Discovering Spatial Weighted Frequent Itemsets in Spatiotemporal Databases

P. P. C. Reddy[1,+], R. Uday Kiran[2,3,*], Koji Zettsu[2], Masashi Toyoda[3], Masaru Kitsuregawa[3,4], P. Krishna Reddy[1]
[1]International Institute of Information Technology-Hyderabad, Telangana, India
[2]National Institute of Information and Communications Technology, Tokyo, Japan
[3]The University of Tokyo, Tokyo, Japan
[4]National Institute of Informatics, Tokyo, Japan
{uday_rage,toyoda,kitsure}@tkl.iis.u-tokyo.ac.jp, pradeepchandra.p@research.iiit.ac.in, zettsu@nict.go.jp, pkreddy@iiit.ac.in

***Abstract*—Weighted Frequent Itemset (WFI) mining is an important model in data mining. It aims to discover all itemsets whose weighted sum in a transactional database is no less than the user-specified threshold value. Most previous works focused on finding WFIs in a transactional database and did not recognize the spatiotemporal characteristics of an item within the data. This paper proposes a more flexible model of Spatial Weighted Frequent Itemset (SWFI) that may exist in a spatiotemporal database. The recommended patterns may be found very useful in many real-world applications. For instance, an SWFI generated from an air pollution database indicates a geographical region where people have been exposed to high levels of an air pollutant, say $PM2.5$. The generated SWFIs do not satisfy the anti-monotonic property. Two new measures have been presented to effectively reduce the search space and the computational cost of finding the desired patterns. A pattern-growth algorithm, called Spatial Weighted Frequent Pattern-growth, has also been presented to find all SWFIs in a spatiotemporal database. Experimental results demonstrate that the proposed algorithm is efficient. We also describe a case study in which our model has been used to find useful information in air pollution database.**

***Index Terms*—Data mining, pattern mining, weighted frequent itemset, pattern-growth, spatiotemporal data**

I. Introduction

Frequent Itemset Mining (FIM) is a famous data mining model [2], [3], [7] with many real-world applications [1]. FIM aims to discover all itemsets in a transactional database that satisfy the user-specified minimum support ($minSup$) constraint. The $minSup$ controls the minimum number of transactions that an itemset must cover within the data. Since only a single $minSup$ is used for the whole data, the model implicitly assumes that all items within the data have the uniform frequency. However, this is the seldom case in many real-world applications. In many applications, some items appear very frequently within the data, while others rarely appear. If the frequencies of items vary a great deal, then we encounter the following two problems:

1) If $minSup$ is set too high, we miss those itemsets that involve rare items in the data.
2) To find the itemsets that include both frequent and rare items, we have to set $minSup$ very low. However, this may cause a combinatorial explosion, producing too many itemsets, because those frequent items associate with one another in all possible ways and many of them are meaningless depending upon the user or application requirements.

This dilemma is known as the *rare item problem* [26]. When confronted with this problem in real-world applications, researchers have tried to find frequent itemsets using multiple $minSups$ [18], where the $minSup$ of an itemset is expressed with *minimum item support* of its items. An open problem of this extended model is the methodology to determine the items' *minimum item supports*.

Cai et al. [5] introduced Weighted Frequent Itemset Mining (WFIM) to address the rare item problem. WFIM takes into account the weights (or importance) of items and tries to find all Weighted Frequent Itemsets (WFIs) in a transactional database that satisfy the user-specified weight constraint. Several weight constraints (e.g., weighted sum, weighted support, and a weighted average) have been discussed in the literature to determine the interestingness of an itemset in a transactional database. Selecting an appropriate weight constraint depends on the user or application requirements. Some of the practical applications of WFIM include market-basket analytics [5], spectral signature analytics in astronomical databases [6], and event analytics in Twitter data [12].

This paper argues that though studies on WFIM consider the importance of items within the data, they disregard the spatiotemporal characteristics of an item. Consequently, WFIM is insufficient to find only those WFIs that have items close (or neighbors) to one another in a spatiotemporal database. This paper introduces Spatial Weighted Frequent Itemset Mining (SWFIM) to address this issue. Before we discuss the contributions of this paper, we describe an essential application of SWFIM.

Air pollution is a significant factor for many cardio-respiratory problems found in the people living in Japan. In this context, the Atmospheric Environmental Regional Observation System (AEROS) constituting of several monitoring

+ Developed fast algorithm and contributed significantly in experiments
*Proposed the model, drafted the work, and corresponding author

stations has been set up by the Ministry of Environment, Japan. The data generated by these stations represent a non-binary spatiotemporal database. An SWFI found in this pollution database provides the information regarding the geographical region (or set of stations) where people exposed to high levels of an air pollutant. This information is useful for the users of the pollution control board in devising appropriate policies to control the industrial emissions.

High Utility Itemset Mining [9], [14], [27] generalizes WFIM by taking into account the items' internal utility and external utility values. Consequently, WFIs can be generated using HUIM algorithms. However, such an approach is inefficient. It is because we need to transform a binary transactional database into a non-binary transactional database by adding one as the internal utility for every item in the data. Consequently, the resultant database size increases significantly (approximately 1.5 to 2 times), which in turn increases the memory and runtime requirements of a HUIM algorithm.

This paper proposes a more flexible model of SWFI that may exist in a spatiotemporal database. An itemset in a spatiotemporal database is considered as an SWFI if it satisfies the user-specified *minimum weighted sum* and *maximum distance* constraints. The generated SWFIs do not satisfy the anti-monotonic property. Two upper bound measures, called *estimated weighted sum* (EWS) and *cumulative neighborhood weighted sum* ($CNWS$), have been employed to reduce the computational cost of desired itemsets. EWS aims to identify candidate items whose supersets may be SWFIs. $CNWS$ seeks to identify those items that have to be projected (or build conditional pattern bases) to find all SWFIs. A pattern-growth algorithm, called Spatial Weighted Frequent Pattern-growth (SWFP-growth), has also been presented to find all SWFIs in STD efficiently. Experimental results demonstrate that SWFP-growth is efficient. We also describe a case study in which we apply our model to find useful information in air pollution database.

The remainder of this paper is organized as follows. Section 2 discusses the previous literature related to a problem. Section 3 introduces the proposed model of SWFI in a STD. Section 4 describes the SWFP-growth. Experimental results are reported in Section 5. Section 6 concludes the paper with future research directions.

II. Related Work

A. Weighted itemset mining

Cai et al. [5] introduced WFIM to address the *rare item problem* in FIM. Two Apriori algorithms, called MinWAL(O) and MinWAL(M), have been discussed for finding WFIs in a transactional database. Unfortunately, both algorithms suffer from the performance issues involving multiple database scans and the generation of too many candidate itemsets. Yun and John [28] discussed a pattern-growth algorithm, called WFIM, to find the weighted frequent itemsets. Uday et al. [12] described an improved WFIM based on the concept of *cutoff weight*, which represents the maximum weight among all weighted items.

Cai et al. [6] used a variant of WFIM algorithm to find weighted frequent itemsets in an astronomical database. An entropy-based weighting function has been employed to determine the interestingness of an itemset.

In the literature, researchers have studied WFIM by taking into account other parameters. Tao et al. [21] proposed a weighted association rule model by taking into account the weight of a transaction. An Apriori-like algorithm, called WARM (Weighted Association Rule Mining) algorithm, was discussed to find to the itemsets. Vo et al. [25] proposed a Weighted Itemset Tidset tree (WIT-tree) for mining the itemsets and used a Diffset strategy to speed up the computation for finding the itemsets. Lin et al. [16] studied the problem of finding weighted frequent itemsets by taking into account the occurrence time of the transactions. The discovered itemsets are known as recency weighted frequent itemsets. Furthermore, Lin et al. [17] extended the basic weighted frequent itemset model [5] to handle uncertain databases. Chowdhury et al. [4] discussed a weighted frequent itemset model with an assumption that weights of items can vary with time and proposed the algorithm AWFPM (Adaptive Weighted Frequent Pattern Mining). Please note that though some of the above studies consider the temporal occurrence information of items within the data, they completely disregard the spatial information of the items. On the contrary, the proposed study investigates the problem of finding SWFIs in STD by taking into account the spatiotemporal characteristics of the items within the data.

B. High utility itemset mining

Yao et al. [27] introduced HUIM by taking into account the items' internal utility (i.e., number of occurrences of an item within a transaction) and external utility (i.e., weight of an item in the database) values. Since then, the problem of finding HUIs from the data has received a great deal of attention [9], [11], [14], [30]. As HUIM generalizes WFIM, WFIs can be generated using HUIM algorithms by transforming a binary database into a non-binary database. This paper argues that such an approach to finding WFIs using HUIM algorithms is inefficient because of two main reasons:

1) The process of transforming a huge binary database into a non-binary database is a costly operation concerning to both memory and runtime.
2) The size of the resultant non-binary transactional database is generally much more substantial (approximately 1.5 to 2 times) than that of the actual binary database. Consequently, HUIM algorithms have to find WFIs from much larger databases consuming memory and runtime.

In practice, a WFIM algorithm (respectively, FIM algorithm) is generally faster thana HUIM algorithm for mining WFIs (respectively, FIs) in a binary transactional database. It is because they are more optimized for that specific problem.

Uday et al. [15] discussed an algorithm, called Spatial High Utility Itemset Miner (SHUIMiner), to find all spatial high utility itemsets in a non-binary spatiotemporal database.

Unfortunately, finding the proposed SWFIs using SHUIMiner turns out to be costly due to the above mentioned reasons.

C. Spatial co-occurrence itemset mining

The problem of finding spatiotemporal co-occurrence itemsets (or association rules) in spatiotemporal databases has received a great deal of attention [8], [10], [19], [22]. These algorithms can be broadly classified into distance-based approaches [8], [10] and transaction-based approaches [19], [22]. A distance-based approach typically uses a parameter, called the prevalence, to determine how interesting the spatiotemporal co-occurrences are in the data. A transaction-based approach initially cluster the data over space and time and then apply traditional association rule mining algorithms on each cluster to find useful information. Unfortunately, all spatiotemporal co-occurrence itemset mining algorithms determine the interestingness of an itemset by taking into account only its $support$ and disregard the internal and external utility values of an item. Moreover, most of these algorithms cannot handle numeric data. On the contrary, the proposed model considers internal and external utility values of an item and handles numeric data.

Overall, the proposed model of finding SWFIs in a spatiotemporal database is novel and distinct from current studies.

III. Proposed Model

A. Model of Spatial Weighted Frequent Itemset

Let $I = \{i_1, i_2, \cdots, i_m\}, m \geq 1$, be the set of items. Let $X \subseteq I$ be an itemset (or a pattern). An itemset X containing k number of items is called a k-itemset. A transaction, denoted as $T_{ts} = (ts, Y)$, where $ts \in \mathbb{R}^+$ represents the transactional identifier (or timestamp) of the corresponding transaction and $Y \subseteq I$ is an itemset. A (binary) temporal database, denoted as $TDB = \{T_1, T_2, \cdots, T_n\}, n \geq 1$. Let $w(i_j, T_{ts})$, $1 \leq ts \leq n$, denote the **weight** of an item i_j in a transaction T_{ts}. Let $W(i_j) = \{w(i_j, T_1), w(i_j, T_2), \cdots, w(i_j, T_n)\}$ denote the set of all weights of i_j in a temporal database. The **items' weight database**, WD, is the set of weights of all items in I. That is, $WD = \bigcup\limits_{i_j \in I} W(i_j)$. A spatial database, denoted as $SD = \bigcup\limits_{i_j \in I} (i_j,\ (lat_{i_j}, long_{i_j}))$ is a collection of location points of all items in I. The terms lat_{i_j} and $long_{i_j}$ respectively denote the latitude and longitude information of an item i_j. (A spatiotemporal database is a combination of TDB and SD. For brevity, we describe SWFIM using TDB, WD and SD.)

Example 1. Let $I = \{a, b, c, d, e, f, g\}$ be the set of items (or air pollution monitoring station identifiers). The set of items 'a' and 'b,' i.e., $\{a, b\}$ (or ab, in short) is an itemset. This itemset contains two items. Therefore, it is a 2-itemset. A temporal database generated from I is shown in Table I. A spatial database of all items in Table I is shown in Table II. These two databases jointly represent a spatiotemporal database. The items' weight database is shown in Table III. Each transaction in this database represents the measurement of an air pollutant, say PM2.5, by a sensor for a particular time period. The weight of an item a in the first transaction, i.e., $w(a, T_1) = 20$. In other words, station a located at $(0, 0)$ has recorded $20\mu g/m^3$ of PM2.5 at the timestamp of 1.

Definition 1. (**The** $support$ **of** X **in a temporal database.**) If $X \subseteq T_k.Y$, $1 \leq k \leq n$, it is said that X occurs in transaction T_k (or T_k contains X). Let $TDB^X \subseteq TDB$ denote the set of all transactions containing X in TDB. The $support$ of X in TDB, denoted as $S(X) = |TDB^X|$.

Example 2. The itemset $ab \subseteq T_1.abgf$. Thus, the first transaction contains the itemset ab. Similarly, the sixth transaction also constaints the itemset ab. The set of all transactions containing ab in Table I, i.e., $TDB^{ab} = \{T_1, T_6\}$. The $support$ of ab in Table I, i.e., $S(ab) = |TDB^{ab}| = 2$.

Definition 2. (**Weighted sum of an itemset** X **in a transaction.**) The weighted sum of an itemset X in T_k, denoted as $WS(X, T_k)$, is the sum of weights of all items of X in T_k. That is, $WS(X, T_k) = \sum_{i_j \in X} w(i_j, T_k)$. If $X \not\subseteq T_k.Y$, then $WS(X, T_k) = 0$.

Example 3. The weighted sum of ab in T_1, i.e., $WS(ab, T_1) = w(a, T_1) + w(b, T_1) = 20 + 15 = 35$. The itemset ab does not occur in the second transaction. It means the stations a and b have cumulatively recorded 35 $\mu g/m^3$ of PM2.5 at the timestamp 1.

Definition 3. (**Weighted sum of an itemset** X **in a temporal database.**) The weighted sum of X in TDB, denoted as $WS(X) = \sum_{T_{ts} \in TDB^X} WS(X, T_{ts})$.

Example 4. The weighted sum of ab in Table I, i.e., $WS(ab) = \sum_{T_{ts} \in TDB^{ab}} WS(ab, T_{ts}) = WS(ab, T_1) + WS(ab, T_6) = (20 + 15) + (10 + 20) = 35 + 30 = 65$. Similarly, for the itemset cd, $TDB^{cd} = \{T_4, T_5\}$, $S(cd) = 2$ and $WS(cd) = \sum_{T_{ts} \in TDB^{cd}} WS(cd, T_{ts}) = WS(cd, T_4) + WS(cd, T_5) = (80 + 10) + (40 + 20) = 150$.

Definition 4. (**Weighted Frequent Itemset** X**.**) An itemset X is a weighted frequent itemset if $WS(X) \geq minWS$, where $minWS$ represents the user-specified minimum weighted sum.

Example 5. If the user-specified $minWS = 150$, then ab is not a weighted frequent itemset because $WS(ab) \ngeq minWS$. On the other hand, the itemset cd is a weighted frequent itemset because $WS(cd) \geq minWS$.

Definition 5. (**Spatial Weighted Frequent Itemset** X**.**) A weighted frequent itemset X is said to be a spatial weighted frequent itemset if the distance between any two items in X is no more than the user-specified maximum distance ($maxDist$). That is, X is a SWFI if $\forall i_a, i_b \in X, a \neq$

ts	Items
1	$abgf$
2	$acfg$
3	dfg
4	bcd
5	$bcde$
6	$abceg$

TABLE I: Temporal database

Items	location
a	$(0,0)$
b	$(3,4)$
c	$(3,-4)$
d	$(6,0)$
e	$(3,0)$
f	$(9,0)$
g	$(12,0)$

TABLE II: Spatial database

ts/Item	a	b	c	d	e	f	g
1	20	15	0	0	0	20	20
2	5	0	30	0	0	20	10
3	0	0	0	30	0	20	15
4	0	60	80	10	0	0	0
5	0	60	40	20	5	0	0
6	10	20	10	0	45	0	20

TABLE III: Items' weight database

$b, Dist(i_a, i_b) \leq maxDist$, where $Dist(.)$ is a distance function such as Euclidean distance.

Example 6. The Euclidean distance between c and d items, i.e., $Dist(c,d) = 5$. If the user-specified $maxDist = 5$, then the weighted frequent itemset cd is a spatial weighted frequent itemset because $Dist(c,d) \leq maxDist$. The complete set of SWFIs generated from Table I are shown in Table IV.

Definition 6. (**Problem Definition.**) Given a temporal database (TDB), items' weight database (WD) and items' spatial database (SD), the problem of spatial weighted frequent itemset mining involves discovering all itemsets in TDB that have weighted sum no less than the user-specified minimum weighted sum ($minWS$) and the distance between any two of its items is no more than the user-specified $maxDist$. It is interesting to note that WFIM is a special case of the problem SWFIM when $maxDist = \infty$ (or very large). For brevity, we have considered spatial items as points. However, the proposed model is generic and allows spatial items to be represented with other geometric forms such as lines and polygons.

B. A small discussion.

In our model, we have set a strict constraint that all items in an SWFI must be close (or neighbors) to one another. If we relax this constraint, then too many uninteresting itemsets with items far away from the rest can be generated as SWFIs. Example 7 illustrates the importance of employing a strict spatial constraint on SWFIs.

Example 7. Let $l = (0,0)$, $m = (2,0)$, $n = (4,0)$ and $o = (6,0)$ be four items located on a straight line. Let $maxDist = 2$. If we relax the constraint that all items in a SWFI need not be close to each other, then we may find $lmno$ as a SWFI. Unfortunately, this itemset may be uninteresting to the user as the items n and o are located far away from l.

To reduce the number of input parameters, the proposed model does not determine the interestingness of an itemset using $minSup$ constraint. However, if an application demands, the user can employ $minSup$ as an additional constraint to find SWFIs. Please note that significant changes are not needed for our SWFP-growth algorithm as it inherently records the $support$ information of an itemset.

IV. Proposed Algorithm

The space of items in a database gives rise to a subset lattice. The itemset lattice is a conceptualization of the search space when mining SWFIs. The proposed SWFP-growth is a variant of UP-growth [23], which performs a depth-first search on this itemset lattice to find all SWFIs in TDB. The reason for choosing pattern-growth algorithm over other algorithms (e.g., Apriori [3], Eclat [29], or LCM [24]) is because pattern-growth algorithms can be easily extended to develop disk-based algorithms and parallel algorithms [13]. Due to page limitation, this paper confines only to the sequential memory-based pattern-growth algorithm.

In this section, we first introduce the basic idea of SWFP-growth algorithm. Next, we describe the working of SWFP-growth using the database shown in Table I.

A. Basic idea

The *weighted sum* of an ordered itemset can be more or less than the *weighted sum* of its ordered superset. In other words, the SWFIs generated from the data do not satisfy the *convertible anti-monotonic*, *convertible monotonic*, or *convertible succinct properties* [20]. This increases the search space, which in turn increases the computational cost of finding the SWFIs. Two upper bound measures, called *optimized estimated weighted sum* ($OEWS$) and *cumulative neighborhood weighted sum* ($CNWS$), have been presented to reduce the search space and the computational cost. These two measures aim to identify itemsets (or items) whose supersets may yield SWFIs. We now describe each of these measures.

1) Optimized estimated weighted sum: The key objective of $OEWS$ measure is to identify items whose supersets may yield SWFIs. The items whose $OEWS$ value is no less than the user-specified $minWS$ are called as *candidate items*. Definitions 7 and 8 define the *estimated weighted sum* (EWS) of an itemset in a transaction and temporal database, respectively. Definitions 9 and 10 respectively define the candidate item and candidate itemsets. Pruning technique to remove itemsets whose supersets may not yield any SWFI is given in Property 1. Definition 11 defines the calculation of optimized EWS value of an item based on the prior knowledge regarding the pattern-growth technique.

Definition 7. (**Estimated Weighted Sum of an item i_j in a transaction.**) Let N_{i_j} denote the set of all neighbors of an item $i_j \in I$. That is, $\forall i_k \in N_{i_j}, dist(i_j, i_k) \leq maxDist$. The *estimated weighted sum* (EWS) of an item i_j in a transaction T_{ts}, denoted as $EWS(i_j, T_{ts})$, represents the sum

Itemset	$weighted\ sum$
c	160
b	155
cd	150
bd	150

TABLE IV: SWFIs generated from Table I at $minWS = 150$ and $maxDist = 5$

Item	Neighbours
a	bce
b	ade
c	ade
d	$bcef$
e	$abcd$
f	dg
g	f

TABLE V: Neighbors of each item at $maxDist = 5$

of weights of i_j and its neighboring items in T_{ts}. That is, $EWS(i_j, T_{ts}) = w(i_j, T_{ts}) + \sum_{i_k \in T_{ts}.Y \cap i_k \in N_{i_j}} w(i_k, T_{ts})$.

Example 8. Consider the item a in Table I. The neighbors of a, i.e., $N_a = \{bce\}$ (see Table V). The *estimated weighted sum* of a in T_1 is the sum of weights of a and its neighboring items in T_1. That is, $EWS(a, T_1) = w(a, T_1) + w(b, T_1) = 20 + 15 = 35$. Please note that the weights of remaining items (i.e., g and f) in T_1 are not used in the calculation of $EWS(a, T_1)$. It is because these two items are not neighbors of a. The above definition of EWS captures the maximum weighted sum of a and its neighboring items in a transaction. We now extend this definition by taking into account a set of transactions (or a temporal database).

Definition 8. (**EWS of an item in a temporal database).** Let TDB^{i_j} denote the set of all transactions containing i_j in TDB. The EWS of an item i_j in TDB, denoted as $EWS(i_j)$, represents the sum of *estimated weighted sum* of i_j in all transactions of TDB^{i_j}. That is, $EWS(i_j) = \sum_{T_k \in TDB^{i_j}} EWS(i_j, T_k)$.

Example 9. The transactions containing a in Table I are: T_1, T_2 and T_6. Therefore, $TBD^a = \{T_1, T_2, T_6\}$. The EWS of a in T_1, i.e., $EWS(a, T_1) = 35$. Similarly, $EWS(a, T_2) = 35$ and $EWS(a, T_6) = 85$. The EWS of a in the entire database, i.e., $EWS(a) = EWS(a, T_1) + EWS(a, T_2) + EWS(a, T_6) = 35+35+85 = 155$. In other words, $EWS(a)$ provide the information that an item a with all its neighboring items has resulted in a maximum weighted sum of 155 $\mu g/m^3$ in the entire database. Henceforth, this value can be used as a upper-bound constraint to identify candidate items whose supersets may yield SWFIs. The above definition captures the maximum weighted support an item and its supersets (constituting of its neighboring items) can have in the entire spatiotemporal database with respect to its neighboring items. Thus, EWS acts as a weighted sum upper bound on the items. For an item $i_j \in I$, if $EWS(i_j) < minWS$, then neither i_j nor its supersets will result in SWFIs. So only those items whose EWS is no less than $minWS$ will generate SWFIs at higher order. We call these items as candidate items and defined in Definition 9.

Definition 9. (**Candidate item.**) An item i_j in TDB is said to be a candidate item if $EWS(i_j) \geq minWS$.

Example 10. Continuing with the previous example, the item a in Table I is a candidate item because $EWS(a) \geq minWS$.
We now generalize the above definition by taking into account the notion of itemset. This generalization facilitates uses to push the above pruning technique to the lower levels of itemset lattice.

Definition 10. (**Candidate itemset.**) Let α be a suffix itemset. Let $TDB^{\alpha} \subseteq TDB$ be the conditional pattern base (or projected database) of α. (If $\alpha = \emptyset$, then $TDB^{\alpha} = TDB$.) Let $WS(\alpha)$ be the weighted sum of α in TDB. Let i_j be an item in TDB^{α}. Let $\widehat{EWS(i_j)}$ denote the EWS value of an item i_j in $TDB^{\alpha \cup i_j}$. If $\widehat{EWS(i_j)} + WS(\alpha) \geq minWS$, then $\alpha \cup i_j$ is a candidate itemset (or i_j is a candidate item in TDB^{α}). Otherwise, i_j is an uninteresting item that can be pruned from TDB^{α}. The proposed SWFP-growth employs the above definition to identify candidate itemsets whose supersets may yield SWFIs.

Property 1. **(Pruning technique).** For an itemset X, if $EWS(X) \leq minWS$, then neither X nor its supersets can be SWFIs.

Definition 11. **(Calculating the optimized EWS value of an item using the prior knowledge regarding the pattern-growth technique).** In the pattern-growth technique, the *conditional pattern base* (or CPB) of a suffix item does not include any previous suffix items. For example, let a, b, c and d be the sorted list of items in a lexicographical order. In the pattern-growth technique, the search space of finding SWFIs from these four items can be divided into four smaller search spaces: (i) d's conditional pattern base (or d-CPB), (ii) c-CPB excluding d (which is after c in the sorted list), (iii) b-CPB excluding c and d and (iv) a-CPB excluding b, c and d. Thus, given a sorted transaction, $\widehat{T_k} = (ts, \{i_1, i_2, \cdots, i_k\})$, the optimized EWS value of an item i_p in $\widehat{T_k}$, denoted as $OEWS(i_p, \widehat{T_k})$, is the summation of weighted sum of i_j and neighboring items before i_p in $\widehat{T_k}$. That is, $OEWS(i_p, \widehat{T_k}) = w(i_p, \widehat{T_k}) + \sum_{i_a \in \{i_p\text{-}CPB \cap N_{i_p}\}} w(i_a, \widehat{T_k})$, where i_p-CPB denote the set of items that include in the conditional pattern base of i_p and N_{i_p} represent the neighboring items of i_p.

Example 11. Let us consider the first transaction T_1 in Table I. The lexicographical sorted order of items in this transaction is $abfg$. Let us consider the item g, which is the last item in the sorted transaction. The *conditional pattern base* of g, i.e., g-$CPB = \{abf\} \cap N_g = \{abf\} \cap \{f\} = \{f\}$. Therefore, the EWS of g in T_1, i.e., $OEWS(g, T_1) = w(g, T_1) + w(f, T_1) = 20 + 20 = 40$. Similarly, for the item f, f-$CPB = \{ab\}$ and $N_f = \{dg\}$. The $OEWS$ of f in T_1, i.e., $OEWS(f, T_1) = w(f, T_1) + \sum_{i_k \in \{f\text{-}CPB \cap N_f\}} w(i_k, T_1) = w(f, T_1) = 20$.

Property 2. For an itemset X, $EWS(X, \widehat{T_k}) \geq OEWS(X, \widehat{T_k})$. In other words, $OEWS$ is the more tighter constraint than EWS.

The SWFP-growth employs EWS measure to find candidate items. After finding candidate items and sorting them with respect to EWS descending order, items' $OEWS$ values in every transaction are used to find candidate itemsets effectively.

2) Cumulative neighborhood weighted sum: The candidate items constitute of both weighted frequent items and uninteresting items whose supersets may generate SWFIs. We have observed that constructing projected databases (or conditional pattern bases) for all uninteresting items is a costly operation. In this context, we exploit another weight upper bound measure, called *cumulative neighborhood weighted sum* ($CNWS$), to identify those candidate items whose projections will only SWFIs.

Definition 12. (Cumulative neighborhood weighted sum) Let $S = \{i_1, i_2, \cdots, i_k\} \subseteq I$ be an ordered list of candidate items such that $EWS(i_1) \leq EWS(i_2) \leq \cdots \leq EWS(i_k)$. The *cumulative neighborhood weighted sum* of an item $i_j \in S$, denoted as $EWS(i_j)$, is the sum of weighted sum of remaining items in the list which are neighbors of i_j. That is, $CNWS(i_j) = \sum_{p=j+1}^{|S|} WS(i_p)$ if $i_p \in N(i_p)$. For the last item in S, $cnws(i_k) = 0$.

Example 12. Let us order the candidate items in increasing order of their EWS values. Let $\succ$ denote this order of items. The candidate items in $\succ$ order are a, e, c, b and d. Let us consider item a, which is the first item in $\succ$ order. The neighbors of this item are b, c and e (see Table V). Thus, the item a will generate SWFIs by combining with the items b, c and e. Thus, the *cumulative neighborhood weighted sum* of a, i.e., $CNWS(a) = WS(b) + WS(c) + WS(e) = 365$. The $CNWS$ of a provides the crucial information that the item a and its supersets containing only a's neighborhood items can at most have the maximum weighted sum of 365 in the entire database. This information can be used to determine whether a suffix item in the tree needs to be projected or not. If sum of weighted support of $suffixitemset$ and $CNWS$ of a suffix itemset is less than the user-specified $minWS$, then we can prevent the depth-first search (or construction of conditional pattern bases) to find SWFIs. Thus, significantly reducing the search space.

Property 3. (**Additive property.**) For an itemset X, $WS(X) \leq \sum_{i_j \in X} WS(i_j)$.

B. SWFP-growth

The proposed SWFP-growth algorithm is presented in Algorithms 1 and 2. Briefly, SWFP-growth algorithm involves the following steps: (i) finding candidate items (ii) constructing Spatial Weighted Frequent Pattern-tree (SWFP-tree) by compressing the spatiotemporal database using candidate items (iii) Recursively mining SWFP-tree to find all candidate itemsets and (iv) finding all SWFIs from candidate itemsets by performing another scan on the spatiotemporal database. Before we explain each of these steps, we describe the structure of SWFP-tree.

1) Structure of SWFP-tree: In SWFP-tree, each node N includes $N.name$, $N.support$, $N.oews$, $N.parent$, $N.hlink$ and a set of child nodes. The details are as follows. $N.name$ is the item name of the node. $N.support$ represents the $support$ of an item in node N. $N.oews$ represents the $OEWS$ value of an item in node N. $N.parent$ records the parent node of the node. $N.hlink$ is a node link which points to a node whose item name is the same as $N.name$.

Header table is employed to facilitate the travel of SWFP-tree. In this table, each entry is composed of an item name, $OEWS$ value, and a link. The link points to the last occurrence of the node which has the same item as the entry in the SWFP-tree. By following the link in the header table and the nodes in SWFP-tree, the nodes whose item names are the same can be traversed efficiently.

2) Finding candidate items: In the first database scan, we calculate the EWS, minimum weight sum and $weightedsum$ of each item in database TDB. The calculated EWS values for all items in Table I are shown in Fig. 1(a). From these items, the candidate items are generated by pruning all items that have EWS value less than the user-specified $minWS$. The candidate items are later sorted in descending order of their EWS value. Let this sorted list of candidate items be denoted as L. The sorted list of candidate items generated from Table I for the user-specified $minWS = 150$ is shown in Fig. 1(b). (The above process can be repeated until no more items get pruned from the temporal database. However, for computational reasons we recommend limiting this step to single scan on the database.)

3) Construction of SWFP-tree: Using the generated candidate items, we scan the temporal database for the second time and generate SWFP-tree by following the procedure similar to that Frequent Pattern-tree (or FP-tree). It has to be noted that we will maintaining both $support$ and $OEWS$ value of an item at each node.

The sorted transactional database constituting of only candidate items is shown in Fig. 1(c). The scan on the first sorted transaction, "1: *ba*," generates a branch $\langle b : 1 : 15 \rangle, \langle a : 1 : 35 \rangle$ (format is $\langle item : support : OEWS \rangle$). Fig. 2(a) shows the branch generated after scanning first transaction. The scan on the second sorted transaction, "2:ca," generates another branch $\langle c : 1 : 30 \rangle, \langle a : 1 : 35 \rangle$ (see Fig. 2(b)). Simiarl process is repeated for remaining transactions and SWFP-tree is updated accordingly. The final SWFP-tree generated after scanning entire temporal database is shown in Fig. 2(c). For brevity, we are not showing the node-links. However, they exist as in FP-tree.

4) Recursive mining of SWFP-tree: After constructing SWFP-tree, we start with the last item in the header table. Choosing this item as a suffix itemset, we determine its $CNWS$. If the sum of weighted support of the suffix item and its $CNWS$ value is more than the user-specified $minWS$, then we construct its conditional pattern base constituting of neighboring items of suffix itemset, construct its conditional SWFP-tree, and generate all candidate itemsets. If $CNWS$ value of a suffix item is less than the user-specified $minWS$,

Item	a	b	c	d	e	f	g
EWS	155	265	255	325	210	105	125

(a) EWS values for all items

Item	d	b	c	e	a
EWS	325	265	255	210	155

(b) Sorted list of candidate items

ts	Items
1	ba
2	ca
3	d
4	dbc
5	dbce
6	bcea

(c) temporal database

Fig. 1: Generating temporal database containing only candidate items

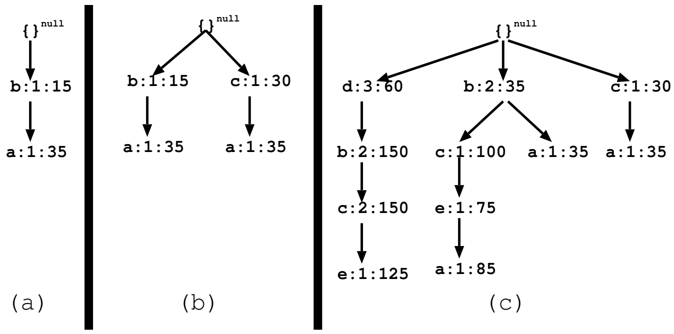


Fig. 2: Construction of SWFP-tree. (a) After scanning first transaction (b) After scanning second transaction and (c) After scanning the entire database

then we skip the construction of conditional pattern bases and move to the next item in the header table. Similar process is repeated for the other items in the header table.

5) Generating all SWFIs from candidate itemsets: After finding all candidate items from SWFP-tree, we perform third scan on the database and calculate actual weighted support for each candidate itemset. The candidate itemset that has weighted support no less than the user-specified $minWS$ will be generated as SWFI. The complete set of SWFIs generated from Table I for the user-specified $minWS$ of 150 is shown in Table IV.

V. Experimental Results

Since there exists no algorithm to mine SWFIs in a binary spatiotemporal database, we only evaluate the proposed algorithm using various databases. The SWFP-growth algorithm has been written in java and executed on i7 1.5 GHz processor having 8GB of memory. The experiments have been conducted using synthetic (T10I4D100K) and real-world (Retail, Chess and PM2.5) databases.

The **T10I4D100K** [3] is a sparse synthetic database, which is widely used for evaluating various pattern mining algorithms. This transactional database is converted into a temporal database by considering tids as timestamps. A spatial database for all the items in T10I4D100K has been generated by assigning random coordinates between $(0, 0)$ to $(100, 100)$. The coordinates of these items in a Cartesian coordinate system is shown in Fig. 3a. It can be observed that items have non-uniformly spread throughout the region. The statistical details of this database were provided in Table VI.

The **Retail** is a sparse real-world transactional database, which is widely used for evaluating various pattern mining algorithms. This database is converted into a temporal database by considering $tids$ as timestamps. A spatial database for all the items has been generated by assigning random coordinates between $(0, 0)$ to $(200, 200)$. The coordinates of these items in a Cartesian coordinate system is shown in Fig. 3b. It can be observed that items have non-uniformly spread throughout the region. The statistical details of this database were provided in the third row of Table VI.

AEROS consists of several air pollution measuring stations located throughout Japan. Each station measures several air pollution concentrates (e.g., NO, NO_2, PM2.5 and SO_2) over hourly intervals. In this paper, we only consider PM2.5 pollution concentrate. The pollution data is generated at 1 hour time interval for 24 hours of a day. For our experiments, we are using air pollution data of 6 months (i.e., from 01-12-2018 to 04-06-2019). The PM2.5 database contained 5366157 data points and 1065 items (or station ids). UTC time is used to record the transactions. Without loss of generality, the pollution database was split into a temporal database, spatial database and items weight database. PM2.5 is a **dense high dimensional** database. The statistical details of this database are shown in Table VI.

The **Chess** is a dense real-world transactional database, which is widely used for evaluating various pattern mining algorithms. This database is converted into a temporal database by considering $tids$ as timestamps. A spatial database for all the items has been generated by assigning random coordinates between $(0, 0)$ to $(20, 20)$. The coordinates of these items in a Cartesian coordinate system is shown in Fig. 3d. It can be observed that items have non-uniformly spread throughout the region. The statistical details of this database were provided in the fourth row of Table VI.

TABLE VI: Statistics of the datasets

Database	Type	Items	Size	Transaction length		
				min.	avg.	max.
T10I4D100K	sparse	870	4.5 MB	1	10	29
Retail	sparse	16470	4.6 MB	1	10	76
PM2.5	dense	1065	30.1 MB	50	950	1055
Chess	dense	75	354 KB	37	37	37

Figs. 4a, 4b, 4c and 4d show the number of SWFIs generated in T10I4D100K, Retail, PM2.5 and Chess databases at different $minWS$ and $maxDist$ values, respectively. The following observations can be drawn from these two figures : (i) increase in $minWS$ causes a decrease in SWFIs as many itemsets fail to satisfy the increased $minWS$ value and (ii) increase in $maxDist$ causes increase in SWFIs as higher $maxDist$ facilitates the items to increase their neighborhood sizes. It can be observed that at higher $maxDist$ values, too many SWFIs are getting generated. It is because of the increase in neighborhood size facilitates items to combine with far away items and generate SWFIs. Many SWFIs generated at high $maxDist$ may found to be uninteresting to the users.

S.No.	Pattern	WS	Location
1	{5587,5605,5611,5617,5624}	154,583	Sapporo
2	{4249,4255,4275,4282,4331,4348,-4354,4391,4396}	381,348	Tokyo
3	{2079,2091,2102,2106}	164,538	Osaka
4	{1197,1229,1265,1270}	198,402	Okayama

TABLE VII: Some of the interesting SWFIs generated in pollution database

Figs. 5a, 5b, 5c and 5d show the memory requirements of SWFP-growth (in megabytes) on T10I4D100K, Retail, PM2.5 and Chess databases at different $minWS$ and $maxDist$ values, respectively. The following observations can be drawn from these two figures : (i) increase in $minWS$ results in the decrease of memory as relatively less number of SWFIs get generated and (ii) increase in $maxDist$ results in increase of memory required to find SWFIs. It is because a large number of SWFIs get generated at higher $maxDist$ values.

Figs. 6a, 6b, 6c and 6d show the runtime requirements of SWFP-growth algorithm on T10I4D100K, Retail, PM2.5 and Chess databases at different $minWS$ and $maxDist$ values, respectively. The following observations can be drawn from these two figures : (i) increase in $minWS$ results in a decrease of runtime as fewer SWFIs are getting generated and (ii) increase in $maxDist$ results in the increase of runtime.

1) A case study: identifying highly polluted regions of PM2.5: Table VII shows the SWFIs generated in the PM2.5 database at $maxDist = 5$ kilometers and $minWS = 10,000\mu g/m^3$. The spatial location of these stations is shown in Fig. 7(a). The spatial location of the sensors present in each spatial weighted frequent itemset are shown in Fig. 7(b). These patterns indicate the geographical areas where people have been exposed to high levels of PM2.5 pollutant. This information can be found very useful in devising policies to control pollution.

VI. Conclusion

In this paper, we have introduced a flexible model of spatial weighted frequent itemset that exist in a spatiotemporal database. Two novel measures have been introduced to reduce the search space effectively. A pattern-growth algorithm has also been presented to find all desired itemsets in a spatiotemporal database. Experimental results demonstrate that the proposed algorithm is efficient. Finally, we have also demonstrated the usefulness of the proposed model with a real-world case study on air pollution data.

In this paper, we have studied the problem of finding SWFIs by taking into account positive weights for the items in a spatiotemporal database. As a part of future work, we would like to investigate finding SWFIs in a spatiotemporal database using both positive and negative weights for the items. Additionally, we would like to investigate disk-based and parallel algorithms to find SWFIs.

Algorithm 1 SWFP-tree (TDB: temporal database, I: items in a database, SD: spatial database, WD: weight database, $minWS$: minimum weighted sum, $minDist$: minimum distance)

1: Scan the spatial database SD and identify neighbors for each item i_j in I. Let $N(i_j)$ denote the neighbors for item i_j in I.

2: Scan the database TDB and calculate EWS, WS and $minimumwieghts$ for each item i_j in I. Prune all items in I that have EWS less than the user-specified $minWS$. Consider the remaining items in I as candidate items and sort them in descending order of their EWS values. Let L denote this sorted list of candidate items.

3: Create the root node of SWFP-tree T and label it as "$null$". Scan the temporal database TDB for the second time and update SWFP-tree as follows. For each transaction $T_{ts} \in TDB$ do the following. Identify and sort the candidate items in T_{ts} in L order. Let $\widehat{T_{ts}}$ denote the sorted transaction of T_{ts} containing only candidate items. Let the sorted candidate item list in $\widehat{T_{ts}}$ be $[p|P]$, where p is the first element and P is the remaining list. Call *insert_tree(*$[p|P], T$*)*, which is performed as follows. If T has a child N such that $N.item\text{-}name = p.item\text{-}name$, then increment the $N.support$ value by 1, calculate the $OEWS$ value of p in $\widehat{T_{ts}}$ and add this value to the existing $N.oews$ value. If T has a child N such that $N.item\text{-}name \neq p.item\text{-}name$, then create a new node N, set its $support$ count to 1, calculate the $OEWS$ value of p in $\widehat{T_{ts}}$ and set this value as $N.oews$. Next, its parent link is linked to T, and its node-link to the nodes with the same *item-name* via the node-link structure. If P is non-empty, call *insert_tree(*P*,* N*)* recursively.

Algorithm 2 SWFP-growth

1: **input** : T_X: SWFP-tree, H_X: header table for T_X, X: an itemset

2: **output**: all candidate weighted frequent itemsets in T_X

3: **for** each $item\ a_i \in H_X$ **do**

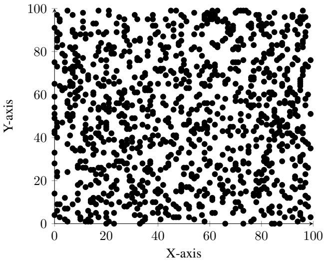
4: generate an itemset $Y = X \cup a_i$. The $EWS(Y)$ is set as $a_i.oews$ in H_X.

5: if WeightedSum(Y) + $CNWS(a_i)$ is no less than $minWS$ then construct Y's conditional pattern base constituting of only neighbors of a_i. Next, recalculate each node's $oews$ value. Consider items having $oews$ value greater than $minWS$ as candidate items in $Y\text{-}CPB$ and put them in H_Y. Readjust the $oews$ values for the items by removing non-candidate items in $Y\text{-}CPB$. Create a new tree T_Y by calling *insert_tree(*$[p|P], T_Y$*)*. If $T_y \neq null$, call $SWFP-growth(T_Y, H_Y, Y)$.

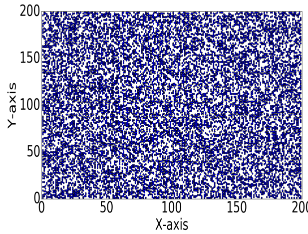
6: **end for**

References

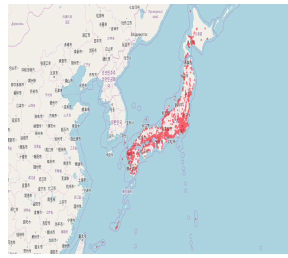
[1] Aggarwal, C.C., Han, J.: Frequent Pattern Mining. Springer Publishing Company, Incorporated (2014)



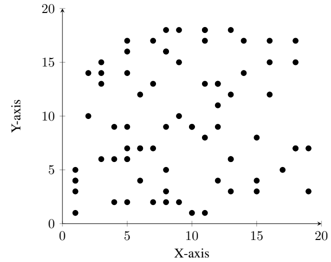
(a) T10I4D100K



(b) Retail

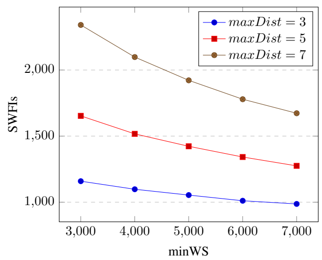


(c) PM2.5

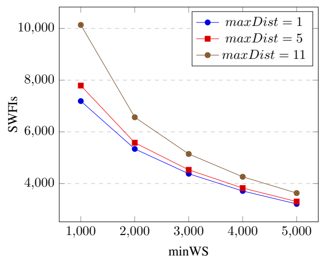


(d) Chess

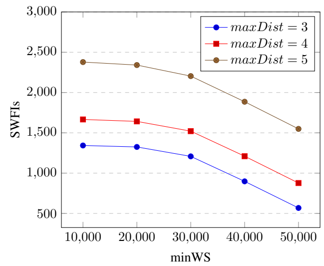
Fig. 3: Spatial visualization of items in various databases



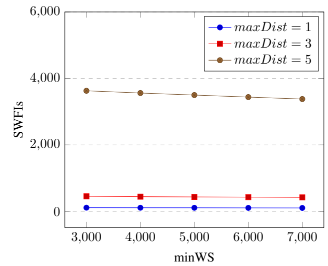
(a) T10I4D100K



(b) Retail

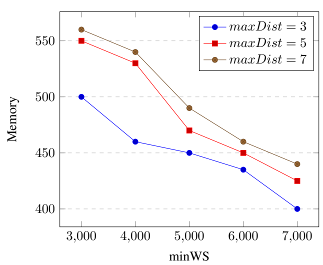


(c) PM2.5

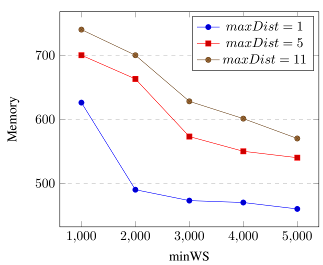


(d) Chess

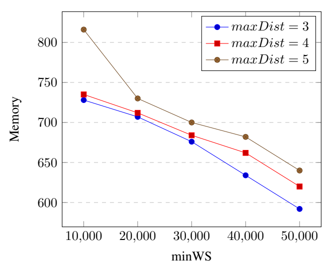
Fig. 4: SWFIs generated by SWFP-growth algorithm at different $minWS$ and $maxDist$ values in various databases



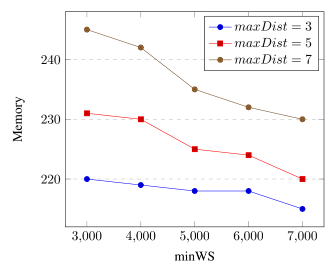
(a) T10I4D100K



(b) Retail



(c) PM2.5



(d) Chess

Fig. 5: Memory requirements of SWFP-growth in various databases at different $minWS$ and $MaxDist$ values

[2] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Acm sigmod record. vol. 22, pp. 207–216 (1993)

[3] Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. vol. 1215, pp. 487–499 (1994)

[4] Ahmed, C., Tanbeer, S., Jeong, B.S., Lee, Y.K.: Mining weighted frequent patterns using adaptive weights. Intelligent Data Engineering and Automated Learning–IDEAL 2008 pp. 258–265 (2008)

[5] Cai, C.H., Fu, A.W.C., Cheng, C., Kwong, W.: Mining association rules with weighted items. In: Database Engineering and Applications Symposium, 1998. Proceedings. IDEAS'98. International. pp. 68–77. IEEE (1998)

[6] Cai, J.H., Zhao, X.J., Sun, S.W., Zhang, J.F., Yang, H.F.: Stellar spectra association rule mining method based on the weighted frequent pattern tree. Research in Astronomy and Astrophysics **13**(3), 334 (2013)

[7] Cheng, H., Han, J.: Pattern-growth methods. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 1–5. Springer New York, New York, NY (2016)

[8] Ding, W., Eick, C.F., Wang, J., Yuan, X.: A framework for regional association rule mining in spatial datasets. In: Proceedings of the Sixth International Conference on Data Mining. pp. 851–856. ICDM '06 (2006)

[9] Duong, Q., Fournier-Viger, P., Ramampiaro, H., Nørvåg, K., Dam, T.: Efficient high utility itemset mining using buffered utility-lists. Appl. Intell. **48**(7), 1859–1877 (2018)

[10] Eick, C.F., Parmar, R., Ding, W., Stepinski, T.F., Nicot, J.P.: Finding regional co-location patterns for sets of continuous variables in spatial datasets. In: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 30:1–30:10. GIS '08 (2008)

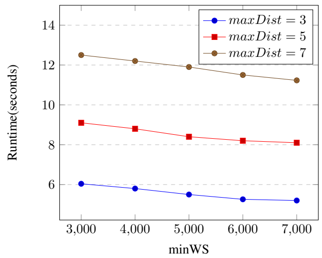
[11] Gan, W., Lin, J.C., Fournier-Viger, P., Chao, H., Hong, T., Fujita, H.: A survey of incremental high-utility itemset mining. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **8**(2) (2018)

[12] Kiran, R.U., Kotni, A., Reddy, P.K., Toyoda, M., Bhalla, S., Kitsuregawa, M.: Efficient discovery of weighted frequent itemsets in very large transactional databases: A re-visit. In: IEEE BigData. pp. 723–732 (2018)

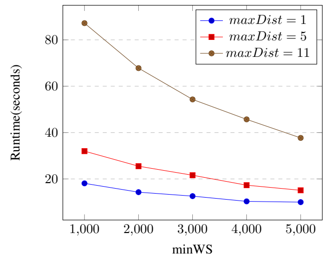
[13] Kiran, R.U., Kotni, A., Reddy, P.K., Toyoda, M., Bhalla, S., Kitsuregawa, M.: Efficient discovery of weighted frequent itemsets in very large transactional databases: A re-visit. In: IEEE Big Data. pp. 723–732 (2018)

[14] Kiran, R.U., Reddy, T.Y., Fournier-Viger, P., Toyoda, M., Reddy, P.K., Kitsuregawa, M.: Efficiently finding high utility-frequent itemsets using cutoff and suffix utility. In: Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part II. pp. 191–203 (2019)

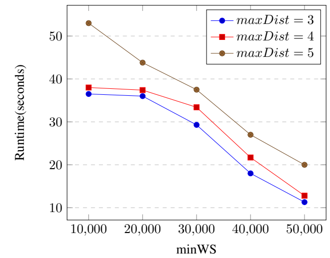
[15] Kiran, R.U., Zettsu, K., Toyoda, M., Fournier-Viger, P., Reddy, P.K.,



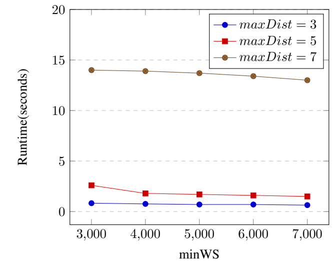
(a) T10I4D100K



(b) Retail

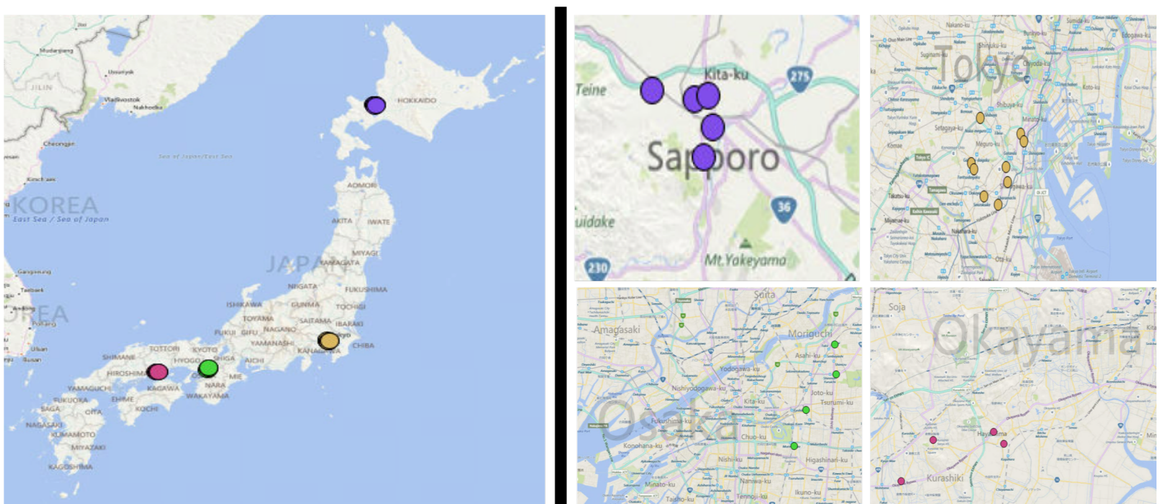


(c) PM2.5



(d) Chess

Fig. 6: Runtime requirements of SWFP-growth in various databases at different $minWS$ and $MaxDist$ values



(a) (b)

Fig. 7: Spatial location of sensors that have measured high levels of PM2.5. (a) Japan and (b) Zoomed pictures

Kitsuregawa, M.: Discovering spatial high utility itemsets in spatiotemporal databases. In: Proceedings of the 31st International Conference on Scientific and Statistical Database Management. pp. 49–60. SSDBM '19 (2019)

[16] Lin, J.C.W., Gan, W., Fournier-Viger, P., Chao, H.C., Hong, T.P.: Efficiently mining frequent itemsets with weight and recency constraints. Applied Intelligence pp. 1–24 (2017)

[17] Lin, J.C.W., Gan, W., Fournier-Viger, P., Hong, T.P., Tseng, V.S.: Weighted frequent itemset mining over uncertain databases. Applied Intelligence **44**(1), 232–250 (2016)

[18] Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 337–341. ACM (1999)

[19] Mohan, P., Shekhar, S., Shine, J.A., Rogers, J.P., Jiang, Z., Wayant, N.: A neighborhood graph based approach to regional co-location pattern discovery: A summary of results. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 122–132. GIS '11 (2011)

[20] Pei, J., Han, J.: Can we push more constraints into frequent pattern mining? In: KDD. pp. 350–354. ACM (2000)

[21] Tao, F., Murtagh, F., Farid, M.: Weighted association rule mining using weighted support and significance framework. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 661–666. ACM (2003)

[22] Tran-The, H., Zettsu, K.: Discovering co-occurrence patterns of heterogeneous events from unevenly-distributed spatiotemporal data. In: 2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017. pp. 1006–1011 (2017)

[23] Tseng, V.S., Shie, B.E., Wu, C.W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. on Knowl. and Data Eng. **25**(8), 1772–1786 (Aug 2013)

[24] Uno, T., Asai, T., Uchida, Y., Arimura, H.: LCM: an efficient algorithm for enumerating frequent closed item sets. In: FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA (2003)

[25] Vo, B., Coenen, F., Le, B.: A new method for mining frequent weighted itemsets based on wit-trees. Expert Systems with Applications **40**(4), 1256–1264 (2013)

[26] Weiss, G.M.: Mining with rarity: a unifying framework. SIGKDD Explorations **6**(1), 7–19 (2004)

[27] Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: SIAM. pp. 482–486 (2004)

[28] Yun, U., Leggett, J.J.: Wfim: weighted frequent itemset mining with a weight range and a minimum weight. In: Proceedings of the 2005 SIAM International Conference on Data Mining. pp. 636–640. SIAM (2005)

[29] Zaki, M.J.: Scalable algorithms for association mining. IEEE Trans. Knowl. Data Eng. **12**(3), 372–390 (2000). https://doi.org/10.1109/69.846291, https://doi.org/10.1109/69.846291

[30] Zhang, C., Almpanidis, G., Wang, W., Liu, C.: An empirical evaluation of high utility itemset mining algorithms. Expert Syst. Appl. **101**, 91–115 (2018)