Discovering Spatial High Utility Itemsets in Spatiotemporal Databases

R. Uday Kiran
National Institute of Information and Communications Technology, Japan
The University of Tokyo, Japan
uday_rage@tkl.iis.u-tokyo.ac.jp

Koji Zettsu
National Institute of Information and Communications Technology, Japan
zettsu@nict.gov.jp

Masashi Toyoda
The University of Tokyo, Japan
toyoda@tkl.iis.u-tokyo.ac.jp

Philippe Fournier-Viger
Harbin Institute of Technology, Shenzhen, China
philfv8@hit.edu.cn

P. Krishna Reddy
International Institute of Information Technology, Hyderabad, India
pkreddy@iiit.ac.in

Masaru Kitsuregawa
National Institute of Informatics, Japan
The University of Tokyo, Japan
kitsure@tkl.iis.u-tokyo.ac.jp

ABSTRACT

In real-world databases, high utility itemset (HUI) is an important class of regularities. Most previous studies have focused on mining HUIs in transactional databases and did not consider the spatiotemporal characteristics of items. In this study, a more flexible model of spatial HUIs (SHUIs) that exist in spatiotemporal databases is proposed. In a spatiotemporal database (*STD*), an itemset is said to be an SHUI if its *utility* is not less than a user-specified *minimum utility* and the distance between any two of its items is not more than a user-specified *maximum distance*. Identifying SHUIs is very challenging because the generated itemsets do not satisfy the anti-monotonic property. In this study, we present two novel pruning techniques for reducing computational costs. Moreover, a fast single scan algorithm is presented for effectively evaluating all SHUIs in a *STD*. Furthermore, two case studies are presented, in which the proposed model is used to identify useful information in traffic congestion data and air pollution data.

CCS CONCEPTS

• **Information systems → Association rules**.

KEYWORDS

Data mining, utility itemset mining, pattern mining and spatiotemporal databases

ACM Reference Format:
R. Uday Kiran, Koji Zettsu, Masashi Toyoda, Philippe Fournier-Viger, P. Krishna Reddy, and Masaru Kitsuregawa. 2019. Discovering Spatial High Utility Itemsets in Spatiotemporal Databases. In *31st International Conference on Scientific and Statistical Database Management (SSDBM '19), July 23–25, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3335783.3335789

SSDBM '19, July 23–25, 2019, Santa Cruz, CA, USA

ACM ISBN 978-1-4503-6216-0/19/07…$15.00
https://doi.org/10.1145/3335783.3335789

1 INTRODUCTION

In data mining, frequent itemset mining (FIM) [1] is an important model, and its mining algorithms can be used to discover all itemsets in a transactional database that satisfies a user-specified *minimum support* (*minSup*) constraint. The latter indicates that the minimum number of transactions that an itemset must cover in the data. Because the *support* measure is only used to determine the interestingness of an itemset, the model implicitly assumes that all items within the data have uniform weight and that an item at most appears only once in a transaction. Both these assumptions limit the applicability of FIM because the items in many real-world databases have non-uniform weights and can occur multiple times in a transaction. For example, in a supermarket, items have different prices and a customer can purchase multiple quantities of an item such as three bottles of milk, six bread rolls, and a bottle of wine. Because FIM algorithms disregard the purchase quantities of items and their unit prices, they may encounter many uninteresting frequent itemsets. Therefore, the generation of revenue would be low and many rare itemsets that could result in the generation of high revenue would be missed. This problem is referred to as the "rare item problem" [12].

To address the rare item problem, Yao et al. [25] proposed high utility itemset mining (HUIM), which extends considering the items' internal utility (such as the number of occurrences of an item within a transaction) and external utility (such as the weight of an item). Given a (non-binary) transactional database, HUIM aims to find all high utility itemsets (HUIs), i.e., itemsets having a *utility* that is not less than a user-specified *minimum utility* (*minUtil*) constraint. The *utility* of an itemset is the sum of its utilities in all transactions. HUIM has a number of applications such as market-basket analysis [24], mobile commerce [23], click stream analysis [2], biomedicine [16], and cross-marketing [9]. Moreover, mining HUIs have inspired several other important data mining tasks such as high utility sequential pattern mining [26] and high utility periodic pattern mining [8].

Although HUIM and its variants consider an item's weight and its occurrence *frequency* in a transaction, they disregard spatiotemporal characteristics of the item. Consequently, they fail to discover only the HUIs that have items close to one another in a spatiotemporal database. To address this issue, we introduced spatial HUI

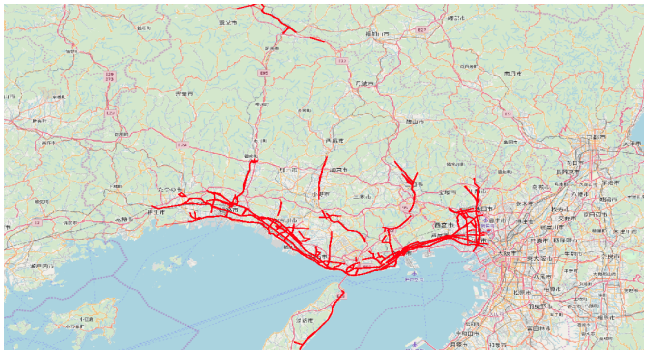

Figure 1: Spatial visualization of roads in Kobe, Japan

time	Mesh code	RID	Pt	CL
00:00	5234252242	2171	p1	100
00:00	5234068814	1048	p2	120
00:00	5234253124	2171	p1	100
00:00	5234068814	1181	p3	50
00:05	5234072511	486	p4	61
00:05	5234072511	489	p5	61
00:05	5234072511	486	p4	61
00:05	5234252244	2171	p1	90
...	...	...	...	...

(a) real-world sensor data

Spatiotemporal database

time	transactions
00:00	(2,2171),(1,1048),(1,1181)
00:05	(2,486),(1,489),(1,2171)

(b) temporal database

Pt\RID	2171	1048	1181	486	489
	p1	p2	p3	p4	p5

(c) spatial database

time	2171	1048	1181	486	489
00:00	100	120	50	0	0
00:05	90	0	0	61	61

(d) external utility database

Figure 2: Representing real-world sensor data as a spatiotemporal database and an external utility database. The terms "Pt," "RID," and "CL" represent the point, road identifier, and congestion length in meters, respectively.

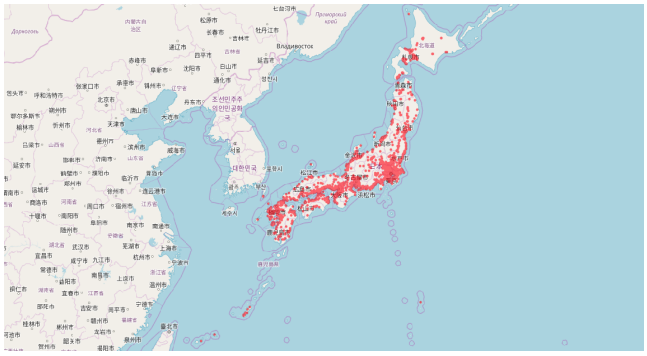

Figure 3: Spatial visualization of AEROS sensors

time	SID	Pt	$PM_{2.5}$
00:00	1	p1	19
00:00	2	p2	7
00:00	3	p3	7
00:00	5	p4	5
01:00	1	p1	21
01:00	2	p2	6
01:00	3	p3	22
01:00	4	p4	32
...	...	...	...

(a) real-world sensor data

Spatiotemporal database

time	transactions
00:00	(1,1),(1,2),(1,3),(1,5)
01:00	(1,1),(1,2),(1,3),(1,4)

(b) temporal database

Pt\SID	1	2	3	4	5
	p1	p2	p3	p4	p5

(c) spatial database

time	1	2	3	4	5
00:00	19	7	7	0	5
00:05	21	6	22	32	0

(d) external utility database

Figure 4: Representing the real-world sensor data as a spatiotemporal database and an external utility database. The terms "Pt" and "SID" represent the point and sensor identifier, respectively. $PM_{2.5}$ is measured in micro-gram per cubic-meter (i.e., $\mu g/m^3$)

mining (SHUIM). Herein we present two business applications of SHUIM.

The first application of SHUIM is improving the road traffic safety in smart cities. To measure and monitor congestion at various road segments, Japan Road Traffic Information Center (JARTIC) has set up a sensor network. Figure 1 shows the road network covered by congestion measuring sensors in Kobe, Japan. The data generated by these sensors represent a spatiotemporal database and an external utility database. Figure 2(a) shows a portion of the actual data generated by the sensors. Figures 2(b), (c), and (d) show the temporal database, spatial database, and external utility database generated from Figure 2(a), respectively. An SHUI generated from these databases, such as a total of 5 km of congestion, was observed on the set of roads $\{2171, 1181\}$ and provided information regarding the set of neighboring roads where a significant length of congestion was observed. At the time of disasters, the above mentioned information can be very useful to users for various purposes such as diverting the traffic, alerting the pedestrians, and suggesting a patrol path for the police. Japanese Industrial Standard X 0410 regional mesh code (or simply, mesh code) [4] represents a portion of the square grid on the Earth's surface. Because a road segment can pass through multiple mesh codes, the congestion length observed by a sensor is equally split among its mesh codes. Thus, the internal utility of an item in the temporal database represents the number of occurrences of a road segment with various mesh codes at a particular timestamp. The terms "$p1$" to "$p5$" in Figure 2 represent the points (i.e., latitude and longitude) of the corresponding road segments.

The second application of SHUIM is improving public health by identifying highly polluted geographical regions. Air pollution is a major cause of cardio-respiratory problems for people living in Japan. Thus, to tackle pollution, the Japanese Ministry of Environment has set up the Atmospheric Environmental Regional Observation System (AEROS). This system comprises air pollution measuring sensors located all over Japan (see Figure 3). The data generated by these sensors is a spatiotemporal database and an external utility database, as shown in Figure 4. An SHUI identified in this pollution database indicates that a total of $1250 \mu g/m^3$ of $PM_{2.5}$[1] was observed by a set of sensors $\{1, 4, 5\}$ that are close to one other. This provides information regarding neighboring sensors (or geographical regions) where large quantities of air pollution have been observed. This information is useful to users of pollution control board and helps them take appropriate steps such as increasing the vegetation, alerting the people to remain indoors

[1] $PM_{2.5}$ refers to atmospheric particulate matter that have a diameter of <2.5 micrometers.

at certain times of the day, and proposing new policies to regulate industrial emissions.

HUIM algorithms cannot be directly employed for identifying SHUIs in spatiotemporal databases because of the following reasons:

- HUIM algorithms implicitly assume that transactions occur at a fixed time interval and the external utility of an item remains unchanged in the entire data. However, a SHUIM algorithm has to consider that transactions within the data can occur at irregular time intervals and the external utilities of items (such as congestion length on a road segment and $PM_{2.5}$ value recorded by a sensor) can vary over time.
- HUIM algorithms determine the interestingness of an itemset without considering the spatial information of its items. Alternatively, a SHUIM algorithm has to determine the interestingness of an itemset by considering both its *utility* and the closeness (or proximity) of its items.
- Identifying SHUIs in spatiotemporal databases is a computationally expensive process because the *utility* function does not satisfy the *convertible anti-monotone*, *convertible monotone*, or *convertible succinct properties* [19]. The pruning techniques of existing HUIM algorithms can be employed to reduce computational costs; however, such techniques are inefficient because they disregard the spatial information of items.

With this motivation, we propose a more flexible model of SHUIs in spatiotemporal databases. In a spatiotemporal database (*STD*), an itemset is said to be an SHUI if its *utility* is not less than a user-specified *minimum utility* and the distance between any two of its items is not more than a user-specified *maximum distance*. To calculate an upper bound on the *utility* of itemsets, two new measures, *Probable Maximum Utility* (*PMU*) and *Neighborhood Sub-tree Utility* (*NSU*), are proposed. Moreover, to effectively reduce the search space, two novel pruning techniques employing the *PMU* and *NSU* measures have been proposed. Furthermore, a fast single scan algorithm, called SHUI-Miner is presented to effectively determine all SHUIs in a spatiotemporal database. The proposed SHUI-Miner algorithm is based on the renowned efficient high-utility itemset mining (EFIM) algorithm [27]. Experimental results show that SHUI-Miner is efficient. We also present two case studies in which we applied the proposed model to identify useful information in traffic congestion and air pollution data.

The remainder of this study is organized as follows. In Section 2, we discuss the literature on spatial frequent itemset mining and HUI mining. In Section 3, we introduce the proposed model of SHUI. In Section 4, we introduce the SHUI-Miner algorithm. In Section 5, we present the experimental results. Finally, in Section 6, we provide conclusion of this study and discuss future research directions.

2 RELATED WORK

2.1 Spatial co-occurrence itemset mining

Since the introduction of frequent itemsets in [1], the problem of identifying spatiotemporal co-occurrence itemsets (or association rules) in spatiotemporal databases has received considerable attention of researchers [6, 7, 17, 22]. These algorithms can be broadly classified into distance-based approaches [6, 7] and transaction-based approaches [17, 22]. Typically, a distance-based approach uses a parameter called prevalence to determine how interesting spatiotemporal co-occurrences occur in the data. Initially, a transaction-based approach clusters data over space and time (i.e., spatial clustering) and then applies traditional association rule mining algorithms to cluster and determine useful associations. Unfortunately, all spatiotemporal co-occurrence itemset mining algorithms determine the interestingness of an itemset by considering only its *support* and disregarding the internal and external utility values of an item. Consequently, most of these algorithms cannot handle numeric data. Nevertheless, the proposed SHUIM model considers the internal and external utility values of items and handles numeric data.

2.2 HUI mining

To address the rare item problem in FIM, Cai et al. [5] proposed weighted frequent itemset mining (WFIM). WFIM algorithms [5, 10] can be used to identify all interesting itemsets in a binary transactional database using the weight (or importance) of items. A major limitation of WFIM algorithms is that they do not consider the number of occurrences of an item in a transaction. Yao et al. [25] proposed HUIM, which considers the importance of items and their occurrence *frequency* in every transaction. To circumvent the fact that *utility* is not anti-monotonic and to identify all HUIs, algorithms such as IHUP [3], PB [11], Two-Phase [15], BAHUI [21], UP-Growth and UP-Growth+ [13, 14], and MU-Growth [28] utilize the *transaction weighted utility* (*TWU*) to prune the search space. *TWU* is an upper bound on the utility of itemsets. Note that the abovementioned algorithms operate in two phases. In the first phase, they identify candidates of HUIs by calculating their TWUs to prune the search space. If an itemset has a *TWU* greater than *minUtil*, then it is considered as a candidate itemset because the itemset or its supersets may be HUIs. Otherwise, if an itemset has a *TWU* less than *minUtil*, it is discarded. In the second phase, these algorithms scan the database to calculate the exact utility of all candidates to filter those that are low-utility itemsets; unfortunately, all of the abovementioned algorithms suffer from the problem of generating too many candidates.

Recently, to avoid the problem of candidate generation, single phase HUI mining algorithms (such as d2HUP [13], EFIM [27], and HU-FIMi [24]) were developed. These algorithms use upper bounds that are tighter than the *TWU* to prune the search space and can immediately obtain the exact utility of any itemset to determine if it should be outputted. Unfortunately, all HUIM algorithms disregard the spatiotemporal characteristics of items within the data. The proposed SHUIM model generalizes HUIM by considering thespatiotemporal characteristics of the items. Thus, the proposed SHUIM model is novel and distinct from HUIM.

3 PROPOSED MODEL

Let $I = \{i_1, i_2, \cdots, i_m\}$, $m \geq 1$, be a set of items. Let $p(i_j, ts)$ denote the **external utility** of an item $i_j \in I$ at a timestamp $1 \leq ts \leq n$. Let $P(i_j) = p(i_j, 1) \cup \cdots \cup p(i_j, n)$ denote the set of all external utility values of i_j in the data. The (external) **utility database**, *UD*, refers to the set of external utility values of all items in I,

ts	Items
1	$(a,2),(b,3),(g,2),(f,1)$
2	$(a,1),(c,3),(f,4),(g,1)$
3	$(d,3),(f,2),(g,3)$
4	$(b,3),(c,4),(d,1)$
6	$(b,4),(c,4),(d,1),(e,1)$
9	$(a,1),(b,2),(c,2),(e,3),(g,1)$

Table 1: Temporal database

ts/Item	a	b	c	d	e	f	g
1	100	50	0	0	0	100	200
2	50	0	100	0	0	50	100
3	0	0	0	100	0	100	50
4	0	200	200	100	0	0	0
6	0	150	100	200	50	0	0
9	100	100	50	0	150	0	200

Table 2: External utility 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 3: Spatial database

i.e., $UD = \bigcup_{i_j \in I} P(i_j)$. A **temporal database**, $D = \{T_1, T_2, \cdots, T_n\}$, where $T_{ts} \subseteq I$ denotes a transaction. Each transaction T_{ts} is associated with a timestamp $ts \in (1,n)$. Every item $i_j \in T_{ts}$ has a positive number $q(i_j, T_{ts})$, called its **internal utility**. Generally, the internal utility of an item represents its $frequency$ in a transaction. A spatial database, $SD = \bigcup_{i_j \in I}(i_j,\ (lat_{i_j}, long_{i_j}))$ is a collection of location points of all items in I. The terms lat_{i_j} and $long_{i_j}$ denote the latitude and longitude information of an item i_j, respectively. (A spatiotemporal database STD is a combination of D and SD. For brevity, we describe SHUIM using UD, D, and SD.)

Example 1. Let $I = \{a,b,c,d,e,f,g\}$ be a set of all items (or road identifiers). An irregular temporal database generated from I is shown in Table 1. Table 2 presents the external utilities (or congestion lengths) of all items at various timestamps. Let the unit for measuring congestion length be m. The external utility of item a at timestamp 1 is $p(a,1) = 100$, and the internal utility of item a at timestamp 1 is $q(a,1) = 2$. The spatial database shown in Table 3 provides the location information of all items in Table 1. (For brevity, internal utilities have been presented along with the items in Table 1; however, these values can be treated as a separate database similar to that of the external utility database.)

Definition 1. (Utility of an item in a transaction) The utility of an item i_j in a transaction T_{ts}, denoted as $u(i_j, T_{ts})$, representing the product of its internal and external utility values, i.e., $u(i_j, T_{ts}) = p(i_j, T_{ts}) \times q(i_j, T_{ts})$.

Example 2. Continuing with the previous example, the $utility$ of an item a (i.e., congestion length at road a) in the first transaction is $u(a, T_1) = p(a, T_1) \times q(a, T_1) = 2 \times 100 = 200$ m.

Definition 2. (Utility of an itemset in a transaction) Let $X \subseteq I$ be an itemset. An itemset is a k-itemset if it contains k items. The utility of an itemset X in a transaction T_{ts} is $u(X, T_{ts}) = \Sigma_{i_j \in X} u(i_j, T_{ts})$ if $X \subseteq T_{ts}$; otherwise, $u(X, T_{ts}) = 0$.

Example 3. The set of items "a" and "b," i.e., $\{a,b\}$ (or "ab" in short) is an itemset, which contains two items; therefore, it is a 2-itemset. The utility of itemset ab (or the total length of congestion observed on the roads a and b at the timestamp 1) in the transaction T_1 is $u(ab, T_1) = u(a, T_1) + u(b, T_1) = 200 + 150 = 350$ m.

Definition 3. (Utility of an itemset in a database) The utility of an itemset X in a database D is defined as $u(X) = \Sigma_{T_{ts} \in g(X)} u(X, T_{ts})$, where $g(X)$ denotes the set of all transactions containing X.

Example 4. In Table 1, the itemset ab appears in the transactions T_1 and T_9; therefore, $g(x) = \{T_1, T_9\}$. The $utility$ of itemset ab (or the total length of congestion observed on the roads a and b) in the entire database is $u(ab) = u(ab, T_1) + u(ab, T_9) = 350 + 300 = 650$ m. Similarly, the $utility$ of itemset cd in the entire database is $u(cd) = u(cd, T_4) + u(cd, T_6) = 900 + 600 = 1,500$ m.

Definition 4. ***(High utility itemset)*** An itemset X is said to be a HUI if $u(X) \geq minUtil$, where $minUtil$ represents the user-specified minimum utility value.

Example 5. If the user-specified $minUtil = 1,500$ m, then ab is not a HUI because $u(ab) \ngeq minUtil$, i.e., ab is an uninteresting low utility itemset. However, cd is a HUI because $u(cd) \geq minUtil$.

Definition 5. ***(Spatial high utility itemset)*** A HUI X is said to be a spatial high utility itemset if and only if the distance between its items is not more than the user-specified *maximum distance* ($maxDist$). Thus, an itemset X is a SHUI if $u(X) \geq minUtil$ and $\forall i_a, i_b \in X, a \neq b, Dist(i_a, i_b) \leq maxDist$, where $Dist(.)$ is a distance function (e.g. Euclidean distance and Geodesic distance) and $maxDist$ denotes a user-specified maximum distance constraint.

Example 6. Continuing with the previous example, the HUI cd is also an SHUI because $Dist(c,d) \leq maxDist$. This SHUI can be expressed as cd $[utility = 1500\ m]$. The set of all SHUIs generated from Table 1 are shown in Table 4. Note that Table 5 lists the neighbors of every item at $maxDist = 5$.

Note that all items in a SHUI must be close to one another (i.e., the distance between any two items in a SHUI must always be less than or equal to the user-specified $maxDist$.). If we relax this constraint, then uninteresting itemsets with items far away from the rest can be generated as SHUIs.

Example 7. Let $p = (0,0)$, $q = (4,0)$, $r = (8,0)$, and $s = (12,0)$ be four items located on a straight line and let $maxDist = 4$. If we relax the constraint that all items in a SHUI need not be close to each other, then we may identify $pqrs$ as a SHUI because $dist(p,q) \leq maxDis$, $dist(q,r) \leq maxDist$ and $dist(r,s) \leq maxDist$. Clearly, the itemset $pqrs$ is uninteresting because the items r and s are far away from p.

Itemset	*utility*
c	1600
b	1550
cd	1500
bd	1500

Table 4: SHUIs generated from Table 1 at $minUtil = 1500$ **and** $maxDist = 5$

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

Table 5: Neighbors of each item at $maxDist = 5$

Definition 6. (Problem definition) Given a temporal database D, spatial database SD, and a utility database UD, the problem of SHUIM is to identify all itemsets that have a *utility* not less than the user-specified $minUtil$ and the distance between its items is not more than the user-specified $maxDist$. Interestingly, HUIM is a special case of SHUIM problem when $maxDist = \infty$ (or for a very large value).

In the next section, we present the SHUI-Miner algorithm for efficiently identifying all SHUIs in a spatiotemporal database.

4 SHUI-MINER

In this section, we first present the basic idea of SHUI-Miner. Next, we explain the algorithm using the database shown in Table 1.

4.1 Basic idea

The space of items in a database gives rise to a subset lattice. When mining SHUIs, the itemset lattice is a conceptualization of the search space. However, reducing this search space is a challenging task because the generated SHUIs do not satisfy the *convertible anti-monotonic*, *convertible monotonic*, or *convertible succinct properties* [18]. To address this challenge, SHUI-Miner employs two new *utility* upper bound measures, i.e., *Probable Maximum Utility* (*PMU*) and *Neighborhood Sub-tree Utility* (*NSU*), to identify itemsets (or items) whose supersets may generate SHUIs. Both *PMU* and *NSU* measures internally utilize the information on the neighborhood of items (or distance between the items) to identify itemsets whose supersets may generate SHUIs. We will now present each of these two measures.

4.1.1 Probable Maximum Utility. PMU aims to prune items whose supersets in a database (or projected database) cannot be SHUIs. The *PMU* of an itemset in a database is defined in Definition 12, and is based on Definitions 7, 8, 9, 10, and 11.

Definition 7. (Probable Maximum Utility of an item in a transaction). Let N_{i_j} denote the neighbors of an item $i_j \in I$, i.e., $\forall i_k \in N_{i_j}, dist(i_j, i_k) \leq maxDist$. The *probable maximum utility* (*PMU*) of an item in a transaction T_{ts}, denoted as $pmu(i_j, T_{ts})$, represents the sum of utilities of i_j and its neighboring items in T_{ts}, i.e., $pmu(i_j, T_{ts}) = u(i_j, T_{ts}) + \sum_{i_k \in T_{ts} \cap i_k \in N_{i_j}} u(i_k, T_{ts})$.

Example 8. Consider the item a in Table 1. The neighbors of a, i.e., $N_a = \{bce\}$ (see Table 5). The *probable maximum utility* of a in T_1 is the sum of utilities of a and its neighboring items in T_1, i.e., $pmu(a, T_1) = u(a, T_1) + u(b, T_1) = 200 + 150 = 350$ m. Note that the utilities of the remaining items (i.e., g and f) in T_1 are not used while calculating $pmu(a, T_1)$ because these two items are not neighbors of a. The above definition of *PMU* captures the maximum utility of a and its neighboring items in T_1. We now extend this definition by considering a set of transactions (or a database).

Definition 8. (*PMU* of an item in a database). Let $g(i_j)$ denote the set of all transactions containing an item i_j. The *PMU* of an item in a database, denoted as $pmu(i_j)$, represents the sum of *probable maximum utility* of i_j in all transactions containing i_j, i.e., $pmu(i_j) = \sum_{T_{ts} \in g(i_j)} pmu(i_j, T_{ts})$.

Example 9. The transactions containing a in Table 1 are as follows: T_1, T_2 and T_9; therefore, $g(a) = \{T_1, T_2, T_9\}$. The *PMU* of a in T_1 is $pmu(a, T_1) = 350$ m. Similarly, $pmu(a, T_2) = 350$ m and $pmu(a, T_9) = 850$ m. Note that the *PMU* of a in the entire database is $pmu(a) = pmu(a, T_1) + pmu(a, T_2) + pmu(a, T_9) = 350 + 350 + 850 = 1550$ m, i.e., a with all its neighbors has resulted in a *utility* (or congestion length) of 1,550 m.

The above definition captures the *utility* upper bound of an item with respect to its neighbors. Property 1 defines a pruning technique based on Definition 8. The correctness of this pruning technique is shown in Theorem 4.1, which is based on Properties 2 and 3.

Property 1. (Pruning items using *PMU*) For an item $i_j \in I$, if $pmu(i_j) < minUtil$, then i_j can be pruned from the database because neither i_j nor its supersets will be SHUIs.

Example 10. The *pmu* values for the items a, b, c, d, e, f, and g in Table 1 are 1550, 2650, 2550, 3250, 2100, 1450 and 1350, respectively. If the user-specified $minUtil = 1500$, then the items "f" and "g" can be pruned from the database because $pmu(f) < minUtil$ and $pmu(g) < minUtil$.

Property 2. Let $\widehat{D} \subseteq D$ be a projected database containing items $\{i_j \cup N_{i_j}\}$. Let $\widehat{I} \subseteq I$ be a set of all items in $\widehat{D}$. The *utility* of i_j in $\widehat{D}$ will always be less than or equal to $pmu(i_j)$, i.e., $u(i_j) \leq pmu(i_j)$.

Property 3. (Utility upper bound of PMU.) In the projected database $\widehat{D}$, $u(X) \leq pmu(i_j)$, as $i_k \subseteq X$.

Theorem 4.1. *For an item $i_j \in I$, if $pmu(i_j) < minUtil$, then neither i_j nor its supersets can be SHUIs.*

Proof. In $\widehat{D}$, if $pmu(i_j) < minUtil$, then $u(i_j) < minUtil$ (see Property 2); thus, i_j cannot be a SHUI. Moreover, for every itemset X in $\widehat{D}$, $u(X) \leq minUtil$ (see Property 3); thus, all supersets of i_j will not generate any SHUI. Hence, the validity of the algorithm is proved. □

The items that are not pruned by Property 1 are known as **secondary items**. Definition 9 defines the secondary items. The SHUI-Miner algorithm does a depth-first search on the itemset lattice of secondary items to identify all SHUIs in the data.

Definition 9. (Secondary items.) An item $i_j \in I$ is a secondary item if $pmu(i_j) \geq minUtil$.

Example 11. Continuing with the previous example, the secondary items in Table 1 are a, b, c, d, and e.

We now generalize Definition 8 by considering the notion of itemset. This generalization facilitates pushing the above pruning technique to the lower levels of the itemset lattice.

Definition 10. (Neighboring items that can extend an itemset.) Let SI denote the set of secondary items. Let $\succ$ denote an order of secondary items in PMU in an ascending order. Let α be an itemset. Let $E(\alpha)$ denote the set of all secondary items that can be used to extend α according to the depth-first search, i.e., $E(\alpha) = \{z | z \in SI \wedge z \succ x, \forall x \in \alpha\}$. Let $N(\alpha) = (\cap_{i_k \in \alpha} N_{i_k}) \cap E(\alpha)$ denote the set of all neighboring items that can be used to extend α according to the depth-first search to identify SHUIs.

Example 12. The PMU in ascending order of secondary items in Table 1 is $SI = \{a, e, c, b, d\}$. Let $\alpha = a$ be an itemset. The $E(a) = \{e, c, b, d\}$. The Neighborhood of a is $N(a) = \{ecb\}$ (see Table 5). Let $\alpha = ae$, then $E(ae) = \{cbd\}$. The Neighboring items that can extend ae, i.e., $N(ae) = N(a) \cap N(e) \cap E(ae) = \{ecb\} \cap \{acbd\} \cap \{cbd\} = \{cb\}$, i.e., the itemset ae will be extended by only considering the items c and b. The item d will not be utilized for extending ae because d is not a neighbor of a.

Definition 11. (PMU of an itemset in a transaction) The PMU of an itemset α in a transaction T_{ts} is defined as $PMU(\alpha, T_{ts}) = u(\alpha, T_{ts}) + \sum_{i \in T_{ts} \wedge i \in N(\alpha) \wedge i \succ x \forall x \in \alpha} u(i, T_{ts})$.

Example 13. In Table 1, ae has appeared in the transaction T_9, which in PMU in an ascending order of items is $\{(a, 1), (e, 3), (c, 2), (b, 2), (g, 1)\}$. The PMU of ae in T_9 is the sum of utilities of ae and the neighboring items that are appearing in the sorted T_9 after e, i.e., $pmu(ae, T_9) = u(ae, T_9) + u(c, T_9) + u(b, T_9) = 550 + 100 + 200 =$ 850 m.

Definition 12. (PMU of an itemset in a database) The PMU of an itemset α in a database is $PMU(\alpha) = \sum_{T_{ts} \in g(\alpha)} pmu(\alpha, T_{ts})$.

Example 14. In Table 1, ae has appeared only in the transaction T_9; therefore, $g(ae) = \{T_9\}$. The pmu of ae in the entire database, i.e., $pmu(ae) = \sum_{T_9} pmu(ae, T_9) = 850$, i.e., the maximum utility any superset of ae (containing only the neighboring items of a and e) can have in Table 1 is 850 m.

The above definition represents the maximum utility achievable by any superset of an itemset (or an item) in the data. Thus, if the pmu of an itemset fails to satisfy $minUtil$, then we stop exploring its supersets. The accuracy of this algorithm is based on Property 4 as shown in Theorem 4.2. This theorem generalizes Property 1.

Property 4. (Overestimation using the PMU of an itemset) Let α be an itemset and z be an item in $N(\alpha)$. Let β be an extension of α such that $z \in \beta$. The relationship $PMU(\alpha \cup z) \geq u(\beta)$ holds (in the projected database of α), i.e., PMU is an overestimating function.

Theorem 4.2. ***(Pruning an item in all sub-trees using the PMU).*** *Let α be an itemset and $z \in N(\alpha)$. If $PMU(\alpha, z) < minUtil$, then all extensions of α containing z have low utility, i.e., item z can be ignored while exploring all sub-trees of α.*

Proof. The correctness is straightforward to prove from Property 4. □

4.1.2 Neighborhood sub-tree utility. The PMU measure identifies secondary items (or itemsets) whose supersets may be SHUIs in the database. We use a variant of *sub-tree utility*[27], called *Neighborhood sub-tree utility* (NSU), to identify these secondary items whose projections (or depth-first search in the itemset lattice) will result in identifying all SHUIs. The NSU is defined in Definition 13 and illustrated in Example 15. The secondary items whose projections will result in identifying SHUIs are referred to as *primary items*. The primary items are defined in Definition 14 and illustrated in Example 16. The relationship between PMU and NSU is given in Property 5.

Definition 13. (Neighborhood sub-tree utility) Let α be an itemset and $z \in N(\alpha)$ be an item that can extend α according to the depth-first search in the itemset lattice (or set enumeration tree). The neighboring sub-tree utility of z with respect to α, denoted as $NSU(\alpha, z) = \sum_{T_{ts} \in g(\alpha \cup z)} [u(\alpha, T_{ts}) + u(z, T_{ts}) + \sum_{i \in T_{ts} \wedge i \in N(\alpha \cup z)} u(i, T_{ts})]$.

Example 15. The PMU in an ascending order of secondary items in Table 1 is a, e, c, b, and d. Let us consider item d, which has pmu value of 3,250. Let $\alpha = \emptyset$. The *neighborhood sub-tree utility* of $\alpha \cup d$, $NSU(\alpha, d) = [0 + 300] + [0 + 100] + [0 + 200] = 600$. Because $pmu(d) \geq minUtil$ and $NSU(d) \ngeq minUtil$, it turns out that d will generate SHUIs in the projections of another items. The projection (or depth-first search) on d will not result in any SHUI.

Property 5. For an itemset X, $PMU(X) \geq NSU(X)$.

Definition 14. (Primary items). Given the PMU in an ascending order of secondary items, a secondary item $i_j \in SI$ is said to be a primary item if $NSU(i_j) \geq minUtil$.

Example 16. Consider the secondary items a and d, which have minimum and maximum pmu value in Table 1. The NSU values for these two secondary items, i.e., $NSU(a) = 1550$ and $NSU(d) = 600$. Because $NSU(a) \geq minUtil$, we consider a as a primary item. We do not consider d as a primary item because $NSU(d) \ngeq minUtil$.

4.1.3 Comparison of PMU and NSU measures against existing pruning measures. In the literature, researchers have employed *Transaction Weighted Utility* (TWU) [15], *local utility* (LU), and *sub-tree utility* (STU) [27] as utility upper bound measures to reduce the computational cost of identifying HUIs in (non-binary) transactional databases. These measures are defined in Definitions 15, 16, and 17. The proposed PMU measure is a tighter utility upper bound compared to TWU and LU measures (see Theorem 4.3). Similarly, the proposed NSU measure is a tighter utility upper bound compared to STU measure (see Theorem 4.4), i.e., the proposed PMU and NSU measures can help in determining the SHUIs by generating fewer secondary items and primary items compared to the existing HUIM algorithms [9, 27].

In the next subsection, we present SHUI-Miner algorithm using primary and secondary items.

Definition 15. (Transaction Weighted Utility of an itemset.) In a database D, the TWU of an itemset α, denoted as $TWU(\alpha)$,

represents the sum of utilities of all transactions containing α, i.e., $TWU(\alpha) = \sum_{T_{ts} \in g(\alpha)} \sum_{i_j \in T_{ts}} u(i_j, T_{ts})$, where $g(\alpha)$ represents the set of transactions containing α.

Definition 16. (Local utility of an itemset.) In a database D, the LU of an itemset $\alpha \cup z$,denoted as $LU(\alpha) = \sum_{T_{ts} \in g(\alpha \cup z)} u(\alpha, T_{ts}) + \sum_{i \in T_{ts} \wedge i \in E(\alpha) > x \forall x \in \alpha} u(i, T_{ts})$.

Definition 17. (Sub-tree utility) Let α be an itemset and z be an item that can extend α according to the depth-first search in the itemset lattice (or set enumeration tree). The sub-tree utility of z with respect to α, denoted as $STU(\alpha, z) = \sum_{T_{ts} \in g(\alpha \cup z)} [u(\alpha, T_{ts}) + u(z, T_{ts}) + \sum_{i \in T_{ts} \wedge i \in E(\alpha \cup z)} u(i, T_{ts})]$.

Property 6. For an itemset α, $TWU(\alpha) \geq LU(\alpha)$ [27].

Theorem 4.3. *For an itemset α, $TWU(\alpha) \geq LU(\alpha) \geq PMU(\alpha)$.*

Proof. For an itemset α, $N(\alpha) \subseteq E(\alpha)$ (see Definition 10); thus, $LU(\alpha) \geq PMU(\alpha)$. From Property 6, it turns out that $TWU(\alpha) \geq PMU(\alpha)$; thus, $TWU(\alpha) \geq LU(\alpha) \geq PMU(\alpha)$. Hence, the validity of the algorithm is proved. □

Theorem 4.4. *For an itemset $\alpha \cup z$, $STU(\alpha, z) \geq NSU(\alpha, z)$.*

Proof. Because $N(\alpha \cup Z) \subseteq E(\alpha \cup z)$, it turns out that $\sum_{T_{ts} \in g(\alpha \cup z)} [u(\alpha, T_{ts}) + u(z, T_{ts}) + \sum_{i \in T_{ts} \wedge i \in N(\alpha \cup z)} u(i, T_{ts})] \leq \sum_{T_{ts} \in g(\alpha \cup z)} [u(\alpha, T_{ts}) + u(z, T_{ts}) + \sum_{i \in T_{ts} \wedge i \in E(\alpha \cup z)} u(i, T_{ts})]$, i.e., $NSU(\alpha, z) \leq STU(\alpha, z)$. Hence, this confirms the validity of the algorithm. □

Algorithm 1 SHUI

1: **input**: D is a temporal database, SD is a spatial database, UD is an external utility database, $minUtil$ is a user-specified minimum utility constraint, and $maxDist$ is a user-specified maximum distance constraint
2: **output**: A set of SHUIs
3: $N = Neighbors(SD, maxDist)$. Let $N(i_j)$ denote the set of all neighboring items for $i_j \in I$.
4: Let α denote an itemset that needs to be extended. Initially, set $\alpha = \emptyset$;
5: Scan the temporal database to determine the PMU for every item $i_j \in I$.
6: $Secondary(\alpha) = \{i_j | i_j \in I \wedge PMU(i_j) \geq minUtil\}$
7: Let $>$ be the total order of PMU in an ascending order of their values in $Secondary(\alpha)$;
8: Scan D to remove each item $i \notin Secondary(\alpha)$ from the transactions, sort items in each transaction according to $>$, and delete empty transactions;
9: Calculate *neighborhood sub-tree utility* for all items in $secondary(\alpha)$ by scanning the database D once using utility-bin array;
10: $Primary(\alpha)$=$\{z \in secondary(\alpha) | NSU(\alpha, z) \geq minUtil\}$;
11: $RecursiveSearch(\alpha, D, Primary(\alpha), Secondary(\alpha), minUtil)$;

Algorithm 2 Neighbors

1: **input:** SD:spatial database; $maxDist$: maximum distance
2: **output:** $N < item, neighbors >$: A two dimensional hashmap with records of items and their neighbors
3: **for** each item $i_j \in SD$ **do**
4: **for** each item $i_k \in SD$ **do**
5: **if** $(i_j \neq i_k)$ and $(Dist(i_j, i_k) \leq maxDist)$ **then**
6: Add i_k as a neighbor of i_j in N.
7: **end if**
8: **end for**
9: **end for**

Algorithm 3 RecursiveSearch

1: **input** : α is an itemset, $\alpha - D$ is the α projected database, $Primary(\alpha)$ is the primary items of α, $Secondary(\alpha)$ is the secondary items of α, and $minUtil$ is minimum utility
2: **output**: the set of all SHUIs that are extensions of α
3: **for** each *item* $i \in Primary(\alpha)$ **do**
4: $\beta = \alpha \cup \{i\}$;
5: Scan α-D to calculate $u(\beta)$. Create a projected database, $\beta - D$.
6: **if** $u(\beta) \geq minutil$ **then**
7: output β as a spatial high utility itemset
8: **end if**
9: Calculate $stu(\beta, z)$ and $lu(\beta, z)$ for all item $z \in Secondary(\alpha)$ by scanning β-D once, using two utility-bin array;
10: $Primary(\beta) = \{z \in Secondary(\alpha) | NSU(\beta, z) \geq minutil$ and $z \in N_\beta\}$;
11: $Secondary(\beta) = \{z \in Secondary(\alpha) | PMU(\beta, z) \geq minutil$ and $z \in N_\beta\}$;
12: RecursiveSearch(β, β-D, $Primary(\beta)$, $Secondary(\alpha)$, $minutil$, $minSup$);
13: **end for**

4.2 Proposed algorithm

The SHUI-Miner is presented in Algorithms 1, 2, and 3. Because the calculation of PMU and NSU for a projected database of α is straightforward, we have not presented the related algorithms in this study. The terms "projected transaction" and "projected database" are defined in Definitions 18 and 19, respectively. We now illustrate these three algorithms using the databases shown in Tables 1, 2, and 3. Let $minUtil = 1500$ and $maxDist = 5$.

Definition 18. (Projected transaction.) The projection of a transaction $T \in D$ using an itemset α is denoted as $\alpha - T$ and defined as $\alpha - T = \cup_{\forall i \in T \wedge i \in N(\alpha)} i$.

Example 17. Let $\alpha = a$. In Table 1, the item a appears in T_1; therefore, the projected transaction of a, i.e., $a - T = \{(b, 3)\}$. Other items in T_1 are not included in $a - T$ because they are not neighbors of a.

Definition 19. (Projected database.) The projection of a database D using an itemset α is denoted as $\alpha - D$ and defined as the multiset $\alpha - D = \cup_{\forall T \in D \wedge \alpha - T \neq \emptyset} \{\alpha - T\}$.

ts	Items
1	$(a, 2), (b, 3)$
2	$(a, 1), (c, 3)$
3	$(d, 3)$
4	$(c, 4), (b, 3), (d, 1)$
6	$(e, 1), (c, 4), (b, 4), (d, 1)$
9	$(a, 1), (e, 3), (c, 2), (b, 2)$

Table 6: Sorted temporal database generated from Table 1

Example 18. In Table 1, item a appears in T_1, T_2, and T_9; therefore, $a - D = \{\{(b, 3)\}, \{(c, 3)\}, \{(e, 3), (c, 2), (b, 2)\}\}$.

To identify neighbors for each item in the data (Line 3 in Algorithm 1 and entire Algorithm 2), the spatial database shown in Table 3 is used. The list of items and their corresponding neighbors are shown in Table 5. Next, we set $\alpha = \emptyset$ and calculate *PMU* values for each item in Table 1. The list of *PMU* values is $\{\{a : 1550\}, \{b : 2650\}, \{c : 2550\}, \{d : 3250\}, \{e : 2100\}, \{f : 1450\}, \{g : 1350\}\}$ (lines 4 and 5 in Algorithm 1). Next, secondary items with respect to α is generated by comparing *PMU* of each item with *minUtil* (line 6 in Algorithm 1). The generated secondary items are sorted in *PMU* in an ascending order $\succ$; thus, $Secondary(\alpha) = \{a, e, c, b, d\}$ (line 7 in Algorithm 1). Next, we prune all non-secondary items in D, sort the remaining items in each transaction in $\succ$ order, and delete empty transactions. The resulting sorted temporal database containing only secondary items is shown in Table 6 (line 8 in Algorithm 1). Next, we calculate *NSU* for each item in $Secondary(\alpha)$. The *NSU* values for the ordered secondary items is $\{\{a : 1550\}, \{e : 2000\}, \{c : 3300\}, \{b : 1850\}, \{d : 600\}\}$ (line 9 in Algorithm 1). The items that have *NSU* value greater than *minUtil* are considered as *primary items*, i.e., $Primary(\alpha) = \{a, e, c, b\}$ (line 10 in Algorithm 1). Next, we call RecursiveSearch function to generate all SHUIs from the database (line 11 in Algorithm 1).

The RecursiveSearch (i.e., Algorithm 3) works as follows. First, start with the first item a in the $Primary(\alpha)$; moreover, because $\alpha = \emptyset$, $\beta = a$ and we scan D (i.e., $\alpha - D$) and calculate $u(\beta) = u(a) = 350$. Next, we construct a projected database $\beta - D$ (i.e., $a - D$) with only neighboring items of a. Figure 5 (a) shows the projected database of a generated from Table 6. As $u(a) \not\geq minUtil$, a will not be generated as a SHUI. Next, for each item z in $\beta - D$, we calculate *PMU* and *NSU* values. Figure 5(b) shows the *PMU* and *NSU* values calculated for all items in $a - D$. The items having *NSU* greater than or equal to *minUtil* are considered as primary items (i.e., $primary(\beta)$). As all items in $a - D$ have *NSU* value less than *minUtil*, we stop creating the projected databases for the primary item a and move to the next primary item $e \in Primary(\alpha)$. The projected database of e, i.e., $e - D$, is shown in Figure 5(c). Next, we calculate *PMU* and *NSU* values for all items in $e - D$. The items c and b in $e - D$ have $NSU \geq minUtil$; thus, we consider $primary(e) = \{c, b\}$. Next, we construct a projected database for ec, i.e., $(ec - D)$, as shown in Figure 5(e). We then calculated *PMU* and *NSU* for all neighbouring items in $ec - D$, as shown in Figure 5(f). It can be observed from Figure 5(f) that we have not calculated *PMU* and *NSU* for the item b in $ec - D$ because b is not a neighbor of c. The above process is repeated for all primary items. All SHUIs

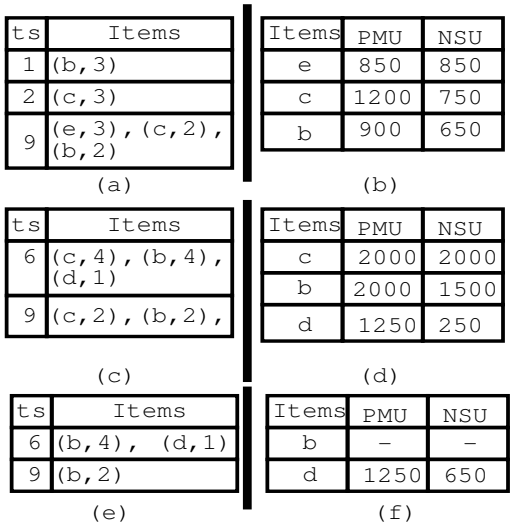

Figure 5: Recursive mining of SHUI-Miner. (a) Projected database of a (i.e., $a-D$), (b) *PMU* and *NSU* values of the items in $a - D$, (c) projected database of e (i.e., $e - D$), (d) *PMU* and *NSU* values of the items in $e - D$, (e) projected database of ec (i.e., $ec - D$), and (f) *PMU* and *NSU* values of the items in $ec - D$

generated by SHUI-Miner for the database shown in Table 1 are presented in Table 4.

5 EXPERIMENTAL RESULTS

Because there exists no algorithm for identifying SHUIs in spatiotemporal databases, we only evaluate the performance of SHUI-Miner. The SHUI-Miner algorithm was written in C++ and executed on a machine with 1.5 GHz processor and 4GB RAM. The experiments have been conducted on two real-world databases, namely, *congestion database* and *pollution database*. The XRAIN database was used to demonstrate the usefulness of SHUIs generated from congestion database. The open source application, QGIS, is used to display some of the interesting SHUIs generated from both databases. We now explain all three databases.

The (traffic) congestion database is provided by JARTIC. The attributes of the congestion database is shown in Table 7. Sample data of JARTIC is shown in Figure 2(a). The data from each sensor was collected over an interval of 5 min. The primary attributes for this database are as follows: *mesh code* and *Road ID*. The distance between the road segments was calculated using geodesic distance. In the experiment, we used road congestion data generated on July 17, 2015, for Kobe, Japan. On this day, Kobe was hit by Typhoon Nangka, which brought over 2 feet of rain and caused 550,000 people to leave Kobe. The source database contained 39,873 transactions and 2,412 items (or RIDs). To record the transactions, we used UTC time. Without loss of generality, original congestion database was split into temporal database, spatial database, and external utility database. The internal utility for the items in the temporal database was determined by counting the number of occurrences of Road ID (RID) at a particular timestamp.

Attribute	Type of attribute
Timestamp	interval data
Mesh code	categorical data
RID	categorical data
Location	line
Con. length	continuous data

Table 7: Attributes of congestion database

Attribute	Type of attribute
Timestamp	interval data
Sensor ID	categorical data
Sensor location	point
$PM_{2.5}$	continuous data

Table 8: Attributes of pollution database

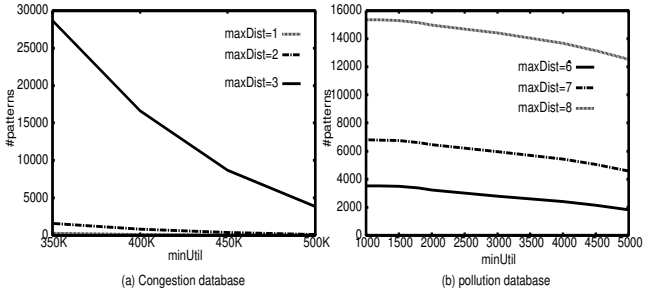

Figure 6: SHUIs generated in pollution database

The air pollution, particularly $PM_{2.5}$ (or simply, PM2.5) concentrate, is the major cause for many cardio-respiratory problems reported in Japan. In particular, the health of elderly people was severely affected by PM2.5. To control and monitor air pollution, AEROS was setup by the Ministry of Environment, Japan. The data generated by various sensors in AEROS helped develop the air pollution database [20]. In this study, we focused on PM2.5, and similar studies were performed for other pollutant concentrates such as NO_2 and O_3. The unit for measuring PM2.5 is microgram per cubicmeter ($\mu g/m^3$). The attributes of pollution database are presented in Table 8. The sample data of this table is shown in Figure 4(a). The air pollution data was generated at an interval of 1 h for 24 h a day. For our experiments, we used air pollution data of the following three days: April 21, 2018; April 22, 2018; and April 23, 2018. The pollution database has 97,651 transactions and 1,026 items (or sensor ids). To record the transactions, we used UTC time. Without loss of generality, the pollution database was split into temporal, spatial, and external utility databases. Since each sensor produces only one value in an hour (i.e., transaction), the internal utility for the items in temporal database is set as 1.

XRAIN data is a high-resolution precipitation raster data provided by Japan Meteorological Agency (JMA). In this study, we used the precipitation data of Typhoon Nangka, which struck Japan on July, 17, 2015.

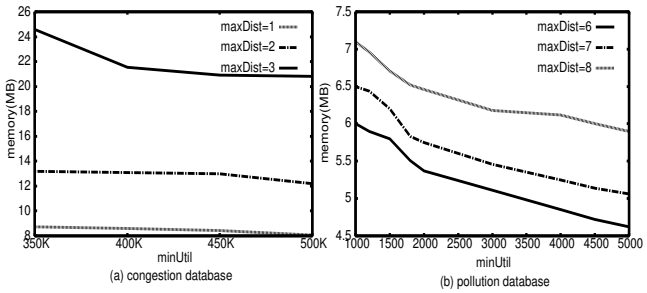

Figure 7: Memory consumed by SHUI-Miner

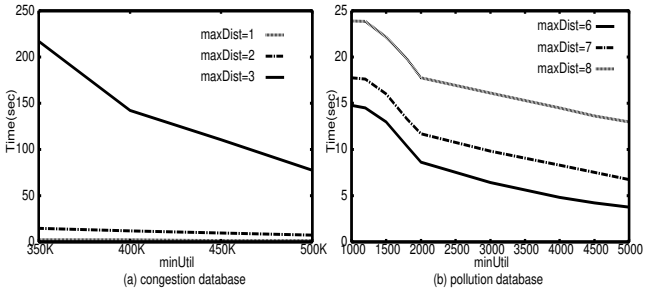

Figure 8: Runtime requirements of SHUI-Miner

Figures 6(a) and (b) show the number of SHUIs generated in congestion and pollution databases at different $minUtil$ and $maxDist$ values, respectively. The following observations can be drawn from these two figures. (*i*) Increase in $maxDist$ results in increase in SHUIs because increase in $maxDist$ increases the neighborhood size of the items, which thereby enables items to combine with more items and generate SHUIs and (*ii*) Increase in $minUtil$ results in decrease in SHUIs. This is because many itemsets fail to satisfy the increased $minUtil$ value.

Figures 7(a) and (b) show the memory requirements of SHUI-Miner in congestion and pollution databases at different $minUtil$ and $maxDist$ values, respectively. The following observations can be drawn from these two figures. (*i*) The memory requirements of SHUI-Miner increases with an increase in $maxDist$ because increase in SHUIs results in an increase in $maxDist$. (*ii*) The memory requirements of SHUI-Miner decreases with increase in $minUtil$ because fewer SHUIs get generated with increase in $minUtil$. (*iii*) Moreover, it is observed that SHUI-Miner takes less time to identify SHUIs in congestion and pollution databases at large or small threshold values.

Figures 8(a) and (b) show the runtime requirements of SHUI-Miner algorithm in congestion and pollution databases at different $minUtil$ and $maxDist$ values. The following observations can be drawn from these two figures. (*i*) The runtime of SHUI-Miner increases with an increase in $maxDist$ because of the increase in the number of SHUIs. (*ii*) The runtime of SHUI-Miner decreases with the increase in $minUtil$ because of the decrease in SHUIs as a result of an increase in $minUtil$. (*iii*) Moreover, it can be observed that SHUI-Miner is efficient for large or small threshold values.

S.No.	SHUI	$Utility$	Area
1	{807, 799, 792, 790, 813, 812}	13959	Tokyo
2	{175, 177, , 176, 174, 173}	8616	Hiroshima
3	{516, 520, 517, 519}	8556	Nagoya
4	{352, 354, 356, 343}	7822	Osaka

Table 9: Some of the interesting SHUIs generated from pollution database

5.1 Case studies

5.1.1 Improving traffic safety. To demonstrate the usefulness of SHUIs in traffic congestion data, we have divided the congestion data of July, 17, 2015, into hourly intervals and applied SHUI-Miner algorithm to each hourly congestion data to identify SHUIs (i.e., sets of road segments where a lot of congestion was observed within the traffic network). The $maxDist$ = 1 km and $minUtil$ = 10000 m (=10 km), i.e., a total of at least 10 km of congestion should be observed in a set of neighboring roads in an hour. Figure 9 shown hourly congestion reported from 6:00 hours to 14:00 hours. The black lines in each of these figures represent the road segments where congestion was observed by a sensor in the entire one hour interval of time. On these black lines, we have overlaid colored lines that represent road segments (or SHUIs) generated in hourly congestion data. Road segments with same color represent a SHUI. Moreover, hourly XRAIN data of the typhoon Nangka was overlaid on the road segments. The following observations can be drawn from these figures: (*i*) roads with heavy congestion (i.e., SHUIs) vary with time (*ii*) roads with heavy congestion were observed in areas with high precipitation (Figures 9(d)-(f)). Thus, the knowledge of SHUIs (or heavy congested roads) can be very useful to the traffic control room in diverting the traffic.

Some of the authors of this paper have previously developed a model to predict congestion on roads using XRAIN data [22]. Applying SHUIM algorithm on this predicted congestion database provides prior information regarding the road segments (SHUIs) where heavy congestion might happen. The users in traffic control room can utilize this knowledge for various purposes such as diverting the traffic, alerting the pedestrians to execute caution while crossing the roads, and suggesting patrol paths for the police.

5.1.2 Identifying highly air polluted locations. Table 9 shows some of the SHUIs generated from air pollution database at $maxDist$ = 7 kilometers and $minUtil$ = 5000 $\mu g/m^3$. The spatial location of these sensors is shown in Figure 10. The exact location of the sensors presented in each SHUI is shown in Figures 11-14. It can be observed in these figures that large quantities of PM2.5 were observed at the sea ports (situated to the east coast of Japan). This information can be found very useful for devising policies for pollution control.

6 CONCLUSIONS AND FUTURE WORK

In this study, we exploited the spatiotemporal characteristics of the items within the data and proposed a flexible model for evaluating SHUI that exists in a spatiotemporal database. Two novel $utility$ upper bound measures, *PMU* and *NSU*, have been introduced to reduce the search space of SHUIM. Moreover, we have theoretically shown that the proposed measures are tighter than the current utility upper bound measures when it comes to mining SHUIs. A single pass algorithm has been proposed to effectively determine all SHUIs in spatiotemporal databases. Note that experimental results demonstrate that the proposed algorithm is efficient, and we also presented two case studies to demonstrate the effectiveness of SHUIs in real-world applications.

In this study, we examined SHUIM by considering only positive external utility values. As a part of our future work, we would like to extend the proposed model of SHUI by considering both positive and negative external utility values. We focused on identifying SHUIs in static spatiotemporal data, and it would be interesting to investigate SHUIM in graphs, data streams, and symbolic databases in future.

7 ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to Mr. Yuji Uchiyama, Picolab Co., Ltd., Tokyo, Japan for development of the experimental system at National Institute of Information and Communication Technologies, Tokyo, Japan.

REFERENCES

[1] Rakesh Agrawal, Ramakrishnan Srikant, et al. 1994. Fast algorithms for mining association rules. In *VLDB*, Vol. 1215. 487–499.
[2] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, and Byeong-Soo Jeong. 2010. Mining High Utility Web Access Sequences in Dynamic Web Log Data. In *International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD '10)*. 76–81.
[3] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee. 2009. Efficient tree structures for high utility pattern mining in incremental databases. *IEEE TKDE* 21, 12 (2009), 1708–1721.
[4] Statistics Bureau. [Online accessed on 28-Feb-2019]. Japanese Industrial Standard X0410. http://www.stat.go.jp/english/data/mesh/02.html.
[5] C. H. Cai, A. W. C. Fu, C. H. Cheng, and W. W. Kwong. 1998. Mining association rules with weighted items. In *International Database Engineering and Applications Symposium*. 68–77.
[6] Wei Ding, Christoph F. Eick, Jing Wang, and Xiaojing Yuan. 2006. A Framework for Regional Association Rule Mining in Spatial Datasets. In *Proceedings of the Sixth International Conference on Data Mining (ICDM '06)*. 851–856.
[7] Christoph F. Eick, Rachana Parmar, Wei Ding, Tomasz F. Stepinski, and Jean-Philippe Nicot. 2008. 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)*. Article 30, 10 pages.
[8] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Quang-Huy Duong, and Thu-Lan Dam. 2016. PHM: Mining Periodic High-Utility Itemsets. In *Industrial Conference on Data Mining*. 64–79.
[9] Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, Tzung-Pei Hong, and Hamido Fujita. 2018. A survey of incremental high-utility itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 2 (2018).
[10] R. Uday Kiran, Amulya Kotni, P. Krishna Reddy, Masashi Toyoda, Subhash Bhalla, and Masaru Kitsuregawa. 2018. Efficient Discovery of Weighted Frequent Itemsets in Very Large Transactional Databases: A Re-visit. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*. 723–732.
[11] Guo-Cheng Lan, Tzung-Pei Hong, and Vincent S Tseng. 2014. An efficient projection-based indexing approach for mining high utility itemsets. *Knowledge and information systems* 38, 1 (2014), 85–107.
[12] Bing Liu, Wynne Hsu, and Yiming Ma. 1999. Mining association rules with multiple minimum supports. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 337–341.
[13] Junqiang Liu, Ke Wang, and Benjamin CM Fung. 2012. Direct discovery of high utility itemsets without candidate generation. In *ICDM*. IEEE, 984–989.
[14] Mengchi Liu and Junfeng Qu. 2012. Mining high utility itemsets without candidate generation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 55–64.
[15] Ying Liu, Wei-keng Liao, and Alok Choudhary. 2005. A two-phase algorithm for fast discovery of high utility itemsets. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 689–695.
[16] Yu-Cheng Liu, Chun-Pei Cheng, and Vincent S. Tseng. 2013. Mining differential top-k co-expression patterns from time course comparative gene expression

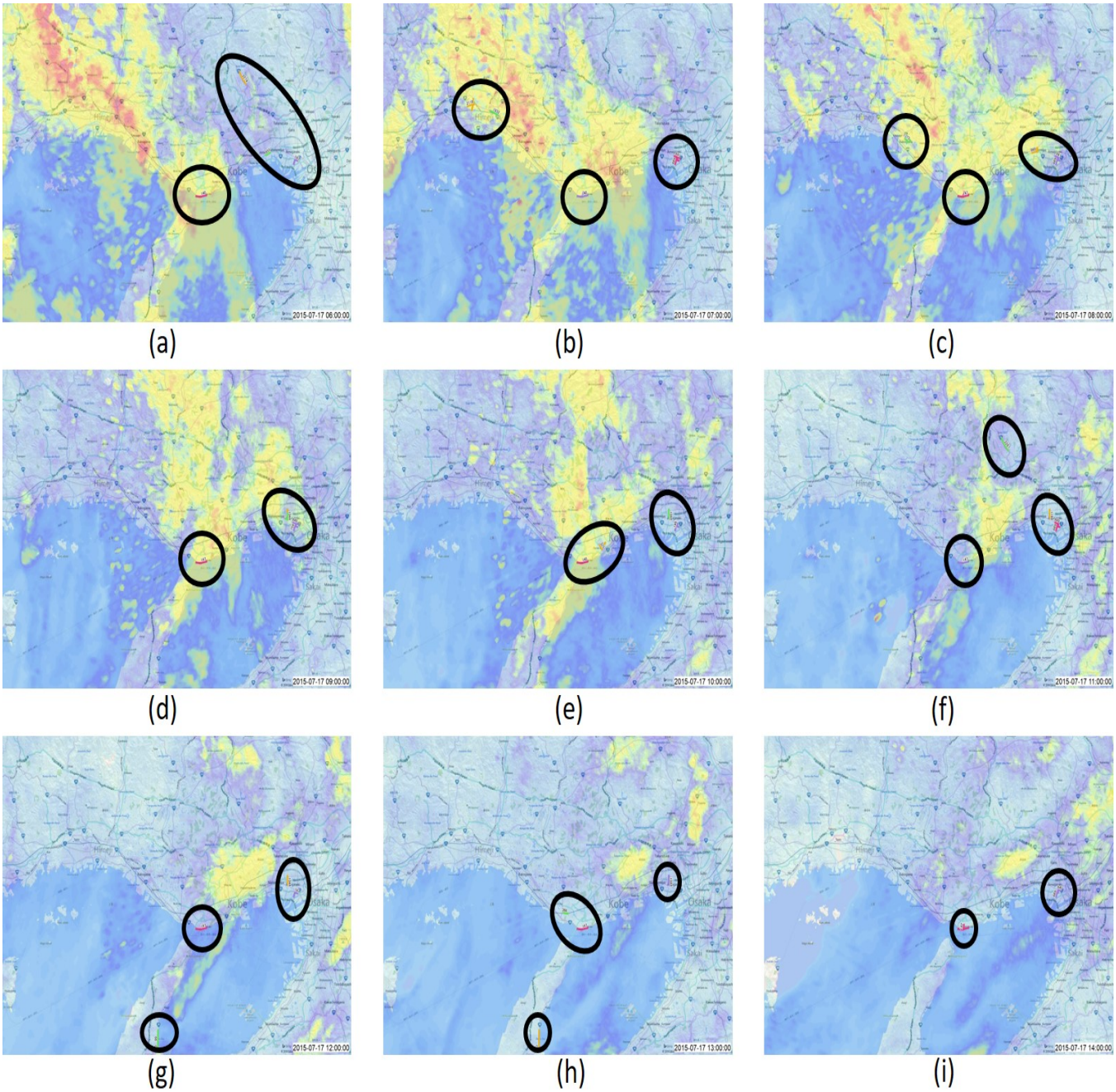

Figure 9: SHUIs generated from the hourly congestion data. XRain (or precipitation) data is overlaid at hourly intervals. High congested road segments have been shown within the dark ovals. (a) to (i) show some of the SHUIs generated from the hourly congestion data starting from 6 to 14 h (UTC time)

datasets. *BMC Bioinformatics* 14, 1 (21 Jul 2013), 230.

[17] Pradeep Mohan, Shashi Shekhar, James A. Shine, James P. Rogers, Zhe Jiang, and Nicole Wayant. 2011. 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)*. 122–132.

[18] Jian Pei and Jiawei Han. 2000. Can We Push More Constraints into Frequent Pattern Mining?. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00)*. 350–354.

[19] Jian Pei, Jiawei Han, and Wei Wang. 2007. Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems* 28, 2 (01 Apr 2007), 133–160.

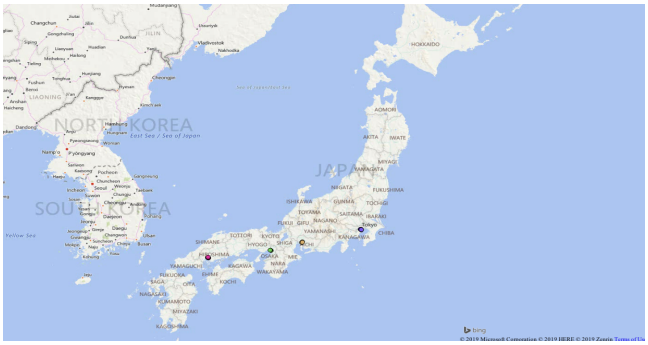

Figure 10: Spatial location of SHUIs generated in pollution data

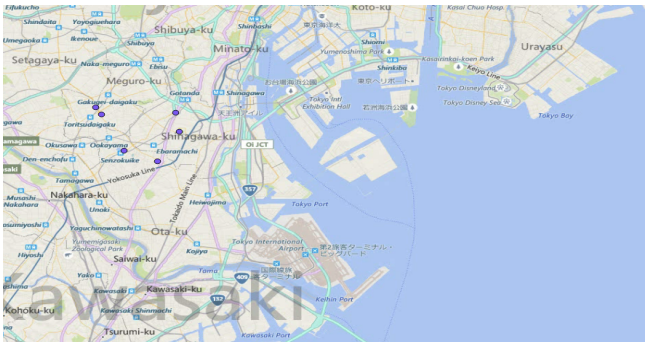

Figure 11: Sensors that recorded high quantities of PM2.5 in Tokyo area

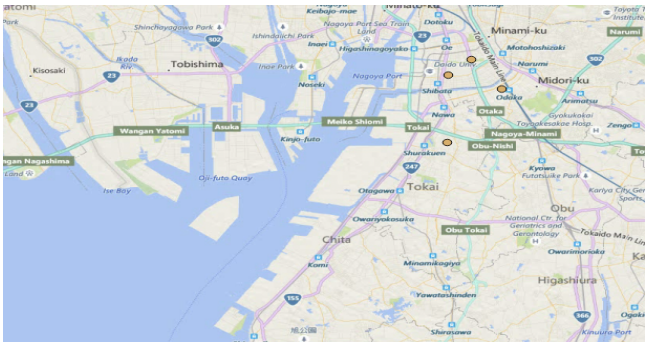

Figure 12: Sensors that recorded high quantities of PM2.5 in Nagoya area

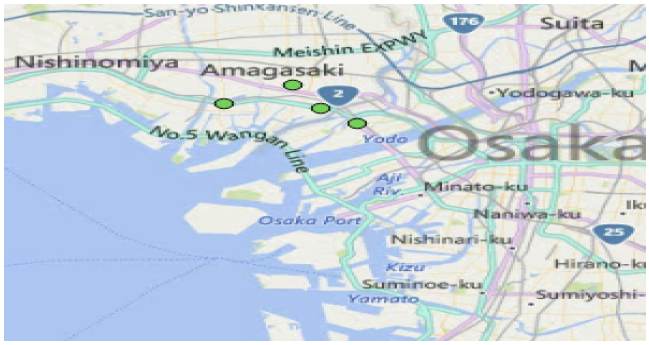

Figure 13: Sensors that recorded high quantities of PM2.5 in Osaka area

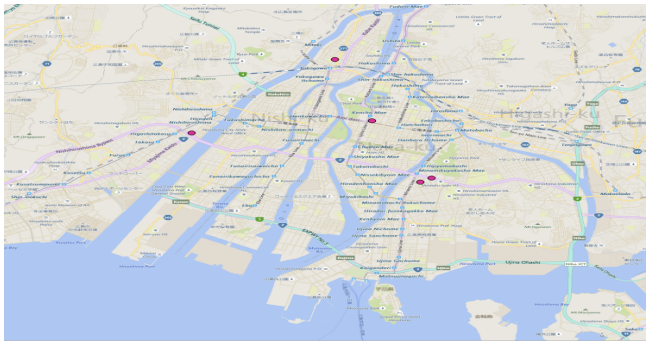

Figure 14: Sensors that recorded high quantities of PM2.5 in Hiroshima area

[20] Tomohiro Sato, Minh-Son Dao, Kota Kuribayashi, and Koji Zettsu. 2019. SEPHLA: Challenges and Opportunities Within Environment - Personal Health Archives. In *MultiMedia Modeling - 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8-11, 2019, Proceedings, Part I*. 325–337.

[21] Wei Song, Yu Liu, and Jinhong Li. 2014. BAHUI: fast and memory efficient mining of high utility itemsets based on bitmap. *International Journal of Data Warehousing and Mining (IJDWM)* 10, 1 (2014), 1–15.

[22] Hung Tran-The and Koji Zettsu. 2017. 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*. 1006–1011.

[23] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu. 2013. Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases. *IEEE Trans. on Knowl. and Data Eng.* 25, 8 (Aug. 2013), 1772–1786.

[24] R. Uday Kiran, T. Yashwanth Reddy, Philippe Fournier-Viger, Masashi Toyoda, P. Krishna Reddy, and Masaru Kitsuregawa. 2019. Efficiently Finding High Utility-Frequent Itemsets Using Cutoff and Suffix Utility. In *Advances in Knowledge Discovery and Data Mining*, Qiang Yang, Zhi-Hua Zhou, Zhiguo Gong, Min-Ling Zhang, and Sheng-Jun Huang (Eds.). Springer International Publishing, Cham, 191–203.

[25] Hong Yao, Howard J Hamilton, and Cory J Butz. 2004. A foundational approach to mining itemset utilities from databases. In *SIAM*. 482–486.

[26] Junfu Yin, Zhigang Zheng, and Longbing Cao. 2012. USpan: An Efficient Algorithm for Mining High Utility Sequential Patterns. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. 660–668.

[27] Souleymane Zida, Philippe Fournier-Viger, Jerry Chun-Wei Lin, Cheng-Wei Wu, and Vincent S Tseng. 2017. EFIM: a fast and memory efficient algorithm for high-utility itemset mining. *Knowledge and Information Systems* 51, 2 (2017), 595–625.

[28] Souleymane Zida, Philippe Fournier-Viger, Cheng-Wei Wu, Jerry Chun-Wei Lin, and Vincent S Tseng. 2015. Efficient mining of high-utility sequential rules. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. 157–171.