softw. 84 (9), 1524–1539. https://doi.org/10.1016/j.jss.2011.02.047.

Zhu, Q., Tang, X., Elahi, A., 2021. Application of the novel harmony search optimization algorithm for DBSCAN clustering. Expert Syst. Appl. 178, (April). https://doi.org/10.1016/j.eswa.2021.115054 115054.