Noname manuscript No.
(will be inserted by the editor)

Variabillity Analysis on Pattern Mining Algorithms over Data Streams.

the date of receipt and acceptance should be inserted later

1 Introduction

Pattern Mining over Data Stream (PMDS) is one of the most challenging tasks in data mining. Then the problem is extensively studied in the last two decades. Current studies are mainly focused on a descriptive approach to present the algorithms and the result is a lack of depth in the analysis. In this sense, an analytical understanding is necessary to document what is common and what varies between the algorithms over time. Supporting this in-depth analysis, it is interesting to draw in variability modelling, a focal point of Software Product Line Engineering (SPLE) that helps to facilitate the management (building and maintenance) of a collection of similar software systems by factorising their common parts and identifying their specific ones.

2 Background

2.1 Pattern mining in the support framework

The problem was proposed in the early nineties in *Basket Market Analisys* [1]. The task consists of discovering groups of *items* (*itemsets*) that are frequently purchased together by customers. The frequency or *support* of an itemset is defined as the number or percentage of customer *transactions* containing the pattern as subset. *Frequent itemset* (FI) is one having support no less than a minimum threshold. The support is a *monotone* measure, that is if an itemset is infrequent, all its supersets are also infrequent, and thus do not need to be explored. This property is very powerful and can greatly reduce the

Address(es) of author(s) should be given

search space. *Apriori* [1], the first heuristic to find frequent itemsets followed a levelwise breadth-first search methodology where, in each level, candidate itemsets were formed from the already mined (frequent) itemsets. To address the main limitations of this approach, *FP-Growth* [21] the first *pattern-growth* algorithm was proposed. Its main improvments are keeping track the frequent itemsets without candidate generation phase (maximum two scan over the database is required) and storing compressed information about patterns in a tree structure called *FP-tree*. In constrast, algorithms such as *Eclat* [67] use a levelwise depth-first search strategy and a *vertical database representation* called *TidList* [67], which indicates the list of transactions where each item appears. These various algorithms may find a huge amount of itemsets for low support threshold and dense databases. To reduce the number of itemsets and present more meaningful itemsets to the user, *concise representations* of frequent itemsets have been proposed. The most popular concise representations of frequent itemsets are *closed itemset* (CI[52]) and *maximal itemset* (MFI[4]).

2.2 Pattern mining in the utility framework

In support framework, purchased quantities are assumed to be binary, i.e., either an itemset appears in a transaction or not and all items are treated with the same importance/weight/price. However, in the real world, each item in a supermarket has a different importance/price and one customer can buy multiple copies of an item. Thus the framework is extended to develop a mining model [59] discovering the *High-Utiliy Itemset*. In the utility framework, for each item is given the *internal utility*, i.e., number of units bought in each transactions and the *external utility*, i.e., the unit profit value. High-Utiliy Itemsets (HUI) are all itemsets that have a *utility* higher than a given threshold in a database (i.e. itemsets generating a high profit). A major challenge in HUI mining is that the *anti-monotonicity-property* does not serve to prune the search space with the utility measure. To solve this problem, the concept of *Transaction-Weighted Utilization* (TWU) is introduced in [46]. A popular variation of the HUI mining problem is to discover *high average-utility itemsets* (HAUIs), where an alternative measure called the *average-utility* (au) is used to evaluate the utility of itemsets by considering their lengths [23].

2.3 Pattern mining in the uncertain framework

In the support and the utility frameworks, patterns are discovered from data in which users definitely know whether an item is present in, or absent from, a transaction in the database. However, there are situations in which users are uncertain about the presence or absence of items. Uncertainty plays a role in several real-life applications since data collected is often imperfect, inaccurate, or may be collected through noisy sensors. Existing work on frequent itemset mining over uncertain data falls into two categories: *expected-support frequent* *itemsets* (EFI [14]) and *probabilistic frequent itemsets* (PFI [5]). In addition, it is possible to distinguish between the *tuple uncertainty model* and *attribute uncertainty model*. The former considers that each tuple or transaction is associated with an *existential probability* (a value in [0,1]), which indicates the chance that the transaction exists in the database. The latter associates to each attribute or item appearing in a transaction an *existential probability* representing the chance that this item appeared in the transaction.

2.4 Data stream models

A *data stream* $\mathcal{S} =< T_1, T_2, \ldots, T_i, \ldots >$ is a continuous sequence of transactions ordered according to their arrival time. The main challenges for frequent pattern mining in data stream are: (i) single pass constraint; (ii) limited processing time; (iii) limited memory. The prerequisite is therefore to find a suitable data window model. A window is a subsequence between the i-th and j-th transactions, denoted as $W[i, j] =< T_i, T_{i+1}, \ldots, T_j >$, $i \leq j$. Several window models exist in the literature and the best known are [31]: *landmark*, *sliding*, *damped*, *tilted*.

Landmark window: In this model $W[s, t]$, transactions are considered from a starting time point s, called landmark, to the current time point t. This model is suitable when transactions are treated as equally importance. A special case is when $s = 1$ (the full data stream is the window).

Sliding window: In this model, transactions are considered in a window $W[t-w+1, t]$, where w is the window size from the current time point t. As time goes, each window moves along with the current time point and transactions arriving before the time point $t - w + 1$ are discarded.

Damped window: In this model, each transaction is associated with a weight. The highest weight is assigned to more recently arrived transactions. A typically way is to define a *decay factor* d, $0 < d < 1$. As each new data transaction arrives, the support count of the previous mined patterns is multiplied by d to reduce their contribution.

Tilted window: In this model, transactions are considered in a set of varying size windows. Each window corresponds to different granularity. The most recent transactions are kept at the finest granularity, whereas the old ones are registered at a coarser granularity. In the simplest version, each window size is twice of that more recent neighbor.

In the different window models, the algorithms can be characterized by the number of transactions ($N \geq 1$) handled in each update operation. This leads to treat a single transaction (*Per transaction update*) or a batch of N transactions (*Per batch update*). Another charaterization is consider the update frequency in terms of number (e.g., every 1000 transactions) or in terms of time (e.g., weekly, monthly). The first is referred as *Transaction-Sensitive* and the later as *Time-Sensitive*. In the sliding model, we have two variants: *fixed-size windows* (FsW) and *variable-size windows* (Vsw). It is possible to have algorithms combining different characteristics.

Landmark, *damped* and c models all maintain the entire history of the stream with adding operations. In our description, we proposed to regroup these different models in a unique one called *incremental*.

Given a window model and a mining framework (support, utility, uncertain, etc.), the PMDS problem is to find the indicated pattern (frequent, maximal, closed, probabilistic, high utility, expected-support, high-average utility, etc.) in the current window.

3 Analysis of PMDS algorithms

In the search process, about 400 papers published between 2002 and 2019 are collected through scientific databases. Filtering all, a total of **58 algorithms** is selected as the most relevant. Fig. 1 presents these algorithms with a holistic view in a structurally logical and thematically coherent manner that combines the three well-known frameworks (support, utility, uncertain), the three well-accepted mining approaches (Apriori, FP-Growth, hybrid) and the two classic window models (incremental, sliding). Figures on the lines or in brackets indicate the number of algorithms for a given framework, approach or window model (red color indicates the highest number). The following sections provide a deep description and a fine analysis of selected algorithms that show the key design parameters and the technical relationship.

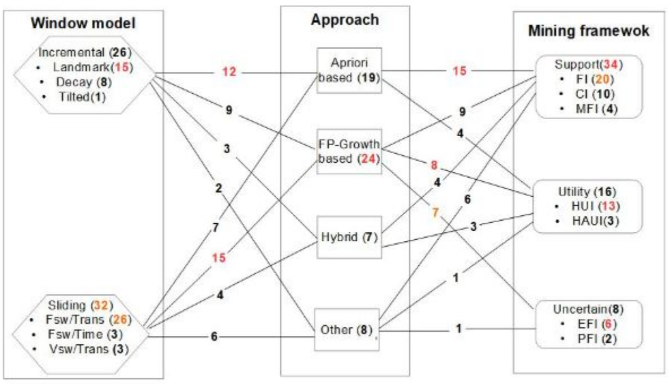

Fig. 1 The PMDS state-of-the-art algorithms by framework, approach and window model.

3.1 Pattern mining over incremental window

As a summary of pattern mining over incremental window, the Table 1 shows the most well known algorithms among the research community. *Lossy Counting*, *FP-Stream*, *esDec*, *FDPM*, *DSM-FI* and *FUFP-Tree* appear in most of the comparative studies in PMDS and they are, by far, the key approaches used in incremental window. *Lossy Counting* (LC) is recognized as the first PMDS algorithm. It serves as authority and comparison model to the first generation of algorithms, so it is the most cited. The *FP-Stream* and *esDec* algorithms are highly cited since, respectively they introduced for the first time, the tilted and the decay window model. *FDPM* is the lone *False-negative* algorithm in this window model. *FUFP-Tree* is one of the most original proposal since LC as an *exact* algorithm and introducing first the *Pattern-Growth* approach. The research community paid more atttention to support-based algorithms in the first decade. In the last decade, mining algorithms in utility and uncertain framework are emerging. It is also obvious that the landmark window model and *Apriori* approach are desirable for the research community.

Algorithm	**Year**	**Framework (pattern)**	**Approach**	**Window (sub) model**	**Accuracy**	**# Citations**
Lossy Counting [47]	2002	Support (FI)	Apriori	Landmark	False-positive	1863
FP-Stream [19]	2003	Support (FI)	FP-Growth	Titled	False-positive	773
estDec [8]	2003	Support (FI)	Apriori	Decay	False-positive	458
FDPM [61]	2004	Support (FI)	Apriori	Landmark	False-negative	222
DSM-FI [39]	2004	Support (FI)	Apriori	Landmark	False-positive	197
INSTANT [48]	2007	Support (MFI)	Apriori	Landmark	Exact	74
FUFP-Tree [24]	2008	Support (FI)	Apriori	Landmark	Exact	208
Pre-FUFP [43]	2009	Support (FI)	Apriori	Landmark	Exact	113
EstMax [58]	2009	Support (MFI)	Apriori	Decay	False-positive	31
CLICI [20]	2010	Support (CI)	Other	Decay	False-positive	22
TUF-Streaming [33]	2011	Uncertain (EFI)	FP-Growth	Decay	False-positive	49
GUIDE [54]	2012	Utility (HUI)	Other	Landmark	Approximative	77
FUP-HU [44]	2012	Utility (HUI)	Apriori	Landmark	Exact	104
UHS-Stream [22]	2013	Uncertain (EFI)	FP-Growth	Landmark	False-positive	21
TFUHS-Stream [22]	2013	Uncertain (EFI)	FP-Growth	Decay	False-positive	21
PRE-HUI [42]	2014	Utility (HUI)	Apriori	Landmark	Exact	41
HUPID [65]	2015	Utility (HUI)	FP-Growth	Landmark	Exact	68
HUI-LIST-INS [45]	2015	Utility (HUI)	Hybrid	Landmark	Exact	26
GENHUI [26]	2016	Utility (HUI)	FP-Growth	Decay	Exact	23
FCIMining [7]	2016	Support (CI)	FP-Growth	Decay	False-positive	11
TDUP [40]	2017	Support (FI)	Apriori	Landmark	False-positive	41
LIHUP [66]	2017	Utility (HUI)	Hybrid	Landmark	Exact	56
IMHAUI [27]	2017	Utility (HAUI)	FP-Growth	Landmark	Exact	20
MPM [63]	2018	Utility (HAUI)	FP-Growth	Decay	Exact	64
PIHUP [30]	2018	Utility (HUI)	Apriori	Landmark	Exact	16
IIHUM [64]	2019	Utility (HUI)	Hybrid	Landmark	Exact	18

Table 1 List of the most PMDS state-of-the-art algorithms in incremental window model ordered by year. # Citations is from GoogleScholar [57]. Last updated january 25th, 2021.

3.2 Pattern mining over sliding window

As a summary of pattern mining over sliding window, Table 2 illustrates the most well known algorithms among the research community. *Moment*, *IHUP*, *DStree*, *MFI-TransSW* and *MFI-TimeSW* appear in most of the comparative studies in PMDS and they are, by far, the key approaches used in sliding window. The *Moment* algorithm is recognized as the first closed itemset mining algorithm in sliding windows. It serves as authority and comparison model to the later algorithms. *IHUP* is the most cited algorithm in this window model. It introduces for the first time, the *Pattern-Growth* approach for HUI mining.

DStree is a FI mining algorithm expandable to handle different pattern types (CI and MFI). The *MFI-TransSW* and *MFI-TimeSW* algorithms are also FIs mining adopting the *Eclat* algorithm framework, thus they appear in the top-five algorithms as the first algorithm proposing this approach. It is observed that the *FsW/TransSW* window model, *Pattern Growth* approach and *exact* algorithms from accuracy standpoint are desirable for the research community.

Algorithm	Year	Framework (pattern)	Approach	Window (sub) model	Accuracy	# Citations
Moment [12]	2004	Support (CI)	FP-Growth	FsW/TransSW	Exact	409
SWM [9]	2004	Support (FI)	Apriori	FsW/TransSW	False-positive	188
PFP [41]	2005	Support (FI)	Other	**FsW/TimeSW**	Approximative	185
DSTree [34]	2006	Support (FI)	FP-Growth	FsW/TransSW	Exact	212
CFI-Stream [25]	2006	Support (CI)	FP-Growth	FsW/TransSW	Exact	168
THUI-Mine [13]	2008	Utility (HUI)	Apriori	FsW/TransSW	Exact	192
MHUI [37]	2008	Utility (HUI)	Hybrid	FsW/TransSW	Exact	135
SWIM [49]	2008	Support (FI)	Other	FsW/TransSW	Exact	121
Incmine [11]	2008	Support (CI)	Apriori	**FsW/TimeSW**	False-positive	50
NewMoment [36]	2009	Support (CI)	FP-Growth	FsW/TransSW	Exact	75
CPS-tree [55]	2009	Support (FI)	FP-Growth	FsW/TransSW	Exact	166
IHUP [2]	2009	Utility (HUI)	FP-Growth	FsW/TransSW	Exact	618
MFI-TransSW [38]	2009	Support (FI)	Hybrid	FsW/TransSW	Exact	227
MFI-TimeSW [38]	2009	Support (FI)	Hybrid	**FsW/TimeSW**	Exact	227
UF-Streaming [32]	2009	Uncertain (EFI)	FP-Growth	FsW/TransSW	False-positive	122
SUF-Growth [32]	2009	Uncertain (EFI)	FP-Growth	FsW/TransSW	Exact	122
FCDT [28]	2009	Support (FI)	Other	**VsW/TransSW**	False-positive	16
Clostream [60]	2011	Support (CI)	Apriori	FsW/TransSW	Exact	46
VSW [16]	2012	Support (FI)	Other	**VsW/TransSW**	Exact	70
SWP-Tree [10]	2012	Support (FI)	FP-Growth	**VsW/TransSW**	False-positive	50
LDS [15]	2012	Support (FI)	Hybrid	FsW/TransSW	Exact	30
MAX-FISM [18]	2012	Support (MFI)	Apriori	FsW/TransSW	Exact	33
StreamFCI [56]	2012	Support (CI)	FP-Growth	FsW/TransSW	Exact	18
Tmoment [51]	2013	Support (CI)	FP-Growth	FsW/TransSW	Exact	72
pWin [17]	2013	Support (FI)	Apriori	FsW/TransSW	Approximative	16
FEMP [3]	2013	Uncertain (PFI)	Other	FsW/TransSW	Exact	6
UDS-FIM [29]	2014	Uncertain (EFI)	FP-Growth	FsW/TransSW	Exact	8
SHU-Growth [53]	2016	Utility (HUI)	FP-Growth	FsW/TransSW	Exact	64
SHAU [62]	2016	Utility (HAUI)	FP-Growth	FsW/TransSW	Exact	20
PFIMoS [35]	2018	Uncertain (PFI)	FP-Growth	FsW/TransSW	Exact	12
ConPatSet [50]	2018	Support (CI)	Apriori	FsW/TransSW	Exact	10
RMFIsM [6]	2019	Support (MFI)	Other	FsW/TransSW	Exact	3

Table 2 List of PMDS state-of-the-art algorithms in sliding window model ordered by year. # Citations is from GoogleScholar [57]. Last updated january 25th, 2021.

3.3 Advanced design parameters of PMDS algorithms

In addition to features showed in Tables 1 and 2, an in-depth analysis highlights four key parameters that have major impact and introduce variability in the algorithm design: the steps followed, the update type, the maintained sets and the data structures used.

Update types: algorithms do update operations (addition or deletion) *Per Batch* (PB) or *Per Transaction* (PT). The distribution is substantially the same as shown in Table 3: PB (30) and PT (28). In the incremental window model, the preference is for the first mode (16 algorithms) whereas in the sliding window model, the favor is for second mode (18 algorithms).

Data sets: In PMDS, the main challenge is selecting the data sets to maintain in memory. The data sets must be informative enough to mine all the patterns and small to fit in memory. The algorithms devise three different strategies as reported in Table 3:

- Maintain patterns exclusively: case of algorithms in classes **C1** up to **C10** (24 algorithms). The mined patterns fall in three types: Interesting[1] Patterns (IP), Significant[2] Patterns (SP) or inFrequent Patterns ($\neg$FP).
- Maintain the Transactions in data Stream (TS) exclusively: case of algorithms in classes **C11** up to **C18** (22 algorithms).
- Maintain both patterns and the transactions: case of algorithms in classes **C19** up to **C24** (12 algorithms).

In these different situations, adopting an efficient pruning strategy is critical to the performance of algorithms that heavily rely on selected and maintained sets.

Data structures: Trees, lists and tables are essentially the data structures used to maintain incoming transactions and patterns. Trees, in different variants (lexicographic, prefix, canonical, forest) are the most used (by 40 algorithms) data structure. The lists are mainly used in the sliding model with vertical representations of the transactions. In the incremental model, it is unrealistic to maintain transactions in lists and they are mainly used to keep patterns information. Hash tables are only used in the sliding model to maintain closed patterns.

Algorithms steps: Five distinct steps are identified. Steps 2 and 4 are not required for all algorithms as shown in Table 3:

- **step 1:** It is the preprocessing phase of the incremental database by reading (in memory) and sorting the incoming transactions.
- **step 2:** It is the patterns generation phase from the incremental database if the patterns are maintained.
- **step 3:** It is the update phase of the data structure used to maintain the patterns and/or the transactions.
- **step 4:** It is the pruning phase (of the data structure updated in step 3) if the pruning condition is verified.
- **step 5:** It is the generation phase of the interesting patterns in the updated database when requested by the user.

4 Conclusion and research perspectives

In this work is proposed an approach to organize families of PMDS algorithms so that they are easier to understand and design. It consists of two steps. The

[1] depend on framework: support(FI, CI, MFI), utility (HUI, HAUI), uncerain (EFI, PFI)
[2] given a significant measure threshold

Algorithm classes [references]	Data sets				Structures		Update		Steps				
	TS	SP	IP	¬FP	TR	LT	PT	PB	1	2	3	4	5
C1 [13,44]			×			×		×	×	×	×		×
C2 [35]			×		×		×		×	×	×		×
C3 [48,50,60]			×	×		×	×		×	×	×		×
C4 [12,25,28]			×	×	×		×		×	×	×		×
C5 [7,20]			×	×	×		×		×	×	×	×	×
C6 [40–42]		×	×			×		×	×	×	×		×
C7 [11,61]		×	×			×		×	×	×	×	×	×
C8 [19,22,22]		×	×		×			×	×	×	×	×	×
C9 [32,33,47]		×	×		×			×	×	×	×		×
C10 [8,58]		×	×		×		×		×	×	×	×	×
C11 [15,38,45,66]	×					×		×	×		×		×
C12 [6,38,64]	×					×	×		×		×		×
C13 [24,32,34,53,55,62]	×				×			×	×		×		×
C14 [39]	×				×			×	×		×	×	×
C15 [2,26,27]	×				×		×		×		×		×
C16 [10,54]	×				×		×		×		×	×	×
C17 [29]	×				×	×		×	×		×		×
C18 [63,65]	×				×	×	×		×		×		×
C19 [18,51]	×		×	×	×	×	×		×	×	×		×
C20 [3]	×		×			×	×		×	×	×		×
C21 [36,37,56]	×		×		×	×	×		×	×	×		×
C22 [30]	×	×	×		×			×	×	×	×		×
C23 [16,17,43,49]	×	×	×		×	×		×	×	×	×		×
C24 [9]	×	×	×		×	×	×		×	×	×		×
Total	**34**	**19**	**36**	**10**	**40**	**31**	**28**	**30**	**58**	**36**	**58**	**13**	**58**

TS: maintaining incoming Transactions
SP: maintaining significant patterns
LT: Lists or tables
¬FP: maintaining infrequent patterns
IP: maintaining interesting patterns
TR: Trees
PB: per batch upadate
PT: per transaction update

Table 3 Product-by-feature matrix extracted from PMDS state-of-the-art algorithms.

first step is conducting a Systematic Literature Review of PMDS algorithms to constitute a representative sample covering any framework (support, utility, uncertain), any window model (incremental, sliding) and any technical approach (Apriori, pattern growth, hybrid, etc). The second step takes as input the set of selected algorithms described (in textual or pseudo-code format) in the literature, and manually extract (by an expert) the algorithms characteristic features. These features are structured in tables similar to product-by-feature matrix used in SPLE.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Database mining: A performance perspective. IEEE Transactions on Knowledge and Data Engineering **5**(6), 914–925 (1993)
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Lee, Y.K.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Transactions on Knowledge and Data Engineering **21**(12), 1708–1721 (2009)
3. Akbarinia, R., Masseglia, F.: Fast and exact mining of probabilistic data streams. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 493–508. Prague, Czech Republic (2013)
4. Bayardo Jr, R.J.: Efficiently mining long patterns from databases. ACM Sigmod Record **27**(2), 85–93 (1998)
5. Bernecker, T., Kriegel, H.P., Renz, M., Verhein, F., Zuefle, A.: Probabilistic frequent itemset mining in uncertain databases. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, pp. 119–128. Paris, France (2009)
6. Cai, S., Hao, S., Sun, R., Wu, G.: Mining recent maximal frequent itemsets over data streams with sliding window. Int. Arab J. Inf. Technol. **16**(6), 961–969 (2019)

7. Caiyan, D., Ling, C.: An algorithm for mining frequent closed itemsets with density from data streams. International Journal of Computational Science and Engineering **12**(2-3), 146–154 (2016)
8. Chang, J.H., Lee, W.S.: Finding recent frequent itemsets adaptively over online data streams. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 487–492. Washington, DC, U.S.A (2003)
9. Chang, J.H., Lee, W.S.: A sliding window method for finding recently frequent itemsets over online data streams. Journal of Information science and Engineering **20**(4), 753–762 (2004)
10. Chen, H., Shu, L., Xia, J., Deng, Q.: Mining frequent patterns in a varying-size sliding window of online transactional data streams. Information Sciences **215**, 15–36 (2012)
11. Cheng, J., Ke, Y., Ng, W.: Maintaining frequent closed itemsets over a sliding window. Journal of Intelligent Information Systems **31**(3), 191–215 (2008)
12. Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Moment: Maintaining closed frequent itemsets over a stream sliding window. In: Proceedings of the 4th IEEE International Conference on Data Mining, pp. 59–66. Brighton, UK (2004)
13. Chu, C.J., Tseng, V.S., Liang, T.: An efficient algorithm for mining temporal high utility itemsets from data streams. Journal of Systems and Software **81**(7), 1105–1117 (2008)
14. Chui, C.K., Kao, B., Hung, E.: Mining frequent itemsets from uncertain data. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 47–58. Nanjing, China (2007)
15. Deypir, M., Sadreddini, M.H.: A dynamic layout of sliding window for frequent itemset mining over data streams. Journal of Systems and Software **85**(3), 746–759 (2012)
16. Deypir, M., Sadreddini, M.H., Hashemi, S.: Towards a variable size sliding window model for frequent itemset mining over data streams. Computers & Industrial Engineering **63**(1), 161–172 (2012)
17. Deypir, M., Sadreddini, M.H., Tarahomi, M.: An efficient sliding window based algorithm for adaptive frequent itemset mining over data streams. Journal of Information Science and Engineering **29**(5), 1001–1020 (2013)
18. Farzanyar, Z., Kangavari, M., Cercone, N.: Max-fism: Mining (recently) maximal frequent itemsets over data streams using the sliding window model. Computers & Mathematics with Applications **64**(6), 1706–1718 (2012)
19. Giannella, C., Han, J., Pei, J., Yan, X., Yu, P.S.: Mining frequent patterns in data streams at multiple time granularities. Next Generation of Data Mining **212**, 191–212 (2003)
20. Gupta, A., Bhatnagar, V., Kumar, N.: Mining closed itemsets in data stream using formal concept analysis. In: Proceedings of the 12th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), pp. 285–296. Bilbao, Spain (2010)
21. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. ACM Sigmod Record **29**(2), 1–12 (2000)
22. HewaNadungodage, C., Xia, Y., Lee, J.J., Tu, Y.C.: Hyper-structure mining of frequent patterns in uncertain data streams. Knowledge and information systems **37**(1), 219–244 (2013)
23. Hong, T.P., Lee, C.H., Wang, S.L.: Mining high average-utility itemsets. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 2526–2530. San Antonio, TX, USA (2009)
24. Hong, T.P., Lin, C.W., Wu, Y.L.: Incrementally fast updated frequent pattern trees. Expert Systems with Applications **34**(4), 2424–2435 (2008)
25. Jiang, N., Gruenwald, L.: Cfi-stream: mining closed frequent itemsets in data streams. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 592–597. Philadelphia, PA, USA (2006)
26. Kim, D., Yun, U.: Mining high utility itemsets based on the time decaying model. Intelligent Data Analysis **20**(5), 1157–1180 (2016)
27. Kim, D., Yun, U.: Efficient algorithm for mining high average-utility itemsets in incremental transaction databases. Applied Intelligence **47**(1), 114–131 (2017)
28. Koh, J.L., Lin, C.Y.: Concept shift detection for frequent itemsets from sliding windows over data streams. In: Proceedings of the 14th International Conference on Database Systems for Advanced Applications, pp. 334–348. Brisbane, Australia (2009)

29. Le Wang, L.F., Wu, M.: Uds-fim: an efficient algorithm of frequent itemsets mining over uncertain transaction data streams. Journal of Software **9**(1), 44–56 (2014)
30. Lee, J., Yun, U., Lee, G., Yoon, E.: Efficient incremental high utility pattern mining based on pre-large concept. Engineering Applications of Artificial Intelligence **72**, 111–123 (2018)
31. Lee, V.E., Jin, R., Agrawal, G.: Frequent pattern mining in data streams. In: Frequent Pattern Mining, pp. 199–224. Springer (2014)
32. Leung, C.K.S., Hao, B.: Mining of frequent itemsets from streams of uncertain data. In: Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE'09), pp. 1663–1670. Shanghai, China (2009)
33. Leung, C.K.S., Jiang, F.: Frequent pattern mining from time-fading streams of uncertain data. In: International Conference on Data Warehousing and Knowledge Discovery, pp. 252–264. Toulouse, France (2011)
34. Leung, C.K.S., Khan, Q.I.: Dstree: a tree structure for the mining of frequent sets from data streams. In: Proceedings of the 6th IEEE International Conference on Data Mining, pp. 928–932. Hong Kong, China (2006)
35. Li, H., Zhang, N., Zhu, J., Wang, Y., Cao, H.: Probabilistic frequent itemset mining over uncertain data streams. Expert Systems with Applications **112**, 274–287 (2018)
36. Li, H.F., Ho, C.C., Lee, S.Y.: Incremental updates of closed frequent itemsets over continuous data streams. Expert Systems with Applications **36**(2), 2451–2458 (2009)
37. Li, H.F., Huang, H.Y., Chen, Y.C., Liu, Y.J., Lee, S.Y.: Fast and memory efficient mining of high utility itemsets in data streams. In: Proceedings of the 8th IEEE International Conference on Data Mining, pp. 881–886. Pisa, Italy (2008)
38. Li, H.F., Lee, S.Y.: Mining frequent itemsets over data streams using efficient window sliding techniques. Expert systems with applications **36**(2), 1466–1477 (2009)
39. Li, H.F., Lee, S.Y., Shan, M.K.: An efficient algorithm for mining frequent itemsets over the entire history of data streams. In: Proceedings of 1st International Workshop on Knowledge Discovery in Data Streams (2004)
40. Li, Y., Zhang, Z.H., Chen, W.B., Min, F.: Tdup: an approach to incremental mining of frequent itemsets with three-way-decision pattern updating. International Journal of Machine Learning and Cybernetics **8**(2), 441–453 (2017)
41. Lin, C.H., Chiu, D.Y., Wu, Y.H., Chen, A.L.: Mining frequent itemsets from data streams with a time-sensitive sliding window. In: Proceedings of the SIAM International Conference on Data Mining, pp. 68–79. Newport Beach, CA, USA (2005)
42. Lin, C.W., Hong, T.P., Lan, G.C., Wong, J.W., Lin, W.Y.: Incrementally mining high utility patterns based on pre-large concept. Applied Intelligence **40**(2), 343–357 (2014)
43. Lin, C.W., Hong, T.P., Lu, W.H.: The pre-fufp algorithm for incremental mining. Expert Systems with Applications **36**(5), 9498–9505 (2009)
44. Lin, C.W., Lan, G.C., Hong, T.P.: An incremental mining algorithm for high utility itemsets. Expert Systems with Applications **39**(8), 7173–7180 (2012)
45. Lin, J.C.W., Gan, W., Hong, T.P., Zhang, B.: An incremental high-utility mining algorithm with transaction insertion. The Scientific World Journal **2015** (2015)
46. Liu, Y., Liao, W.K., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 689–695. Hanoi, Vietnam (2005)
47. Manku, G.S., Motwani, R.: Approximate frequency counts over data streams. In: Proceedings of the 28th International Conference on Very Large Databases (VLDB), pp. 346–357. Hong Kong, China (2002)
48. Mao, G., Wu, X., Zhu, X., Chen, G., Liu, C.: Mining maximal frequent itemsets from data streams. Journal of Information Science **33**(3), 251–262 (2007)
49. Mozafari, B., Thakkar, H., Zaniolo, C.: Verifying and mining frequent patterns from large windows over data streams. In: Proceedings of the 24th IEEE International Conference on Data Engineering, pp. 179–188. Cancun, Mexico (2008)
50. Nguyen, T.: Mining incrementally closed item sets with constructive pattern set. Expert Systems with Applications **100**, 41– 67 (2018)
51. Nori, F., Deypir, M., Sadreddini, M.H.: A sliding window based algorithm for frequent closed itemset mining over data streams. Journal of Systems and Software **86**(3), 615–623 (2013)

52. Pei, J., Han, J., Mao, R., et al.: Closet: An efficient algorithm for mining frequent closed itemsets. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery **4**(2), 21–30 (2000)
53. Ryang, H., Yun, U.: High utility pattern mining over data streams with sliding window technique. Expert Systems with Applications **57**, 214–231 (2016)
54. Shie, B.E., Philip, S.Y., Tseng, V.S.: Efficient algorithms for mining maximal high utility itemsets from data streams with different models. Expert Systems with Applications **39**(17), 12947–12960 (2012)
55. Tanbeer, S.K., Ahmed, C.F., Jeong, B.S., Lee, Y.K.: Sliding window-based frequent pattern mining over data streams. Information sciences **179**(22), 3843–3865 (2009)
56. Tang, K., Dai, C., Chen, L.: A novel strategy for mining frequent closed itemsets in data streams. Journal of Computers **7**(7), 1564–1573 (2012)
57. URL: "https://google.scholar.fr" (accessed April 10, 2020)
58. Woo, H.J., Lee, W.S.: Estmax: Tracing maximal frequent item sets instantly over online transactional data streams. IEEE Transactions on Knowledge and Data Engineering **21**(10), 1418–1431 (2009)
59. Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: Proceedings of the 4th SIAM International Conference on Data Mining, pp. 482–486. Lake Buena Vista, Florida, USA (2004)
60. Yen, S.J., Wu, C.W., Lee, Y.S., Tseng, V.S., Hsieh, C.H.: A fast algorithm for mining frequent closed itemsets over stream sliding window. In: Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ), pp. 996–1002. Taipei, Taiwan (2011)
61. Yu, J.X., Chong, Z., Lu, H., Zhou, A.: False positive or false negative: mining frequent itemsets from high speed transactional data streams. In: Proceedings of the 30th International Conference on Very Large Data Bases, pp. 204–215. Toronto, Canada (2004)
62. Yun, U., Kim, D., Ryang, H., Lee, G., Lee, K.M.: Mining recent high average utility patterns based on sliding window from stream data. Journal of Intelligent & Fuzzy Systems **30**(6), 3605–3617 (2016)
63. Yun, U., Kim, D., Yoon, E., Fujita, H.: Damped window based high average utility pattern mining over data streams. Knowledge-Based Systems **144**, 188–205 (2018)
64. Yun, U., Nam, H., Lee, G., Yoon, E.: Efficient approach for incremental high utility pattern mining with indexed list structure. Future Generation Computer Systems **95**, 221–239 (2019)
65. Yun, U., Ryang, H.: Incremental high utility pattern mining with static and dynamic databases. Applied Intelligence **42**(2), 323–352 (2015)
66. Yun, U., Ryang, H., Lee, G., Fujita, H.: An efficient algorithm for mining high utility patterns from incremental databases with one database scan. Knowledge-Based Systems **124**, 188–206 (2017)
67. Zaki, M.J.: Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering **12**(3), 372–390 (2000)