Complex & Intelligent Systems (2022) 8:1737–1761
https://doi.org/10.1007/s40747-021-00634-0

ORIGINAL ARTICLE

Hierarchical clustering for multiple nominal data streams with evolving behaviour

Jerry W. Sangma[1] · Mekhla Sarkar[2] · Vipin Pal[1] · Amit Agrawal[3] · Yogita[1]

Received: 11 June 2021 / Accepted: 17 December 2021 / Published online: 7 January 2022

Abstract
Over the decade, a number of attempts have been made towards data stream clustering, but most of the works fall under clustering by example approach. There are a number of applications where clustering by variable approach is required which involves clustering of multiple data streams as opposed to clustering data examples in a data stream. Furthermore, a few works have been presented for clustering multiple data streams and these are applicable to numeric data streams only. Hence, this research gap has motivated current research work. In the present work, a hierarchical clustering technique has been proposed to cluster multiple data streams where data are nominal. To address the concept changes in the data streams splitting and merging of the clusters in the hierarchical structure are performed. The decision to split or merge is based on the entropy measure, representing the cluster's degree of disparity. The performance of the proposed technique has been analysed and compared to Agglomerative Nesting clustering technique on synthetic as well as a real-world dataset in terms of Dunn Index, Modified Hubert Γ statistic, Cophenetic Correlation Coefficient, and Purity. The proposed technique outperforms Agglomerative Nesting clustering technique for concept evolving data streams. Furthermore, the effect of concept evolution on clustering structure and average entropy has been visualised for detailed analysis and understanding.

Keywords Data streams · Hierarchical clustering · Concept evolution · Nominal data

Introduction

Nowadays, data are generated continuously from different sources, such as sensors, web-browsing activities, network routers, etc. These continuously flowing data happen to be of very large volume and the patterns prevailing in it keep on changing with time. These patterns actually represent the behaviour of the underlying source from where the data are being generated. This form of data where new patterns keep on evolving and the size of data is ever increasing has been termed as a data stream. Application of traditional data-mining techniques which have been designed with an assumption that entire data are available for processing at a time and whose behaviour is static results in poor performance on data streams. Therefore, designing data-mining techniques for data streams is the need of the hour.

Clustering is a data-mining technique which groups the data objects into different groups in such a way that intra-group similarity between objects is maximised and inter-group similarity is minimised. In the context of data streams, the clustering problem can be formulated as per the two different approaches. In the first approach, the data examples of a single data stream are clustered into different groups; whereas, as per the second approach, different data streams in itself are clustered into different groups. The first approach is referred to as clustering by example and the second approach is referred to as clustering by variable. Overall, it can be said that in case of clustering by example the main focus is on profiling the relationship between different data instances of a

✉ Jerry W. Sangma
jerry.sangma@nitm.ac.in

Mekhla Sarkar
d0829005@cgu.edu.tw

Vipin Pal
vipinpal@nitm.ac.in

Amit Agrawal
amitagrawal1909@gmail.com

Yogita
yogitathakran@nitm.ac.in

[1] National Institute of Technology Meghalaya, Shillong, Meghalaya, India

[2] Chang Gung University, Guishan 33302, Taiwan

[3] Wells Fargo & Company, Bengaluru, India

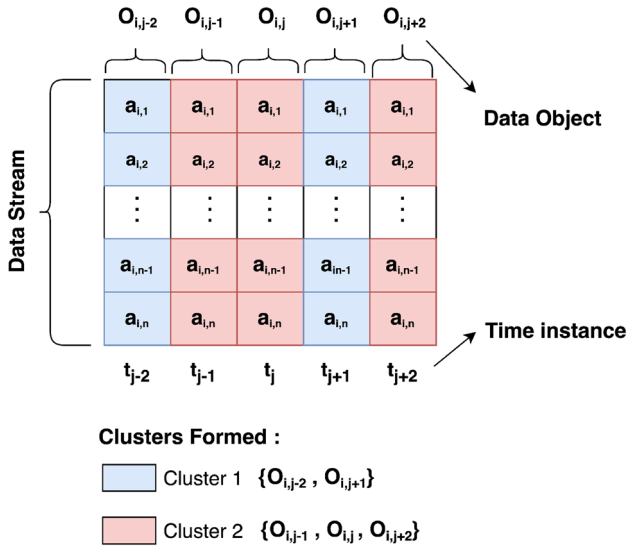

Fig. 1 Clustering by example approach

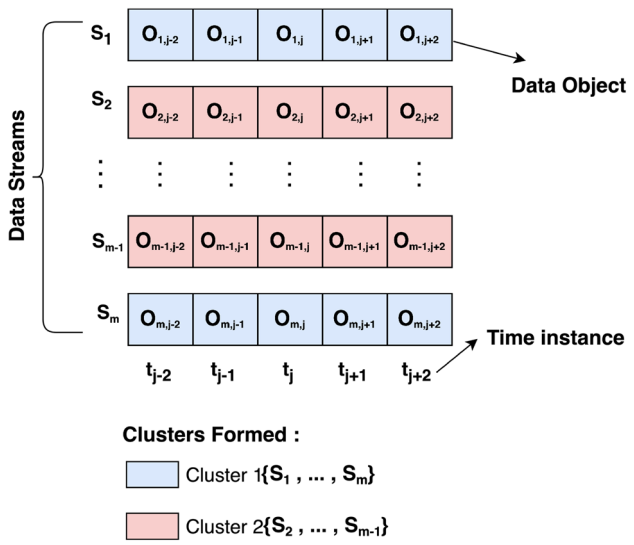

Fig. 2 Clustering by variable approach

single data stream; whereas, in the case of clustering by variable, the main focus is on profiling the relationship between multiple data streams. Clustering by variable approach has its own set of applications in a real-world scenario. For example, providing targeted ads for a group of customers based on their product purchase and browsing histories, grouping the users based on their preferred genres of music and providing similar music suggestions, grouping the web-users based on their web-browsing behaviours for promotional advertisement, endorsement advertisement, bandwagon advertisement, etc.

The clustering by example and clustering by variable approaches are diagrammatically shown in Figs. 1 and 2, respectively. In the above figures, $O_{i,j} = \{a_{i,1}, a_{i,2}, \ldots, a_{i,n}\}$ is a data object containing n attributes which arrived at jth time instant in the ith data stream. In Fig. 1, two clusters has been created using clustering by example approach, where cluster 1 = $\{O_{1,j-2}, O_{4,j+1}\}$ and cluster 2 = $\{O_{2,j-1}, O_{3,j}, O_{5,j+2}\}$. In case of clustering by example approach, each of the data streams maintain its individuality in terms of clusters, and in an application, there may be only one relevant data stream, or there may be many more. In case of clustering by variable approach, it can be seen from Fig. 2 that two clusters have been formed by grouping the m data streams (S_i's), such that cluster 1 = $\{S_1, \ldots, S_m\}$ and cluster 2 = $\{S_2, \ldots, S_{m-1}\}$.

On surveying the literature, it has been found that most of the existing works have addressed the problem of clustering by example [1–4,7,8,10,16,17,19,21,23,25,27,30,33] and a few attempts have been made towards the problem of clustering by variable [5,6,9,11,22,28,29]. Furthermore, these works are mainly suitable for numeric data streams. However, in several applications, other types of data have also come in the form of data streams such as nominal, text, web-data, etc. In the case of data with nominal attributes, there is no ordering between nominal attributes' values. However, for the sake of using the clustering approaches proposed for numeric attributes, conversion of nominal attributes to numeric attributes introduces unnecessary orderings between the values of numeric attributes representing the values of nominal attributes. Hence, after converting to numeric attributes, the operation performed on nominal attributes is not meaningful and leads to misleading results.

In totality, based on the literature survey, no work has been presented for clustering multiple nominal data streams. Therefore, there is a scope and need to work on clustering techniques for multiple data streams in case of nominal and other types of data.

In the present work, a hierarchical clustering by variable technique for multiple nominal data streams has been proposed. It is an integrative technique in the sense that it employs cosine distance for measuring the dissimilarity between data streams and the entropy for computing the degree of disparity within a cluster. The extent to which the data observations in a cluster show disorderedness or randomness signifies the cluster's disparity with higher values indicating higher disorderedness or randomness. To deal with the continuously flowing nature of the data streams, the proposed technique processes the data incrementally where the increment interval is equal to the size of the sliding window. Furthermore, it adapts the hierarchical structure of clusters by splitting and/or merging the clusters to incorporate the evolving behaviour of data streams where new concepts keep on coming and old may fade out.

The performance of the proposed technique has been analysed on synthetic datasets as well as a real-world dataset and compared to Agglomerative Nesting (AGNES) clustering technique in terms of four clustering validation measures,

viz., Dunn Index (DI) [24], Modified Hubert Γ statistic (MHΓ) [24], Cophenetic Correlation Coefficient (CPCC) [14], and Purity [31].

The main contributions of the proposed work have been given below:

- A method has been proposed for clustering multiple nominal data streams using a hierarchical clustering by variable approach.
- The proposed method is able to handle the concept drifts in the data streams through the merge/split operations of the nodes in the hierarchical clustering structure.

The remainder of the paper is organised as follows. In the section "Literature review", literature review has been discussed. In the section "Problem formulation and preliminaries", the problem formulation and preliminaries has been given followed by the presentation of the proposed method in the section "Proposed technique". In the section "Datasets and performance measures", the datasets and performance measures has been presented. In the section "Experimental results and analysis", experimental results has been discussed. Finally, in "Conclusion", concluding remarks has been made.

Literature review

Over the years, many techniques for clustering data streams have been proposed. Among those proposed techniques, the discussion relating to clustering by variable techniques has been discussed in this section, which is the focus of the current work.

Dai et al. [11] have presented a clustering on demand (COD) framework which worked in two phases. The online phase stored statistics for the incoming data observations in terms of sliding windows, whereas in the offline phase, stored statistics were used for generating clusters. The advantage of COD framework is in its ability to process the data observations from multiple data streams in a single pass and off loading the actual clustering process to the the offline phase, thus staying true to the constraints in the data stream scenario. Balzanella et al. [5] have proposed a graph-based technique for clustering multiple data streams which collects data observations from the data streams in terms of sliding window and creates summaries out of it. It maintains an undirected graph whose adjacency matrix stores the similarity between the data streams and is updated on every new window of data by applying Dynamic Clustering Algorithm [12] on it. The final clustering structure of the data streams is obtained by applying a partition based clustering technique on the summaries stored online.

Ling et al. have proposed a spectral component-based clustering technique for clustering multiple data streams called SPE-cluster [9]. Here, the data from the data streams are taken in sequential non-overlapping sliding windows where in each window, the data sequences of the respective data streams are represented as the sum of the spectral components. This technique addresses the lag-correlation between the data streams while computing the similarity between the data streams which is ignored in other data stream clustering techniques using Euclidean distance. This technique also works in an online–offline phase. In the online phase, it calculates the spectral components of the data streams, while the offline phase employs dynamic k-means for clustering the most recent sliding window. In [29], the author has proposed a Kendall correlation-based clustering technique for multiple data streams. Here, the sliding window technique is used to gather data observations from the incoming data streams. For clustering the data streams, it uses a modified k-means algorithm which can adjust the number of clusters to reflect the evolving changes in the data streams.

Bones et al. [6] proposed a data stream clustering technique which clusters similar data streams based on the correlation of the attribute values. It uses a sliding window technique, and for each window, a fractal value is calculated in a fractal dimension, which is a reduced dimension of the original dimension of the data streams. This fractal value represents the correlation of the attribute values from the original dimension and is found to cluster the data streams better. Laurinec and Lucka [22] have proposed ClipStream. This technique consisted of two phases, online (data abstraction) and offline phases. In the data abstraction phase, the data from the data streams are processed window wise and a reduced feature vector called FeaClip is constructed from the original feature space, thus representing a clipped version of the data streams. The clipped representation of the data streams captured two behaviours of a data stream: global statistics of the data stream and local behaviour of the data stream. In the offline phase, clustering is performed using the k-medoid clustering technique. Since offline clustering is time-consuming, the change detection module of the ClipStream executes when the data streams evolve.

Online Divisive-Agglomerative Clustering (ODAC) [28] was proposed by Rodrigues et. al. It is a hierarchical clustering approach for multiple data streams and creates a hierarchy of tree nodes. In this technique, each node of the hierarchical tree comprises of data streams, where the leave nodes represent the clusters. For handling concept evolution, the nodes of the hierarchical tree are split and/or merged. The decision for either splitting or merging is done based on the diameter of the cluster and Hoeffding bound [20]. This method is suitable only for numerical data streams as it is dependent upon Pearson's correlation coefficient [26] which is used as the

similarity measure and the entire clustering process is based on this measure.

It can be observed that over approximately 15 years, very few works have been presented under the clustering by variable category. Those mainly fit numerical data streams. Moreover, nominal values do not have any exact order and are not quantitative [18]. Therefore, converting nominal values to numeric values does not make sense. Any effort to perform mathematical operations on nominal attributes after converting them to numerical attributes will not be meaningful. For example, a nominal attribute colour will have values *red*, *green*, *blue*, etc. Assigning numerical values to these values, for example, *red=1*, *green=2*, and *blue=3*, will not make any sense, since the values for colours are not quantitative. Hence, finding mean, median, or any other statistic on such numerical representations of the nominal values will not be meaningful. In this present work, we have proposed a hierarchical clustering technique for multiple nominal data streams. The main difference between ODAC [28] and the clustering technique proposed in the present work lies in the similarity measure used and its computation and the type of data that each method can handle. The technique proposed in the current paper is targeted at multiple nominal data streams. In contrast, ODAC [28] is focused explicitly on numerical data streams and is not suitable for nominal data streams.

Problem formulation and preliminaries

In this section, the problem of clustering by variable for multiple data streams has been introduced. Furthermore, the processes for calculating the dissimilarity measures between the data streams and the entropy values for the clusters have been discussed. Also, the notations used throughout the paper have been described.

Problem formulation

A data stream consists of data observations produced at different time instances. Let $DS = \{S_1, S_2, \ldots, S_i, \ldots, S_m\}$ represent the set of data streams where S_i is the ith data stream in the set DS which comprises in total m data streams. Each $S_i = \{o_{i,1}, o_{i,2}, \ldots, o_{i,j}, \ldots, o_{i,\infty}\}$ where $o_{i,j}$ is a data observation observed in the jth time instance(t_j) belonging to the ith data stream (S_i). The clustering of multiple data streams using clustering by variable approach aims to group together those data streams which are producing similar observations over the time. However, additional challenges need to be addressed in clustering data streams, such as the continuous arrival of data in the data streams. Hence, in the proposed work, a snapshot of the data streams' data is taken to handle this ever-increasing size. A data snapshot is extracted using a sliding window technique, as shown in Fig. 3, and

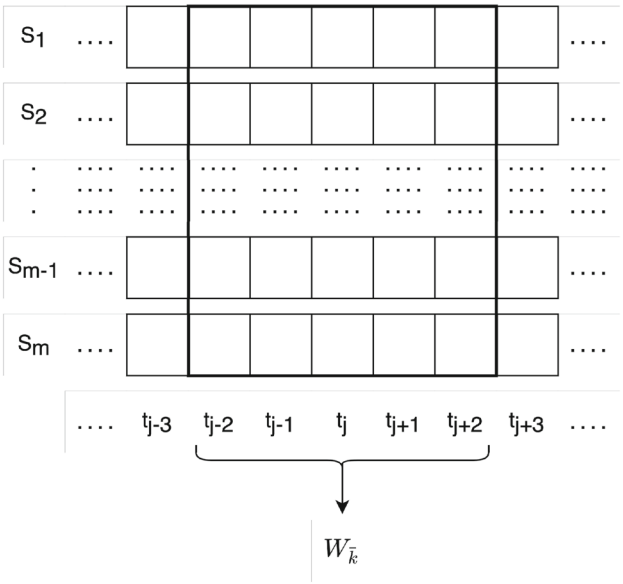

Fig. 3 Example of a sliding window ($W_{\bar{k}}$) with window size(w) equal to 5 operating on data streams S_1 to S_m through time t_{j-2} to time t_{j+2}

processed. In Fig. 3, $W_{\bar{k}}$ is the $\bar{k}$th sliding window containing examples in the time frame t_{j-w+1} to t_j from m data streams where w is the size of the sliding window. Furthermore, there may be concept evolution as new data observations keep on adding to the data streams which requires update in the clustering structure generated using previous sliding window's data. In the proposed work, the update in the clustering structure has been considered by allowing merge and/or split operations on clusters (for detail, refer "Merge sub-module" and "Split sub-module").

Hierarchical clustering requires no prior information on the number of clusters and maintains a hierarchical tree of clusters at different levels. Each node in the hierarchical tree represents a cluster. Except for the leaf nodes, all other clusters are a combination of their child clusters. The hierarchical tree can be cut at any level to obtain a set of clusters. Agglomerative and divisive are the two strategies for generating hierarchical clustering. The first one follows a bottom–up approach by starting with single data observation and then iteratively merging them to form larger clusters. The second one follows a top–down strategy by starting with a single cluster comprising all the data observations and then iteratively splitting them into smaller clusters. The merge and split operations are final in the case of traditional hierarchical clustering.

In the context of clustering multiple data streams, the hierarchical clustering structure needs to be updated over time. Updation becomes necessary due to the evolution of new concepts in streaming data which may involve the combination of both split and merge operations based on some clusters parameters.

Table 1 Selected data streams

S_i	$o_{i,1}$	$o_{i,2}$	$o_{i,3}$	$\cdots$	$o_{i,w}$
S_j	$o_{j,1}$	$o_{j,2}$	$o_{j,3}$	$\cdots$	$o_{j,w}$

In the present work, a hierarchical clustering technique for multiple nominal data streams has been proposed. The proposed technique integrates both agglomerative and divisive strategies for updating the hierarchical structure on the arrival of a window of new data observations based on the sliding window technique. Furthermore, it employs cluster entropy as a parameter for deciding whether to split/merge or not to split/merge the clusters. Under the proposed technique, the cluster results can be viewed and analysed at any time, depending on the user's requirement.

Notations

This section describes the notations used throughout the paper.

- Let $DS = \{S_i, 1 \leq i \leq m\}$ where
  - DS is the set of data streams.
  - S_i is the ith data stream in the set DS.
  - m is the number of data streams in the set DS.
- Let $N = \{C_i, 1 \leq i \leq \mid N \mid\}$ where
  - N is the set of nodes in the hierarchical tree.
  - C_i is the ith node in the set N.
  - C_i^l is the ith leaf node in the hierarchical tree and $C_i^l \epsilon N$.
  - $\mid N \mid$ is the number of nodes in the set N.
  - $\mid K \mid$ is the number of leaf nodes in the set $\mid N \mid$.
- $\mid C_i^l \mid$ is the number of data streams in the leaf node C_i^l.
- C_i^p is the immediate parent node of C_i^l in the hierarchical tree and $C_i^p \epsilon N$.
- D_r is the latest data snapshot.
- w is the size of the sliding window.
- d_{init} is the number of initial sliding windows.
- $E_i = \{e_i, 1 \leq i \leq \mid N \mid\}$ where
  - e^i is the entropy for an ith node(C_i) in the hierarchical tree.

Computation of dissimilarity between data streams

In the proposed technique, the dissimilarity between any two data stream is calculated by applying cosine distance over a data snapshot extracted corresponding to a sliding window as discussed below.

Table 2 Frequency matrix for the selected data streams

	o_1	o_2	o_3	$\cdots$	$o_{\bar{w}}$
S_i	$f_{i,1}$	$f_{i,2}$	$f_{i,3}$	$\cdots$	$f_{i,\bar{w}}$
S_j	$f_{j,1}$	$f_{j,2}$	$f_{j,3}$	$\cdots$	$f_{j,\bar{w}}$

Step 1: A data snapshot is extracted from the data streams. Next, two data streams are selected whose distance is to be calculated (say S_i and S_j), as shown in Table 1.
Step 2: Next, a frequency matrix (F) for S_i and S_j is created, as shown in Table 2.
In the above frequency matrix (F), as shown in Table 2, f_{ik} and f_{jk} are the frequency of occurrence of the value o_k in S_i and S_j respectively, and $1 \leq k \leq \bar{w}$, where $\bar{w}$ is the number of unique values occurring in S_i and S_j.
Step 3: Finally, the cosine distance is calculated as given in Eq. (1)

$$\text{cosine}(S_i, S_j) = \frac{\sum f_{i,k} X f_{j,k}}{\sqrt{\sum f_{S_i}^2} X \sqrt{\sum f_{S_j}^2}}. \tag{1}$$

Computing entropy of clusters

For calculating the entropy of a cluster, the following steps are followed.

- *Step 1*: Let a cluster C_i comprise of a number of data streams where each data stream comprises b data instances. Let us say all those data observations are stored in a matrix A^i where the dimension of A^i is $(a \times b)$.
- *Step 2*: Next, the unique values in A^i are extracted and stored in a vector U^i whose size is equal to the number of unique values appearing in A^i.
- *Step 3*: For each unique value stored in U^i, its corresponding count of occurrence in A^i is taken and stored in a vector $\bar{U}^i$ of size $\mid U^i \mid$.
- *Step 4*: Finally, the entropy (e^i) for the ith cluster (C_i) is calculated as shown in Eq. (2)

$$e^i = -\sum_{k=1}^{|\bar{U}^i|} \left(\frac{\bar{U}_k^i}{\sum_{j=1}^{|\bar{U}^i|} \bar{U}_j^i} \log \frac{\bar{U}_k^i}{\sum_{j=1}^{|\bar{U}^i|} \bar{U}_j^i} \right). \tag{2}$$

For a node containing only a single data stream, the entropy value is set to zero. However, the entropy value for a node containing two or more than two streams can range from zero to $\log \mid U^i \mid$.

Table 3 Example of an ith data snapshot (D_i)

user1	bbc	cnn	netflix	youtube
user2	youtube	cnn	cnn	google
user3	medium	udemy	google	hulu
user4	hbo	IEEE	google	steam
user5	springer	IEEE	twitter	google

Proposed technique

The overall working of the proposed method is shown in Fig. 4 with the help of a flowchart and explained in the subsequent sections. It comprises three main modules, *viz.,* initialisation module, accommodation sub-module, and update module. These modules are highlighted in Fig. 4 with the help of dotted lines. The initialisation module executes only once for creating the initial hierarchical clustering structure, whereas the accommodation and update modules keep on repeating as the new data snapshots keep on arriving. From Fig. 4, it can be seen that the proposed method first executes the sub-modules in the initialisation module, followed by the sub-modules in the update module. In the initialisation module, the proposed method acquires the initial data snapshot from the data streams. This initial data snapshot is then used to create an initial hierarchical clustering structure, after which the entropies for the nodes of the resulting initial hierarchical clustering structure are calculated. The proposed method then executes the update module, where the next data snapshot is incorporated into the existing hierarchical clustering structure. Again, the entropies are re-calculated for the nodes of the hierarchical tree using the newly acquired data snapshot. The hierarchical clustering structure is then tested for modification when the changes in the node entropies are significant. The changes to the hierarchical clustering structure are done through the merge/split operations. The above process for the update module is then re-iterated for the subsequent data snapshots.

Data snapshot

A data snapshot from the data streams is obtained using the sliding window technique. The sliding window technique extracts data from m data streams by obtaining w data objects from each of the m data streams. In the proposed work, w is understood as the size of the sliding window. Hence, a data snapshot is a matrix of size $(m \times w)$ containing data objects belonging to m data streams. An example of an ith data snapshot (D_i) is shown in Table 3. Each row in D_i represents a user generating a data stream of values. The generated values are that of the websites visited by the respective users. The size of D_i is (5×4), where the number of users is five and the number of visited websites is four by each user, respectively.

The data stream processing in the proposed work has been done data snapshot wise. It helps in handling the ever-increasing size of the data streams as it is not possible to make the entire data streams available in one go for processing.

Initialisation module: initialisation of the hierarchical clustering structure

The main essence of the initialisation step is to capture the clustering structure prevailing over some initial data snapshots, so that it can be used as a foundation structure and can be updated when more data keep on streaming. For creating the initial hierarchical clustering structure, the data corresponding to d_{init} number of initial data snapshots have been used. Hence, in total $d_{\text{init}} \times w$, the number of data observations from each of the m data streams is used for constructing the initial hierarchical clustering structure. The creation of the hierarchical tree structure using the data of d_{init} data snapshots is as given below:

- *Step 1*: For the construction of the hierarchical clustering structure, Agglomerative Nesting (AGNES) has been used along with the average linkage method as a measure for merging clusters and cosine distance as a dissimilarity measure. In the hierarchical clustering structure, each node represents a cluster consisting of one or more data streams.
- *Step 2*: Furthermore, the entropy corresponding to each of the node in the hierarchical clustering structure is computed as discussed in the section "Computing entropy of clusters".
- *Step 3*: Next, the entropy corresponding to each level of the hierarchical tree is calculated. The entropy of a level is defined as the average of the entropies of nodes available at a particular level as given in Eq. 3

$$\bar{E}_q = \frac{1}{\mid q \mid} \sum_{j}^{\mid q \mid} e_j^i. \tag{3}$$

  In the above equation, $\bar{E}_q$ represents the average entropy of nodes at the qth level of the hierarchical tree. e_j^i represents the entropy of an ith node (C_i) belonging to the qth level of the hierarchical tree and $\mid q \mid$ represents the total number of nodes (C_i) at the qth level of the hierarchical tree.
- *Step 4*: The hierarchical tree is cut with the help of an elbow method depending upon the change in the entropy value from one level to another. Let us say there is a maximum decrement in entropy value from qth $-$ 1 to qth level, then a cut is marked below the qth level, and the nodes at the qth level are considered as leaf nodes. This step is performed to prune out that part of the hierar-

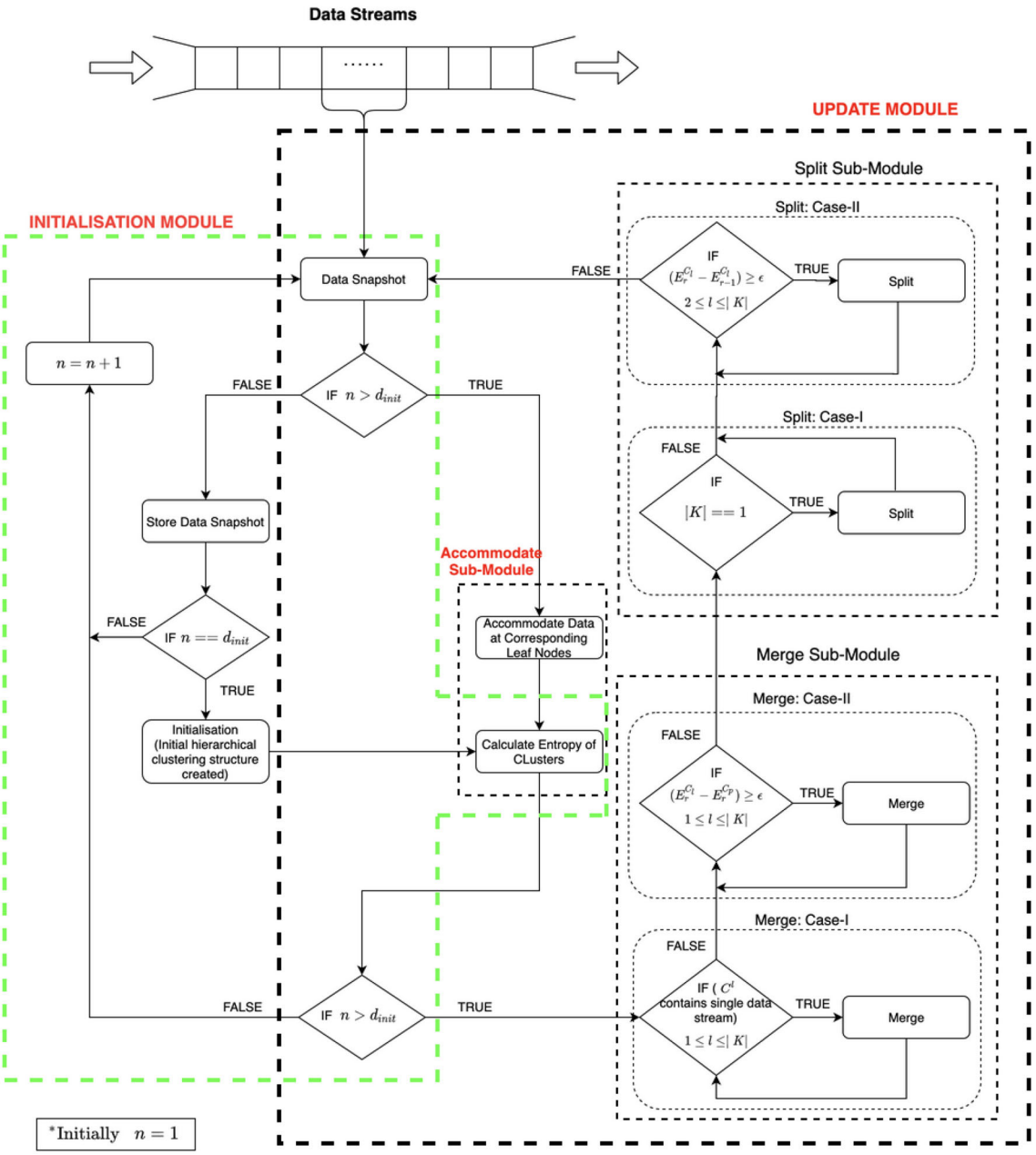

Fig. 4 Working of the proposed technique

chical tree where the changes in the entropy level of the hierarchical tree are only marginal.

– *Step 5*: The hierarchical clustering structure after being cut is then used as a base hierarchical clustering structure for processing later incoming data snapshots as discussed in the next section.

Update module: updating the initial hierarchical clustering structure

In the case of data streams, concept evolution may occur over time. The clustering structure as per the concept evolution is updated through merge and split operations. After the initial hierarchical clustering structure has been created as explained in the initialisation module (refer to the section "Initialisation module: initialisation of the hierarchical clustering structure"), the update module processes the next incoming data snapshots one after another. The update mod-

ule mainly intends to update the clustering structure with incoming data snapshots, so that the concept evolving nature of data streams can be reflected in clustering. The update module comprises three sub-modules, *viz.,* accommodation sub-module, merge sub-module, and split sub-module. The functioning of these three sub-modules has been discussed next.

Accommodation sub-module

This sub-module on receiving the new data snapshot let say ith data snapshot (D_i), assigns the data instances from the respective data streams of D_i to the leaf nodes (clusters) where the corresponding data stream lies and discards the data instances of the D_{i-1} data snapshot from the leaf nodes (clusters), but it keeps the entropy value e^i_{i-1} of each node C_i corresponding to D_{i-1} data snapshot for further processing.

Merge sub-module

The merge operation in the proposed method handles two cases, i.e., *merge case-I* and *merge case-II*. In *merge case-I*, the clusters containing a single data stream are merged, and in *merge case-II*, the clusters containing at least two data streams are merged. The merging operation is performed based on the difference in the entropy of the parent and child node. However, in the case of a cluster having a single data stream, the entropy of a cluster happens to be zero. In this scenario, it is not logical to decide on the merge operation depending upon the difference in the entropy of the parent and child node. Hence, to address this exceptional scenario, the *split case-I* is used. As per the proposed framework *merge case-I* is tested first followed by the second case(*merge case-II*).

- *Merge case-I*: For all the leaf nodes(clusters) containing a single data stream (C^l_s), the steps below are executed:
  - *Step-1*: Calculate average entropy for the entire hierarchical clustering structure denoted by ξ_r. ξ_r is defined as the average of the summation of entropies of all the nodes in the hierarchical tree denoted for the rth data snapshot(D_r), as shown in Eq. (4)

$$\xi_r(\text{original hierarchical tree}) = \sum_{i=1}^{|N|} e^i_r. \tag{4}$$

  - *Step-2*: Using the average linkage method, calculate the distance between C^l_s and other leaf nodes (C^l_i) to identify the target leaf node (say C^l_{target}) which is closest to C^l_s in terms of distance.
  - *Step-3*: On finding C^l_{target}, the sibling and intermediate nodes to C^l_s and C^l_{target} are iteratively merged starting from the bottom of the hierarchical tree until both C^l_s and C^l_{target} get merged into a common immediate parent node, as shown in Fig. 5.
  - *Step-4*: The average entropy of the modified hierarchical clustering structure denoted by $\acute{\xi}_r$ (generated by *step-3*) is calculated and compared with the average entropy of the original (unmodified) hierarchical clustering structure represented by ξ_r for the rth data snapshot (D_r), as shown in Eq. (5). If Eq. (5) is satisfied, then the modified hierarchical clustering structure is used for further processing; otherwise, the modification in the original hierarchical clustering structure is considered null and void and the original hierarchical clustering structure is used for further processing

$$\acute{\xi}_r < \xi_r. \tag{5}$$

- *Merge case-II*: It focuses on all those leaf nodes that comprise more than one data stream. The steps for this operation drafted below are executed for all such leaf nodes.
  - *Step-1*: Calculate entropy for the leaf node C^l_i. Let the entropy of C^l_i for the rth data snapshot be represented by $\dot{e}^i_r$. Let C^p_i be the immediate parent node of the leaf node C^l_i. Let the entropy of C^p_i for the $(r-1)$th data snapshot be represented by $\ddot{e}^i_{r-1}$.
  - *Step-2*: If the entropy of the leaf node ($\dot{e}^i_r$) is higher than its immediate parent node's entropy ($\ddot{e}^i_{r-1}$) by a factor greater than equal to ϵ, as shown in Eq. (6), then the leaf node and its sibling nodes are merged into its immediate parent node, as shown in Fig. 6. Following the merging process of C^l_i and its sibling nodes into their immediate parent node C^p_i, the immediate parent node C^p_i becomes the new leaf node

$$(\dot{e}^i_r - \ddot{e}^i_{r-1}) \geq \epsilon. \tag{6}$$

    Step-2 is based on the fact that if Eq. (6) is satisfied, then it implies that the data streams in C^l_i are having a high variation to one another as compared to the data streams in C^p_i due to the data instances in the r^{th} data snapshot (D_r) as compared to the data instances in the $(r-1)$th data snapshot (D_{r-1}). High entropy of the leaf node (C^l_i) in comparison to its immediate parent node (C^p_i) represents a distorted hierarchical clustering structure requiring a merge operation for rectification. Moreover, in *step-2*, the ϵ represents the threshold to decide whether to merge, or to not merge the leaf node (C^l_i) and its sibling node to their parent

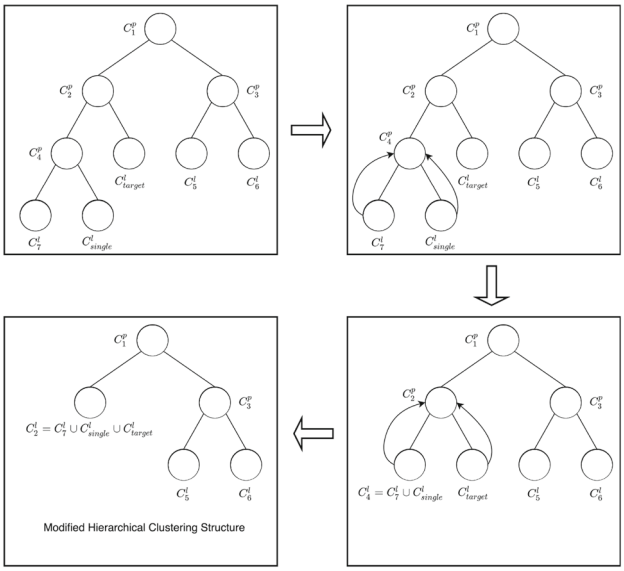

Fig. 5 Example of *merge case-I*

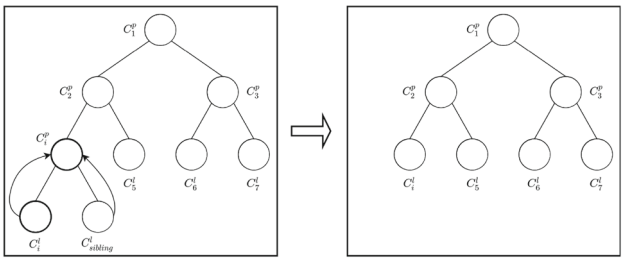

Fig. 6 Example of *merge case-II*

node (C_i^p). The value of ϵ has been decided based on Hoeffding bound as discussed in [28] and further explained in the section "Threshold (ϵ)".

Split sub-module

The split operation is executed for the rth data snapshot D_r once the merge operation is completed for the same data snapshot (D_r). The split operation in the proposed method handles two cases, i.e., split case-I and split case-II. Split case-I is used for splitting only when the hierarchical clustering structure contains a single node following the single or multiple merge operations. Split case-II is executed in all other scenarios. The split case-I captures the exceptional scenario where the complete restructuring of the hierarchical clustering structure is required due to changes in underlying concepts.

- *Split case-I*: For splitting a hierarchical clustering structure containing only a single cluster (say C_s^l), the following steps are taken:
  - *Step-1*: Calculate entropy for the single cluster (C_s^l) and let its entropy be represented by $\dot{e}_r^s$.
  - *Step-2:* Next, the two most dissimilar data streams in C_s^l in terms of cosine distance is found. Let the two most dissimilar data streams be denoted by S_a and S_b, respectively.
  - *Step-3:* Create two child nodes for C_s^l. In one of the two newly created child nodes S_a is added, whereas in the other child node, S_b is added.
  - *Step-4:* For each data stream (say S_i) in C_s^l, excluding S_a and S_b, the cosine distance between S_i and S_a, and S_i and S_b is calculated. S_i is then added to the child node containing S_a if its cosine distance to S_a is the least, else S_i is added to the leaf node containing S_b.
  - *Step-5:* Calculate average entropy for the entire hierarchical clustering structure obtained after *step-4* which is defined as the average of the summation of entropies of all nodes in the hierarchical tree denoted by ξ_r for the rth data snapshot (D_r), as shown in Eq. (7)

$$\xi_r = \frac{1}{\mid N \mid} \sum_{i=1}^{|N|} e_r^i. \tag{7}$$

  - *Step-6:* Check the difference between the average entropies of the hierarchical clustering structure containing only a single cluster and the modified hierarchical clustering structure obtained after *step-4* represented by e_r^s and ξ_r, respectively. If Eq. (8) is satisfied, then the modified hierarchical clustering structure is used for further processing; otherwise, the modification in the original hierarchical clustering structure is considered null and void and the original hierarchical clustering structure is used for further processing

$$(\xi_r - \dot{e}_r^s) \leq \epsilon. \tag{8}$$

- *Split case-II*: For splitting a hierarchical clustering structure containing two or more clusters, the following steps are taken:
  - *Step-1*: Calculate entropy for the leaf node C_i^l. Let the entropy of C_i^l for the rth data snapshot be represented by $\dot{e}_r^i$ and $\dot{e}_{r-1}^i$ represents the entropy for the leaf node C_i^l for the $(r-1)^{th}$ data snapshot which is already calculated and stored as discussed in the section "Accommodation sub-module" and does not require to be calculated again.
  - *Step-2*: If $\dot{e}_r^i$ is greater than equal to $\dot{e}_{r-1}^i$ by a factor of ϵ, as shown in Eq. (9), then the leaf node(C_i^l) is

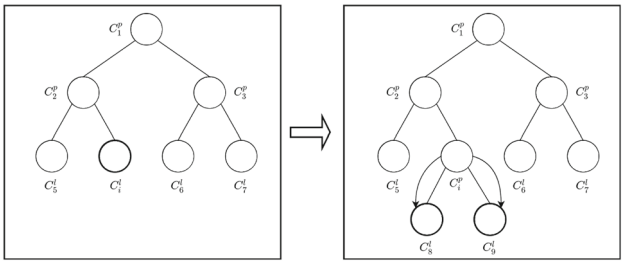

Fig. 7 Example of a node split

splitted, as shown in Fig. 7

$$(\dot{e}_r^i - \dot{e}_{r-1}^i) \geq \epsilon. \tag{9}$$

The steps for splitting C_i^l have been detailed below:

Step-2a: First, the two most dissimilar data streams in C_i^l in terms of cosine distance are found. Let the two most dissimilar data streams be denoted by S_a and S_b, respectively.

Step-2b: Next, two child nodes for C_i^l are created. In one of the two newly created child nodes, S_a is added, whereas in the other child node, S_b is added.

Step-2c: For each data stream (say S_i) in C_i^l, excluding S_a and S_b, the cosine distance between S_i and S_a, and S_i and S_b is calculated. S_i is then added to the child node containing S_a if its cosine distance to S_a is the least, else S_i is added to the leaf node containing S_b.

The splitting of a leaf node (C_i^l) on satisfying Eq. (9) as described in *step-2* above implies that the data streams in C_i^l is having a high variation to one another due to the data observations in the rth data snapshot (D_r) as compared to the instance when C_i^l had data observations from the $(r-1)$th data snapshot (D_{r-1}). High entropy of the leaf node (C_i^l) due to the rth data snapshot (D_r) indicates incompatibility between the data streams in C_i^l and hence requiring a split operation for rectification.

Threshold (ϵ)

The threshold (ϵ) used in Eqs. (6), (8), and (9) is set using the Hoeffding bound [20]. Hoeffding bound is a statistical bound which states that after observing x observations from a random variable v having a range R, the actual mean will be at least $(\bar{v} - \epsilon)$ where $\bar{v}$ is the mean calculated from the observed x observations and this can be stated with (1-δ) confidence. The advantage of the Hoeffding bound lies in the fact that it is unaffected by the distribution generating the observations and has been highly referenced for parameter setting [13,15, 28,32]. The equation for calculating the Hoeffding bound is as given in Eq. (10)

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2x}}, \tag{10}$$

where

- δ is the margin of error and $0 < \delta \leq 1$.
- ϵ is the threshold decided by the Hoeffding bound.

In the proposed technique, entropy of a cluster represents whether the data instances within the cluster are in conformity to one another. Higher conformity leads to a lower cluster entropy, whereas lower conformity leads to a higher cluster entropy. A high cluster entropy satisfying the conditions as discussed in the section "Merge sub-module" and the section "Split sub-module" calls for restructuring of the clustering structure through merge and split operations. The parameters for the calculation of the threshold (ϵ) as given in Eq. (10) for deciding on merging or splitting of a cluster in case of the proposed technique is as follows:

- *Merge:* For merging a cluster (say C_i^l) whose parent node is (say C_i^p), the difference between the entropies of C_i^l and C_i^p for the rth and $(r-1)$th data snapshot, i.e., $\dot{e}_r^i$ and $\ddot{e}_r^i$, is considered as the random variable (v) where range (R) is as given in Eq. (11)

$$\begin{aligned} R^{'} &= \max\{\ddot{e}_{r-1}^i, \dot{e}_r^i\} \\ R &= [-R^{'}, +R^{'}]. \end{aligned} \tag{11}$$

- *Split:* For splitting a cluster (say C_i^l), the difference between the entropies of C_i^l for the $(r-1)$th and rth data snapshot, i.e., $\dot{e}_{r-1}^i$ and $\dot{e}_r^i$, is considered as the random variable (v) where range (R) is as given in Eq. (12)

$$\begin{aligned} R^{'} &= \max\{\dot{e}_{r-1}^i, \dot{e}_r^i\} \\ R &= [-R^{'}, +R^{'}]. \end{aligned} \tag{12}$$

For both the above cases, the size of the data snapshot is taken as the number of data observations (x).

Algorithm for the proposed technique

The algorithm for the proposed technique has been given under *Algorithm* 1, while the algorithm for the construction of the initial hierarchical clustering structure has been given under *Algorithm* 2.

Algorithm 1: proposed technique

In *line 2* of *Algorithm*1, the *MAIN* procedure calls the *INITIALISE* procedure in *Algorithm* 2. The *INITIALISE* procedure creates the initial hierarchical clustering structure ("Initialisation module: initialisation of the hierarchical clustering structure"). The *INITIALISE* procedure returns $tree$, $leaves$, and E_i, which are the initial hierarchical clustering structure, the leaf nodes in the $tree$ and the set containing the entropy of each node in the $tree$, respectively. Next, *lines 3–25* are repeated where newer data snapshots are processed, and necessary actions are taken to handle the concept changes. In *lines 4–5* of *Algorithm* 1, the proposed method acquires the next data snapshot (D_i) from the data streams. It places the data instances in D_i into the respective nodes of the $tree$ (refer to the section "Accommodation sub-module"). In *line 7*, the entropy is calculated for the $tree's$ nodes for D_i by calling on the *getEntropy()* method (refer to "Computing entropy of clusters") which returns E_i. In line 6, the entropy calculated for D_{i-1} is stored in E_{i-1}.

The algorithm in *line 10* merges each cluster (refer to merge case-I of the section "Merge Sub-Module") and returns a new set of leaf nodes ($leaves$) and a modified clustering structure ($tree$) on satisfying the condition specified in *line 9*. In *lines 13–18*, the clusters containing two or more data streams are tested for merging. In *line 14*, the threshold (ϵ) for merging cluster C^l is calculated by calling on the *calculateThreshold()* method (refer to the section "Threshold (ϵ)"). For the clusters with more than one data stream satisfying the condition in *line 15*, the *doMerge()* procedure is executed (refer to merge case-II of "Merge sub-module"). Similarly, the nodes in the hierarchical tree are tested for splitting. On satisfying the condition in *line 21*, the proposed method splits the clusters as detailed in the section "Split sub-module". When the hierarchical clustering structure contains only the root node as a cluster (leaf node), *split case-I* (refer to split case-I of "Split sub-module") is executed, else *split case-II* (refer to split case-II of "Split sub-module") is performed. The threshold (ϵ) is calculated using the *calculateThreshold()* procedure (refer to the section "Threshold (ϵ)"). *Lines 4–24* are repeated for the entire length of the data streams.

Algorithm 1 Proposed technique

```
1: procedure MAIN(w)
2:     tree, leaves, E_i = INITIALISE(w · d_init)
3:     while TRUE do
4:         D_i = getNextDataSnapshot(w)
5:         accommodateData(tree, D_i)
6:         E_{i-1} = E_i
7:         E_i = getEntropy(tree)
8:         for each C^l in leaves do
9:             if checkSingleStream(C^l) == TRUE then
10:                leaves, tree = doMerge(tree, leaves)
11:            end if
12:        end for
13:        for each C^l in leaves do
14:            ϵ = calculateThreshold(tree, C^l)
15:            if (E_i^{C^l} - E_i^{C^p}) ≥ ϵ then
16:                leaves, tree = doMerge(tree, leaves)
17:            end if
18:        end for
19:        for each C^l in leaves do
20:            ϵ = calculateThreshold(tree, C^l)
21:            if (E_i^{C^l} - E_{i-1}^{C^l}) ≥ ϵ then
22:                leaves, tree = doSplit(tree, leaves)
23:            end if
24:        end for
25:    end while
26: end procedure
```

In *Line 5*, the hierarchical tree is cut at a level as decided in *step-4* of the section "Initialisation module: initialisation of the hierarchical clustering structure". The *INITIALISE* procedure, in *line 6*, returns the hierarchical structure (*tree*), the leaf nodes (*leaves*) in the *tree*, and the set containing the entropy for each node in the *tree* (E_i).

Algorithm 2: initialisation of the hierarchical clustering structure

The procedure *INITIALISE* under *Algorithm 2* takes the size of the initial data snapshot to be used to construct a hierarchical clustering structure as discussed in the section "Initialisation module: initialisation of the hierarchical clustering structure". In *Algorithm 2*, *line 2* is responsible for acquiring the initial data snapshot followed by the construction of the hierarchical clustering structure in *line 3*. In *line 4*, the entropy for each node of the hierarchical tree is calculated.

Algorithm 2 Initialisation of the hierarchical clustering structure

```
1: procedure INITIALISE(size)
2:     D_i = getNextDataSnapshot(size)
3:     tree = createHierarchicalTree(D_i)
4:     E_i = getEntropy(tree)
5:     tree, leaves = cutTree(tree, E_i)
6:     return tree, leaves, E_i
7: end procedure
```

Time and space complexity

The time and space complexity for the proposed technique has been discussed in the following two sub-sections.

Time complexity

The proposed method consists of two modules: the initialisation module and the update module. The initialisation module is performed only once with a limited number of data snapshots, irrespective of the size of the data streams. Therefore,

the computational time of this step has not been included in the further analysis as it can be understood as a constant factor. The update module works on a single data snapshot at a time in an incremental manner. Thus, after processing the current data snapshot, it discards it by just keeping the summary statistics of the data snapshot. Therefore, the computation of time complexity of the update module is as follows:

- The entropy of all the nodes in the hierarchical tree is computed. In the worst-case scenario, computation for the entropy of a node is $(mw \log mw)$, where m is the number of data streams and w is the size of the data window. In a hierarchical tree, at max, there can be $(2m-1)$ number of nodes, so the computational time complexity is given as $((2m-1)\times(mw \log mw))$, i.e., $(m^2 w \log mw)$. Therefore, the term $(m^2 w \log mw)$ can be represented as a constant Φ.
- The *doMerge()* operation in *line 10* of *Algorithm 1* takes a total of m^2 computational time, and since this operation is executed for all the leaf nodes in the hierarchical clustering structure; hence, *lines 8–12* uses a computational time complexity of $((2m-1)\ m^2)$, i.e., m^3. *Lines 13–18* calculates the threshold and performs the *doMerge()* operation (*merge case-II*) for each leaf node. Both the *calculateThreshold()* method and *doMerge()* method is of the order m each. Hence, *lines 13–18* takes $((2m-1)\times m\times m)$, i.e., m^3 computational time. Similarly, *lines 19–24* which performs the split operations takes $(mw \log mw)$ computational time.

Overall, the time complexity for *lines 4–24* is $((m^2 w \log mw) + m^3 + m^3 + (mw \log mw))$. As the value of m and w is constant irrespective of the size of the data streams, the term $((m^2 w \log mw) + m^3 + m^3 + (mw \log mw))$ can be represented as a constant Φ. Let us say the length of the data stream is h, and then, the time complexity for processing it will be $((h/mw)\times\Phi)$, which can be represented as $\mathcal{O}(h)$.

Space complexity

At any point in time, the proposed technique holds the summary statistics of D_{i-1} data snapshot and the data objects of the data snapshot D_i, which amounts to the size of mw each, i.e., $2mw$. The proposed technique also maintains a representation of the hierarchical clustering structure, which takes $(2m-1)$ space. For making decisions during the merge and split operations, the entropy information is stored for the current and previous clustering structure constructed using D_i and D_{i-1}, which requires $((2m-1)+(2m-1))$ space each. Overall, the total space required by the proposed technique is $(2mw + (2m-1)+(2m-1)+(2m-1))$, i.e., $(2mw + 3(2m-1))$ or $(mw+m)$. Since the value of m and w is constant irrespective of the data streams' size, the term $(mw+m)$ can be represented as a constant ϕ. Hence, the space complexity for the proposed technique can be represented in a *big-oh* notation as $\mathcal{O}(\phi)$.

Datasets and performance measures

In this section, the synthetic and real-world datasets used for the performance analysis of the proposed technique have been presented. Furthermore, the different performance measures used to validate the proposed technique's performance have also been discussed.

Datasets

For the performance analysis of the proposed work, two synthetic datasets and one real-world dataset that represents the browsing habits of different students have been taken. The characteristics of these three datasets have been discussed next.

Synthetic stationary dataset

This dataset is stationary in nature and there is no concept evolution in it. It aims to analyse the performance of the proposed technique in such a scenario where overall data pattern is not changing over time. This dataset further comprises of four sub-datasets, as shown in Table 5, *viz.,* synthetic stationary 2C dataset, synthetic stationary 3C dataset, synthetic stationary 5C dataset, and synthetic stationary 7C dataset, where XC specifies the X number of clusters in the dataset. For generating data for a data stream falling under a specific cluster, the uniform distribution U as given in Eq. 13 can be used. In Eq. 13, the uniform distribution U is executed h times to generate a data stream (say S_i) of length h

$$S_i = h \times (U(a,b)), \tag{13}$$

where

- a, b : integers, $b \geq a$ and $n = b - a + 1$.
- n : number of discrete values.
- h : length of the data stream.

So for creating data streams belonging to different clusters, different values for the parameters a and b of U have been taken. The same value for the parameter h can be taken to generate same length data streams. These parameters values are presented in Table 4.

For generating 20 two cluster data streams, each data stream either chooses one of the following parameters as given in *rows(1–2)* of Table 4, and are placed into respective clusters: $\{(S_1, S_2, S_3, S_4, S_5, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}),$

Table 4 Parameter values for uniform distribution (U) for generating data streams belonging to different clusters

Cluster number	Parameter a	Parameter b
1	1	10
2	11	20
3	21	30
4	31	40
5	41	50
6	51	60
7	61	70

(S_6, S_7, S_8, S_9, S_{10}, S_{16}, S_{17}, S_{18}, S_{19}, S_{20})}. Similarly, for 20 three cluster data streams, *rows(1–3)* of Table 4 are used and the streams are placed accordingly: {(S_0, S_1, S_2),(S_3, S_4, S_{10}), (S_{11}, S_{12}, S_{13}, S_{14}, S_5, S_6, S_7, S_8, S_9, S_{15}, S_{16}, S_{17}, S_{18}, S_{19})}. For 20 five cluster data streams, *rows(1–5)* of Table 4 are used leading to the following configuration: {(S_0, S_1, S_2),(S_3, S_4, S_{10}), (S_{11}, S_{12}, S_{13}), (S_{14}, S_5, S_6), (S_7, S_8, S_9, S_{15}, S_{16}, S_{17}, S_{18}, S_{19})}. Finally, for 20 seven cluster data streams, *rows(1–7)* of Table 4 are used and the assignment of the data streams are as follows: {(S_0, S_1, S_2), (S_3, $_4$, S_{10}), (S_{11}, S_{12}, S_{13}), (S_{14}, S_5, S_6), (S_7, S_8, S_9), (S_{15}, S_{16}, S_{17}), (S_{18}, $_{19}$)}.

Synthetic concept evolving dataset

This dataset has been generated in such a way that new concepts keep on evolving in it and older may fade out over time. It aims to test the performance of the proposed technique under the concept evolving scenario. This dataset also comprises of 20 data streams and each data stream contains 100,000 data observations. In this dataset, after an interval of every 25,000 points, a concept evolution has been introduced as given in Table 6 and the data corresponding to each of the cluster have been generated using a uniform distribution as discussed in the Section "Synthetic stationary dataset"

For a proper representation of the effect of concept evolution on the performance of the proposed technique, synthetic concept evolving dataset has been organised into four sub-datasets, *viz.,* synthetic concept evolving H1 dataset, synthetic concept evolving H2 dataset, synthetic concept evolving H3 dataset, and synthetic concept evolving H4 dataset. The synthetic concept evolving H1 dataset comprises data instances from 1 to 25,000 from the entire synthetic concept evolving H4 dataset. Similarly, synthetic concept evolving H2 dataset and synthetic concept evolving H3 dataset consists of data instances from 1 to 50,000 and 1 to 75,000, respectively, from the entire synthetic concept evolving H4 dataset. These variations introduced in the dataset is made to reflect concept evolution which may occur in a real-world scenario. The above information is also given in Table 7.

Table 6 Synthetic concept evolving dataset

Data instances range	Number of clusters	Remark
1–25,000	3	Here, each cluster represents a concept (pattern). It can be seen that from 1 to 25,000, the number of concepts is 3 which has been increased to 5 from 25,001–50,000. Again, the number of concepts is decreased to 2 from 50,001–75,000 and further increased to 7 from 75,001–100,000
25,001–50,000	5	
50,001–75,000	2	
75,001–100,000	7	

Web browsing dataset

The real-world dataset represents the browsing behaviour of 20 students of *National Institute of Technology Meghalaya, India*. Here, the data are generated from the browsing behaviour of each student, where each student corresponds to a data stream, so it can be said that there are 20 data streams. Corresponding to each student, 10,000 observations have been recorded. Initially, from 1st to 35th data snapshot, the students were grouped into two groups. Furthermore, for collecting the concept evolution related information, students were divided into three groups from the 36th to 45th data snapshot. Each group was then asked to browse some similar websites and the data from this exercise were recorded. Then

Table 5 Synthetic stationary dataset

Dataset	Number of data streams	Size of each data stream	Number of clusters
Synthetic stationary 2C	20	100,000	2
Synthetic stationary 3C	20	100,000	3
Synthetic stationary 5C	20	100,000	5
Synthetic stationary 7C	20	100,000	7

Table 7 Minimum and maximum number of concepts in the synthetic concept evolving dataset

Dataset	Number of data instances	Number of minimum concepts	Number of maximum concepts
Synthetic concept evolving H1	25,000	3	3
Synthetic concept evolving H2	50,000	3	5
Synthetic concept evolving H3	75,000	2	5
Synthetic concept evolving H4	100,000	2	7

Table 8 Change in the number of groups of students over different data snapshots

Data snapshots from	Data snapshots to	Number of groups
1	35	2
36	45	3
46	65	4
66	82	5
83	100	4

again, students were divided into four groups, five groups, and again into four groups for further data snapshots, as shown in Table 8.

The main reason for considering this grouping of students during collection of data is to collect the labelled data in terms of the number of concepts (clusters) that has been prevailing over a set of data snapshots. This helps in unbiased comparison of the proposed technique with AGNES.

Performance measures

The performance measures discussed below have been used to validate the proposed technique against AGNES.

Dunn index [24]: This index is based on the inter-cluster distance and intra-cluster distance of clusters. It is a ratio of the minimum distance between clusters to the maximum intra-cluster distance, as shown in Eq. (14). A high value of this index indicates good clustering, where the clusters formed are compact and well separated

$$\begin{aligned} \mathrm{DI} &= \frac{\min_{1\le i\le j\le |K|}\Delta_{\mathrm{inter}}(C_i, C_j)}{\max_{1\le k\le |K|}\Delta_{\mathrm{intra}}(C_k)} \\ \Delta_{\mathrm{intra}}(C_k) &= \sum_{o\in C_k}\mathrm{cosine}(o, \bar{C_k}) \\ \Delta_{\mathrm{inter}}(C_i, C_j) &= \frac{1}{\mid C_i \mid\mid C_j \mid}\sum_{o_i\in C_i}\sum_{o_j\in C_j}\mathrm{cosine}(o_i, o_j). \end{aligned} \tag{14}$$

In Eq. (14), $\Delta_{\mathrm{intra}}(C_k)$ represents the inter-cluster distance of the kth cluster and $\bar{C_k}$ represents the centroid of C_k. $\Delta_{\mathrm{inter}}(C_i, C_j)$ represents the inter-cluster distance between two clusters C_i and C_j, respectively.

Modified Hubert's Γ

Statistic [24]: This statistic is based on the proximity between the data objects in a dataset and the proximity between the cluster centres. This metric is used to describe the extent to which the clusters formed fit the data. It is calculated as given in Eq. (15) and a high value of this statistic indicates good clustering

$$\mathrm{MH}\Gamma = \frac{1}{m}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}(\mathrm{cosine}(S_i, S_j))(\Delta_{\mathrm{inter}}(C_i^l, C_j^l)). \tag{15}$$

Cophenetic correlation coefficient [14]: This metric is used to measure the correlation coefficient between a distance matrix obtained from the original data points against the distance matrix obtained after modelling the same original data points into a dendrogram-based hierarchical structure. High values for this metric, as shown in Eq. (16), suggest well-formed clusters

$$\begin{aligned} \mathrm{CPCC} &= \frac{\mathrm{numr}}{\mathrm{deno}}, \\ numr &= \sum_{i<j}(\mathrm{cosine}(S_i, S_j) - \mu_P) \\ &\quad \sum_{i<j}(\Delta_{\mathrm{den}}(C_i, C_j) - \mu_{\mathrm{den}}) \\ \mathrm{deno} &= \sqrt{d_1 * d_2} \\ d_1 &= [\sum_{i<j}(\mathrm{cosine}(S_i, S_j) - \mu_P)]^2 \\ d_2 &= [\sum_{i<j}(\Delta_{\mathrm{den}}(C_i, C_j) - \mu_{\mathrm{den}})]^2. \end{aligned} \tag{16}$$

In Eq. (16), $\Delta_{\mathrm{prox}}(S_i, S_j)$ represents the distance between the ith and jth data stream S_i and S_j, respectively. μ_P represents the average pairwise distance between m data streams and μ_{den} represents the average pairwise inter-cluster distance between K clusters.

Purity [31]: It represents the degree to which the clusters formed contain data objects from a single class. The more each cluster contains data objects from a single class, the higher is the purity value. Equation (17) shows the process of purity calculation where L is the set of class labels

$$\text{Purity} = \frac{1}{m}\sum_{i}^{|K|}\max(C_i^l \cap L). \tag{17}$$

Experimental results and analysis

The performance of the proposed technique has been analysed and compared to AGNES technique on synthetic as well as real-world datasets in terms of performance measures which have been presented in the section "Performance measures". AGNES is a hierarchical clustering technique, which has also been used for creating the initial hierarchical structure in the case of the proposed technique. The proposed technique also follows the hierarchical clustering by variable approach; hence, AGNES has been preferred over other traditional clustering techniques for the performance comparison. Indeed, AGNES cannot deal with data streams, but it has been used as a baseline technique for analysing the performance of the proposed technique. The use of traditional clustering techniques as a baseline technique has also been done in the literature related to the data streams [3,22,28,33]. Most of the existing data stream clustering technique follows the clustering by example approach; hence, comparing the proposed method, which follows a clustering by variable approach to a method following a clustering by example approach, is incompatible. Moreover, most of the work has tailored their processing in the clustering by variable domain depending on the similarity or dissimilarity measure. Hence, this makes it hard to apply the computation of dissimilarity between data streams proposed in the current work in the setting of existing works.

AGNES technique requires the availability of the entire dataset at one go for processing, so this condition has been maintained even for obtaining the results for comparison. As opposed to this, the proposed technique processes the data in terms of data chunks which have been obtained from data streams using a sliding window of fixed size. Different sliding window sizes have been tried for experimental analysis and the results corresponding to a window size of $w = 100$ have been presented in this section. For the initialisation of the clustering structure, D_{init} has been set to 20 which means that initial 20 windows have been used. Furthermore, margin of error (δ) has been set to 0.05 for the calculation of the threshold (ϵ).

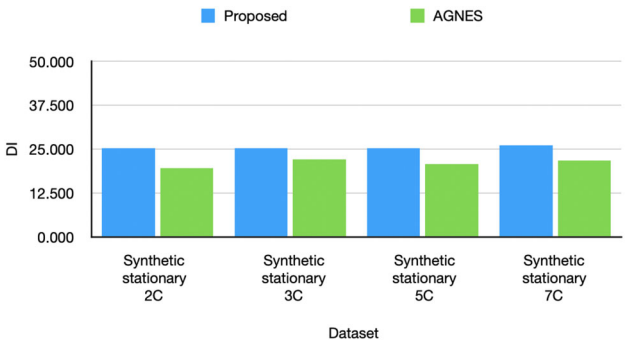

Fig. 8 DI score of the proposed technique and AGNES on four synthetic stationary datasets

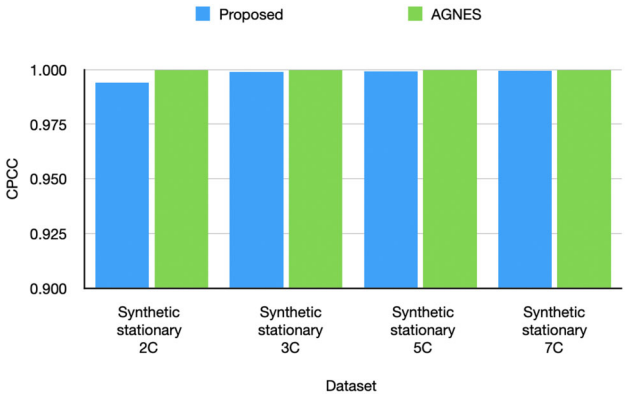

Fig. 9 CPCC score of the proposed technique and AGNES on four synthetic stationary datasets

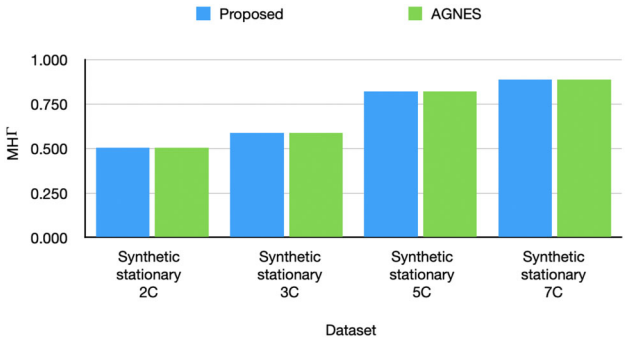

Fig. 10 MHΓ score of the proposed technique and AGNES on four synthetic stationary datasets

Results for synthetic stationary datasets

The performance of the proposed technique and AGNES on synthetic stationary 2C dataset, synthetic stationary 3C dataset, synthetic stationary 5C dataset, and synthetic stationary 7C dataset in terms of DI, CPCC, MHΓ, and Purity is given in Figs. 8, 9, 10, and 11, respectively.

It can be observed from Fig. 8 that the average of the DI scores obtained by the proposed technique on every window is comparatively better in comparison to the average DI

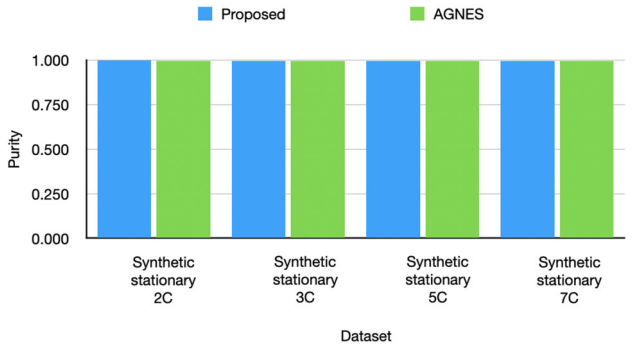

Fig. 11 Purity score of the proposed technique and AGNES on four synthetic stationary datasets

score obtained by AGNES on all the four synthetic stationary datasets, viz., synthetic stationary 2C dataset, synthetic stationary 3C dataset, synthetic stationary 5C dataset, and synthetic stationary 7C dataset. The average DI score for AGNES is calculated by dividing the score achieved by AGNES by the same number of windows processed by the proposed technique. The comparatively better performance of the proposed technique on all the four synthetic stationary datasets can be attributed to the fact that the hierarchical clustering structure which is incrementally modelled after every data snapshot can reflect the local changes in the data streams better than AGNES.

Furthermore, from Fig. 9, it can be seen that the proposed technique performs almost identical to AGNES in terms of the CPCC values obtained on all the four synthetic stationary datasets. However, in Figs. 10 and 11, the performance of the proposed technique and AGNES is exactly the same on all the four synthetic stationary datasets.

Overall, it can be said that for a non-concept evolving data streams, the performance of both the techniques, *viz.*, proposed technique and AGNES, is approximately the same, but the main advantage of the proposed technique is that it has performed comparatively to AGNES while processing data in the form of windows instead of asking for availability of the entire data, which proves its advantage over AGNES in terms of memory requirements.

Results on synthetic concept evolving datasets

The proposed technique has been evaluated and compared to AGNES on four synthetic concept evolving datasets, namely, synthetic concept evolving H1 dataset, synthetic concept evolving H2 dataset, synthetic concept evolving H3 dataset, and synthetic concept evolving H4 dataset as described in Tables 6 and 7 to analyse their performance in presence of multiple concepts in the data streams.

The movement of the different data streams from one cluster to another cluster on evolution of new concepts in the data streams is shown in Fig. 12 for the synthetic concept evolving H4 dataset. Altogether there has been three changes in the concepts in the synthetic concept evolving H4 dataset as detailed in the section "Synthetic concept evolving dataset". Furthermore, the different operations performed by the proposed technique to handle the evolving concepts in the data streams are shown in Fig. 13a–c.

Initially, the first 20 windows (*windows 1–20*) were used by the proposed technique for creating the initial clustering structure as it has been shown in light colours (light blue, light green, and light red) in Fig. 12a. There are three clusters at the time of initialisation where *cluster 1* comprises of $(S_1 - S_4)$ data streams, *cluster 2* comprises of $(S_5 - S_8)$ data streams, and *cluster 3* comprises of $(S_9 - S_{20})$ data streams, and the corresponding tree structure is also depicted in Fig. 12b. After the initialisation, from *window 21–250*, the initial assignment of the data streams into three clusters does not change as no concept evolution occurs in *windows 21–250*, as shown in Fig. 12a, and the same thing is also shown in Fig. 12c. However, on the 251st window, a concept change occurs and the clustering structure changes from three cluster to five cluster due to the split operation, as shown in Fig. 12d and e. Hence, some data streams, i.e., $(S_{15} - S_{18})$ and $(S_{19} - S_{20})$ in *cluster 3* are assigned into two new clusters, viz., *cluster 4* and *cluster 5*, respectively. Therefore, after the split operation on 251st window *cluster 1*, *cluster 2*, *cluster 3*, *cluster 4* and *cluster 5* comprises of $(S_1 - S_4)$, $(S_5 - S_8)$, $(S_9 - S_{14})$, $(S_{15} - S_{18})$ and $(S_{19} - S_{20})$ data streams, respectively. This assignment stays the same until 500th window. Next, concept evolution happens in 501st and 502nd windows and the clustering structure changes from five cluster to two cluster. The different operations for changing the clustering structure from five cluster to two cluster are shown in Fig. 12f where *node* n_3 containing six data streams is split into two clusters named as *cluster 3* and *cluster 6*, as shown in Fig. 12a and f, each containing three data streams. In Fig. 12g, *node* n_5 containing four data streams is splitted into two new clusters named *cluster 4* and *cluster 7*, as depicted in Fig. 12g, containing one data stream and three data streams, respectively. In Fig. 12h, the process of merging of *cluster 4* and *cluster 7* takes place after which the clustering structure contains six clusters in total, as shown in Fig. 12i. This process of merging continues where in Fig. 12j, *cluster 4* and *cluster 5* merges followed by the merging of *cluster 3*, *cluster 6*, and *cluster 4*, as illustrated in Fig. 12k. Again, all three clusters, i.e., *cluster 1*, *cluster 2*, and *cluster 3*, merge into its parent node(*denoted as* n_1) as shown in Fig. 12l to produce a single cluster as depicted in Fig. 12m. Finally, the parent node (n_1) is split into two clusters: *cluster 1* and *cluster 2*, as shown in Fig. 12n containing $(S_1 - S_4, S_{12} - S_{15})$ and $(S_5 - S_{11}, S_{16} - S_{20})$ data streams, respectively. The same clustering structure in Fig. 12n is maintained till the 750th window until a concept evolution is encountered at the 751st win-

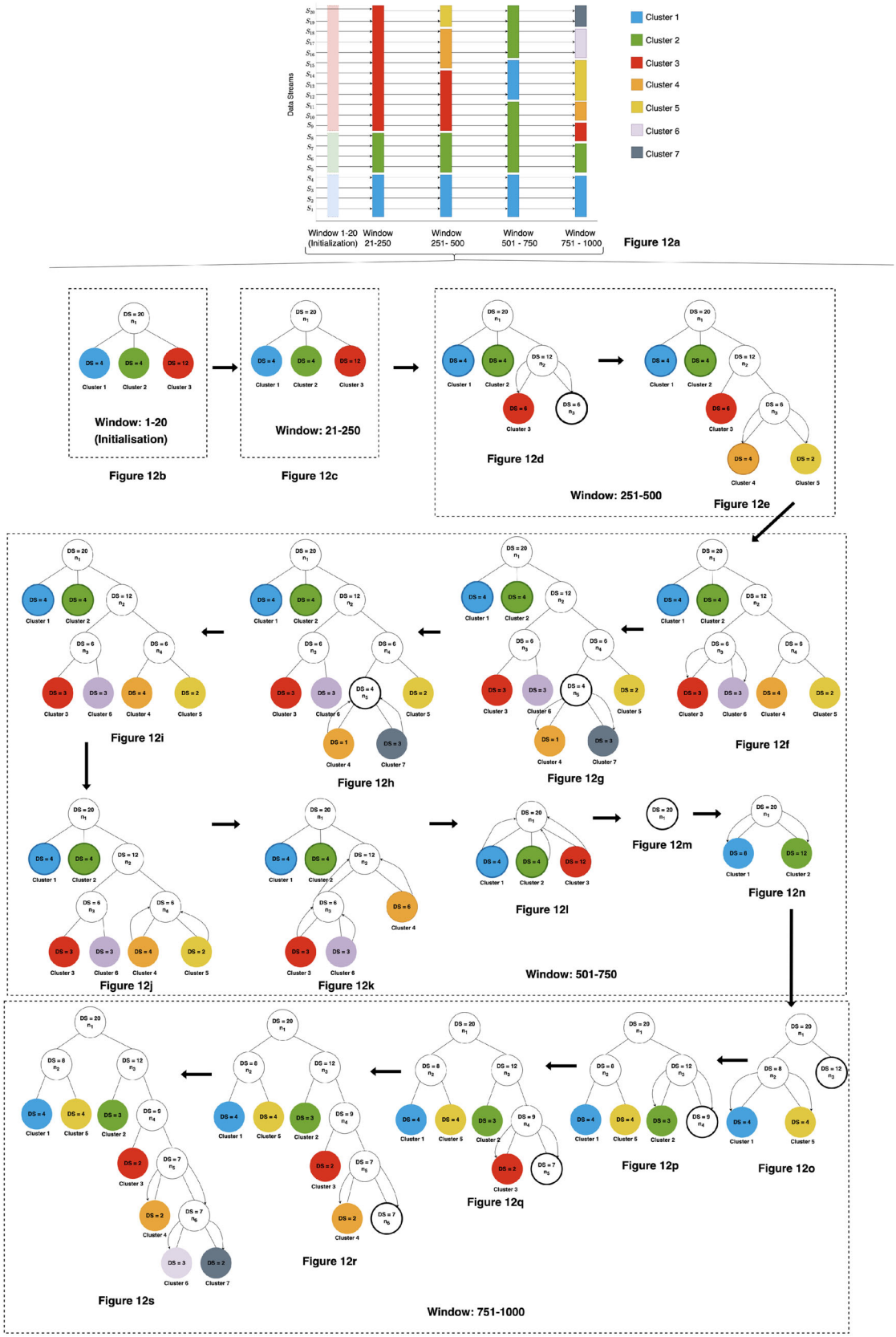

Fig. 12 The movement of data streams from one cluster to another on evolution of concepts

dow. Due to the concept evolution at the 751st window, *node* n_2 is split into two clusters, as shown in Fig. 12o, followed by the splitting of node n_3 into another two clusters, as illustrated in Fig. 12p. Again, node n_4 is splitted into two clusters, as illustrated in Fig. 12q, followed by the splitting of node n_5 into yet another two clusters, as illustrated in Fig. 12r. Finally, node n_6 is splitted into two clusters, as depicted in Fig. 12s, where *cluster 1*, *cluster 2*, *cluster 3*, *cluster 4*, *cluster 5*, *cluster 6* and *cluster 7* contains $(S_1 - S_4)$, $(S_5 - S_7)$, $(S_8 - S_9)$, $(S_{10} - S_{11})$, $(S_{12} - S_{15})$, $(S_{16} - S_{18})$, and $(S_{19} - S_{20})$ data streams, respectively. The clustering structure obtained after the split operations on the 751th window remains unchanged till the last window (i.e., 1000th window). Overall, it can be observed that different concepts in the synthetic concept evolving H4 dataset have been accurately captured in the clustering structure by the proposed technique.

The change in the average entropy of clustering structure (tree) is shown in Fig. 13a–c, respectively, for first concept evolution at 251st window, second concept evolution at 501st window, and third concept evolution at 751st window corresponding to the synthetic concept evolving H4 dataset. It can be observed from Fig. 13a that the average entropy of clustering structure at the time of 250th window is 3.314, but on accommodating the data instances of 251st window, the average entropy increases to 3.797 which represents an unstable clustering structure. This increase in entropy is because of the new concepts arriving in the data streams due to the 251st window. To capture the new concepts in the clustering structure, different split operations are performed by the proposed technique which decreases the average entropy of the clustering structure (tree) to 3.305 (the different split operations performed are shown in Fig. 12d and e). In Fig. 13b, the average entropy of clustering structure at the time of 500th window is 3.306. However, after accommodating the newer concepts which are spread over 501th and 502th window, the average entropy of the clustering structure increases to 3.663 representing an unstable clustering structure. Therefore, to stabilise the clustering structure, multiple split and merge operations on the data instances corresponding to 501th and 502th window are executed which lowers the average entropy of the clustering structure to 3.299 (the different split and merge operations performed are shown in Fig. 12f–n). The third and final concept evolution occurs at 751st window where the average entropy of the clustering structure increases from 3.314 on 700th window to 4.945 on accommodating the 751st window, as shown in Fig. 13c. However, after the split operations performed by the proposed technique on the data instances of 751st window, the average entropy of the clustering structure reduces to 3.219 (the different split operations performed are shown in Fig. 12o–s).

The DI score, CPCC score, MHΓ score, and Purity score achieved by the proposed technique are shown in Figs 14, 15, 16, and 17, respectively. The DI score has been plotted

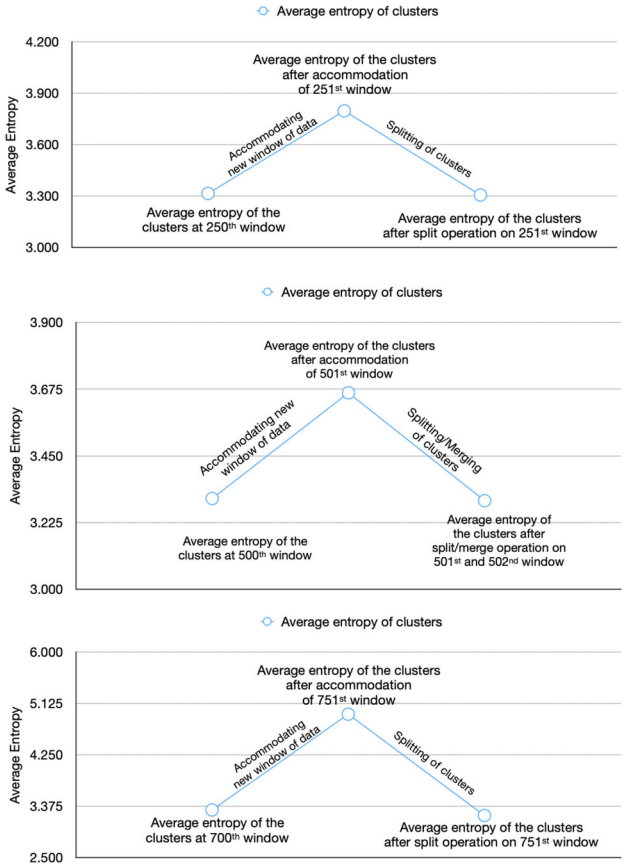

Fig. 13 Change in the average entropy of clustering structure corresponding to different concept evolution

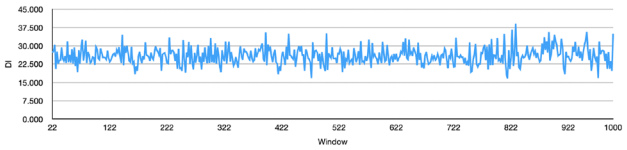

Fig. 14 DI score of the proposed technique

with an interval of 2 windows; similarly, an interval of two windows has been taken for plotting the CPCC, MHΓ and Purity scores. It can be observed from Fig. 14 that some fluctuations in the DI score are due to the changes in the values of the data instances of different data windows. Whereas, the changes in CPCC and MHΓ are only corresponding to those windows where changes in concepts occur and Purity score is 1 corresponding to all the windows. In totality, it can be said based on the DI, CPCC, MHΓ, and Purity scores, as shown in Figs. 14, 15, 16, and 17 that the performance of the proposed technique is very promising for concept evolving datasets.

The comparative analysis of the proposed technique and AGNES on synthetic concept evolving H1, synthetic concept evolving H2, synthetic concept evolving H3 and synthetic concept evolving H4 datasets in terms of DI, CPCC, MHΓ,

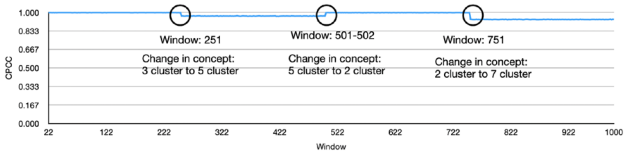

Fig. 15 CPCC score of the proposed technique

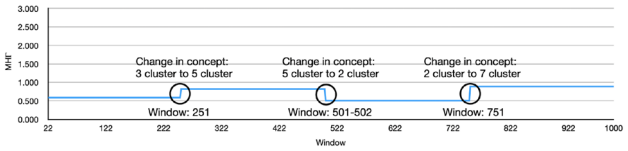

Fig. 16 MHΓ score of the proposed technique

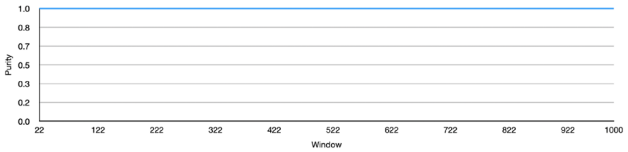

Fig. 17 Purity score of the proposed technique

and Purity is shown, respectively, in Figs. 18, 19, 20, and 21. It can be observed from Fig. 18 that the average of the DI scores obtained by the proposed technique on every window and the average DI score achieved by AGNES is nearly the same for synthetic concept evolving H1 dataset due to the fact that synthetic concept evolving H1 dataset contains data instances from a single concept. However, the performance of the proposed technique is far better compared to AGNES on synthetic concept evolving H2, synthetic concept evolving H3, and synthetic concept evolving H4 datasets as these three datasets consist of data instances belonging to multiple concepts which are correctly captured by the proposed technique but not by AGNES. Similarly, it can be observed from Figs. 19 and 20 that the performance of the proposed technique and AGNES is roughly the same for synthetic concept evolving H1 dataset, but the performance of the proposed technique is better than AGNES for synthetic concept evolving H2, synthetic concept evolving H3, and synthetic concept evolving H4 datasets. This type of behaviour of the proposed technique and AGNES in terms of CPCC and MHΓ is again accounted to the capability of the proposed technique to address the change in concepts by adjusting the clustering structure via different split and merge operations, whereas AGNES is unable to handle it. However, it can be seen from Fig. 21 that the value of Purity is 1 for both the proposed technique and AGNES. However, the Purity value for AGNES is misleading as it is shown in Fig. 22 that a very large number of clusters have been generated by AGNES as compared to the actual number of clusters. Whereas, in the case of proposed technique, the number of clusters generated and the actual number of clusters is the same which proves that the proposed

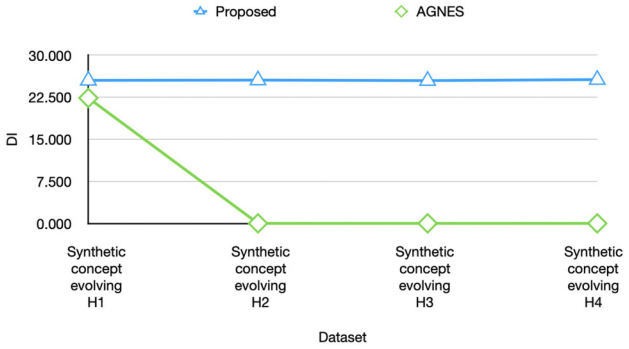

Fig. 18 DI score of the proposed technique and AGNES on four synthetic concept evolving datasets

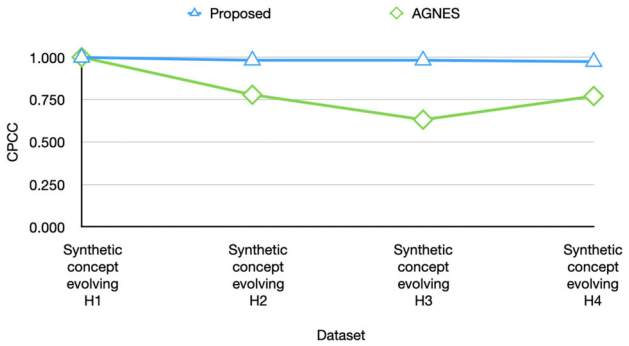

Fig. 19 CPCC score of the proposed technique and AGNES on four synthetic concept evolving datasets

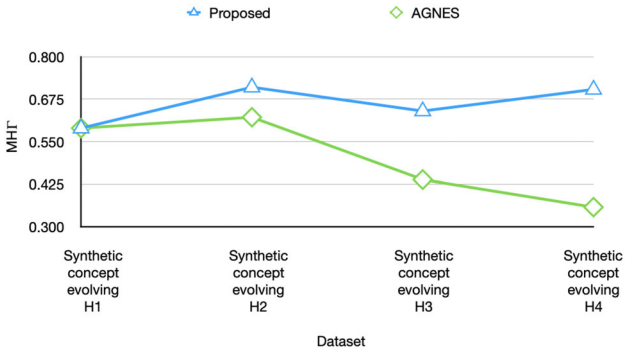

Fig. 20 MH score of the proposed technique and AGNES on four synthetic concept evolving datasets

technique accurately captures the concept evolution in the clustering structure. Overall, the proposed technique's superior performance indicates the proposed technique's ability to identify and reflect the concept changes in the data streams through proper updation of the clustering structure. However, the same cannot be said about AGNES, as it fails in recognising the evolving changes in the data streams.

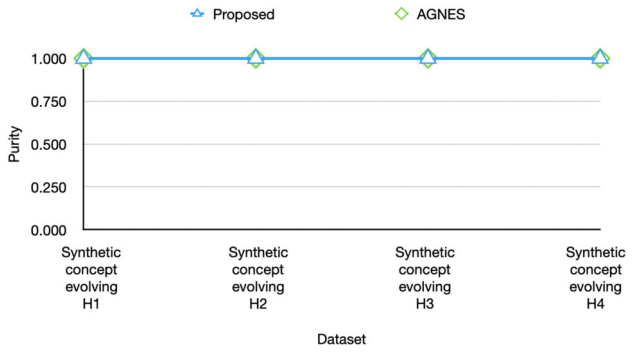

Fig. 21 Purity score of the proposed technique and AGNES on four synthetic concept evolving datasets

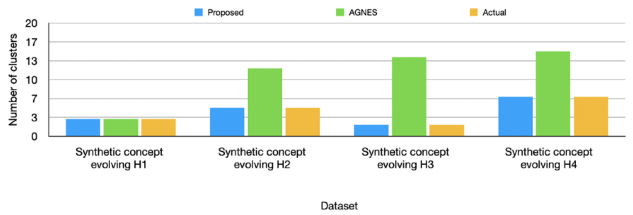

Fig. 22 Number of clusters generated by the proposed technique and AGNES vs. actual number of clusters for synthetic concept evolving datasets

Performance on web browsing dataset

The proposed technique has also been evaluated and compared to AGNES on web browsing dataset as described in Table8 to analyse their performance in a real-world scenario.

The movement of the different data streams from one cluster to another cluster on evolution of new concepts in the data streams is shown in Fig. 23 for the web browsing dataset. Overall, there has been four changes in the concepts in the web browsing dataset as detailed in the section "Web browsing dataset". Initially, the first 20 windows (*windows 1–20*) were used by the proposed technique for creating the initial clustering structure as it has been shown in light colours (light blue, light green, and light red) in Fig. 23a. There are two clusters at the time of initialisation where *cluster 1* comprises of $(S_1 - S_{10})$ data streams and *cluster 2* comprises of $(S_{11} - S_{20})$ data streams, and the corresponding tree structure is also depicted in Fig. 23b. After the initialisation, from *window 21–35*, the initial assignment of the data streams into two clusters does not change as no concept evolution occurs in *windows 21–35*, as shown in Fig. 23a, and the same thing is also shown in Fig. 23c. However, on the 36th window, a concept change occurs and the clustering structure changes from two cluster to three cluster due to the split operation, as shown in Fig. 23d. Hence, the data streams $(S_1 - S_{10})$ in *node* n_2 are assigned to two new clusters, i.e., the data streams $(S_1 - S_5)$ and $(S_6 - S_{10})$ are assigned to *cluster 1* and *cluster 3*, respectively. Therefore, after the split operation on 36st window, *cluster 1*, *cluster 2*, and *cluster 3* comprises $(S_1 - S_5)$, $(S_{11} - S_{20})$, and $(S_6 - S_{10})$ data streams, respectively. This assignment stays the same till the 45th window. Next, concept evolution happens in 46th window and the clustering structure changes from three cluster to four cluster. The split operation for changing the clustering structure from three cluster to four cluster is shown in Fig. 23 where *node* n_3 containing ten data streams is split into two clusters named as *cluster 2* and *cluster 4* containing three and seven data streams, respectively, as shown in Fig. 23a and e. This new assignment of the data streams also remains the same till the 65th window. In Fig. 23f, *node* n_4 containing seven data streams is split into two new clusters, viz., *cluster 4* and *cluster 5*, as depicted in Fig. 23f, containing three and four data streams, respectively, on processing the 66th window where another concept change occurs. The clustering structure so obtained after the split operation stays unchanged till the 82nd window where *cluster 1*, *cluster 2*, *cluster 3*, *cluster 4*, and *cluster 5* contains $(S_1 - S_5)$, $(S_{11} - S_{13})$, $(S_6 - S_{10})$, $(S_{14} - S_{16})$, and $(S_{17} - S_{20})$ data streams, respectively. Finally, on encountering the 83^{rd} window where yet again another concept change was detected, *cluster 4* and *cluster 5* were merged into its parent cluster (*node* n_4 as shown in Fig. 23f) by the proposed technique to produce a clustering structure accommodating four clusters, namely, *cluster 1*, *cluster 2*, *cluster 3*, and *cluster 4* containing $(S_1 - S_5)$, $(S_{11} - S_{13})$, $(S_6 - S_{10})$, and $(S_{14} - S_{20})$ data streams, respectively. The clustering structure obtained after the merge operation on the 83rd window remains unchanged till the last window (i.e., 100th window) and can be observed from Fig. 23a and g. Overall, it can be ascertained that different concepts in the web browsing dataset have been accurately captured in the clustering structure by the proposed technique.

The DI score, CPCC score, MHΓ score, and Purity score achieved by the proposed technique are shown in Figs. 24, 25, 26, and 27, respectively. The DI score has been plotted with an interval of 2 windows; similarly, an interval of two windows has been taken for plotting the CPCC, MHΓ, and Purity scores. It can be observed from Fig. 24 that some fluctuations in the DI score are due to the changes in the values of the data instances of different data windows. On examining the CPCC score from Fig. 25, it can be seen that the changes occurring in the CPCC score correspond to the concept changes in the 46th and 83rd windows, while the changes corresponding to the concept changes in the 36th and 66th windows are negligible. Whereas, the changes in MHΓ are only corresponding to those windows where changes in concepts occur and Purity score is 1 corresponding to all the windows. Hence, it can be said based on the DI, CPCC, MHΓ, and Purity scores as shown in Figs. 24, 25, 26, and 27 that for a dataset representing a real-world scenario promising performance by the proposed technique can be observed.

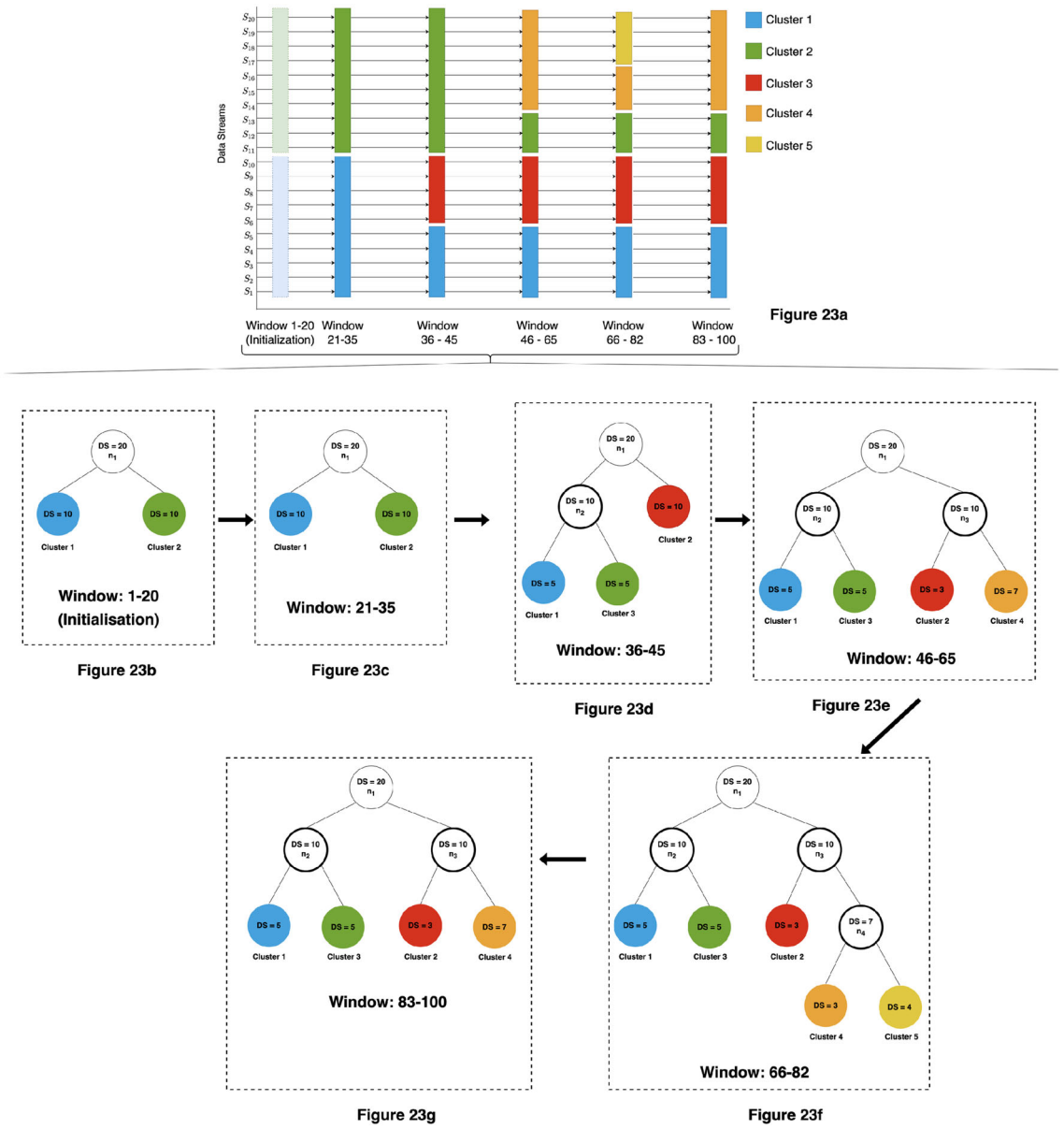

Fig. 23 The movement of data streams from one cluster to another on evolution of concepts

The comparative analysis of the proposed technique and AGNES on web browsing dataset in terms of DI, CPCC, MHΓ, and Purity has been done in fractions of the web browsing dataset, i.e., for *windows 21–35*, *windows 21–45*, *windows 21–65*, *windows 21–82*, and *windows 21–100* containing concepts in the range one to five as discussed in the section "Web browsing dataset" to observe the effect of multiple concepts on both the techniques as shown, respectively, in Figs. 28, 29, 30, and 31.

It can be observed from Fig. 28 that the average of the DI scores obtained by the proposed technique on every window and the average DI score achieved by AGNES is nearly the same for *windows 21–35* due to the fact that *windows 21–35* contain data instances from a single concept. However, the performance of the proposed technique is far better compared to AGNES on *windows 21–45*, *windows 21–65*, *windows 21–82*, and *windows 21–100* as these windows consist of data instances belonging to multiple concepts which are correctly captured by the proposed technique but not by AGNES.

Furthermore, it can be observed from Fig. 29 that the performance of the proposed technique and AGNES is roughly the same for *windows 21–35* and *windows 21–45*. However, the performance of AGNES drops in comparison to the proposed technique as the number of concepts starts increasing further in *windows 21–65*, *windows 21–82*, and *windows 21–100*. Similarly, in case of MHΓ, the perfor-

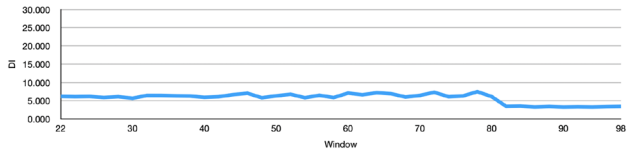

Fig. 24 DI score of the proposed technique

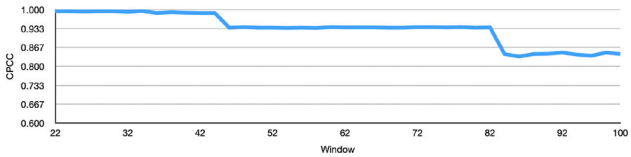

Fig. 25 CPCC score of the proposed technique

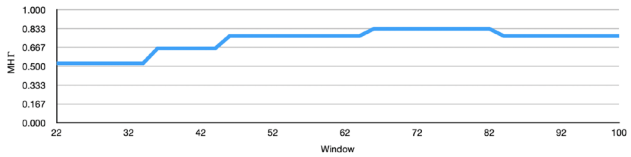

Fig. 26 MHΓ score of the proposed technique

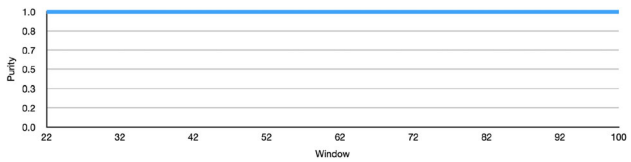

Fig. 27 Purity score of the proposed technique

mance of the proposed technique and AGNES is roughly the same for *windows 21–35*, but the performance of AGNES starts decreasing in comparison to the proposed technique for *windows 21–45*, *windows 21–65*, *windows 21–82*, and *windows 21–100* as can be seen from Fig. 30. Again, this type of behaviour of the proposed technique and AGNES in terms of CPCC and MHΓ can be accounted to the capability of the proposed technique to address the changes in concepts by adjusting the clustering structure via different split and merge operations, whereas AGNES falls short in such capabilities of handling it. However, it can be seen from Fig. 31 that the value of Purity is 1 for both the proposed technique and AGNES. However, the Purity value for AGNES is misleading as it is shown in Fig. 32 that a very large number of clusters have been generated by AGNES as compared to the actual number of clusters. Whereas, in case of the proposed technique, the number of clusters generated and the actual number of clusters are the same which proves that the proposed technique accurately captures the concept evolution in the clustering structure. Overall, the performance of the proposed technique validated using the validity indices indicates that it can correctly group the students into an optimal number of clusters. Moreover, the clusters produced by the proposed technique are compact and well separated, as

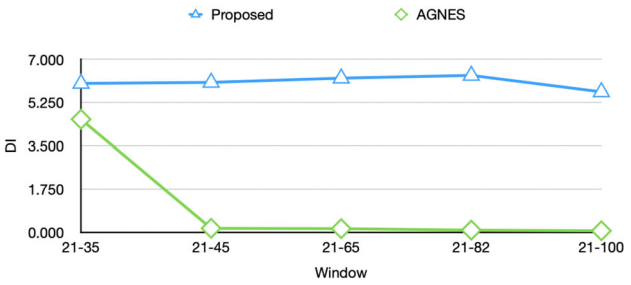

Fig. 28 DI score of the proposed technique and AGNES on web browsing dataset

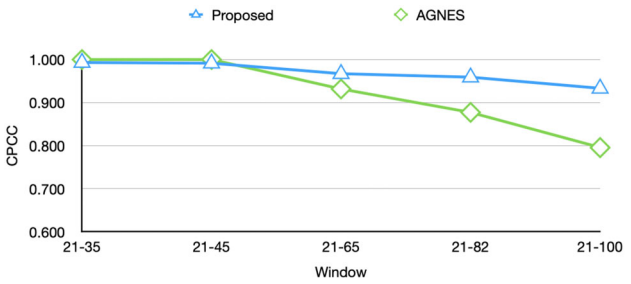

Fig. 29 CPCC score of the proposed technique and AGNES on web browsing dataset

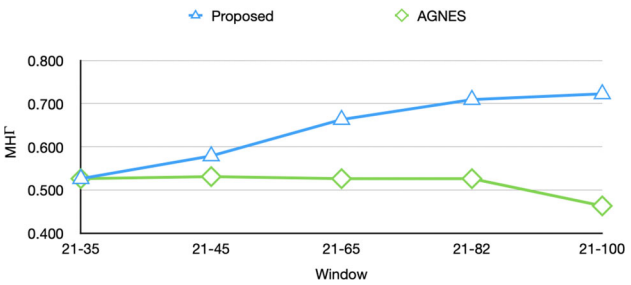

Fig. 30 MHΓ score of the proposed technique and AGNES on web browsing dataset

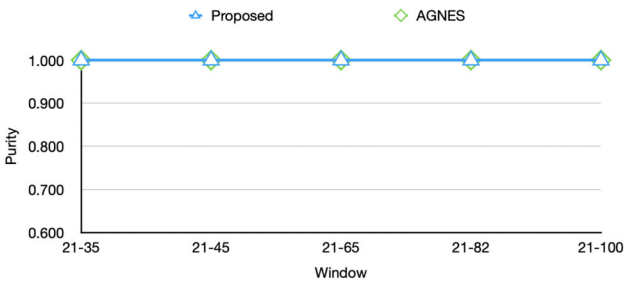

Fig. 31 Purity score of the proposed technique and AGNES on web browsing dataset

suggested by the scores obtained. However, the same cannot be implied in the case of AGNES as it could not handle the concept changes in the data streams. Consequently, AGNES proceeded with a non-optimal assignment of students into clusters.

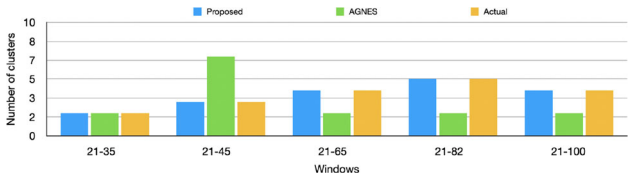

Fig. 32 Number of clusters generated by the proposed technique and AGNES vs. actual number of clusters for web browsing dataset

Time comparison

The proposed technique has also been compared to AGNES based on the computation time required by both the techniques for processing the synthetic stationary datasets, synthetic concept evolving datasets and web browsing dataset, as shown in Figs. 33, 34, and 35, respectively.

From Fig. 33, it is clearly evident that the processing time required by the proposed technique on all the synthetic stationary datasets, namely, synthetic stationary 2C dataset, synthetic stationary 3C dataset, synthetic stationary 5C dataset, and synthetic stationary 7C dataset, is significantly less as compared to the processing time required by AGNES for processing the same datasets. Again, from Fig. 34, it can observed that AGNES requires more processing time compared to the proposed technique as the number of data instances increases on every consecutive synthetic concept evolving datasets (i.e., number of data instances in synthetic concept evolving H1 dataset $<$ number of data instances in synthetic concept evolving H2 dataset $<$ number of data instances in synthetic concept evolving H3 dataset $<$ number of data instances in synthetic concept evolving H4 dataset). Similarly, from Fig. 35, in the case of web browsing dataset as the number of windows (i.e., number of data instances) increases, so does the processing time required by AGNES which is very significant in comparison to the processing time required by the proposed technique. Hence, from Figs. 33, 34, and 35, it can be said that the proposed technique requires significantly less computation time compared to AGNES which satisfies the time constraint in processing the data streams.

Conclusion

In the present paper, a hierarchical clustering technique for multiple nominal data streams has been presented. It measures the quality of a cluster by calculating its entropy. The proposed technique is capable of addressing the concept evolving nature of the data streams by adapting the clustering structure with the help of merge and split operations. The experimental analysis has been performed on two synthetic and a real application dataset, namely synthetic stationary datasets, synthetic concept evolving datasets, and

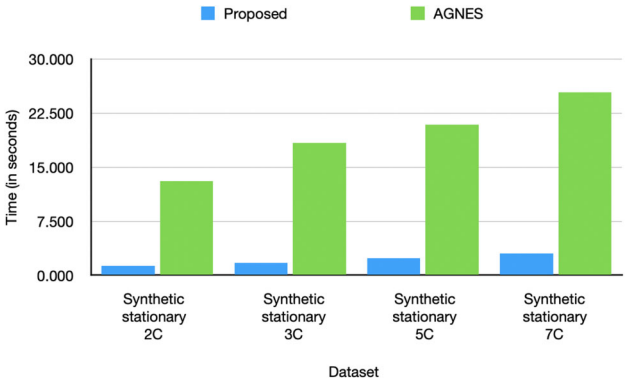

Fig. 33 Time taken by the proposed technique and AGNES on synthetic stationary datasets

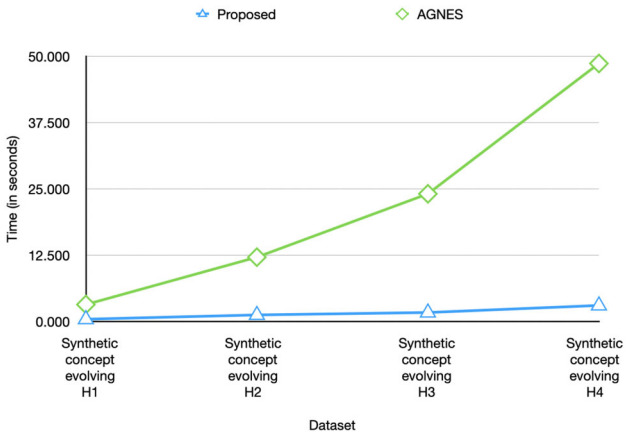

Fig. 34 Time taken by the proposed technique and AGNES on synthetic concept evolving datasets

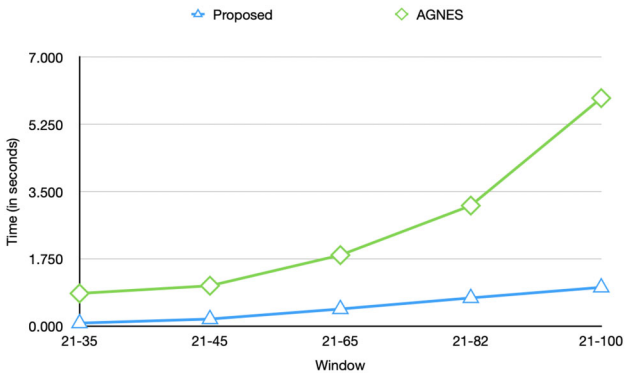

Fig. 35 Time taken by the proposed technique and AGNES on web browsing dataset

web browsing dataset in terms of performance measures, viz., DI, CPCC, MHΓ, and Purity. The proposed technique has achieved the highest DI score of 25.642, CPCC score of 0.974, MHΓ score of 0.704, and Purity score of 1 for synthetic concept evolving H4 dataset. For the web browsing dataset, the DI score of 5.678, CPCC score of 0.993, MHΓ score of 0.723, and Purity score of 1 has been attained by

the proposed technique. Overall, it can be concluded from the experimental results that the proposed technique has performed approximately the same on synthetic stationary datasets. However, the proposed technique has outperformed the AGNES technique on synthetic concept evolving datasets as well as web browsing dataset. Furthermore, it can be stated that the proposed technique requires much less computation time in comparison to AGNES.

Acknowledgements We would like to thank *National Institute of Technology Meghalaya, India* for providing access to their student web browsing dataset.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Ackermann MR, Märtens M, Raupach C, Swierkot K, Lammersen C, Sohler C (2012) Streamkm++ a clustering algorithm for data streams. J Exp Algorithmics (JEA) 17:1–2
2. Aggarwal CC, Han J, Wang J, Yu PS (2004) A framework for projected clustering of high dimensional data streams. Proceedings of the Thirtieth international conference on Very large data bases-Volume 30:852–863
3. Aggarwal CC, Philip SY, Han J, Wang J (2003) A framework for clustering evolving data streams. In: Proceedings 2003 VLDB conference, Elsevier, pp 81–92
4. Aggarwal CC, Philip SY (2010) On clustering massive text and categorical data streams. Knowl Inf Syst 24(2):171–196
5. Balzanella A, Lechevallier Y, Verde R (2011) Clustering multiple data streams. In: New perspectives in statistical modeling and data analysis, Springer, pp 247–254
6. Bones CC, Romani LA, de Sousa EP (2016) Improving multivariate data streams clustering. In: Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE), Procedia Computer Science, pp 461–471
7. Cao F, Estert M, Qian W, Zhou A (2006) Density-based clustering over an evolving data stream with noise. In: Proceedings of the 2006 SIAM international conference on data mining, SIAM, pp 328–339
8. Chen HL, Chen MS, Lin SC (2008) Catching the trend: a framework for clustering concept-drifting categorical data. IEEE Trans Knowl Data Eng 21(5):652–665
9. Chen L, Zou LJ, Tu L (2012) A clustering algorithm for multiple data streams based on spectral component similarity. Inf Sci 183(1):35–47
10. Chen Y, Tu L (2007) Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 133–142
11. Dai BR, Huang JW, Yeh MY, Chen MS (2004) Clustering on demand for multiple data streams. In: Fourth IEEE International Conference on Data Mining (ICDM'04), IEEE, pp 367–370
12. Diday E (1971) Une nouvelle méthode en classification automatique et reconnaissance des formes la méthode des nuées dynamiques. Revue de statistique appliquée 19(2):19–33
13. Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 71–80
14. Farris JS (1969) On the cophenetic correlation coefficient. Syst Zool 18(3):279–285
15. Gama J, Medas P, Rocha R (2004) Forest trees for on-line data. In: Proceedings of the 2004 ACM symposium on Applied computing, pp 632–636
16. Guha S, Meyerson A, Mishra N, Motwani R, O'Callaghan L (2003) Clustering data streams: theory and practice. IEEE Trans Knowl Data Eng 15(3):515–528
17. Guha S, Mishra N, Motwani R, et al. (2000) Clustering data streams. In: focs, IEEE, p 359
18. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier, Amsterdam
19. Hassani M, Spaus P, Seidl T (2014) Adaptive multiple-resolution stream clustering. In: International Workshop on Machine Learning and Data Mining in Pattern Recognition, Springer, pp 134–148
20. Hoeffding W (1994) Probability inequalities for sums of bounded random variables. In: The Collected Works of Wassily Hoeffding, Springer, pp 409–426
21. Khalilian M, Mustapha N, Sulaiman N (2016) Data stream clustering by divide and conquer approach based on vector model. J Big Data 3(1):1
22. Laurinec P, Lucká M (2019) Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting. Data Min Knowl Disc 33(2):413–445
23. Li Y, Li D, Wang S, Zhai Y (2014) Incremental entropy-based clustering on categorical data streams with concept drift. Knowl-Based Syst 59:33–47
24. Liu Y, Li Z, Xiong H, Gao X, Wu J (2010) Understanding of internal clustering validation measures. In: 2010 IEEE international conference on data mining, IEEE, pp 911–916
25. Meesuksabai W, Kangkachit T, Waiyamai K (2011) Hue-stream: evolution-based clustering technique for heterogeneous data streams with uncertainty. In: International Conference on Advanced Data Mining and Applications, Springer, pp 27–40
26. Pearson K (1896) Vii. mathematical contributions to the theory of evolution. –iii. regression, heredity, and panmixia. Philos Trans Roy Soc Lond Ser A 187:253–318
27. Puschmann D, Barnaghi P, Tafazolli R (2016) Adaptive clustering for dynamic IoT data streams. IEEE Internet Things J 4(1):64–74
28. Rodrigues PP, Gama J, Pedroso J (2008) Hierarchical clustering of time-series data streams. IEEE Trans Knowl Data Eng 20(5):615–627
29. Tu L (2012) Clustering on multiple data streams. In: Advances in Computer Science and Information Engineering, Springer, pp 73–78
30. Udommanetanakit K, Rakthanmanon T, Waiyamai K (2007) E-stream: evolution-based technique for stream clustering. In: International conference on advanced data mining and applications, Springer, pp 605–615

31. Wu J, Xiong H, Chen J (2009) Adapting the right measures for *k*-means clustering. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 877–886
32. Zhao L, Wang L, Dw C (2012) Hoeffding bound based evolutionary algorithm for symbolic regression. Eng Appl Artif Intell 25(5):945–957
33. Zhong S (2005) Efficient streaming text clustering. Neural Netw 18(5–6):790–798