Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Efficient Discovery of Weighted Frequent Neighborhood Itemsets in Very Large Spatiotemporal databases

R. UDAY KIRAN[1,2], P. P. C. REDDY[3], K. ZETTSU[1], MASASHI TOYODA[2], MASARU KITSUREGAWA[2,4] AND P. KRISHNA REDDY[3],

[1]National Institute of Information and Communications Technology, Tokyo, Japan (e-mail: {uday_rage,zettsu}@nict.go.jp)

[2]The University of Tokyo, Tokyo, Japan (e-mail: {uday_rage,toyoda,kitsure}@tkl.iis.u-tokyo.ac.jp)

[3]International Institute of Information Technology-Hyderabad, Telangana, India (e-mail: pradeepchandra.p@research.iiit.ac.in, pkreddy@iiit.ac.in)

[4] National Institute of Informatics, Tokyo, Japan

Corresponding author: R. Uday kiran (e-mail: uday_rage@tkl.iis.u-tokyo.ac.jp).

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 Weighted Frequent Neighborhood Itemsets (WFNI) that may exist in a spatiotemporal database. The recommended patterns may be found very useful in many real-world applications. For instance, an WFNI 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 WFNIs 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 WFNIs 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, weighted frequent itemset, pattern-growth technique, spatiotemporal database

I. INTRODUCTION

FREQUENT Itemset Mining (FIM) is an important data mining model [1]–[3] with many real-world applications [4]. 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 in 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 in 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 [5]. When confronted with this problem in real-world applications, researchers have tried to find frequent itemsets using multiple $minSups$ [6], [7], 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. [8] 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) that satisfy the user-specified weight constraint in a transactional database. 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 [8], spectral signature analytics in astronomical databases [9], and finding events in Twitter data [10].

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 inadequate to find only those WFIs that have items close (or neighbors) to one another in a spatiotemporal database. A naïve approach to tackle this problem involves discovering all WFIs from the data and pruning the WFIs whose items are not neighbors to each others. Unfortunately, this approach is inefficient due to its huge search space and the computational cost. With this motivation, this paper introduces the model of Weighted Frequent Neighborhood Itemsets (WFNI) that may exist in a spatiotemporal database. Before we describe the contributions of this paper, we discuss the usefulness of the proposed itemsets with a real-world application.

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 stations has been set up by the Ministry of Environment, Japan. The data generated by these stations represent a non-binary spatiotemporal database. An WFNI found in this pollution database provides the information regarding the geographical region (or a set of neighboring stations) where people have been 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 (HUIM) [11]–[13] generalizes WFIM (respectively, FIM) by taking into account the items' internal utility and external utility values. However, discovering WFIs (respectively, frequent itemsets) using a HUIM algorithm is inefficient due to the additional cost of transforming a binary spatiotemporal database into a non-binary spatiotemporal database. (This topic is further discussed in latter parts of this paper).

This paper proposes a more flexible model of WFNI that may exist in a spatiotemporal database. An itemset in a spatiotemporal database is considered as an WFNI if it satisfies the user-specified minimum weighted sum and maximum distance constraints. The generated WFNIs 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 search space and the computational cost of finding the desired itemsets. EWS aims to identify candidate items whose supersets may be WFNIs. $CNWS$ seeks to identify those items that have to be projected (or build conditional pattern bases) to find all WFNIs. A pattern-growth algorithm, called Spatial Weighted Frequent Pattern-growth (SWFP-growth), has also been presented to find all WFNIs in a spatiotemporal database efficiently. Experimental results demonstrate that SWFP-growth is not only memory and runtime efficient, but also scalable as well. We also describe a case study in which we apply our model to find useful information in air pollution database.

Reddy et al. [14] proposed the model of WFNI by taking into account the items as points. This paper generalizes the model of WFNI by taking into account items of any geometric form (e.g,, point, line, or polygon). We will also provide the correctness of our algorithm. Furthermore, we strengthen the paper with extensive experiments and describe the real-world application of the proposed model using air pollution database.

The remainder of this paper is organized as follows. Section 2 discusses the previous literature related to the problem. Section 3 introduces the proposed model of WFNI that may exist in a spatiotemporal database. 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. FREQUENT ITEMSET MINING

Frequent itemsets are an important class of regularities that exist in databases. Since it was first introduced in [2], the problem of finding these itemsets has received a great deal of attention. Several algorithms (e.g., Apriori [2], ECLAT [15] and Frequent Pattern-Growth (FP-growth) [3], [16]) have been described in the literature to find frequent itemsets. Though there exists no universally acceptable best algorithm to find frequent itemsets in any database, FP-growth is widely accepted as the best algorithm to mine frequent itemsets in real-world databases [17]. Consequently, several extensions of FP-growth using GPUs, disks and parallel processing have been discussed to find frequent itemsets efficiently.

FP-growth is a depth-first search algorithm that discovers frequent patterns using pattern-growth technique. The pattern-growth technique briefly involves the following two steps: (i) compress the database into a $tree$, and (ii) recursively mine the entire $tree$ to find all frequent itemsets. We also employ a pattern-growth based algorithm to find all WFNIs in a spatiotemporal database. However, it has to be noted that the $tree$ structure and the mining procedure of our algorithm are different from that of the FP-growth algorithm.

B. WEIGHTED ITEMSET MINING

Cai et al. [8] 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 [18] discussed a pattern-growth algorithm, called WFIM, to find the weighted frequent itemsets. Uday et al. [10] described an improved WFIM based on the concept of cutoff weight, which represents the maximum weight among all weighted items.

Cai et al. [9] 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. [19] 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. [20] 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. [21] 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. [22] extended the basic weighted frequent itemset model [8] to handle uncertain databases. Chowdhury et al. [23] 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 WFNIs in spatiotemporal databases by taking into account the spatiotemporal characteristics of the items within the data.

C. HIGH UTILITY ITEMSET MINING

Yao et al. [13] 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 [11], [12], [24], [25]. As HUIM generalizes WFIM (respectively FIM), WFIs (respectively, FIs) can be generated using a HUIM algorithm. **This paper argues that such an approach to finding WFIs using HUIM algorithms is inefficient** because of two main reasons:

1) To employ a HUIM algorithm, we need to transform the binary transactional database into a non-binary transactional database by adding one as the internal utility for every item in a transaction. This 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 substantial larger (approximately 1.5 to 2 times) than the actual size of a binary database. Consequently, HUIM algorithms have to find WFIs from much larger databases consuming more memory and runtime.

In practice, a WFIM algorithm (respectively, FIM algorithm) is generally faster than a 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. [26] 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 WFNIs using SHUIMiner turns out to be costly due to the above mentioned reasons.

D. 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 [27]–[30]. These algorithms can be broadly classified into distance-based approaches [27], [28] and transaction-based approaches [29], [30]. 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 WFNIs in a spatiotemporal database is novel and distinct from current studies.

III. PROPOSED MODEL

Without loss of generality, a spatiotemporal database can be represented as a spatial database and a temporal database. For brevity, we first describe the neighborhood itemset using a spatial database. Next, we introduce weighted frequent neighborhood itemset using a temporal database and items' weight database.

A. NEIGHBORHOOD ITEMSET

Let $I = \{i_1, i_2, \cdots, i_n\}$, $n \geq 1$, be a set of geometric (or spatial) items. Let P_{i_j} denote a set of coordinates for an item $i_j \in I$. The spatial database SD is a collection of items and their coordinates. That is, $SD = \{(i_1, P_{i_1}), (i_2, P_{i_2}), \cdots, (i_n, P_{i_n})\}$. The above notion of spatial database facilitates us to capture items of various geometric forms, such as point, line, or polygon. Two items, $i_p, i_q \in I$, are said to be **neighbors** to each other if $Dist(i_p, i_q)(= Dist(i_q, i_p)) \leq maxDist$, where $Dist(.)$ is

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

(a)

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

(b)

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

(c)

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

(d)

Itemset	WS
c	160
b	155
cd	150
bd	150

(e)

TABLE 1: Running example. (a) spatial database, (b) neighbors of an item, (c) temporal database, (d) items' weight database, and (e) weighted frequent neighborhood itemsets

a distance function and $maxDist$ is a user-specified maximum distance.

Example 1. Let $I = \{a, b, c, d, e, f, g\}$ be the set of items (or air pollution monitoring station identifiers). A spatial database of these items is shown in Table 1a. Given the distance measure as Euclidean, the distance between the items c and d, i.e., $Dist(c, d) = 5$. If the user-specified $maxDist = 5$, then c and d are considered as neighbors because $Dist(c, d) \leq maxDist$. Table 1b lists the neighbors of every item in Table 1a.

Definition 1. (Neighborhood itemset.) Let $X \subseteq I$ be an itemset (or a pattern). If X contains k items, then it is called a k-itemset. An itemset X in SD is said to be a **neighborhood itemset** if the maximum distance between any two of its items is no more than the user-specified $maxDist$. That is, X is a neighborhood itemset if $max(Dist(i_p, i_q)|\forall i_p, i_q \in X) \leq maxDist$.

Example 2. The set of items c and d, i.e., cd is an itemset. This itemset contains two items. Therefore, it is a 2-itemset. The itemset cd is also a neighborhood itemset because $max(Dist(a, b)) \leq maxDist$.

Several distance functions (e.g. Euclidean distance and Geodesic distance) have been described in the literature to compute the distance between the items. Selecting a right distance function depends on the user and/or application requirements. In our example, we have represented spatial items with points and employed Euclidean as the distance function for brevity. However, our model is generic and can be employed with any distance function that satisfies the commutative property (see Property 1) and anti-monotonic property (see Property 2). We now define weighted frequent neighborhood itemset using temporal database and items' weight database.

Property 1. (Commutative property.) $Dist(i_a, i_b) = Dist(i_b, i_a)$.

Property 2. (Anti-monotonic property). If $X \subset Y$, then the maximum distance between any two items in X will always be less than or equal to the maximum distance between any two items in Y. That is, $max(Dist(i_p, i_q)|\forall i_p, i_q \in X) \leq max(Dist(i_r, i_s)|\forall i_r, i_s \in Y)$.

B. WEIGHTED FREQUENT NEIGHBORHOOD 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, $WD = \bigcup_{i_j \in I} W(i_j)$.

Example 3. Continuing with the previous example, a temporal database generated by all items in Table 1a is shown in Table 1c. The items' weight database is shown in Table 1d. Each transaction in this database represents the measurement of an air pollutant, say PM2.5[1], determined by a weather station for a particular time period. The weight of an item c in the second transaction, i.e., $w(c, T_2) = 30$. In other words, station d located at $(3, -4)$ has recorded $30\mu g/m^3$ of PM2.5 at the timestamp of 2.

Definition 2. (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 4. The itemset $cd \subseteq T_4.bcd$. Thus, the fourt transaction contains the itemset cd. Similarly, the fifth transaction also contains the itemset cd. The set of all transactions containing cd in Table 1c, i.e., $TDB^{cd} = \{T_4, T_5\}$. The $support$ of cd in Table 1c, i.e., $S(cd) = |TDB^{cd}| = 2$.

Definition 3. (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 $X \nsubseteq T_k.Y$, then $WS(X, T_k) = 0$.

Example 5. The weighted sum of cd in T_4, i.e., $WS(cd, T_4) = w(c, T_4) + w(d, T_4) = 80 + 10 = 90$. It means

[1]PM2.5 refers to the particle matter of size less than 2.5 microns. The unit of measurement for PM2.5 is $\mu g/m^3$.

the stations c and d have cumulatively recorded $90\mu g/m^3$ of PM2.5 at the timestamp 4.

Definition 4. **(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 6. The weighted sum of cd in Table 1c, i.e., $WS(cd) = \sum_{T_{ts} \in TDB^{cd}} WS(cd, T_{ts}) = WS(cd, T_4) + WS(cd, T_5) = (80 + 10) + (40 + 20) = 90 + 60 = 150$. It means the stations c and d have together recorded $150\mu g/m^3$ of PM2.5 in the entire data.

Definition 5. **(Weighted frequent neighborhood itemset X.)** A neighborhood itemset X is said to be a weighted frequent neighborhood itemset if $WS(X) \geq minWS$, where $minWS$ represents the user-specified minimum weighted sum.

Example 7. If the user-specified $minWS = 150$, then the neighborhood cd is a weighted frequent neighborhood itemset because $WS(cd) \geq minWS$. The complete set of WFNIs generated from the Tables 1a 1c and 1d are shown in Table 1e.

Definition 6. **(Problem Definition.)** Given a temporal database (TDB), items' weight database (WD) and items' spatial database (SD), the problem of Weighted Frequent Neighborhood Itemsets 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 WFNIM when $maxDist = \infty$ (or very large).

C. A SMALL DISCUSSION.

In our model, we have set a strict constraint that all items in an WFNI 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 WFNIs. Example 8 illustrates the importance of employing a strict spatial constraint on WFNIs.

Example 8. 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 WFNI need not be close to each other, then we may find $lmno$ as a WFNI. 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 WFNIs. Please note that significant changes are not needed for our SWFP-growth algorithm as it inherently records the $support$ information of an itemset.

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**

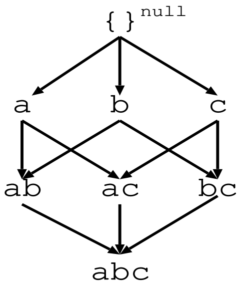


FIGURE 1: Itemset lattice of a, b and c

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 WFNIs. The itemset lattice of the items a, b and c is shown in Figure 1. The proposed SWFP-growth performs a depth-first search on this itemset lattice to find all WFNIs in the data. The main reason for choosing pattern-growth technique is due to the fact that algorithms based on this technique can be easily extended to develop disk-based algorithms and parallel algorithms [31]. In this paper, we confine 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 1c.

A. BASIC IDEA

The weighted sum of an ordered itemset can be more, less, or equal to the weighted sum of its ordered superset (see Property 3). Consequently, the WFNIs generated from the data do not satisfy the convertible anti-monotonic, convertible monotonic, or convertible succinct properties [32]. This increases the search space, which in turn increases the computational cost of finding the WFNIs. 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 WFNIs. We now describe each of these measures.

Property 3. If $X \subset Y$, then $WS(X) \geq WS(Y)$ or $WS(X) \leq WS(Y)$.

1) Optimized estimated weighted sum

The key objective of $OEWS$ measure is to identify items whose supersets may yield WFNIs. 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 WFNI is given in Property 4. Definition 11 defines the calculation of optimized EWS value of an item based on the prior knowledge regarding the pattern-growth technique.

TABLE 2: Neighborhoods of each item at $maxDist = 5$

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

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 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 9. Consider the item a in Table 1c. The neighbors of a, i.e., $N_a = \{bce\}$ (see Table 1b). 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 10. The transactions containing a in Table 1c 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 WFNIs. 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 WFNIs. So only those items whose EWS is no less than $minWS$ will generate WFNIs 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 11. Continuing with the previous example, the item a in Table 1c 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 WFNIs.

Property 4. (Pruning technique). For an itemset X, if $EWS(X) \leq minWS$, then neither X nor its supersets can be WFNIs.

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 WFNIs 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 12. Let us consider the first transaction T_1 in Table 1c. 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 5. 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 WFNIs. 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 WFNIs.

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 13. 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 1b). Thus, the item a will generate WFNIs 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 WFNIs. Thus, significantly reducing the search space.

Property 6. (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 WFNIs 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 1c are shown in Fig. 2(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 1c for the user-specified $minWS = 150$ is shown in Fig. 2(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. 2(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. 3(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. 3(b)). Similar process is repeated for remaining transactions and SWFP-tree is updated accordingly. The $tree$ constructed after scanning the last transaction is shown in Fig. 3(c). To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links. The final SWFP-tree generated after scanning entire temporal database is shown in Fig. 3(d).

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

FIGURE 2: Generating temporal database containing only candidate items

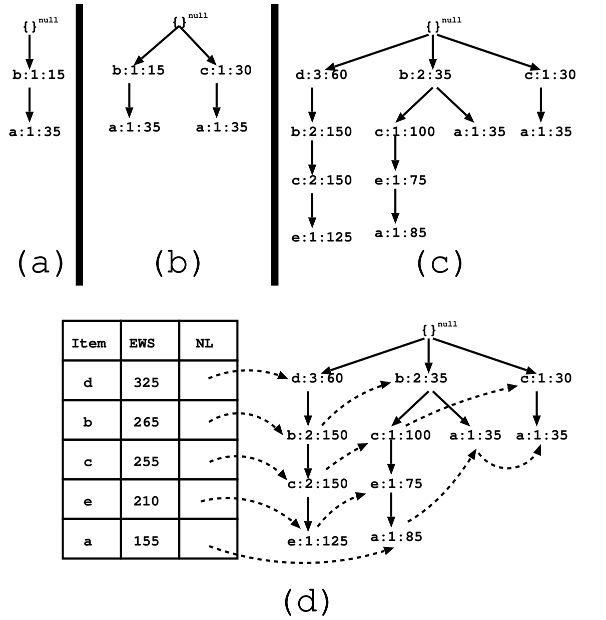


FIGURE 3: Construction of SWFP-tree. (a) After scanning first transaction (b) after scanning second transaction (c) after scanning the entire database and (d) final SWFP-tree

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

TABLE 3: Mining SWFP-tree

suffix item	conditional pattern base	conditional SWFP-tree	candidate itemsets
a	$\langle b, c, e : 1 : 85\rangle, \langle b : 1 : 35\rangle, \langle c : 1 : 35\rangle$	$-$	a
e	$\langle d, b, c : 1 : 125\rangle, \langle b, c : 1 : 75\rangle$	$\langle b, c : 2 : 200\rangle$	eb, ec, e
c	$\langle d : 2 : 150\rangle$	$\langle d : 2 : 150\rangle$	c, cd
b	$\langle d : 2 : 150\rangle$	$\langle d : 2 : 150\rangle$	b, bd

itemsets. If $CNWS$ value of a suffix item is less than the user-specified $minWS$, 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.

Mining of the SWFP-tree is summarized in Table 3 and defined as follows. We first consider a, which is the last item in the SWFP-list. Item a occurs in three branches of the SWFP-tree of Figure 3. (The occurrences of a an easily be found by following its chain of node-links.) The paths formed by these branches are $\langle b, c, e, a : 1 : 85\rangle$, $\langle b, a : 1 : 35\rangle$ and $\langle c, a : 1 : 35\rangle$. Therefore, considering a as a suffix item, its corresponding three prefix are $\langle b, c, e : 1 : 85\rangle$, $\langle b : 1 : 35$ and $\langle c : 1 : 35\rangle$ (format is $\langle item\_1, item\_2, \cdots, item\_k : support : oews\rangle$), which form its conditional pattern base. The $\widehat{OEWS}$ value of the items b, c and e in the conditional pattern base of a (i.e., T^a) is less than the $minWS$. So forth, we prune all items in the conditional pattern base of a, and generate only a as the candidate item. Next, we consider the next item e in the SWFP-list. This item occurs in two branches of the SWFP-tree of Figure 3. The paths formed by these branches are $\langle d, b, c, e : 1 : 125\rangle$ and $\langle b, c, e : 1 : 75\rangle$. Therefore, considering e as a suffix item, its corresponding two prefix are $\langle b, d, c : 1 : 125$ and $\langle b, c : 1 : 75\rangle$, which form its conditional pattern base. The $\widehat{OEWS}$ value of the items b and c are no less than the $minUtil$ value. So forth, conditional SWFP-tree is constructed with the items b and c. From this conditional SWFP-tree, we generate eb, ec and e as candidate itemsets. Similar process is performed for the remaining items in the SWFP-list of Figure 3(d) to find all candidate itemsets. The complete set of candidate itemsets generated from Figure 3(d) are $a, eb, ec, e, c, cd, b, bd$. The correctness of finding all candidate itemsets is shown in Theorem 13.

Theorem 13. Let α be an itemset in SWFP-tree. Let B be the α's conditional pattern base, and β be an item in B. If α is a suffix itemset and $\widehat{OEWS}(\alpha, \beta) + WS(\alpha) \geq minWS$, then $\langle \alpha, \beta\rangle$ is a candidate itemset.

Proof 14. According to the definition of conditional pattern base and compact SWFP-tree, each subset in B occurs under the condition of the occurrence of α in the transactional database. If an item β appears in B, then β appears with α. Thus, $\langle \alpha, \beta\rangle$ is a candidate itemset if $\widehat{OEWS}(\alpha, \beta) + WS(\alpha) \geq minWS$. Hence proved.

5) Generating all WFNIs from candidate itemsets

After finding all candidate itemsets 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 WFNI. The complete set of WFNIs generated from Table 1c for the user-specified $minWS$ of 150 is shown in Table 1e.

V. EXPERIMENTAL RESULTS

Since there exists no algorithm to mine WFNIs in a binary spatiotemporal database, we only evaluate the proposed algorithm using various databases. We show that our algorithm is not only memory and runtime efficient, but also scalable as well.

A. EXPERIMENTAL SETUP

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** [2] 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. 4a. It can be observed that items have non-uniformly spread throughout the region. The statistical details of this database were provided in Table 4.

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. 4b. 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 4.

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 4.

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. 4d. 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 4.

TABLE 4: Statistics of the datasets

Databbase	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

B. EVALUATION OF SWFP-GROWTH AT VARIOUS $MINWS$ VALUES

Figs. 5a, 5b, 5c and 5d show the number of WFNIs 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 WFNIs as many itemsets fail to satisfy the increased $minWS$ value and (ii) increase in $maxDist$ causes increase in WFNIs as higher $maxDist$ facilitates the items to increase their neighborhood sizes. It can be observed that at higher $maxDist$ values, too many WFNIs are getting generated. It is because of the increase in neighborhood size facilitates items to combine with far away items and generate WFNIs. Many WFNIs 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 5: Some of the interesting WFNIs generated in pollution database

Figs. 6a, 6b, 6c and 6d 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 WFNIs get generated and (ii) increase in $maxDist$ results in increase of memory required to find WFNIs. It is because a large number of WFNIs get generated at higher $maxDist$ values.

Figs. 7a, 7b, 7c and 7d 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 WFNIs are getting generated and (ii) increase in $maxDist$ results in the increase of runtime.

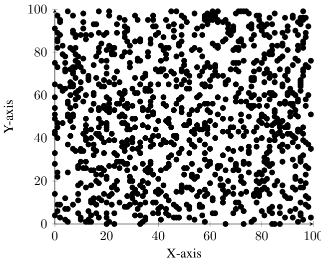
C. SCALABILITY TEST OF SWFP-GROWTH

We study the scalability of proposed algorithm on execution time and required memory by varying the size of T10I4D100K database. We concatenated the T10I4D100K database ten times to produce a very large database, which we call as T10I4D1000K database. Next, we divided this database into five portions of 0.2 million transactions in each part. Then we investigated the performance of our algorithm after accumulating each portion with previous parts while finding SWFIs each time. To find same itemsets as SWFIs with the increase in database sizes, the $minWS$ was doubled to reflect the database size. The $minWS$ for the first database was set at 40,000.

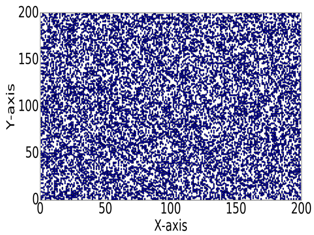
Fig. 8a and 8b respectively show the memory and runtime requirements of SWFP-growth algorithm on T10I4D100K database. It is clear from the graphs that as the database size increases, the memory and runtime requirements of our algorithm increase. However, SWFP-growth has stable performance of about linear increase of runtime and memory consumption with respect to the data size. Thus, SWFP-growth can mine SWFIs over large databases and distinct items with considerable amount of runtime and memory. (We can conduct the above experiment by directly generating T10I4D1000K database using the synthetic database generator. However, such generated results may be misleading as the number of generated itemsets can vary with the database size. In our scaled database, the number of patterns remain the same irrespective of the database size.)

D. A CASE STUDY: IDENTIFYING HIGHLY POLLUTED PM2.5 REGIONS IN JAPAN

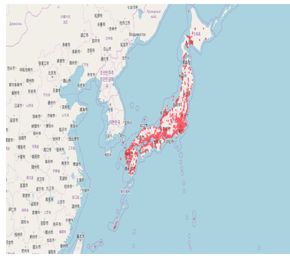
Table 5 shows the WFNIs generated in the PM2.5 database at $maxDist = 5$ kilometers and $minWS = 10,000\mu g/m^3$. The spatial location of all these stations in the entire Japan are shown in Fig. 9. The spatial location of the sensors present in each Weighted Frequent Neighborhood Itemsets are shown in Fig. 10 11 12 and 13. These itemsets in these figures indicate the geographical areas where people have been exposed to high levels of PM2.5 pollutant. It can be observed that high levels of PM2.5 have been observed at the places close to the bay areas (or harbors). This information can be found very useful in devising policies to control pollution at bay areas.



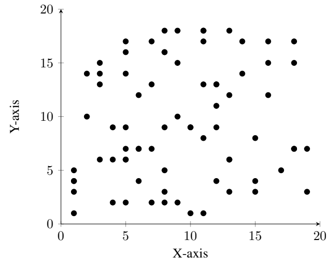
(a) T10I4D100K



(b) Retail

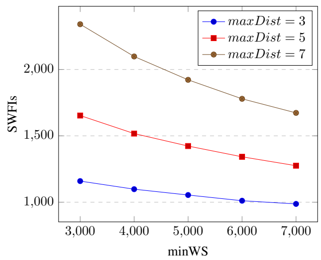


(c) PM2.5

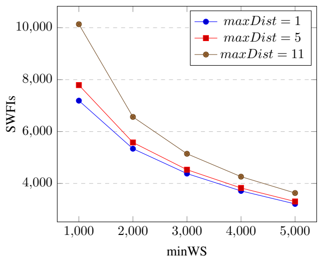


(d) Chess

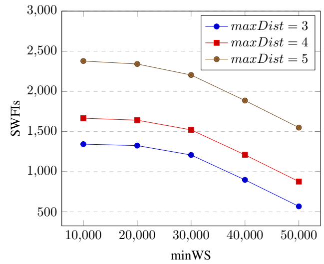
FIGURE 4: Spatial visualization of items in various databases



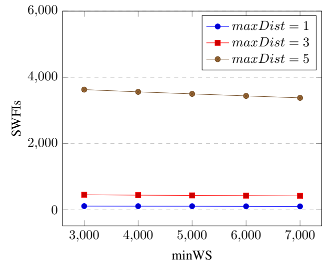
(a) T10I4D100K



(b) Retail

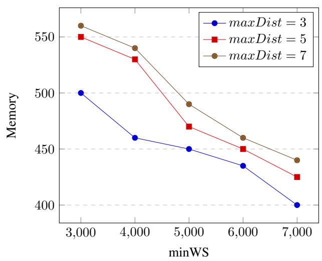


(c) PM2.5

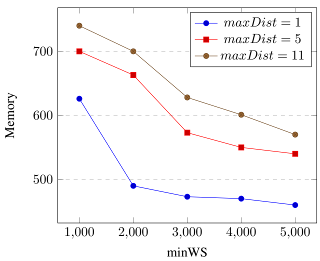


(d) Chess

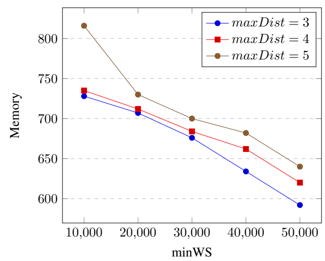
FIGURE 5: WFNIs generated by SWFP-growth algorithm at different $minWS$ and $maxDist$ values in various databases



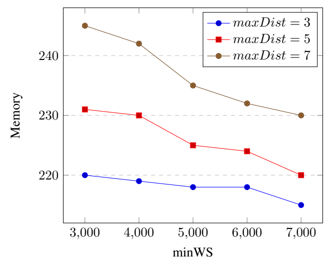
(a) T10I4D100K



(b) Retail



(c) PM2.5



(d) Chess

FIGURE 6: Memory requirements of SWFP-growth in various databases at different $minWS$ and $MaxDist$ values

VI. CONCLUSIONS AND FUTURE WORK

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.

REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami, “Mining association rules between sets of items in large databases,” in Acm sigmod record, vol. 22, pp. 207–216, 1993.

[2] R. Agrawal, R. Srikant, et al., “Fast algorithms for mining association rules,” in Proc. 20th int. conf. very large data bases, VLDB, vol. 1215, pp. 487–499, 1994.

[3] H. Cheng and J. Han, “Pattern-growth methods,” in Encyclopedia of Database Systems, Second Edition, 2018.

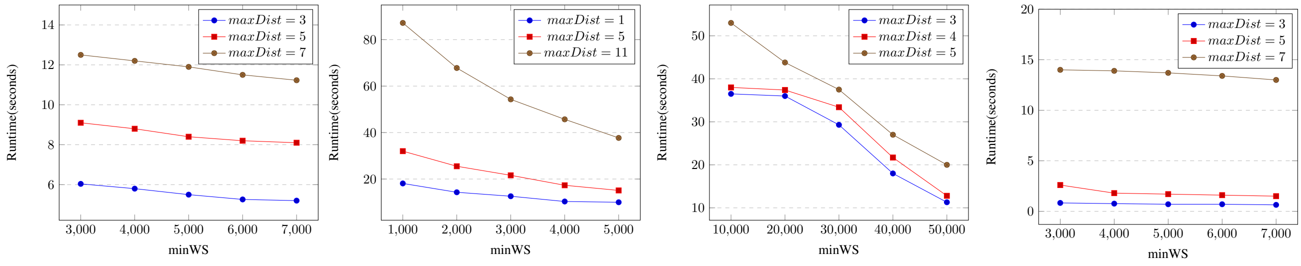
[4] C. C. Aggarwal and J. Han, Frequent Pattern Mining. Springer Publishing Company, Incorporated, 2014.

[5] G. M. Weiss, “Mining with rarity: a unifying framework.,” SIGKDD Explorations, vol. 6, no. 1, pp. 7–19, 2004.

[6] B. Liu, W. Hsu, and Y. Ma, “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.

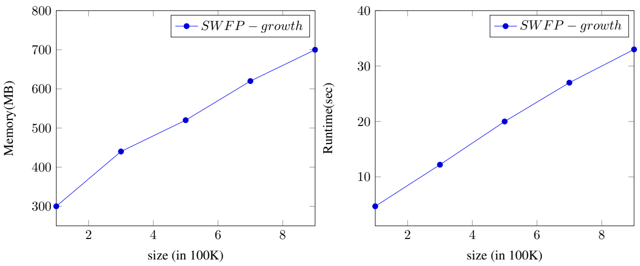
[7] R. U. Kiran and P. K. Reddy, “Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms,” in Proceedings of the 14th International Conference on Extending Database Technology, EDBT/ICDT ’11, pp. 11–20, 2011.

[8] C. H. Cai, A. W.-C. Fu, C. Cheng, and W. Kwong, “Mining association rules with weighted items,” in Database Engineering and Applications



(a) T10I4D100K (b) Retail (c) PM2.5 (d) Chess

FIGURE 7: Runtime requirements of SWFP-growth in various databases at different $minWS$ and $MaxDist$ values



(a) Memory (b) Runtime

FIGURE 8: Scalability of SWFP-growth

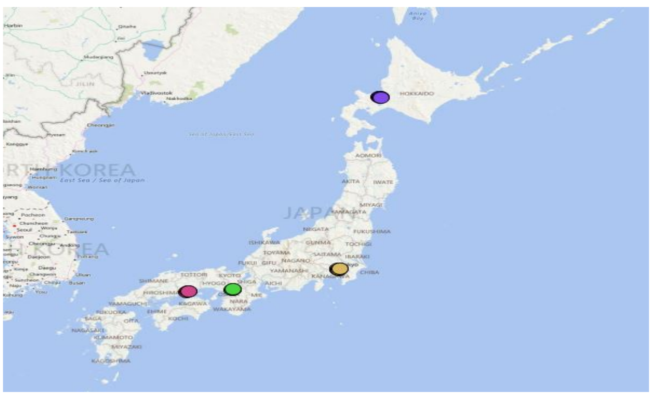


FIGURE 9: Spatial location of sensors that have recorded high levels of PM2.5 values in Japan

Symposium, 1998. Proceedings. IDEAS'98. International, pp. 68–77, IEEE, 1998.

[9] J.-H. Cai, X.-J. Zhao, S.-W. Sun, J.-F. Zhang, and H.-F. Yang, "Stellar spectra association rule mining method based on the weighted frequent pattern tree," Research in Astronomy and Astrophysics, vol. 13, no. 3, p. 334, 2013.

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

[11] Q. Duong, P. Fournier-Viger, H. Ramampiaro, K. Nørvåg, and T. Dam, "Efficient high utility itemset mining using buffered utility-lists," Appl. Intell., vol. 48, no. 7, pp. 1859–1877, 2018.

[12] R. U. Kiran, T. Y. Reddy, P. Fournier-Viger, M. Toyoda, P. K. Reddy, and M. Kitsuregawa, "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.

[13] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," in SIAM, pp. 482–486, 2004.

[14] K. Z. M. T. M. K. P. K. R. P. P. C. Reddy, R. Uday Kiran, "Wfim: weighted frequent itemset mining with a weight range and a minimum weight," in To be appeared in ICDM 2019 UDML workshop, 2019.

[15] M. J. Zaki, "Scalable algorithms for association mining," IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 3, pp. 372–390, 2000.

[16] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," Data mining and knowledge discovery, vol. 8, no. 1, pp. 53–87, 2004.

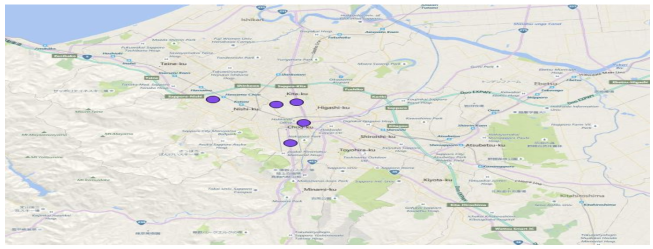


FIGURE 10: Spatial location of sensors that have recorded high levels of PM2.5 values in Sapporo

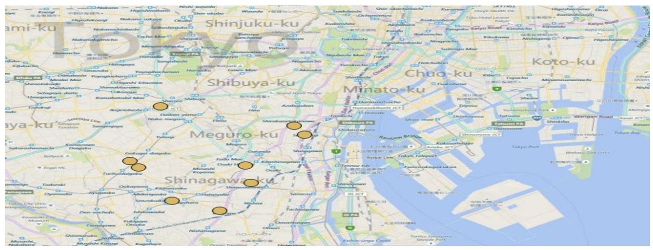


FIGURE 11: Spatial location of sensors that have recorded high levels of PM2.5 values in Tokyo

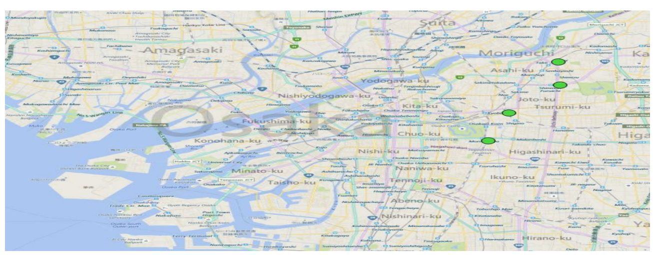


FIGURE 12: Spatial location of sensors that have recorded high levels of PM2.5 values in Osaka

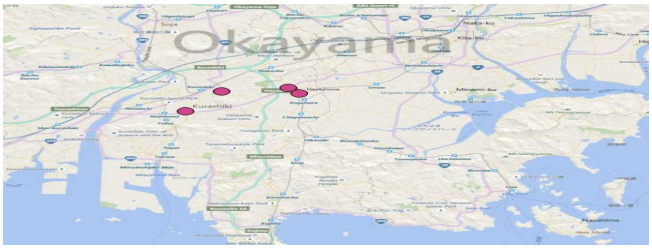


FIGURE 13: Spatial location of sensors that have located high levels of PM2.5 values in Okayama

[17] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," Data Mining and Knowledge Discovery, vol. 15, pp. 55–86, Aug 2007.

[18] U. Yun and J. J. Leggett, "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.

[19] F. Tao, F. Murtagh, and M. Farid, "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.

[20] B. Vo, F. Coenen, and B. Le, "A new method for mining frequent weighted itemsets based on wit-trees," Expert Systems with Applications, vol. 40, no. 4, pp. 1256–1264, 2013.

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

[22] J. C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, and V. S. Tseng, "Weighted frequent itemset mining over uncertain databases," Applied Intelligence, vol. 44, no. 1, pp. 232–250, 2016.

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

[24] W. Gan, J. C. Lin, P. Fournier-Viger, H. Chao, T. Hong, and H. Fujita, "A survey of incremental high-utility itemset mining," Wiley Interdiscip. Rev. Data Min. Knowl. Discov., vol. 8, no. 2, 2018.

[25] C. Zhang, G. Almpanidis, W. Wang, and C. Liu, "An empirical evaluation of high utility itemset mining algorithms," Expert Syst. Appl., vol. 101, pp. 91–115, 2018.

[26] R. U. Kiran, K. Zettsu, M. Toyoda, P. Fournier-Viger, P. K. Reddy, and M. Kitsuregawa, "Discovering spatial high utility itemsets in spatiotemporal databases," in Proceedings of the 31st International Conference on Scientific and Statistical Database Management, SSDBM '19, pp. 49–60, 2019.

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

[28] C. F. Eick, R. Parmar, W. Ding, T. F. Stepinski, and J.-P. Nicot, "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, GIS '08, pp. 30:1–30:10, 2008.

[29] P. Mohan, S. Shekhar, J. A. Shine, J. P. Rogers, Z. Jiang, and N. Wayant, "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, GIS '11, pp. 122–132, 2011.

[30] H. Tran-The and K. Zettsu, "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.

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

[32] J. Pei and J. Han, "Can we push more constraints into frequent pattern mining?," in KDD, pp. 350–354, ACM, 2000.

R. UDAY KIRAN holds the position of Project Assistant Professor at the Kitsuregawa lab, Institute of Industrial Science, The University of Tokyo, Tokyo, Japan. Additionally, he holds the position of Researcher at the Social Big Data Research Collaboration Center, National Institute of Information and Communications Technology, Tokyo, Japan. He received his PhD degree in computer science from International Institute of Information Technology, Hyderabad, Telangana, India. His current research interests include data mining, parallel computation, air pollution data analytics, traffic congestion data analytics, recommender systems and ICTs for Agriculture. He has published over 50 papers in refereed journals and international conferences, such as International Conference on Extending Database Technology (EDBT), International Conference on Scientific and Statistical Database Management (SSDBM), The Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Database Systems for Advanced Applications (DASFAA), International Conference on Database and Expert Systems Applications (DEXA), International Journal of Computational Science and Engineering (IJCSE), Journal of Intelligent Information Systems (JIIS), and Journal of Systems and Software (JSS).

P. P. C. REDDY is pursuing his dual degree (B.Tech. and M.S.) in Computer science from International Institute of Information Technology, Hyderabad.

KOJI ZETTSU is a Director General of Big Data Integration Research Center of National Institute of Information and Communications Technology (NICT). He has been doing research and development of data analytics technology in NICT, and now leading Real Space Information Analytics Project since 2016 to implement smart data platform based on data mining and AI. For promoting industry-academia-government collaboration on the platform, he is also a leader of Cross-Data Collaboration Project of Smart IoT Acceleration Forum in Japan. He received Ph.D. in Informatics from Kyoto University in 2005. His research interests are database systems, data mining, information retrieval and software engineering. He has serviced on numerous academic societies, conference committees and working groups.

MASASHI TOYODA is a professor of Institute of Industrial Science jointly affiliated with the Graduate School of Information Science and Technology at the University of Tokyo, Japan. He received the BS, the MS, and the PhD degrees in computer science from the Tokyo Institute of Technology, Japan, in 1994, 1996, and 1999, respectively. In 1999, he joined the Institute of Industrial Science, the University of Tokyo as a research fellow, and worked as a specially appointed associate professor from 2004 to 2006, and as an associate professor from 2006 to 2018. His research interests include archiving and analysis of Web, social media, and IoT data, information visualization, visual analytics, and user interface.

P. KRISHNA REDDY is a faculty member at IIIT Hyderabad. He is the head of Agricultural Research Center and the member of Data Sciences and Analytics Center research team at IIIT Hyderabad, India. During 2013 to 2015, he has served as a Program Director, ITRA-Agriculture and Food, Information Technology Research Academy (ITRA), Government of India. From 1997 to 2002, he was a research associate at the Center for Conceptual Information Processing Research, Institute of Industrial Science, University of Tokyo. From 1994 to 1996, he was a faculty member at the Division of Computer Engineering, Netaji Subhas Institute of Technology, Delhi. During the summer of 2003, he was a visiting researcher at Institute for Software Research International, School of Computer Science, Carnegie Mellon University, Pittsburg, USA. He has received both M.Tech and Ph.D degrees in computer science from Jawaharlal Nehru University, New Delhi in 1991 and 1994, respectively. His research areas include data mining, database systems and IT for agriculture. He has published about 157 refereed research papers which include 22 journal papers, three book chapters, and six edited books. He is a steering committee member of the pacific-asia knowledge discovery and data mining (PAKDD) conference series and Database Systems for Advanced Applications (DASFAA) conference series. He is a steering committee chair of Big Data Analytics (BDA) conference series since 2017. He was a proceedings chair of COMAD 2008, a workshop chair of KDRS 2010, media and publicity chair of KDD 2015, and general chair of BDA2017. He has organized the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2010), the Third National Conference on Agro-Informatics and Precision Agriculture 2012 (AIPA 2012) and the Fifth International Conference on Big Data Analytics (BDA 2017). He has delivered several invited/panel talks at the reputed conferences and workshops in India and abroad. He has got several awards and recognitions. He has executed research projects by raising the research funding of about 80 million Indian rupees. Since 2004, he has been investigating the building efficient knowledge agricultural knowledge transfer systems by extending developments in IT. He has developed eSagu system, which is an IT-based farm-specific agro-advisory system, which has been field-tested in hundreds of villages on about 50 field and horticultural crops. He has also built eAgromet system, which is an IT-based agro-meteorological advisory system to provide risk mitigation information to farmers. He has conceptualized the notion of Virtual Crop Labs to improve applied skills for extension professionals. Currently, he is investigating the building of Crop Darpan system, which is a crop diagnostic tool for farmers, with the funding support from India-Japan Joint Research Laboratory Program. He has received two best paper awards. The eSagu system, which is an IT based farm-specific agro-advisory system, has got several recognitions including CSI-Nihilent e-Governance Project Award in 2006, Manthan Award in 2008 and finalist in the Stockholm Challenge Award 2008.

MASARU KITSUREGAWA is Director General of National Institute of Informatics and Professor at Institute of Industrial Science, the University of Tokyo. Received Ph.D. degree from the University of Tokyo in 1983. Served in various positions such as President of Information Processing Society of Japan (2013–2015) and Chairman of Committee for Informatics, Science Council of Japan (2014-2016). He has wide research interests, especially in database engineering. He has received many awards including ACM SIGMOD E. F. Codd Innovations Award, IEICE Achievement Award, IPSJ Contribution Award, 21st Century Invention Award of National Commendation for Invention, Japan and C and C Prize. In 2013, he awarded Medal with Purple Ribbon and in 2016, the Chevalier de la Legion D'Honneur. He is a fellow of ACM, IEEE, IEICE and IPSJ.