SN Computer Science (2023) 4:489
https://doi.org/10.1007/s42979-023-01887-x

ORIGINAL RESEARCH

Mining Discriminative Itemsets Over Data Streams Using Efficient Sliding Window

Majid Seyfi[1] · Richi Nayak[1] · Yue Xu[1]

Received: 20 August 2022 / Accepted: 4 May 2023

Abstract

In this paper, we present an efficient novel method for mining discriminative itemsets over data streams using the sliding window model. Discriminative itemsets are the itemsets that are frequent in the target data stream, and their frequency in the target stream is much higher in comparison to their frequency in the rest of the streams. The problem of mining discriminative itemsets has more challenges than mining frequent itemsets, especially in the sliding window model, as during the window frame sliding, the algorithms have to deal with the combinatorial explosion of itemsets in more than one data stream, for the transactions coming in and going out of the sliding window. We propose a single scan algorithm using two novel in-memory data structures for mining discriminative itemsets in a combination of offline and online sliding windows. Offline processing is used for controlling the generation of many unpromising itemsets. Online processing is used for getting more up-to-date and accurate online answers between two offline slidings. The discovered discriminative itemsets are accurately updated in the offline sliding window periodically, and the mining process is continued in the online sliding between two periodic offline slidings. The extensive empirical analysis shows that the proposed algorithm provides efficient time and space complexities with full accuracy. The algorithm can handle large, fast-speed, and complex data streams.

Keywords Data stream mining · Discriminative itemsets · Prefix tree · Sliding window model

Introduction

The data stream is a type of ongoing dataset that is generated and flows continuously in high volume and at fast speed during its period [1]. This has to be quickly processed to extract real-time insight from it. The area of frequent itemset mining in a single data stream is fulfilled with numerous quality algorithms [2]. Working on multiple data streams for mining discriminative items and discriminative itemsets is an emerging topic [3–6]. The discriminative itemsets in the sliding window model are defined as the itemsets that are frequent in the target data stream and have much higher frequencies than that of the same itemsets in other data streams in a fixed recent period. To make it simpler, we use the term 'general data stream' for other data streams. Specifically, we look for the itemsets that are comparatively frequent in the target data stream and infrequent in the general data stream during the fixed-size sliding window frame. The discriminative itemsets in this research problem are fundamentally used for recognizing the target data stream from all other data streams in a finite number of recent transactions or a fixed recent period.

The topic of discriminative itemset mining originated in [3], is different from our proposed method in this paper. First, the method proposed in [3] is looking for the discriminative items while we are looking for discriminative itemsets. Second, the discriminative items are discovered in the landmark window model while we discover the discriminative itemsets in the sliding window model. Third, the discriminative items are discovered approximately while

This article is part of the topical collection "Knowledge Discovery, Knowledge Engineering and Knowledge Management" guest edited by Joaquim Filipe, Ana Fred, Jan Dietz, Ana Salgado and Jorge Bernardino.

✉ Majid Seyfi
majid.seyfi@gmail.com

Richi Nayak
r.nayak@qut.edu.au

Yue Xu
yue.xu@qut.edu.au

[1] Data Science Discipline, Science and Engineering Faculty, Queensland University of Technology, Brisbane, QLD, Australia

Published online: 27 June 2023

we discover the discriminative itemsets with full accuracy. Also, our proposed method has clear differences from the *DISTree* and *DISSparse* methods proposed in [5–7], for mining discriminative itemsets in a static batch of transactions.

H-DISSparse method [8] is proposed for mining discriminative itemsets over data streams but using the tilted-time window model. This has key differences from the proposed method in this paper, using the sliding window model. First, although they are both single-pass algorithms working on data streams, the former works on a better approximation for mining discriminative itemsets through history, while the latter is an exact efficient algorithm for mining discriminative itemsets in a fixed recent sliding time frame. Second, the tilted-time window model is defined in multiple historical different size periods, while the sliding window model is defined in a fixed-size recent period. In *H-DISSparse*, the logarithmic tilted-time window model is applied to the *DISSparse* [7] which is updated by shifting and merging for displaying the recent discriminative itemsets in fine granularities and the historical ones in coarse granularities. The sliding window model, on the other side, has a dynamic start point moving by time focusing only on the recent time frame. The sliding window model should deal with new transactions coming to, and old transactions going out of the sliding window model. Third, they both define prefix-tree structures for holding and updating the discovered patterns with a difference that *H-DISSparse* uses *H-DISStream* with a possible built-in tilted-time window model in each node, while in sliding window model *S-DISStream* (i.e., a simple prefix-tree structure) holds the discovered patterns from the recent transactions in the range of window frame size. Fourth, the *H-DISStream*, with its tilted-time window model is updated in offline mode after processing a new batch of transactions in the pre-defined time intervals. On the other side, the *S-DISStream* in our proposed sliding model is updated in a combination of offline and online modes. It accurately reports the discriminative itemsets in offline sliding windows, while using the online monitoring for showing the more up-to-date patterns with good approximation. This online mode is happening between two offline slidings.

The embedded knowledge in data streams is changing over time. Processing the recent transactions is important in the applications looking for the recent patterns inside the data streams (e.g., anomaly detection and decision making). Discovering the recent patterns in a finite number of transactions or a fixed period in data streams and continuous monitoring of the variations in these patterns gives valuable information for data stream mining based on the recent trends. The transactions for pattern mining should be restricted to the most recent ones in the fixed-size window frame by eliminating the effects of the obsolete transactions in the information [9].

Discriminative itemsets differentiate the target data stream from others, as they ignore the generally available frequent itemsets [3, 5, 6]. Stock market monitoring for discovering the fluctuations in the most recent transactions in the fixed-size sliding window frame can be useful by quickly detecting the discriminative itemsets in the recent trends between different markets. Discriminative itemsets represented in the sliding window model are useful for data stream comparison based on their recent trends by highlighting the high demanding itemsets in the target market compared with the other markets in the fixed recent period. A better web page personalization can be achieved by following the changes in user preferences compared to the common trends in the fixed recent period. The stated sequences of queries in one geographical area that have higher support compared to another area are time-related. Monitoring the changes in the discriminative pattern trends in networks during the last few minutes is very useful for anomaly detection and network interference prediction.

The *Apriori* property defined for the frequent itemsets does not stand true in discriminative itemsets mining, consequently, the frequent itemset mining algorithms cannot be directly applied. Moreover, the explosion in the number of itemsets combinations is worse in the sliding window compared to other window models, as new transactions are coming to the sliding window and old transactions are going out of the sliding window [10]. The sliding window model has to be updated by adding a large number of itemset combinations of recent transactions and deleting the effects of the old transactions from the sliding window frame. Depending on the application, the sliding window frame is defined based on a fixed period or fixed number of transactions, and the discriminative itemsets are represented in offline and online updating states. The greatest challenge is a generation of compact in-memory data structures by processing only the recent potential discriminative itemsets in the offline and online sliding states. Moreover, unsynchronized data streams with different speeds add more challenges to the sliding window updating process.

In this paper, we first define the "discriminative itemsets" mining in data streams with the sliding window model. We develop the novel *S-DISSparse* algorithm for efficiently discovering the discriminative itemsets in the fixed-size sliding window frame. We propose two novel data structures named *S-FP-Tree* and *S-DISStream* based on the basic principles presented in *FP-Growth* [11]. We propose two determinative heuristics for exact and efficient mining of discriminative itemsets using the offline sliding window. *S-DISStream* is updated in an offline state when a new partition, i.e., a group of transactions coming through the time, arrives. *S-DISStream* is also used for online (or real-time) monitoring of discriminative itemsets between two offline slidings. Empirical analysis shows efficient time and space

complexity gained by the *S-DISSparse* algorithm for mining discriminative itemsets in the offline and online sliding window. To the best of our knowledge, the proposed method in this paper is the first algorithm for mining discriminative itemsets in data streams using the sliding window model.

The following are contributed to this paper:

- Developing the single-pass algorithm called *S-DISSparse* (i.e., Sliding DIScriminative Sparse; being sparse is the characteristic of these itemsets) for mining discriminative itemsets in data streams using an efficient sliding window model;
- Introducing efficient offline and online updating of the sliding window using novel in-memory data structures;
- Applying two determinative heuristics to the algorithm for removing the impossible discriminative itemsets from the mining process, only based on the recent transactions in the sliding window frame.
- Conducting extensive experiments on the proposed algorithm in large synthetic and real datasets with different parameter settings;
- Indicating practical tactics and principles for parameter settings following the application domain necessities and dataset characteristics;

The rest of the paper is organized as follows: The related works are discussed in "Related Works". The definition of the research problem is presented in "Problem Statement". The sliding window model and its offline updating process are discussed in "Offline Sliding Window". The online sliding window is discussed in "Online Sliding Window". The *S-DISSparse* method is proposed in Sect. 6. Experimental results are reported in "S-DISSparse Method". Section "Conclusion and Future Works" finalizes and concludes the paper.

Related Works

The *Moment* is a famous method proposed for mining closed frequent patterns from data streams using a sliding window model [12]. Compared to the frequent itemsets, the discriminative itemsets are a much smaller subset and they are a type of contrast pattern [13]. The concept of closed frequent itemsets does not apply to the discriminative itemset mining as it does not follow the *Apriori* property. Our proposed algorithm in this paper assumes that the *Apriori* property does not hold. This is the basic assumption in contrast mining [13]. The *DISSparse* algorithm [6, 7] is an efficient method for mining discriminative itemsets in one batch of transactions. *H-DISSparse* [8] is the most related work to our proposed *S-DISSparse* method. This method uses several well-defined characteristics to improve the approximate support of the discovered discriminative itemsets in the offline logarithmic tilted-time window model. In contrast, *S-DISSparse* is an exact algorithm working on a sliding window model. The discriminative itemsets in the offline state are discovered accurately and efficiently and then used for approximate online sliding between two offline slidings. In the sliding window model, the mining is limited to the recent transactions fitted in the window frame by removing the older transactions out of the window frame. A few simpler and less challenging problems were proposed for mining discriminative items in data streams [3, 4]. As well-known contrasting patterns, the emerging patterns (*EPs*) are presented [14]. They are the patterns whose frequencies grow significantly higher in one dataset in comparison to another one.

The definition of discriminative itemsets is very similar to Emerging Patterns (*EPs*), however, there exist several differences. First, the *EPs* methods use maximal itemsets occurring between two borders to find emerging patterns, and the real support of *EPs* is not recorded. In discriminative itemsets mining, frequencies of itemsets have to be known to distinguish between highly and lowly discriminative itemsets. Second, the *EPs* algorithms are mostly designed for static datasets except for a small number of works in stream mining that are based on the same idea of border definition [15, 16]. Authors in [15] have attempted emerging pattern mining in data streams. This method showed the *EPs* related to each block of transactions, and discard the block from the process. Third, in the discriminative itemsets mining method proposed in this paper, only the promising combinations of itemsets are generated.

The δ-discriminative emerging patterns are defined as a special type of useful emerging patterns [17] for mining the statistically important emerging patterns. The δ-discriminative emerging patterns are the frequent patterns in the target dataset that have minimum frequency (i.e., less than δ) in the other datasets. The proposed *DPM* algorithm eliminates part of the patterns as redundant compared to the discriminative itemsets defined in this paper. The full set of discriminative itemsets is discovered without redundancy, each with explicit frequencies in data streams. Moreover, the Conditional Discriminative Patterns Mining method (*CDPM*) [18] is proposed for discovering a set of significant non-redundant discriminative patterns, without any similar discriminative power in their subsets. Other research tracks are looking for statistically significant patterns [19] and discriminative sequential patterns [20].

As an example, for the importance of discriminative itemset mining and its advantages to the emerging patterns and statistically important emerging patterns, we can refer to the applications that need discriminative itemsets with explicit frequencies. The discriminative itemsets in the market basket datasets are discovered with explicit frequencies and without any redundancy. In these types of applications, the

discriminative itemsets are used for mining the discriminative rules with different supports and different confidences. Also, minimizing the number of discriminative itemsets based on their statistical importance caused a smaller number of discovered discriminative itemsets in the sliding window model and less accuracy.

Depending on the target application domain, the data stream processing can be divided into two categories: (1) offline, i.e., bulk processing of input transactions; and (2) online, i.e., processing by transactions generation through time [21]. The proposed *S-DISSparse* algorithm in this paper is designed based on a combination of offline and online processing. *S-DISSparse* can report the accurate set of discriminative itemsets in the offline sliding window, as well as, it can show the online results with good approximation by online monitoring which is happening between two offline slidings. This approach can be applied in real-world scenarios with multiple unsynchronized fast-speed data streams. The correctness of the proposed algorithm is guaranteed in the offline sliding window model.

The sliding window model is widely used for frequent pattern mining in applications that need recent patterns, ignoring the obsolete information that is currently useless or even invalid [9, 22–29]. These frequent pattern mining methods have been expanded based on the basic *Apriori* [22, 25–27] and *FP-growth* [9, 23, 24, 28, 29]. The concept of closed [23] and maximal itemsets [28] is utilized in some research to deal with the combinatorial problem in the sliding window model. However, this reduced itemsets concept is not applicable in discriminative mining, as the *Apriori* property is not valid in them.

Problem Statement

Let $\sum$ be the alphabet set of items, a transaction $T = \{e_1, \dots e_i, e_{i+1}, \dots, e_n\}$, $e_i \in \sum$, is defined as a set of items in $\sum$. The two data streams S_i and S_j are defined as the target and general data streams; each consists of a different number of transactions, i.e., n_i and n_j, respectively. The proposed algorithm in this paper can be extended to multiple data streams using a specific counter for each data stream. A group of input transactions from two data streams S_i and S_j in the pre-defined period are set as a batch of transactions B_n i.e., $n \geq 1$, and n is the latest batch of transactions. Let P be a partition fitting an input batch of transactions B. The sliding window frame denoted as W is made of a fixed number of partitions P_k i.e., $k \geq 1$ and $P_k \subseteq W$ (e.g., in Fig. 1 the sliding window is made of three partitions) and refers to the fixed recent period containing itemsets made of transactions in two data streams S_i and S_j with the lengths of n_i^w and n_j^w, respectively. All partitions cover the same width of the period but the number of transactions in partitions in the sliding window frame W varies depending on the speed of data streams. This means the partitions can be in different sizes depending on the size of input batches. The time interval for the batch of transactions depends on the partition time frame (i.e., $p = W/k$). The window frame W slides in an offline state by adding the itemsets in the recent partition i.e., P_{new}, and deleting the itemsets in the oldest partition i.e., P_{old}, as in Fig. 1.

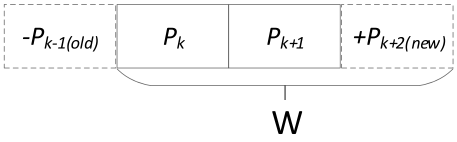


Fig. 1 Sliding window model W made of three partitions P

An itemset I is a set of items in $\sum$ that occur together. The number of transactions containing the itemset is called itemset frequency. We show the frequency of itemset I in each data stream S_i in the window frame W as $f_i^w(I)$ and $r_i^w(I)$ shows the frequency ratio of itemset I in data stream S_i in the window frame W defined as $r_i^w(I) = f_i^w(I)/n_i^w$.

In case of a higher frequency ratio of itemset I in the target data stream S_i in the sliding window frame W than the frequency ratio in the general data stream S_j, i.e., $\frac{r_i^w(I)}{r_j^w(I)} > 1$, the itemset I can be considered as a discriminative itemset. We defined the $R_{ij}^w(I)$ as the ratio between $r_i^w(I)$ and $r_j^w(I)$, i.e., $R_{ij}^w(I) = \frac{r_i^w(I)}{r_j^w(I)}$. With the higher $R_{ij}^w(I)$, the itemset I is more discriminative.

We show the discriminative level by a user-defined threshold $\theta > 1$, with no upper bound. In the case of $R_{ij}^w(I) \geq \theta$, the itemset I, is considered discriminative in the window frame W if, which is formally defined as:

$$R_{ij}^w(I) = \frac{r_i^w(I)}{r_j^w(I)} = \frac{f_i^w(I) n_j^w}{f_j^w(I) n_i^w} \geq \theta \tag{1}$$

The low $f_i^w(I)$ could lead to a very large $R_{ij}^w(I)$. To identify the discriminative itemsets with reasonable frequency in the window frame W, and also in the case of $f_j^w(I) = 0$, another user-specified support threshold, $0 < \varphi < 1/\theta$ is identified. This eliminates the itemsets with very low frequency in the window frame W. Consequently, a discriminative itemset I is defined by its frequency in the window frame W greater than $\varphi\theta n_i$ i.e., $f_i^w(I) \geq \varphi\theta n_i$.

Definition 1. Discriminative itemsets in the sliding window model: Let target data stream S_i and general data stream S_j be with the current size of n_i^w and n_j^w in the sliding window frame W. These data streams contain transactions with

varied lengths made of items in $\sum$. Considering a user-defined discriminative level threshold $\theta > 1$ and a support threshold $\varphi\epsilon(0, 1/\theta)$, we define the set of discriminative itemsets in S_i against S_j in the sliding window model in window frame W, denoted as DI_{ij}^w as:

$$DI_{ij}^w = \left\{ I \subseteq \sum | f_i^w(I) \geq \varphi\theta n_i^w \& R_{ij}^w(I) \geq \theta \right\} \quad (2)$$

Moreover, a relaxation of $\alpha\epsilon(0, 1)$ is defined for sub-discriminative itemsets. The smaller α leads to a greater number of sub-discriminative itemsets. We are interested in the discriminative itemsets; however, we also kept track of the sub-discriminative itemsets during the process as they may be discriminative by sliding window frames. We use this relaxation only for mining the discriminative itemsets with better accuracy in the online sliding window between two offline slidings. The discriminative itemsets in the offline sliding window are discovered with full accuracy and they do not need the definition of relaxation of α.

Definition 2. Sub-discriminative itemsets in the sliding window model: Let target data stream S_i and general data stream S_j be with the current size of n_i^w and n_j^w in the sliding window frame W. These data streams contain transactions with varied lengths made of items in $\sum$. Considering a user-defined discriminative level threshold $\theta > 1$, a support threshold $\varphi\epsilon(0, 1/\theta)$, and a relaxation parameter $\alpha\epsilon(0, 1)$, we define a set of sub-discriminative itemsets in S_i against S_j in the sliding window model W, denoted as SDI_{ij}^w as:

$$SDI_{ij}^w = \left\{ I \subseteq \sum | f_i^w(I) \geq \alpha\varphi\theta n_i^w \& R_{ij}^w \geq \alpha\theta \right\} \quad (3)$$

The itemsets that are not discriminative and not sub-discriminative are defined as non-discriminative itemsets. The non-discriminative itemsets are used for tail pruning in the sliding window model.

The sliding window model is updated in an offline state in the specific time intervals defined for the batch of transactions. The sub-discriminative itemsets are also saved as they may become discriminative in the future by online sliding window frames. Two determinative heuristics are proposed for efficient and exact mining discriminative itemsets in the offline sliding window. The online sliding window is between two offline sliding of the window model with an approximate bound guarantee.

Offline Sliding Window

The sliding window model is made of itemsets from the recent transactions in the range of window frame size in a prefix tree structure (i.e., *S-DISStream* as presented in Fig. 4). The size of the sliding window frame is defined based on the desired output range in the application domain and the limit of main memory. The frequencies of itemsets are held in the full-size sliding window frame and the discriminative itemsets are reported in the offline and online updating sliding windows. The general approach is based on using the offline and online sliding windows together. Offline sliding is happening periodically and online sliding is happening between two offline slidings limited to only the recently discovered discriminative itemsets i.e., itemsets recently exist in *S-DISStream* prefix-tree structure.

The offline sliding window is updated by the new batch of transactions arriving and the oldest batch of transactions going out in offline time intervals in the window frame. The transactions are stored in a prefix-tree structure which is updated in an offline state i.e., *S-FP-Tree* prefix-tree structure. *S-FP-Tree* holds the transactions fitted in the offline sliding window. The offline window is updated periodically based on the recently updated transactions in *S-FP-Tree*. The discovered discriminative itemsets in the offline state are updated in *S-DISStream* accurately which is also used for online sliding between two offline slidings. The online sliding window is updated by adding and deleting the recent and the oldest transaction respectively in the window frame in the real-time frame i.e., limited to only *S-DISStream*. The recent and the oldest transactions update the itemsets' frequencies if they have any subset in *S-DISStream*. The itemsets in *S-DISStream* then change their status from discriminative to non-discriminative or vice versa in the online state. Online sliding is used for more up-to-date and accurate online answers between two offline slidings.

Mining Discriminative Itemsets in Sliding Window Using Prefix Tree

In *FP-Growth* [11], a prefix-tree structure is presented out of the frequent items of each transaction in a dataset. This is a concise prefix tree as it holds the transactions by sharing the branches for their most common frequent items. Each branch starting from root to a node is the prefix of the itemsets ending at a node after that node. The prefix-tree nodes are associated with a counter representing the frequency of the itemset made of the items along the path starting from the root and ending at each specific node. In this paper, we use the sequence from the root to a node to represent an itemset and the two associated values indicate the frequency of the itemset in the target data stream and the general data stream, respectively, e.g., in Fig. 2 the itemset $cb_{7,3}$ shows the frequency of itemset *cb* is 7 in the target data stream and 3 in the general data stream.

Two prefix tree structures are defined for holding the transactions and itemsets in the sliding window model, respectively. The tree structure is applied as it is the most

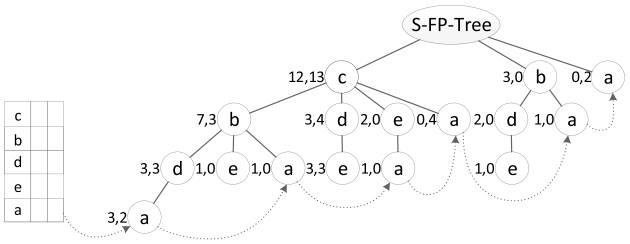


Fig. 2 *Header-Table* and *S-FP-Tree* structures by P_1

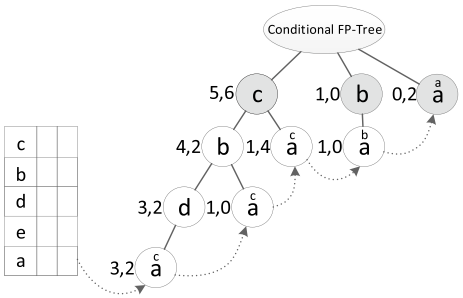


Fig. 3 Conditional *FP-Tree* of *Header-Table* item ***a*** associated with the top ancestor on the first level

efficient way for holding the transactions (or itemsets) by sharing the most common frequent items in the same branch.

S-FP-Tree: The prefix tree structure proposed in [11] is used for holding the items of the transactions, without pruning infrequent items, by sharing the branches for their most common frequent items (i.e., *S-FP-Tree* includes all items). The *S-FP-Tree* is adapted by adding two counters in each node for holding the frequencies of the itemsets in the target data stream S_i and the general data stream S_j, respectively; (e.g., there are two counters associated with each node in the *S-FP-Tree* in Fig. 2). The *S-FP-Tree* is updated during offline window sliding by adding the transactions of the new partition and deleting the transactions of the oldest partition in the sliding window frame W. New paths in the *S-FP-Tree* are added for the new transactions or the frequencies of paths are updated. The frequencies of paths in the *S-FP-Tree* are decreased by deleting the transactions of the oldest partition. The nodes in the *S-FP-Tree* are tagged based on their recent status as stable if the frequencies have not changed by adding the new partition or deleting the oldest partition, respectively, otherwise, they are tagged as updated.

S-DISStream: The *S-DISStream* prefix tree structure is also defined for holding all the recently discovered discriminative and sub-discriminative itemsets as well, in the sliding window. The branches are shared by the most common frequent items of both discriminative and sub-discriminative itemsets in the same *S-DISStream* structure (e.g., as in Fig. 4). The *S-DISStream* paths, starting from the root of the prefix tree structure, may represent subsets of multiple discriminative and sub-discriminative itemsets. Each node in *S-DISStream* is adapted with two counters f_i and f_j, each holding the frequencies of an itemset in the target data stream S_i and the general data stream S_j, respectively in the sliding window frame W. The itemsets in *S-DISStream* are made of transactions from the partitions that fit in the online sliding window frame W.

The *Header-Table* is defined for fast traversing the prefix tree structures using the links holding the itemsets ending with identical items. Each *Header-Table* item node in *S-DISStream* saves the top ancestor on the first level as the root; (e.g., node *c* is the top ancestor of all different *Header-Table* items, including *a*,*e*, *b*, and *c* in the left-most subtree in Fig. 4, and *c* appears in all nodes in the left-most subtree). The nodes in the first level of *S-DISStream* determine different subtrees made of the number of itemsets under the same root in the first level of *S-DISStream* and ending with different *Header-Table* items. Following similar notations as in the conditional *FP-Tree* in *DISSparse* method [6, 7], a subtree in *S-DISStream* is denoted as $Subtree_{root}$; for example, the *S-DISStream* in Fig. 4 has three subtrees under root *c*, *b*, and *e* (i.e., $Subtree_c$, $Subtree_b$, and $Subtree_e$, respectively).

The *Header-Table* item node in conditional *FP-Tree* saves the top ancestor on the first level as the root as well; (e.g., node *c* is the top ancestor of all different *Header-Table* items, including *a*, *b*, and *c* in the left-most subtree in Fig. 3, and *c* appears in all *Header-Table* item nodes in the left-most subtree). The subtrees which have the potential to contain discriminative itemsets are called potential subtrees denoted as $Potential(Subtree_{root})$. The subsets may exist in $Subtree_{root}$ as internal nodes which are in the paths between the *root* of $Subtree_{root}$ and the items in $Header\_Table\_items\left(Subtree_{root}\right)$. The internal nodes are denoted as $Internalnode_{root}$ following similar notations as in [6, 7] (e.g., in the left-most subtree in conditional *FP-Tree* in Fig. 6, $Subtree_c$ has two internal nodes i.e., b_c and d_c, respectively). During processing each header item, the set of *Header-Table* items which are linked under their subtree root node using *Header-Table* links is denoted as $Header\_Table\_items(Subtree_{root})$. The nodes are tagged as discriminative, sub-discriminative or non-discriminative (i.e., a subset of discriminative, or sub-discriminative itemsets). The *S-DISStream* is updated in the sliding window model in offline and online states. The nodes in *S-DISStream* are tagged based on their recent status as stable if the frequencies have not changed by offline window sliding i.e., adding new partitions or deleting the oldest partitions, otherwise, they are tagged as updated. The nodes

Table 1 The first input batch in data streams fits in P_1

S /T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	*abcd*	*abcd*	*abcd*	*abc*	*ab*	*ace*	*bce*	*bc*	*bc*	*bde*	*bd*	*cde*	*cde*	*cde*	*ce*
S_2	*abcd*	*abcd*	*ac*	*ac*	*ac*	*ac*	*a*	*a*	*bcd*	*cde*	*cde*	*cde*	*cd*	*c*	*c*

Table 2 *Desc-Flist* order in S_1 in the first batch

Item/order	*a*	*b*	*c*	*d*	*e*
Frequency	6	10	12	8	7
Order	**4**	**1**	**0**	**2**	**3**

The bold values show the items' order

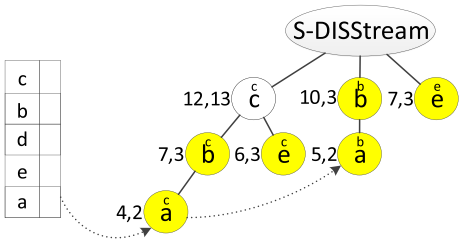


Fig. 4 *Header-Table* and *S-DISStream* structures by P_1

in *S-DISStream* are tagged as online if they are updated during online sliding. The concept of the stable nodes in *S-FP-Tree* and *S-DISStream* refers to the itemsets that do not have any change in their frequencies during the sliding window model.

Initializing the offline sliding window

The *S-FP-Tree* is initialized by the transactions fitting in the first partition P_1; and discriminative itemsets are discovered following the normal process of *DISSparse* algorithm [6, 7] and set to the *S-DISStream*. All the nodes in the *S-FP-Tree* and *S-DISStream* are tagged as stable (i.e., not updated) before adding the recent partition P_{new} and deleting the oldest partition P_{old} i.e., in the full-size window frame W as in Fig. 1. The nodes in *S-FP-Tree* are tagged as updated if the frequencies change during window model sliding; (e.g., by adding new transactions or deleting old transactions).

The conditional *FP-Tree* is made for each *Header-Table* item based on the item's conditional patterns in the *S-FP-Tree*. The *Header-Table* items in the conditional *FP-Tree* hold the same status as following the tags in the *S-FP-Tree*. The tags in *Header_Table_items*($Subtree_{root}$) in the conditional *FP-Tree* can show that the itemsets in a $Subtree_{root}$ are updated or stable during the window model sliding. Similar approaches are applied for the updated or stable itemsets with subsets of $Internalnode_{root}$ in a potential $Subtree_{root}$. Two heuristics are proposed accordingly based on the recently updated itemsets, for mining discriminative itemsets out of the updated potential discriminative subsets in the conditional *FP-Tree*. The stable itemsets are checked in the *S-DISStream* and are tagged based on the recent data stream lengths in the sliding window frame W.

Example 1. The *S-FP-Tree* and *S-DISStream* constructions and updating are graphically monitored using the running example with two batches of transaction fitting in the first two partitions in the sliding window model, respectively. The first batch made of data streams S_1 and S_2 ($n_1 = n_2 = 15$) fits in P_1 and is presented in Table 1. The second batch made of data streams S_1 and S_2 ($n_1 = n_2 = 5$) fits in P_2 is presented in Table 3.

The items in the transaction can be in any order. However, in data stream mining, the transaction items are ordered based on the decreasing frequencies order in the first batch of transactions as in [11] (i.e., *Desc-Flist* order in Table 2 is constructed out of frequent items in Table 1). This will lead to concise data structures and efficient monitoring and processing. The *Desc-Flist* defines the default order used for the two prefix tree structures. Moreover, it shows the bottom-up processing order in the *Header-Table* (e.g., the *Header-Table* in Fig. 2 is processed from item *a* which is the least frequent item in the data stream). This *Desc-Flist* remains the same for monitoring and processing all the upcoming batches in data streams. All the frequent items in each input transaction in the data streams, before adding to the prefix tree structures, are sorted based on the *Desc-Flist* order (e.g., the *S–FP-Tree* and *S-DISStream* in Figs. 2 and 4 are made of transaction items ordered based on *Desc-Flist*).

In our running example, we set the discriminative level threshold to $\theta = 2$ and the support threshold is set to $\varphi = 0.1$. The *S-FP-Tree* and *S-DISStream* structures made of the first partition P_1 are represented in Figs. 2 and 4, respectively. The highlighted nodes in *S-DISStream* in Fig. 4 refer to the discriminative itemset.

The conditional *FP-Tree* is presented in Fig. 3 for the *Header-Table* item *a* which has the lowest frequency). In Fig. 3, we process the subtrees one by one as in [6, 7].

Following the same notation as in [6, 7], the set of itemsets in subtree $Subtree_{root}$ starting from the *root* and ending with a header item $a_{n,m} \in Header\_Table\_items(Subtree_{root})$, is denoted as $itemsets(root, a)$e.g.,$itemsets(c, a) = \{I(a_{3,2}), I(a_{1,0}), I(a_{1,4})\}$ as in Fig. 3. The $\text{Max\_freq}_i(root, a)$ denotes the maximum frequency of $itemsets(root, a)$ in the target data stream S_i This is defined as the sum of the frequencies in S_i of the itemsets in $itemsets(root, a)$ as below.

$$\text{Max\_freq}_i(root, a) = \sum_{b \in itemsets(root,a)} f_i(b) \tag{4}$$

In the equation above, $f_i(b)$ refers to the frequency in S_i of an itemset b in subtree $Subtree_{root}$ (e.g., in Fig. 3 in the left-most subtree $Subtree_c$, which contains three itemsets ending with $a_{3,2}$, $a_{1,0}$, and $a_{1,4}$, the maximum frequency of itemsets in S_i is equal to 5, i.e., $\text{Max\_freq}_i(c, a) = 5$).

If we consider $\mathcal{S}$ as the power set of $itemsets(root, a)$, $\mathcal{S} = 2^{itemsets(root,a)}$, i.e., $\mathcal{S}$ consists of all combinatorial subsets of $itemsets(root, a)$. For $B \in \mathcal{S}$ and $B \neq \{\}$, the discriminative value of each itemset in B is defined below.

$$\text{Dis\_value}(B) = \begin{cases} \frac{r_i(B)}{\theta} \sum_{b \in B} f_j(b) = 0 \\ \frac{r_i(B)}{r_j(B)} \sum_{b \in B} f_j(b) > 0 \end{cases} \tag{5}$$

where $r_i(B) = \frac{\sum_{b \in B} f_i(b)}{n_i}$ and $r_j(B) = \frac{\sum_{b \in B} f_j(b)}{n_j}$. $r_i(B)$ is called the relative support of B inS_i. This is the sum of the relative supports of the itemsets in B in S_i andS_j, respectively.

When the itemsets $b \in B$ do not exist in the dataset S_j, the $\sum_{b \in B} f_j(b) = 0$. Here, $\text{Dis\_value}(B)$ is defined as the ratio between the sum of the relative supports of the itemsets of B in S_i and the discriminative level threshold θ. This means that the itemset should be both frequent and significant in the target data stream.

When at least one of the itemsets in B does exist in the general data stream, the $\sum_{b \in B} f_j(b) > 0$ which indicates that $\text{Dis\_value}(B)$ is defined as the ratio between the sum of the relative supports of the itemsets of B in S_i and the sum of the relative supports of the itemsets of B inS_j. Here, $\frac{r_i(B)}{r_j(B)}$ defines the discriminative value of$B$, denoted as $R_{ij}(B) = \frac{r_i(B)}{r_j(B)}$ (e.g., in the left-most subtree in Fig. 3, for $= \{cbad, cba, ca\}, R_{ij}(B) = \frac{5}{6} * \frac{15}{15}, n_i = n_j = 15$).

By processing the most recent partition, the *S-DISStream* holds the discriminative itemsets in the offline sliding window as in Fig. 4. The *S-FP-Tree* and *S-DISStream* structures are then tagged based on their stable and updated subsets for efficient mining by adding the recent partition P_{new} and deleting the oldest partition P_{old}.

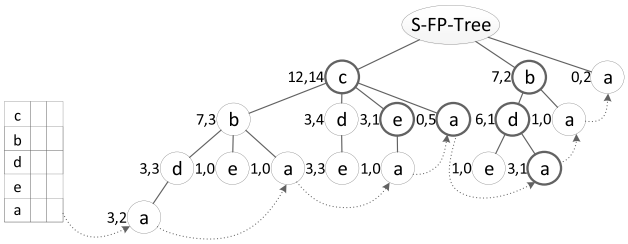


Fig. 5 *Header-Table* and updated *S-FP-Tree* structures by adding P_2

Table 3 The second input batch in data streams fits in P_2

S/T	1	2	3	4	5
S_1	*abd*	*abd*	*abd*	*bd*	*ce*
S_2	*abd*	*ac*	*b*	*ce*	*c*

Incremental Offline Sliding Window

The sliding window model can be simply implemented by adapting the *DISTree* algorithm [5] or *DISSparse* algorithm [6, 7], in the offline updating state. The sliding window model is updated by discriminative itemsets discovered from each new batch of transactions fitting in the new partition added to the sliding window frame W, and deleting the itemsets belonging to the oldest partition out of the full-size sliding window frame W. However, there are several challenges with this naïve approach.

First, the sliding window frame W is made of several partitions, and discovering the discriminative itemsets in a single partition, and merging them with itemsets in the full-size sliding window frame can result in high numbers of false positives and false negatives, which significantly downgrades the output quality. Second, many itemsets can be discriminative in a single partition and non-discriminative in the sliding window frame W, causing an inefficient mining process. Third, two batch processing must be done during the window model sliding i.e., one for adding the discriminative itemsets in the recent batch and one for deleting the itemsets in the oldest batch in the sliding window frame W.

In this paper, two heuristics are proposed based on the *S-FP-Tree* nodes' status (i.e., stable or updated during window model sliding) within efficient time and space use for offline sliding window.

Stable and updated subsets in the offline sliding window

Before processing the next batch of transactions fitting in P_2, all the nodes in the *S-FP-Tree* and *S-DISStream* structures are tagged as stable. Figure 5 shows the *S-FP-Tree* structures after adding the second batch of transactions fits in P_2 (i.e., as in Table 3). The updated nodes in *S-FP-Tree* are represented by thick borders; (e.g., the path $bda_{3,1}$ appears

in the *S-FP-Tree* after adding the new batch of transactions fits in P_2).

A *Potential*($Subtree_{root}$) in the conditional *FP-Tree* in the sliding window frame W satisfies two conditions i.e., $\text{Max\_freq}_i(root, a) \geq \varphi\theta n_i^w$ and $\text{Max\_dis\_value}(root, a) \geq \theta$, where $itemsets(root, a)$ is the set of itemsets in $Subtree_{root}$ ending with a header item $a \in Header\_Table\_items(Subtree_{root})$.

Let $\mathcal{S}$ be the power set of $itemsets(root, a)$, i.e., $\mathcal{S} = 2^{itemsets(root,a)}$, i.e., $\mathcal{S}$ consists of all subsets of $itemsets(root, a)$. For $B \in \mathcal{S}$ and $B \neq \{\}$, the frequency of each itemset in $Subtree_{root}$, concerning the data stream S_i in the sliding window frame W, is defined below.

$$f_i^w(B) = \sum_{b \in B} f_i^w(b) \tag{6}$$

For simplicity, in the equation above, $f_i^w(b)$ refers to the frequency in S_i of an itemset $b \in B$ that belongs to the power set of $itemsets(root, a)$. The frequency of itemset in a subtree is stable if all $b \in B$ are stable during the offline sliding window. The discriminative value of each itemset in $Subtree_{root}$ in the sliding window frame W is defined below.

$$\text{Dis\_value}(B) = \begin{cases} \frac{r_i^w(B)}{\theta} \sum_{b \in B} f_j^w(b) = 0 \\ \frac{r_i^w(B)}{r_j^w} \sum_{b \in B} f_j^w(b) > 0 \end{cases} \tag{7}$$

where $r_i^w(B) = \frac{\sum_{b \in B} f_i^w(b)}{n_i^w}$ and $r_j^w(B) = \frac{\sum_{b \in B} f_j^w(b)}{n_j^w}$. $r_i^w(B)$ is called the relative support of B inS_i. It is the sum of the relative supports of the itemsets in B in S_i andS_j, respectively.

A potential $Subtree_{root}$ contains discriminative itemsets. The potential $Subtree_{root}$ is updated if any of its discriminative itemsets is updated in case of frequencies in S_i or in S_j. A potential $Subtree_{root}$ is stable if all potential discriminative itemsets in $Subtree_{root}$ are stable during the offline sliding window as below.

$$\text{Potentialitemsets} = \left\{ \forall B \in \mathcal{S} \mid \sum_{b \in B} f_i^w(b) \geq \varphi\theta n_i^w \cap \text{Dis\_value}(B) \geq \theta \right\} \tag{8}$$

To find the updated and stable potential discriminative itemsets, all possible itemsets in $Subtree_{root}$ will have to be generated. However, the generation of all possible itemset combinations is time-consuming. In this paper, we propose the simple method for calculating the $\text{Max\_freq}_i(root, a)$ and estimating the $\text{Max\_dis\_value}(root, a)$ in the *S-DISSparse* method.

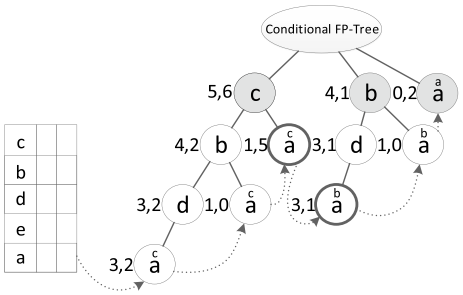


Fig. 6 Conditional *FP-Tree* of *Header-Table* item a updated by P_2

The itemset is stable if it is summed up by the frequencies of only stable itemsets. Let B with maximum $R_{ij}^w(B)$ in $Subtree_{root}$ be defined as B_{max}. Initially, B_{max} is initialized by summing up the $f_i^w(b)$ frequencies of the itemsets b with $f_j^w(b) = 0$. The frequencies of b, with maximum frequency ratio, are summed up by B_{max} only if it increases its discriminative value i.e., $R_{ij}^w(B_{max})$. This B_{max} is either updated or stable. The B_{max} is considered as updated if it is summed up by the frequencies of any updated itemset b. If the discriminative value of B_{max} summed up by any updated b is larger than discriminative level θ, the B_{max} is considered as updated (i.e., the overall frequencies are tested only and not summed up); for example, in Fig. 6, the maximum discriminative value of itemsets in the left-most subtree, $Subtree_c$, is equal to 2, i.e., $\text{Max\_dis\_value}(c, a) = 2$, which is calculated by the sum of frequencies of two stable itemsets ending with the items in $Header\_Table\_items(Subtree_c)$, i.e., $a_{1,0}$ and $a_{3,2}$.

Following Definition 1, if the $f_j^w(b_{max}) = 0$, then $R_{ij}^w(b_{max}) = \frac{f_i^w(b_{max})}{\theta n_i}$. The $\text{Max\_freq}_i(c, a) = 4$, which is calculated by the sum of the frequency of similar stable itemsets i.e., $I(a_{1,0})$ and $I(a_{3,2})$, and the $Subtree_c$, is defined as a stable subtree. As a consequence, the $Subtree_c$ is stable and does not need to be processed for generating all itemset combinations. The algorithm is proposed for calculating the $\text{Max\_freq}_i(root, a)$ and $\text{Max\_dis\_value}(root, a)$ for finding the stable $Subtree_{root}$ in *S-DISSparse* method for mining potential discriminative itemsets. The statement $f^w(b_{max}) += f^w(b)$ in the algorithm is the short way of two statements $f_i^w(b_{max}) += f_i^w(b)$ and $f_j^w(b_{max}) += f_j^w(b)$, for the sake of simplicity.

Algorithm 1 Stable, updated Max_freq$_i(root, a)$, Max_dis_value$(root, a)$
Input: (1) $Subtree_{root}$; (2) header item $a \in Header\_Table\_items(Subtree_{root})$.
Output: (1) stable or updated Max_freq$_i(root, a)$; (2) stable or updated Max_dis_value$(root, a)$.
Begin
1. Max_freq$_i(root, a) = 0$; $F_i^W = 0$; $F_j^W = 0$;
2. **For** each item $b \in Header\_Table\_items(Subtree_{root})$ **do**
3. Max_freq$_i(root, a)$+= $f_i^W(I(b))$;
4. **If** $I(b)$ is updated **then** Tag Max_freq$_i(root, a)$ as updated;
5. **If** $f_j^W(I(b)) = 0$ **then** F_i^W+= $f_i^W(I(b))$; Tag b as checked;
6. **If** $I(b)$ is updated **then** Tag F_i^W as updated; **End if;**
7. **End if;**
8. **End for;**
9. **While** $\exists b \in Header\_Table\_items(Subtree_{root})$ and b are unchecked **do**
10. Find $I(b)$ with maximum $R_{ij}^W(I(b))$;
11. **If** $\frac{F_i^W + f_i^W(I(b))}{F_j^W + f_j^W(I(b))} > \frac{F_i^W}{F_j^W}$ **Or** $\frac{F_i^W + f_i^W(I(b))}{F_j^W + f_j^W(I(b))} > \theta$ **then**
12. F_i^W+= $f_i^W(I(b))$; F_j^W+= $f_j^W(I(b))$;
13. **End if;**
14. Tag b as checked;
15. **If** $I(b)$ is updated **then** Tag F_i^W as updated;
16. **End While;**
17. Max_dis_value$(root, a) = \frac{F_i^W * n_j^W}{F_j^W * n_i^W}$
18. **If** F_i^W is updated **then** Tag Max_dis_value$(root, a)$ as updated;
19. **Return** Stable or updated Max_freq$_i(root, a)$ and Max_dis_value$(root, a)$;
End.

In the above algorithm, the items in *Header_Table_items*($Subtree_{root}$) are scanned in the two separated loops. Each item is checked one time and the updated or stable Max_freq$_i$($root, a$) and Max_dis_value($root, a$) are calculated based on the selected items.

Potential and stable Subtree$_{root}$

All discriminative itemsets in a stable $Subtree_{root}$ are stable in the sliding window frame W compared to the current state in *S-DISStream*. A potential $Subtree_{root}$ can be non-potential before offline sliding with different data stream lengths in the sliding window frame W, and new discriminative itemsets may be discovered in the sliding window frame W; (e.g., by decreasing the length of target data stream S_i or increasing the data streams length ratio, $\frac{n_j^w}{n_i^w}$). A *Potential*($Subtree_{root}$) that satisfies the conditions Max_freq$_i$($root, a$) $\geq \varphi\theta n_i^w$ and Max_dis_value($root, a$) $\geq \theta$ with a smaller frequency in the target data stream S_i, or a smaller frequency ratio in target data stream S_i vs general data stream S_j, compared to the last size of sliding window frame W (i.e., last offline window sliding), is not stable. This part has not been considered in Algorithm 1, considering all partitions are in the same size containing an equal number of transactions. The algorithm has to be modified by holding the length of data streams in the last offline sliding and comparing the Max_freq$_i$($root, a$) and Max_dis_value($root, a$) calculated by the recent and old data stream lengths.

A potential $Subtree_{root}$ in the sliding window model is processed differently.

HEURISTIC 1. A potential $Subtree_{root}$ is stable denoted as *Stable*($Subtree_{root}$) if all potential discriminative itemsets in $Subtree_{root}$ that satisfy the following conditions are stable:

1. Max$_{\text{freq}_i}$($root, a$) $\geq \varphi\theta n_i^w$
2. Max_dis_value($root, a$) $\geq \theta$

where $\theta > 1$ is the discriminative level threshold, $\varphi \epsilon (0, 1/\theta)$ is the support threshold, n_i^w is the size of

target data stream S_i in the sliding window frame W and Max_freq$_i(root, a)$ and Max_dis_value$(root, a)$ are stable if any itemset in $Subtree_{root}$ that satisfies the two conditions is stable.

Lemma 1 (Stable subtree) **HEURISTIC 1** *ensures that any discriminative itemset in a Stable*$(Subtree_{root})$ *is stable in the sliding window model.*

Proof The two conditions in **HEURISTIC 1** ensure that any itemset that is frequent in the target data stream S_i and has a discriminative value larger than the discriminative level θ is stable in the sliding window model. This implies that all discriminative itemsets in a *Stable*$(Subtree_{root})$ exist in the sliding window model by processing the *Potential*$(Subtree_{root})$ during previous offline sliding in the window. The itemset combinations in a *Stable*$(Subtree_{root})$ are tagged as discriminative or non-discriminative in the $Subtree_{root}$ of *S-DISStream*, based on the recent data stream lengths in the sliding window frame W. This implies that the discriminative itemsets are discovered based on the recent data stream lengths that have been changed by adding the new partition and deleting the oldest partition in the sliding window frame W.

For a subtree, if any of the two conditions is updated, the subtree is considered a potential discriminative subtree, and the potential discriminative itemset combinations are generated from the subtree as it may contain new discriminative itemsets.

In Fig. 6, the left-most subtree related to the processing *Header-Table* item a under root node c (i.e., $Subtree_c$) is stable as the only subset of itemsets in the subtree that satisfy the conditions in **HEURISTIC 1** i.e., $I(a_{3,2})$ and $I(a_{1,0})$ by $a_{3,2}, a_{1,0} \in B$, and $B \in \mathcal{S}$, is stable as below.

$$\sum_{b\in B} f_i^w(b) \geq 3+1=4 \geq \left(\varphi\theta n_i^w = 0.1 * 2 * 20 = 4\right) \tag{9}$$

$$\text{Dis\_value}(B) = \frac{\sum_{b\in B} f_i^w(b)}{\sum_{b\in B} f_j^w(b)} * \frac{n_j^w}{n_i^w} = \frac{(3+1)}{(2+0)} * \frac{20}{20} = \frac{4}{2} \geq (\theta = 2) \tag{10}$$

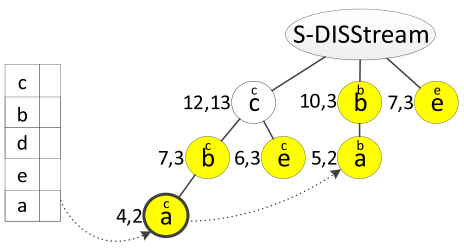


Fig. 7 Updated *S-DISStream* after processing *Stable*$(Subtree_c)$ in conditional *FP-Tree* for *Header-Table* item a in Fig. 6

In this paper, for the sake of simplicity, the dataset lengths are omitted from ratios as $n_1 = n_2$. In the case of data streams with different lengths (i.e., $\frac{n_2}{n_1} \neq 1$), the ratios must be multiplied by the constant of $\frac{n_2}{n_1}$. The conditional *FP-Tree* of *Header-Table* item a made out of *S-FP-Tree* updated with partition P_2 is presented as in Fig. 6.

The *Stable*$(Subtree_{root})$ is ignored from the new itemset combination generation. The current itemset combinations of a *Stable*$(Subtree_{root})$ in $Subtree_{root}$ of *S-DISStream* are traversed using *Header-Table* links and tagged as discriminative or non-discriminative based on the recent data stream lengths in the sliding window frame W; (e.g., in *Stable*$(Subtree_c)$ the itemset $I(a_{4,2})$ i.e., $cba_{4,2}$ in $Subtree_c$ of *S-DISStream* ending with *Header_Table_items*$(Subtree_c)$, is tagged as discriminative as in Fig. 7).

Potential and stable Internalnode$_{\text{root}}$

A *Potential*(in) in a *Potential*$(Subtree_{root})$ (i.e., $in \in Internalnode_{root}$) in the conditional *FP-Tree* in the sliding window frame W satisfies two conditions i.e., Max_freq$_i(root, in, a) \geq \varphi\theta n_i^w$ and Max_dis_value$(root, in, a) \geq \theta$, where *itemsets*$(root, in, a)$ is a set of itemsets in $Subtree_{root}$ ending with a header item $a \in Header\_Table\_items(Subtree_{root})$ with the internal node in as subset.

Let $\mathcal{S}$ be the power set of $Internalnode_{root}$, i.e., $\mathcal{S} = 2^{Internalnode_{root}}$, and itemset I, with the subset of $in \in Internalnode_{root}$, ending with $a \in Header\_Table\_items(Subtree_{root})$ denoted as $I(in)$. The frequency of each itemset in $Subtree_{root}$ with the subset of the internal node in, in respect of the data stream S_i in the sliding window frame W, is defined below (i.e., $B \in \mathcal{S}$).

$$f_i^w(B) = \sum_{b\in B} f_i^w(b) \tag{11}$$

The frequency of an itemset in a subtree is stable if all $b \in B$ are stable during the offline sliding window. A potential internal node $in \in Internalnode_{root}$ is stable if all potential discriminative itemsets in $Subtree_{root}$ with the internal node in as subset are stable during the offline sliding window.

All discriminative itemsets in a $Subtree_{root}$ with a subset of a stable in are stable in the sliding window frame W compared to the current state in *S-DISStream*. A potential internal node $in \in Internalnode_{root}$ can be non-potential before offline sliding with different data stream lengths in the sliding window frame W, and new discriminative itemsets may be discovered in the sliding window frame W; (e.g., by decreasing the length of the target data stream S_i or increasing the data streams' length ratio). A *Potential*(in) that satisfies the conditions Max_freq$_i(root, in, a) \geq \varphi\theta n_i^w$ and Max_dis_value$(root, in, a) \geq \theta$ with smaller frequency in the target data stream S_i, or smaller frequency ratio in

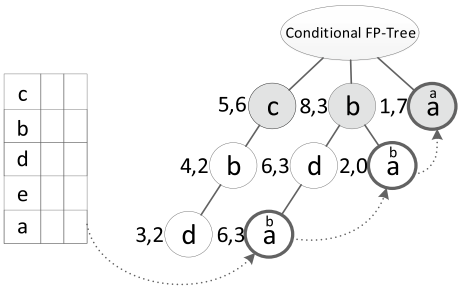


Fig. 8 Expanded conditional *FP-Tree* of *Header-Table* item *a* updated by P_2 after processing the first subtree

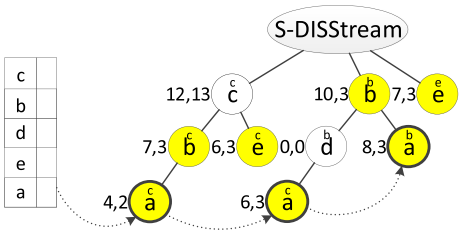


Fig. 9 Updated *S-DISStream* after processing potential discriminative subsets of the left-most subtree in conditional *FP-Tree* for *Header-Table* item *a*

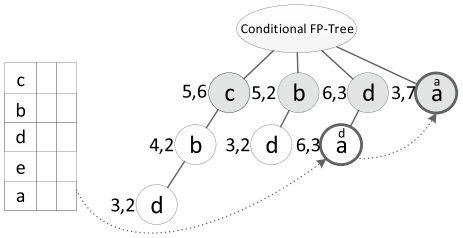


Fig. 10 Expanded conditional *FP-Tree* of *Header-Table* item *a* updated by P_2 after processing the second subtree

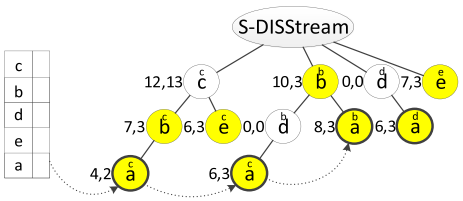


Fig. 11 Updated *S-DISStream* after processing potential discriminative subsets of the left-most subtree in conditional *FP-Tree* for *Header-Table* item *a*

target data stream S_i vs general data stream S_j, compared to the last size of sliding window frame W (i.e., last offline window sliding) is not stable.

The potential internal node $in \in Internalnode_{root}$ in the sliding window model is processed differently.

HEURISTIC 2. An internal node $in \in Internalnode_{root}$ is stable denoted as $Stable(in)$ if all potential discriminative itemsets in $Subtree_{root}$ with the internal node in as subset that satisfy the following conditions are stable.

$$\text{Max\_freq}_i(root, in, a) \geq \varphi\theta n_i^w$$
$$\text{Max\_dis\_value}(root, in, a) \geq \theta$$

where $\theta > 1$ is the discriminative level threshold, $\varphi\epsilon(0, 1/\theta)$ is the support threshold, n_i^w is the size of target data stream S_i in the sliding window frame W and $\text{Max\_freq}_i(root, in, a)$ and $\text{Max\_dis\_value}(root, in, a)$ are stable if any itemset in $Subtree_{root}$ with the internal node in as subset that satisfies the two conditions is stable.

Lemma 2 (Stable internal node) **HEURISTIC 2** *ensures that any discriminative itemset in* $Subtree_{root}$ *with a subset of a* $Stable(in)$ *is stable in the sliding window model.*

Proof The two conditions in **HEURISTIC 2** ensure that any itemset with the subset of internal node $in \in Internalnode_{root}$ that is frequent in the target data stream S_i and has a discriminative value larger than the discriminative level θ is stable in the sliding window model. This implies that all discriminative itemsets with the subset of $Stable(in)$ exist in the sliding window model by processing the $Potential(in)$ in a potential $Subtree_{root}$ during a previous offline sliding of the window. The itemset combinations with a $Stable(in)$ are tagged as discriminative or non-discriminative in the $Subtree_{root}$ of *S-DISStream*, based on the recent data stream lengths in the sliding window frame W. This implies that the discriminative itemsets are discovered based on the recent data stream lengths that have been changed by adding the new partition and deleting the oldest partition in the sliding window frame W.

For a $in \in Internalnode_{root}$, if any of the two conditions is updated the internal node is considered as a potential discriminative internal node and the potential discriminative itemset combinations with the subset of the internal node are generated from the subtree, as it may contain new discriminative itemsets.

Every itemset in a $Stable(in)$ is stable in the sliding window model; for example, in Fig. 6, in the left-most subtree related to the processing *Header-Table* item *a* under root node *c* (i.e., $Subtree_c$), the internal node *b* is stable as the only subset of items in the subtree that satisfy the conditions in **HEURISTIC 2** i.e., made of $I(b_{3,2})$ and $I(b_{1,0})$ as $b_{3,2}, b_{1,0} \in B$ and $B \in \mathcal{S}$, is stable. The $Stable(in)$ is ignored from the new itemset combination generation. The current itemset combinations with a subset of $Stable(in)$ in $Subtree_{root}$ of *S-DISStream* are traversed using *Header-Table* links and tagged as discriminative or non-discriminative based on the recent data stream lengths in the sliding window frame W;

(e.g., in Fig. 7 the itemset $cba_{4,2}$ in $Subtree_c$ of *S-DISStream* ending with $Header\_Table\_items(Subtree_c)$ with the subset of internal node *b*, is tagged as discriminative).

Following the running Example 1, the conditional *FP-Tree* of *Header-Table* item *a* is expanded by sub-branches of $Subtree_c$ (i.e., $bda_{3,2}$, $ba_{1,0}$, and $a_{1,5}$) as in Fig. 8. The $Potential(Subtree_b)$ and its $Potential(in)$ are not stable and *S-DISStream* is updated by new itemset combinations generated out of the potential $Subtree_b$ as in Fig. 9. The discriminative itemsets (i.e., $bda_{4,2}$ and $ba_{8,3}$) are discovered in $Subtree_b$. The potential discriminative itemsets may exist in the *S-DISStream* out of processing the old partitions and be overwritten; (e.g., in Fig. 9 the itemset $ba_{5,2}$ exists in *S-DISStream* out of processing P_1 as in Fig. 4 and is overwritten by the new frequencies as $ba_{8,3}$).

The conditional *FP-Tree* of *Header-Table* item *a* is then expanded by sub-branches of $Subtree_b$ (i.e., $da_{6,3}$ and $a_{2,0}$) as in Fig. 10. The $Potential(Subtree_d)$ is not stable and *S-DISStream* is updated by the discriminative itemset $da_{6,3}$ as in Fig. 10.

The two proposed heuristics significantly decrease search space by ignoring the non-potential or potential but stable itemset combinations. By limiting the search space which accordingly causes smaller tree structures and shorter updating time, both time and space complexity decrease significantly (e.g., in the experiment section) (Fig. 11).

Following the bottom-up order of *Desc-Flist*, the conditional *FP-Tree* is then generated for the rest of *Header-Table* items respectively (i.e., item *e* in Example 1 as in Table 2). The new discriminative itemsets in each potential $Subtree_{root}$ are inserted into *S-DISStream*. The tag of itemsets in *S-DISStream* that belongs to the $Stable(Subtree_{root})$ and $Stable(in)$ are updated based on the recent data stream lengths in the sliding window frame *W*. The frequencies of itemsets in *S-DISStream* that are not updated (i.e., belong to the non-potential $Subtree_{root}$ or with a subset of non-potential $in \in Internalnode_{root}$) must be adjusted based on their appearances in *S-FP-Tree* followed by updating the tag of itemsets.

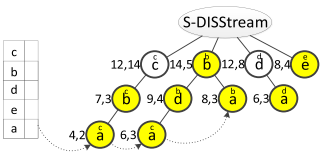
S-DISStream tuning and pruning in the offline sliding window

Against the *Apriori* property and distinguishing the discriminative itemset mining from frequent itemset mining, the non-discriminative itemsets appear as a subset of discriminative itemsets; (e.g., the items *c* and *d* in Example 1 are subsets of discriminative itemsets as in Fig. 12, but they are not discriminative). The frequencies of non-discriminative itemsets appearing as a subset of discriminative itemsets must also be set accordingly using the *S-FP-Tree*. These itemsets may become involved in the online sliding window, as explained in "Incremental Offline Sliding Window".

Lemma 3 (Exact non-discriminative subsets) *tuning the frequencies of the non-discriminative itemsets appearing as a subset of discriminative itemsets using S-FP-Tree ensures the exact frequencies of these itemsets in S-DISStream that may become involved in the online window updating.*

Proof The *S-FP-Tree* is the superset of conditional *FP-Trees* and has a full view of all itemsets in the datasets in the sliding window frame *W*. The exact frequencies of the non-discriminative subsets are collected accurately using their appearances in *S-FP-Tree* by traversing through *Header-Table* links.

The tail pruning in *S-DISStream* is applied for space saving. The itemset in *S-DISStream* in the sliding window model is pruned if it is non-discriminative and stays as a leaf node. The tail pruning ensures that *S-DISStream* only maintains the discriminative itemsets and the non-discriminative subsets in the sliding window frame *W*. The final *S-DISStream* after offline sliding by P_2 as in Table 3 and tail pruning is presented in Fig. 12 with the eight listed discriminative itemsets. The tail pruning is applied in the *S-FP-Tree* following the same process for deleting the old transactions that are out of the sliding window frame *W* with zero frequencies in data streams.



Itemsets	*Frequency*
{*c b a*}	(4,2)
{*c b*}	(7,3)
{*b d a*}	(6,3)
{*b a*}	(8,3)
{*b d*}	(9,4)
{*b*}	(14,5)
{*d a*}	(6,3)
{*e*}	(8,4)

Fig. 12 Final *S-DISStream* after offline sliding based on P_2

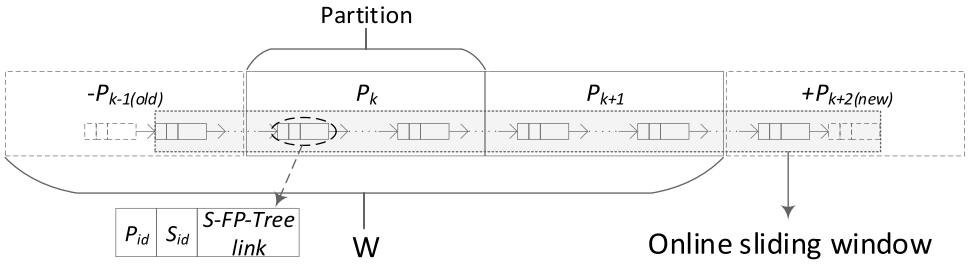


Fig. 13 *Transaction-List* made of partitions fit in the online sliding window frame *W*

Online Sliding Window

Transaction-List: This is a queue structure for keeping track of the transactions in the online sliding window as in Fig. 13. For each transaction that fits in the online sliding window frame W, it holds the partition number; data stream identifier, and a link to the transaction in the *S-FP-Tree* node. The *Transaction-List* contains only the recent transactions that fit in the defined online sliding window frame W. The rest of the transactions are deleted when the window frame W slides as in Fig. 13.

The online sliding window is happening within two offline sliding of the window model limited to the itemsets in *S-DISStream* (i.e., the online sliding window frame W). The itemsets in the *S-DISStream* are the potential itemsets to change their tags during the online sliding window. The frequency and tag of the itemsets existing in *S-DISStream* are updated during online sliding and no new itemset is generated. Each transaction updates the *S-FP-Tree* and is linked by *Transaction-List* for online sliding. During online sliding, every new transaction in the recent partition (i.e., P_{new} as in Fig. 1) in the window frame W is checked for having a subset in the *S-DISStream* by traversing through *Header-Table* links. Subsets of a new transaction that exist in *S-DISStream*, called online itemsets, are used for online sliding by increasing the itemset frequencies and updating the tags in the *S-DISStream*; (e.g., a discriminative itemset may become non-discriminative).

By each new incoming transaction, the oldest transaction in the *Transaction-List* is deleted by its online itemsets out of the sliding window frame W if it belongs to the oldest partition (i.e., P_{old} as in Fig. 1). Online itemsets of the old transaction (i.e., subsets exist in *S-DISStream*) are used for online sliding by decreasing the itemset frequencies and updating the tags in the *S-DISStream*. The online sliding continues for every new transaction until the end of the new partition. The itemsets in the *S-DISStream* that are updated during online sliding are tagged as online. The online itemsets in the *S-DISStream* hold the exact frequencies in the sliding window frame W, however, they must be tagged after offline sliding based on the recent data stream lengths during *S-DISStream* tuning and pruning.

The **HEURISTIC 1** and **HEURISTIC 2** are modified based on the relaxation of α for holding the sub-discriminative itemsets in the sliding window frame W. The sub-discriminative itemsets are the potential discriminative itemsets and are saved in the sliding window model following Definition 2 and based on the relaxation of α.

Property 1. *By modifying* **HEURISTIC 1** *and* **HEURISTIC 2** *based on the relaxation of α the sub-discriminative itemsets will be obtained.*

This property says that a set of non-discriminative itemsets are discovered as the sub-discriminative itemsets by choosing the relaxation of α.

Property 2. *By setting a smaller relaxation of α, the discriminative itemsets in the online sliding window will be discovered with a better approximation.*

This property says by choosing the smaller relaxation of α, more sub-discriminative itemsets are discovered. More numbers of sub-discriminative itemsets in the online sliding window i.e., *S-DISStream* structure, lead to a greater number of discovered discriminative itemsets in the online sliding window model.

Corollary 1. *By modifying the* **HEURISTIC 1** *and* **HEURISTIC 2** *based on the relaxation of α, a refined approximate bound in discriminative itemsets in the online sliding window is obtained.where α is the relaxation threshold for sub-discriminative itemsets and* **HEURISTIC 1** *and* **HEURISTIC 2** *are defined for potential discriminative itemset combination generation during the offline sliding window.*

It has to be considered there is a limit to improving the approximation in the online sliding window. This approximation is highly related to the concept drift that exists in data streams. This means that we can only improve the approximation at a specific level as we will see this in the experiments.

Rational 1. (the highest refined approximate bound in discriminative itemsets in the online sliding window) Corollary 1 ensures that the approximation in discriminative itemsets in the online sliding window may be improved by holding the sub-discriminative itemsets in the *S-DISStream* structure in an online sliding window.

Proof. The sub-discriminative itemsets improve the approximate bound in discriminative itemsets by increasing the number of potential discriminative itemsets under the relaxation of α. We call this the highest refined approximate bound in discriminative itemsets in the online sliding window. Corollary 1 is more efficient when the discriminative itemsets are stable in the neighbor partitions with fewer concept drifts present in the datasets. Based on this, we do not have any false-positive answers as all the reported discriminative itemsets in the online sliding window are discriminative. However, there may be a large number of false negative answers as they are not involved in the online sliding window. These false negatives will be corrected accurately in the offline sliding window.

S-DISSparse Method

The *S-FP-Tree* is updated by adding the transactions from the recent batch of transactions B_n (i.e., the most current batch of transactions) fits in partition P_{new} without pruning infrequent items and by making the *Transaction-List*. The first partition is processed using the *DISSparse* algorithm proposed in [6, 7], and the *S-DISStream* is generated from discriminative itemsets and non-discriminative subsets in transactions fitting in partition P_1. With every new transaction in P_{new}, the *Transaction-List* is updated. The online sliding window is updated by online itemsets in *S-DISStream* (i.e., increasing frequency of online itemsets in P_{new} and decreasing frequency of online itemsets in P_{old}). The online itemsets in the *S-DISStream* are tagged as discriminative or non-discriminative based on their updated frequencies and data stream lengths. By the end of the online sliding of P_{new} the P_{old} is checked for online sliding of the remaining transactions (i.e., when P_{old} has a larger number of transactions than P_{new}).

During offline sliding, the *S-DISStream* is updated by the discriminative itemsets in each *Potential*($Subtree_{root}$) and its *Potential*(*in*) in an offline state. The tags in $Subtree_{root}$ in *S-DISStream* are updated by checking the itemsets in *Stable*($Subtree_{root}$) *Stable*(*in*) in *S-DISStream* based on the recent data stream lengths. By the end of offline sliding of P_{new} the exact frequencies of the non-discriminative subsets not updated in *S-DISStream* are tuned based on their appearance in the *S-FP-Tree* and tail pruning is applied in *S-DISStream* and *S-FP-Tree* structures. The online itemsets in *S-DISStream* are also tagged based on the recent data stream lengths and the process continues by the next partition. The discriminative itemsets are reported in offline time intervals DI_{ij}^{W}.

Algorithm 2. (*S-DISSparse:* **Mining Discriminative Itemsets in Data Streams using Efficient Sliding Window)**

Input: (1) The discriminative level θ; (2) The support threshold φ; (3) The relaxation of α; and (4) incoming batches of transactions fit in partitions P belonging to data streams S_i and S_j.

Output: DI_{ij}^{W}, discriminative itemsets in S_i against S_j in the sliding window frame W (*S-DISStream* structure) in online and offline states.

Begin

1. Make *S-FP-Tree* based on B_1 fits in P_1 and update *Transaction-List*;
2. Process P_1 using *DISSparse* algorithm and make *S-DISStream*;
3. **While** not end of *streams* **do**
4. Untag *S-FP-Tree* and *S-DISStream*;
5. **While** not end of partition P_{new} **do** // *Online sliding window*
6. Update *S-FP-Tree* and *Transaction-List* by the new transaction;
7. **If** the added transaction in P_{new} in W has an online itemset **then**
8. Update online itemsets in *S-DISStream*; // *add frequency*
9. **If** the deleted transaction in P_{old} in W has an online itemset **then**
10. Update online itemsets in *S-DISStream*; // *delete frequency*
11. **End While;**
12. Delete remained transactions in P_{old} in an online state;
13. Update *S-DISStream* by discriminative itemsets in every $Potential(Subtree_{root})$ and $Potential(in)$;
14. Update tags in $Subtree_{root}$ in *S-DISStream* for itemsets in every $Stable(Subtree_{root})$ and $Stable(in)$ based on **HEURISTIC 1** and **HEURISTIC 2** modified by **Corollary 1**;
15. Tune non-discriminative subsets and tag online itemsets in *S-DISStream* and apply tail pruning;
16. Report discriminative itemsets DI_{ij}^{W} in sliding window frame W;
17. **End while;**

End.

In the *S-DISSparse* algorithm, the significant part attracting considerable complexity is related to generating the potential discriminative itemsets and updating the tag of stable itemsets in the *S-DISStream* structure. Tuning the frequencies of non-discriminative subsets in the *S-DISStream* and applying the tail pruning in *S-DISStream* and *S-FP-Tree* have less complexity by considering the sparse discriminative itemsets. The online sliding in lines 5 to line 12 is based on a quick search method on *S-DISStream* structure. The offline sliding is based on the potential subtrees and the potential internal nodes i.e., updated itemsets. The stable itemsets in *S-DISStream* structure are also checked based on a quick search method and tagged as discriminative or non-discriminative.

Theorem 1 (Completeness and correctness of *S-DISSparse*): *Based on* [6, 7], *the DISSparse method discovers the exact set of discriminative itemsets in offline sliding states. Based on Lemma 1 and Lemma 2, the updated potential discriminative itemsets in each potential* $Subtree_{root}$ *in conditional FP-Tree and its* $Internalnode_{root}$ *are generated completely in S-DISStream, and all stable discriminative itemsets are tagged in S-DISStream correctly based on the recent data stream lengths. Based on* **Rational 1**, *the frequencies of non-discriminative itemsets that appear as a subset of discriminative itemsets are collected accurately. These prove the completeness and correctness of the S-DISSparse method by discovering all the discriminative itemsets and their non-discriminative subsets in the offline sliding window.*

Experimental Results

We implemented the algorithms in C++ and ran all the experiments on a desktop computer with an Intel Core (TM) Duo E2640 2.8 GHz CPU and 8 GB main memory running 64-bit Microsoft Windows 7 Enterprise. The data streams made of different numbers of transactions were generated using the IBM synthetic data generator [30]. The synthetic datasets are represented by the format $T\$: I\$: D\$$. Here, T is referred to as the average transaction length, I referred to as the average length of the large itemsets, and D is referred to as the number of transactions. Considering both data streams from the same domain, we used the same T for S_1 and S_2. we generated the S_1 and S_2 with different I. The reason is that there are more maximal potentially large itemsets in S_2 as it is made of several smaller datasets. Moreover, this setting will ensure there is a large enough number of discriminative itemsets in S_1 against S_2. The first dataset called D_1, is generated with S_1 as $T25 : I10 : D60K$ and S_2 as $T25 : I15 : D300K$ limited to $10K$ unique items. To show the unsynchronized behaviour of the data streams in the online sliding window, we made a simple code to mix the two data streams based on their ratio size. The data streams in D_1 are modelled as 30 continuous batches in the same sizes (i.e., for the sake of clarity) with $T25 : I10 : D2K$ and $T25 : I15 : D10K$ belonging to the S_1 and S_2, respectively. The ratio between the size of S_1 and S_2 in D_1 is the same for all 30 batches (i.e., $n_2/n_1 = 5$).

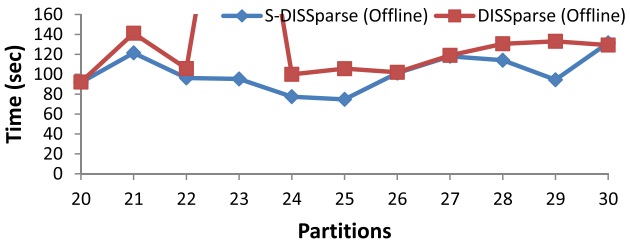


Fig. 14 Time complexity for D_1 (window frame $W = 25$)

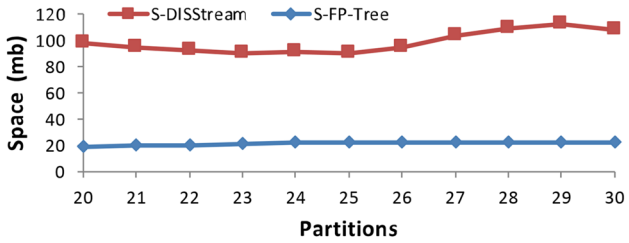


Fig. 15 Space use in offline sliding for D_1 ($W = 25$)

Time and Space Efficiency

In this section, during all experiments, the discriminative level $\theta = 25$ and support threshold $\varphi = 0.002\%$. In the

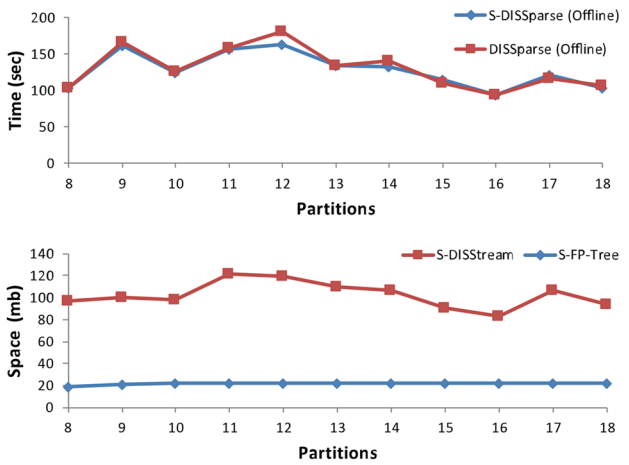


Fig. 16 Time and space use for D_2 ($W = 10$)

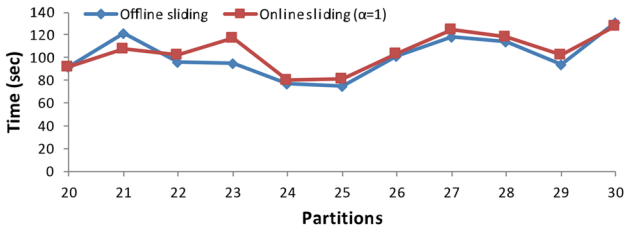


Fig. 17 Time use in online and offline sliding for D_1 ($W = 25$)

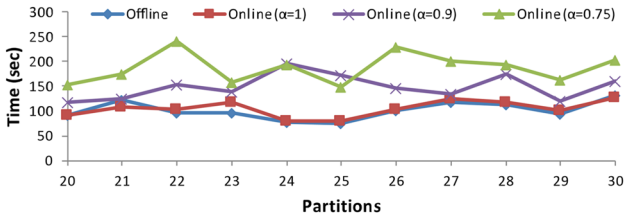


Fig. 18 Time use for online and offline sliding for D_1 ($W = 25$) with different relaxation of α

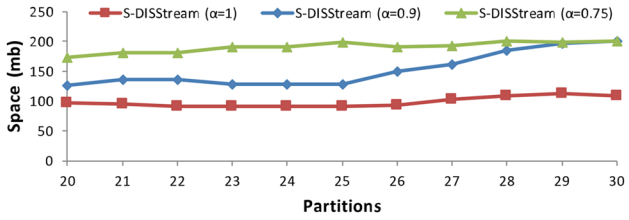


Fig. 19 *S-DISStream* size for D_1 ($W = 25$) by different relaxation of α

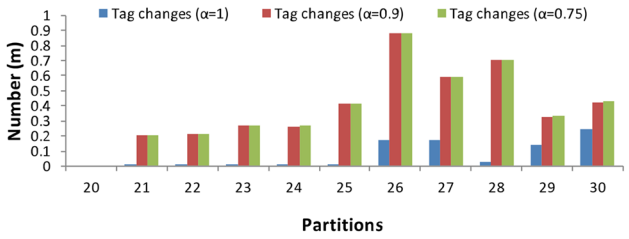


Fig. 20 Number of itemsets that their tag is changed for D_1 ($W = 25$) by different relaxation of α

experiments with D_1, the 25 recent partitions are fitted in the sliding window frame W (i.e., $W = 25$).

The time complexity of the *S-DISSparse* method in the offline updating sliding windows by each partition in D_1, is presented in Fig. 14. The sliding window frame W is initialized by the first 20 partitions in D_1, for the sake of clarity. This is recommended as there may be a large number of discriminative itemsets discovered by processing the initial partitions with small data stream lengths. The sliding window model is then updated in an offline state by every new partition that is fitted in the window frame W. To the best of our knowledge, the *DISSparse* [6, 7] and *H-DISSparse* [8] are the only research works presented for mining discriminative itemsets. The *H-DISSparse* is an approximate method working based on the tilted-time window model and has substantial differences from our proposed *S-DISSparse* method. However, *DISSparse* is an efficient and exact method of working on a single batch of a static dataset. This can be applied as the baseline by processing the whole data stream sizes, as a single static batch of datasets, fitted in the window frame W. The scalability of the *S-DISSparse* algorithm is compared with *DISSparse* algorithm [6, 7] for mining discriminative itemsets in a batch of transactions. The *DISSparse* algorithm is used as a benchmark by processing the full-size window frame W after each offline sliding.

The efficiency of *S-DISSparse* is generally better in comparison to the efficiency of *DISSparse* in the offline sliding window. The time complexity of the offline window sliding using *S-DISSparse* by partition P_{23} shows the most efficiency (i.e., 450 sec in *DISSparse* vs 100 sec in *S-DISSparse*). In this partition, there is concept drift in the data streams showing insensitivity of the *S-DISSparse* to the concept drifts. There are some partitions with the least efficiency as compared with *DISSparse*; (e.g., the offline window sliding by partitions P_{26} and P_{30} shows similar time use in both algorithms). The *S-DISSparse*, in these points, scales the same as *DISSparse* as most of the transactions in the sliding window model are updated during offline sliding by these partitions.

The space use of the *S-DISStream* and *S-FP-Tree* is reported during offline sliding as the largest datasets used in *S-DISSparse* algorithm in Fig. 15.

Following the compact prefix tree structure and by applying the tail pruning, the *S-DISStream* size stays small as the in-memory data structure. The number of discriminative itemsets discovered in the sliding window frame W in the designed experiments is between 2.5 and 3 million (i.e., as in Fig. 21) by considering the different distributions of transactions during the sliding window model.

To show the effects of the different window sizes W, we run the *S-DISSparse* algorithm with the second dataset D_2 which has bigger size partitions. The second dataset called D_2 is generated with S_1 as $T25 : I10 : D90K$ and S_2 as $T25 : I15 : D450K$ limited to $10K$ unique items. The data streams in D_2 are modelled as 18 continuous batches in the same sizes with $T25 : I10 : D5K$ and $T25 : I15 : D25K$ belonging to the S_1 and S_2, respectively. The size of the sliding window frame W is set to the same number of transactions as in the first experiment (i.e., $W = 10$). The sliding window frame W is initialized by the first 8 partitions in D_2. The time and space complexity of the *S-DISSparse* method in the offline updating sliding window by each partition in D_2 is presented in Fig. 16.

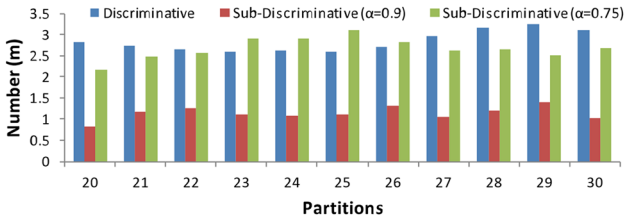


Fig. 21 Number of discriminative and sub-discriminative itemsets for D_1 ($W = 25$) by different relaxation of α

The greater number of transactions in the sliding window frame W is updated by the bigger size partitions. This has to be considered for adjusting a proper sliding window frame size based on the application domains and data stream characteristics. The sliding window frame should be set much bigger in comparison to the average size of a single partition.

The time complexity of the *S-DISSparse* method by offline and online processing in the dataset D_1 is presented

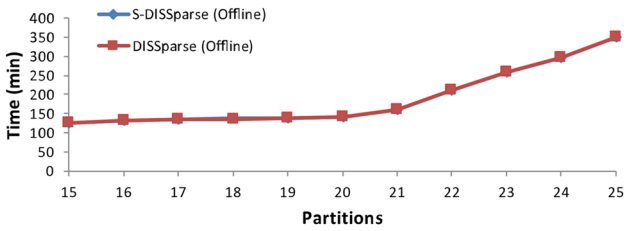


Fig. 22 Time use for Susy dataset (window frame $W = 20$)

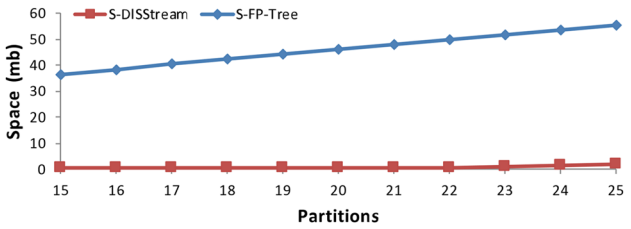


Fig. 23 Space use in offline sliding for Susy dataset ($W = 20$)

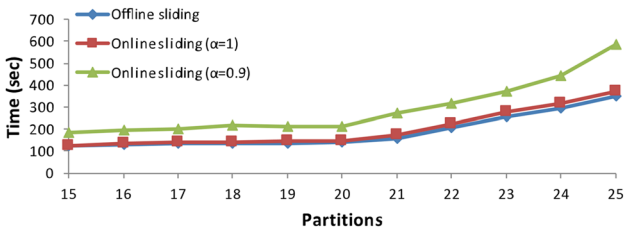


Fig. 24 Time use in online and offline sliding for Susy dataset ($W = 20$)

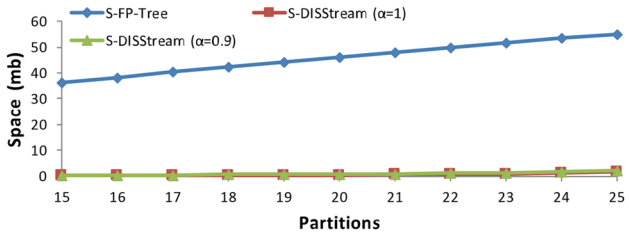


Fig. 25 *S-DISStream* size for Susy dataset ($W = 20$) by different relaxation of α

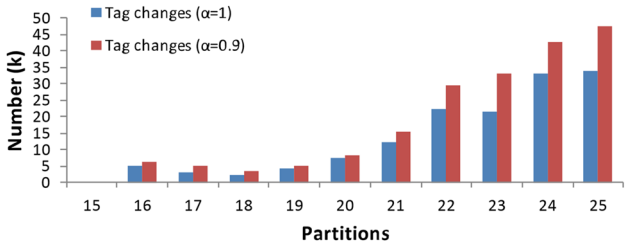


Fig. 26 Number of itemsets that their tag is changed for Susy dataset ($W = 20$) by different relaxation of α

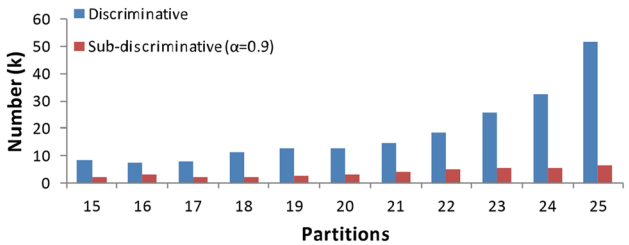


Fig. 27 Number of discriminative and sub-discriminative itemsets for Susy dataset ($W = 20$) by different relaxation of α

in Fig. 17. The non-discriminative subsets (of discriminative itemsets) in *S-DISStream* are updated during online sliding, causing decreasing the time complexity of the *S-DISSparse* algorithm during tuning the frequencies of the non-discriminative subsets in *S-DISStream*.

To improve the approximation of the discriminative itemsets in the online updating sliding window, we set the relaxation of $\alpha = 0.9$ and $\alpha = 0.75$, respectively. The *S-DISSparse* time complexity during offline sliding, online sliding with $\alpha = 1$ (i.e., no sub-discriminative itemsets), and online sliding with $\alpha = 0.9$ and $\alpha = 0.75$, are represented in Fig. 18. The *S-DISSparse* time complexity with $\alpha = 0.75$ is more sensitive to the concept drifts.

The space use of the *S-DISStream* is presented during offline sliding by $\alpha = 1$, $\alpha = 0.9$, and $\alpha = 0.75$ in the *S-DISSparse* algorithm in Fig. 19.

The number of itemsets in which their tag is changed during online sliding (e.g., discriminative to non-discriminative itemsets or non-discriminative to discriminative itemsets) is presented in Fig. 20.

The exact number of discriminative itemsets after each offline sliding window is represented in Fig. 21. The number of sub-discriminative itemsets is also represented by different relaxation of α (i.e., $\alpha = 0.9$ and $\alpha = 0.75$). This shows that choosing smaller relaxation of α may not always improve the approximation in discriminative itemsets in the online sliding window.

Evaluation of Real Datasets

As a real dataset, we used part of the Susy dataset from the UCI repository provided in [31]. The Susy is a dense dataset (i.e., there are no missing values and transactions have values for each attribute), and it has no sparsity characteristics compared to the synthetic datasets. The original dataset is made of five million instances with the first column being the class label followed by eighteen features. Each transaction is made of 190 unique items. We selected the 25 batches each made of twenty thousand instances.

In this section, during all experiments, the discriminative level $\theta = 2.5$ and support threshold $\varphi = 1\%$. In the experiments with Susy dataset, the 20 recent partitions are fitted in the sliding window frame W (i.e., $W = 20$). The sliding window frame W is initialized by the first 15 partitions in the Susy dataset for the sake of clarity. The time complexity of the *S-DISSparse* method in the offline updating sliding window by each partition in the Susy dataset is presented in Fig. 22.

In the Susy dataset, the efficiency of *S-DISSparse* is not better compared to *DISSparse*. Susy has a small number of unique items i.e., 190 items. The sparsity characteristic of the dataset is limited and the subsets are updated by every batch coming in or going out of the sliding window model.

The space use of the *S-DISStream* and *S-FP-Tree* is reported during offline sliding in Fig. 23.

The online sliding does not add high time complexity to the *S-DISSparse* algorithm as in Fig. 24.

The space use of the *S-DISStream* is presented during offline sliding by $\alpha = 1$ and $\alpha = 0.9$ as in Fig. 25.

The number of itemsets in which their tag is changed during online sliding (e.g., discriminative to non-discriminative or vice versa) is presented in Fig. 26.

The exact number of discriminative itemsets after each offline sliding window is represented in Fig. 27. The number of sub-discriminative itemsets is also represented by different relaxation of α (i.e., $\alpha = 0.9$).

Moreover, with the highly parallelized big data technologies in distributed computing, we may achieve a more efficient solution. Our algorithm, on large datasets, would be more scalable such as the proposed work in [32]. This will make the expansion of the algorithm for data stream mining more efficient.

Practical Applications

There are more application scenarios in biomedical image analysis, such as X-ray image analysis [33], Histopathology image analysis [34], Cell image analysis [35], Microorganism image analysis [36], Caner image analysis [37], etc.

The X-ray image analysis [33] has been discovered as a fast and more sensitive alternative screening method with visual indicators for COVID-19 viral infection early diagnosis. Based on the studies, patients affected with COVID-19 present deformities in chest radiographs, and the imaging tool is considered a fast identification of suspected patients in the epidemic area. The computer-aided diagnostic (CAD) can help track COVID-19 infection more efficiently and accurately. Histopathology image analysis [34] enables the automatic global detection of gastric cancer images and demonstrates high global detection performance. Cell image analysis [35] by cervical cytopathology image classification is an important effective method based on deep learning to diagnose cervical cancer. Microorganism image analysis [36] is for the *Environmental Microorganism* (EM) image segmentation task to assist microbiologists in detecting and identifying EMs more effectively. Cancer image analysis [37] is a widely performed screening technique for the early detection of cervical cancer. It uses various deep learning models to capture more potential information to enhance classification performance. Mining discriminative (contrast) patterns in computer-aided diagnostic (CAD) can enhance the performance of the above methods regarding the challenges of low-contrast images and insufficient annotated datasets.

Conclusion and Future Works

In this paper, we proposed an efficient single-pass algorithm for mining discriminative itemsets over data streams using the sliding window model. The algorithm uses two in-memory data structures called *S-FP-Tree* and *S-DISStream*. The offline sliding uses two heuristics based on the *S-FP-Tree* nodes status (i.e., stable or updated during window model sliding) within efficient time and space use. The online sliding is happening between two offline slidings. Empirical analysis shows the efficiency of offline sliding with the accuracy of online sliding. Setting a small size partition compared to the sliding window frame can improve the efficiency in periodic offline sliding. The online sliding by new transactions does not add to the time complexity. The

number of discriminative itemsets generated is significantly less in comparison to frequent itemsets making them more useful for discrimination. In the future, we plan to develop methods of discovering discriminative rules using discriminative itemsets and propose a classifier focusing on distinguishing features of data streams.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data availability The datasets generated and analysed during the current study are available from the corresponding author upon reasonable request.

Declaration

Conflict of interest The research fund of the Queensland University of Technology supported this research.

References

1. Garofalakis M, Gehrke J, Rastogi R. Querying and mining data streams: you only get one look a tutorial, in Proceedings of the 2002 ACM SIGMOD international conference on Management of data. 2002, ACM: Madison, Wisconsin. p. 635–635.
2. Manku GS. Frequent itemset mining over data streams, in *Data Stream Management: Processing High-Speed Data Streams*, M. Garofalakis, J. Gehrke, and R. Rastogi, Editors. 2016, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 209–219.
3. Lin Z, et al. Mining discriminative items in multiple data streams. World Wide Web. 2010;13(4):497–522.
4. Seyfi M. Mining discriminative items in multiple data streams with hierarchical counters approach. in Fourth International Workshop on Advanced Computational Intelligence (IWACI), 2011. 2011. IEEE.
5. Seyfi M, Geva S, Nayak R. Mining discriminative itemsets in data streams. in International Conference on Web Information Systems Engineering. 2014. Springer.
6. Seyfi M, et al. Efficient mining of discriminative itemsets, in Proceedings of the International Conference on Web Intelligence. 2017, ACM: Leipzig, Germany. p. 451–459.
7. Seyfi M, et al. DISSparse: Efficient mining of discriminative itemsets. J Inf Knowl Manag. 2022;21(01):2250009.
8. Seyfi M, et al. Mining discriminative itemsets in data streams using the tilted-time window model. Knowl Inf Syst. 2021;2:1–30.
9. Chang JH, Lee WS. estWin: online data stream mining of recent frequent itemsets by sliding window method. J Inf Sci. 2005;31(2):76–90.
10. Cheng J, Ke Y, Ng W. A survey on algorithms for mining frequent itemsets over data streams. Knowl Inf Syst. 2008;16(1):1–27.
11. Li J, Dong G, Ramamohanarao K. Instance-based classification by emerging patterns. In: Principles of data mining and knowledge discovery. Springer; 2000. p. 191–200.
12. Chi Y, et al. Moment: Maintaining closed frequent itemsets over a stream sliding window. in Fourth IEEE International Conference on Data Mining ICDM '04. 2004.
13. Dong G, Bailey J. Contrast data mining: concepts, algorithms, and applications. Boca Raton: CRC Press; 2012.
14. Dong G, Li J. Efficient mining of emerging patterns: discovering trends and differences. in Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. 1999.
15. Alhammady H, Ramamohanarao K. Mining emerging patterns and classification in data streams. The Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence, 2005: p. 272–275
16. Bailey J, Loekito E. Efficient incremental mining of contrast patterns in changing data. Inf Process Lett. 2010;110(3):88–92.
17. Li J, Liu G, Wong L. Mining statistically important equivalence classes and delta-discriminative emerging patterns. in Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. 2007. ACM.
18. He Z, et al. Conditional discriminative pattern mining. Inform Sci. 2017;375:1–15.
19. Leonardo P, Fabio V. Efficient mining of the most significant patterns with permutation testing. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018, ACM: London, United Kingdom. p. 2070–2079.
20. He Z, et al. Mining conditional discriminative sequential patterns. Inf Sci. 2019;478:524–39.
21. Manku GS, Motwani R. Approximate frequency counts over data streams. In Proceedings of the 28th international conference on Very Large Data Bases. 2002. VLDB Endowment.
22. Lee C-H, Lin C-R, Chen M-S. Sliding-window filtering: an efficient algorithm for incremental mining. In: Proceeding of the 10th Int'l Conference on Information and Knowledge Management. 2001.
23. Chi Y, et al. Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. Knowl Inf Syst. 2006;10(3):265–94.
24. Leung CK-S, Khan QI. DSTree: a tree structure for the mining of frequent sets from data streams. In Sixth International Conference on Data Mining ICDM'06. 2006.
25. Li H-F, Lee S-Y. Mining frequent itemsets over data streams using efficient window sliding techniques. Int J Exp Syst Appl. 2009;36(2):1466–77.
26. Tsai PS. Mining frequent itemsets in data streams using the weighted sliding window model. Expert Syst Appl. 2009;36(9):11617–25.
27. Tanbeer SK, et al. Sliding window-based frequent pattern mining over data streams. Inf Sci. 2009;179(22):3843–65.
28. Farzanyar Z, Kangavari M, Cercone N. Max-FISM: Mining (recently) maximal frequent itemsets over data streams using the sliding window model. Comput Math Appl. 2012;64(6):1706–18.
29. Zhang C, et al. Mining frequent itemsets over tuple-evolving data streams, in Proceedings of the 28th Annual ACM Symposium on Applied Computing. 2013, ACM: Coimbra, Portugal. p. 267–274.
30. Agrawal R, Srikant R. Fast algorithms for mining association rules in large databases. in Proceedings of the 20th International Conference on Very Large Data Bases VLDB. 1994.
31. Fournier-Viger P, et al. The SPMF open-source data mining library version 2. In: Machine learning and knowledge discovery in databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19–23, 2016, Proceedings, Part III. Cham: Springer International Publishing; 2016. p. 36–40.

32. Chunduri RK, Cherukuri AK. Scalable algorithm for generation of attribute implication base using FP-growth and spark. Soft Comput. 2021;25:9219–40.
33. Rahaman MM, et al. Identification of COVID-19 samples from chest X-Ray images using deep learning: a comparison of transfer learning approaches. J Xray Sci Technol. 2020;28(5):821–39.
34. Chen H, et al. GasHis-transformer: a multi-scale visual transformer approach for gastric histopathological image detection. Pattern Recogn. 2022;130: 108827.
35. Liu W, et al. CVM-cervix: a hybrid cervical pap-smear image classification framework using cnn, visual transformer and multilayer perceptron. Pattern Recogn. 2022;2: 108829.
36. Zhang J, et al. LCU-net: a novel low-cost U-net for environmental microorganism image segmentation. Pattern Recogn. 2021;115: 107885.
37. Rahaman MM, et al. DeepCervix: a deep learning-based framework for the classification of cervical cells using hybrid deep feature fusion techniques. Comput Biol Med. 2021;136: 104649.