SMIM framework to generalize high-utility itemset mining

Siddharth Dawar[1], Vikram Goyal[1], and Debajyoti Bera[1]

Indraprastha Institute of Information Technology (IIIT-Delhi), India
{siddharthd,vikram,dbera}@iiitd.ac.in

Abstract. In high-utility itemset mining (HUIM), the utility of a set of items is calculated as the sum of the utilities of the individual items. In this paper, we describe scenarios where utility may be less than this sum for multi-item itemsets. To overcome the limitation of the current itemset mining algorithms for such scenarios, we introduce the SMIM framework for itemset mining in which utilities are constrained to be non-negative subadditive and monotone functions over itemsets. SMIM generalizes HUIM, can be used to analyse transaction databases with multi-item discount schemes, and can further be used to mine interesting patterns in a social network dataset. Finally, we explain how to design algorithms for SMIM with any general subadditive mototone utility function.

1 Introduction

High-utility itemset mining (HUIM) involves a transactional database, in which every transaction consists of a set of weighted items, and the objective is to identify a set of items (*aka.* itemsets) whose total weight in the entire database crosses a threshold [7,24]. The weight of an itemset $X = \{x_1, x_2, \ldots, x_k\}$ in the database $\mathcal{D}$ is the sum of the weights of X in all transactions of $\mathcal{D}$ that contain X, and the weight of X in such a transaction T, usually denoted $u(X,T)$ for utility, has been *traditionally computed as the sum* of the weights (or utilities) of the individual items of X in T, denoted $u(x,T)$.

$$u_{\mathcal{D}}(X) = \sum_{T \subset \mathcal{D}, X \subseteq T} u(X,T) = \sum_{T \subset \mathcal{D}, X \subseteq T} \mathrm{Sum}_{x \in X} u(x,T)$$

Current state-of-the-art algorithms employ highly optimized data structures to process large number of transactions at breathtaking speed on modest hardware. However, they are unsuitable in the scenarios where utility of an itemset in a transaction is *not* defined as the sum of the itemized weights, but some different function.

For example, consider a utility function that defines the utility of an itemset as the sum of its individual utilities discounted by the lowest utility: $u(\{x_1, x_2, \ldots x_k\}, T) = \sum_{i=1}^{k} u(x_i, T) - \min\{x_1, \ldots, x_k\}$. This situation can arise

during discount seasons ("buy 1 shirt and 1 trouser, get the lowest free"), or when hampers are created consisting of items of different nature. Even though HUIM algorithms were proposed for transactions consisting of discounted items, the discounts considered were applicable on only single items ("buy 2 toothpastes, get another toothpaste free") [12,2]. These algorithms are difficult to extend to the former scenario where utility is no longer a simple *sum* of itemwise utilities. Ad-hoc adjustments to HUIM become difficult if the discount schemes are numerous, complex, and use complicated formulæ involving many items.

The focus of this work is not HUIM for discounted transactions, but how far can the idea of a "non-addition" utility function be pushed? Quite far, as can be seen in the list of contributions below.

- We propose an itemset mining problem named SMIM in which the utility function is not necessarily Sum, but any subadditive and monotone (SM) function on itemsets. This means that the utility of an itemset X in some transaction T cannot decrease when a new item from T, say a, is added to X, and when it increases, it does so by at most the utility of a in T.
- We show that SM utility functions have mathematically helpful properties, and can model HUIM, as well as multi-item discounted itemsets in retail transactions. Further, they can be used to identify active and popular users in a Twitter dataset. Treating a Twitter dataset as transactional is easy (e.g., each transaction can include the users active on each day), but identification of complex patterns requires more general utility functions than Sum.
- We design an algorithm named SM-Miner for SMIM where $u(\cdot, T)$ is given as a blackbox. We show how to adapt the existing HUIM algorithms for SMIM, but empirically show that SM-Miner delivers better performance.

One novelty of our solution is a single algorithm for any utility function that can be proved to be subadditive and monotone.

Our framework for SMIM allows us to capture extraneous interactions among the items in an itemset in their joint utility which is not possible in the HUIM framework. Another novelty of our solutions for SMIM is the use of the superfast HUIM algorithms [7] which show that they are still relevant and applicable towards non-trivial utility functions. Future improvements on HUIM algorithms can possibly speedup SMIM too.

2 Problem Statement

Consider a set $\mathcal{U}$ of items where each item has a positive real weight (we do not consider "negative utility") denoted $w(x)$.

Definition 1 (subadditive and monotone function). *A function $f : \mathcal{U} \to \mathbb{R}+$ is defined as subadditive if $\forall X, Y \subseteq \mathcal{U}$, $f(X \cup Y) \leq f(X) + f(Y)$, and is defined as monotone if $\forall X \subseteq Y \subseteq \mathcal{U}$, $f(X) \leq f(Y)$.*

The $Sum()$ function, defined as $Sum(X) = \sum_{x \in X} w(x)$, is a well-known example of such a function. However, $Min(X) = \min_{x \in X} w(x)$ is not monotone (e.g., say $X = \{x_1, x_2\}$, $w(x_1) = 4$ and $w(x_2) = 2$).

Now, let T be some transaction in a database $\mathcal{D}$ with positive weights on items that are denoted $w_T(x)$. We define the utility of a singleton itemset $\{x\}$ as $u(\{x\}, T) = w_T(x)$. We consider the general scenario where utility over the items in T is some monotone and subadditive function over T, denoted $u(X, T)$. The utility of an itemset X in the entire database $\mathcal{D}$, denoted $u_{\mathcal{D}}(X)$, is defined as the sum of $u(X, T)$, summed over all T that contains X. We do not require the monotone and subadditive property to hold for $u_{\mathcal{D}}(.)$.

The SMIM problem considers a transaction database $\mathcal{D}$, and a utility function $u(\cdot, \cdot)$ defined as above, and wants to know the itemsets whose utility over $\mathcal{D}$ is at least a given threshold θ. SMIM generalizes HUIM since Sum is an SM function, and variations of HUIM that were studied by Yao [23] for similar reasons.

3 Related work

Several algorithms have been proposed in the literature for HUIM [1,22,14,6,5,10]. Broadly these algorithms vary in terms of the number of stages they run for (one phase and two-phase), their database representation (tree, utility list, and projected database) and other data structures and heuristics used to prune non-high-utility itemsets effectively. All the HUIM algorithms explore itemsets in a branch-and-bound manner; however, to limit the number of database scans, they use upper bounds functions on the utility function to prune itemsets and decide exploration branches [16,15]. All the above techniques work for a specific utility function, $\sum_{x \in I} u(\{x\}, T)$ to be precise, and do not generalize to arbitrary (or even subadditive monotone) utility functions. The research community has contributed significantly to improve the efficiency of high-utility itemset mining algorithms in the last few years. One of our objectives is to identify protocols to adapt these algorithms to the SMIM framework. Our SMIM algorithms are designed in a similar manner, but with modifications to the upper bound function that are necessitated by the non-Sum nature of SMIM utility functions.

Declaration constraint programming has been used to generalize various notions of itemset mining. In this approach the constraints are expressed in a high-level language and a separate solver is used to identify the appropriate itemsets. Silva et al. [18] proposed a framework for constraint pattern mining that allowed anyone to organize and analyze different algorithms based on the properties of constraints like anti-monotonicity, monotonicity, succinctness, etc. Guns et al. [9] introduced a declarative framework named MiningZinc that can express HUIM. Coussat et al. [4] defined the problem of HUIM in uncertain tensors, and showed that an algorithm called multidupehack [3] could be deployed for a version of the high-utility itemset mining problem where the utility of an item is more general than in HUIM. However, both MiningZinc and multidupehack define the utility of an itemset as the sum of utility of individual items, akin to HUIM, and cannot be generalized to the SMIM scenario.

Subadditive monotone set functions have appeared in quite a few works on identifying important groups of entities, however, we are aware of only one work that is somewhat related to that of ours, but involving sequences of items. Tschiatschek et al. [20] introduced a utility function over a sequence of items, which is

further defined using a submodular monotone function, that captures the ordered preferences among items over arbitrarily long ranges. They proposed a greedy algorithm for selecting a sequence of items under sequence length constraints and showed an application of their algorithm for the task of recommending a sequence of movies to a user.

Our approach is similar to that adopted by Yao et al. [23]. They unified several forms of "utility" measures proposed for itemset mining until 2006, including the ones used in FIM (frequent itemset mining) and HUIM, using a common framework and then identified common mathematical properties that can be used to design efficient pruning strategies. We not only generalize all those utility measures, but go far beyond and present interesting examples along with a case-study that can identify patterns in a social network.

4 Examples of SMIM

We show two scenarios that can be modelled using SMIM.

Example 1 – Discount scheme: First, consider a retail store that is running this multi-product discount scheme: "Buy 1 pencil (P), get 1 eraser (E) free". Observe that utility of any itemset that does not contain both pencil(s) and eraser(s) is once again the sum of the itemwise utilities; for the other itemsets, its utility depends on the quantities of pencils and erasers. Let c_p and c_e denote the costs of 1 pencil and 1 eraser, respectively; let n_p and n_e denote the number of pencils and erasers in T, respectively. We can express the utility function on any transaction T as

$$u(X,T) = \begin{cases} Sum(X,T) & \text{if } \{P,E\} \not\subset T \\ Sum(X \setminus \{P,E\},T) + n_p \times c_p & \text{o/w, if } n_p \geq n_e \\ Sum(X \setminus \{P,E\},T) + n_p \times c_p + (n_e - n_p) \times c_e & \text{o/w, if } n_p < n_e \end{cases}$$

We now prove that the above $u(X,T)$ is subadditive and monotone under the reasonable assumption that $c(E) \leq c(P)$. Note that if there are n_p pencils in T, and X contains pencils, then the quantity of pencils in X is also n_p; in other words, X cannot contain a partial amount of some items.

To show that $u(X,T)$ is monotone, consider some $y \notin X$, and define $Y = X \cup \{y\}$. Now, if y is not a pencil or an eraser, then $u(Y,T) = u(X,T) + u(y,T) \geq u(X,T)$. When y is a pencil or eraser, $u(Y,T)$ may be different depending on whether X already had the other. However, since the cost of 1 eraser is at most that of 1 pencil, the total cost of Y cannot be less than that of X (even when there are an unequal number of pencils and erasers).

A similar line of arguments is used to prove subadditivity for which it is sufficient to consider two disjoint sets of items X and Y. Now, the only situation when $u(X \cup Y,T)$ would be different from $u(X,T) + u(Y,T)$ is when, *wlog.*, X contains pencil(s) and Y contains eraser(s). However, in that scenario, the combined utility $u(X \cup Y,T)$ would be less than $u(X,T) + u(Y,T)$ since 1 or more erasers would now be given for free.

Even though we discuss only one discount scheme, we are hopeful that many other common discount schemes can also be brought into the folds of SMIM.

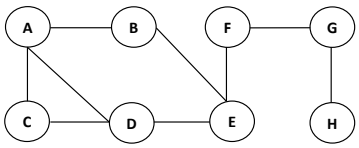
Example 2 – Transactions along with a relationship graph: For the second scenario consider a Twitter dataset containing tweets and retweets of many users and their followers, over a period of several months. We construct a transaction database, denoted $\mathcal{D}$, by fixing (say) 3-hour intervals to construct each transaction that represents the active users during this time period. The weight or utility (denoted $u(x,T)$) of a user x is set to the number of tweets posted by that user in that period. Next, we constructed a directed follower-followee graph, denoted $\mathcal{G}$, on the users by adding a directed edge from a user A to user B if B has retweeted at least one tweet posted by A in any of the time intervals.

Observe that identifying itemsets with high frequency or high utility in $\mathcal{D}$ would fail to account for the "following" relationship exhibited by the retweets. To incorporate this, we use the graph-theoretic notion of coverage of a set of nodes in G which is defined as $Co(X) = |X \cup \bigcup_{x \in X} \text{neighbor(x)}|$ in which the neighbours of x are those users that retweeted at least one tweet of X. We interpret $Co(X)$ as a measure of the *collective influence* of X. We proved that $Co()$ is an SM function and can be used in SMIM; however, $Co()$ does not capture temporal behaviour.

To further identify groups of users who are both "active and possess large influence as a group", an obvious utility function is the following: $sumcov(X) = \sum_{x \in X} u(x,T) \times Co(\{x\})$. *sumcov* defines the itemset utility as sum of values for each item, and just like Sum in HUIM, is subadditive and monotone.

Fig. 1: Comparative illustration of traditional HUIM and SMIM. $u(I)$ denotes the utility of an itemset I as defined in traditional HUIM, whereas $ucov(I)$ denotes its utility whose computation involves both the transaction database and an external graph given below.

TID	**Transaction**	$u(A) = ucov(A)$	$u(C) = ucov(C)$	$u(AC)$	$ucov(AC)$
T_1	$(A:5)\,(C:10)\,(D:2)$	5	10	15	$5Co(AC) + 5Co(C) = 35$
T_2	$(A:10)\,(C:6)\,(E:6)\,(G:5)$	10	6	16	$6Co(AC) + 4CO(C) = 36$
T_3	$(A:10)\,(B:4)\,(D:12)\,(E:6)\,(F:5)$	10	0	0	0
T_4	$(A:5)\,(B:2)\,(C:3)\,(D:2)\,(H:2)$	5	3	8	$3Co(AC) + 2Co(A) = 18$
T_5	$(B:8)\,(C:13)\,(D:6)\,(E:3)$	0	13	0	0
T_6	$(B:4)\,(C:4)\,(E:3)\,(G:2)$	0	4	0	0
T_7	$(F:1)\,(G:2)$	0	0	0	0
T_8	$(F:4)\,(G:3)$	0	0	0	0
	Total utility in database	30	36	39	89



To illustrate the ability to create interesting utility functions, we designed another utility function towards the same objective. For this, it will be helpful to recall that $u(\{x\},T)$ indicates *activity* of x. Let T be a transaction in $\mathcal{D}$ in which q_i

denotes the utility of x_i in T, and the items are ordered such that $q_i \leq q_{i+1}$ for all i. Let $X = \{x_1, x_2, \dots x_k\}$ be an itemset from T. The *coverage utility* of X in T is defined as $ucov(X, T) = q_1 \times Co(X) + \sum_{j=2}^{k}(q_j - q_{j-1}) \times Co(\{x_j \cdots x_k\})$. Figure 1 illustrates $ucov()$ and compares it with $u()$ on an example dataset. For example, $ucov(ACD, T_1)$ in the database in Figure 1 is 37 and $sumcov(\{ACD\}, T_1) = 5 \times Co(\{A\}) + 10 \times Co(\{C\}) + 2 \times Co(\{D\}) = 5 \times 4 + 10 \times 3 + 2 \times 4 = 58$.

We further analysed a Twitter dataset using *ucov* and *sumcov* and showed that the former may be a suitable utility function (see the full version for details). However, it may not be easy to identify high-utility patterns based on a complicated function like *ucov*. Our strategy for this is to first prove that *ucov* is subadditive and monotone, and then use the generic algorithms in the next section that operate on any SM utility function. Proving *ucov* as subadditive and monotone turned out to be quite involved. The proofs are included in the full version of this paper [1]. We discuss the algorithms in the next section.

5 Algorithmic framework for SMIM

Like the specialized HUIM algorithms where the utility function is Sum, specific algorithms have to be designed for specific SM utility functions. Here we lay down a general framework to design SMIM algorithms by adapting HUIM algorithms only relying on the fact that a utility function is subadditive and monotone.

Existing HUIM algorithms can be categorized into list-based, tree-based, and projection-based. These algorithms devise strategies to upper bound certain utility values and employ efficient data structures to explore the space of itemsets with clever pruning strategies using those upper bounds. We show that for SMIM too similar data structures and exploration algorithms can be designed with appropriate upper bound functions. The proofs of the claims in the section are available in the full version of this paper.

5.1 Projection-based algorithms

An essential step in the projection-based algorithms like EFIM [25], D2HUP [13], and MAHI [19] is merging two identical transactions to reduce the costly database scans during the construction of a projected database for an itemset. We explain why this should not be done for arbitrary utility functions.

Let $T = \{(A_1 : q_1), \dots (A_n : q_n)\}$ and $S = \{(A_1 : r_1), \dots (A_n : r_n)\}$ be two transactions with the same items in them, but with possibly different weights. Merging T and S to create a transaction $M = \{(A_1 : q_1 + r_1), \dots (A_n : q_n + r_n)\}$ does not change the utility of the itemset $J = \langle A_1, A_2, \dots A_n \rangle$ in the database for the $Sum()$ utility function that is used for HUIM. However, utility of J may not satisfy the same for an arbitrary SM utility function. For example, $ucov(\{F, G\}, T_7) = 7$ and $ucov(\{F, G\}, T_8) = 15$ for the transactions T_7 and T_8 in Figure 1, but if we merge them to a single transaction $M = \{(F : 5), \dots (G : 5)\}$, then we get $ucov(\{F, G\}, M) = 20$. Therefore, transaction merging should be disabled when adapting projection-based algorithms to SMIM.

[1] `http://www.iiitd.edu.in/~dbera/docs/2021-smim.pdf`

5.2 Tree-based algorithms

A tree-based algorithm like UP-Growth and UPGrowth+ [21] creates a tree data structure from a transaction database which is then used to find potential high utility candidates. They create a global tree structure before starting the mining process. Every node of this tree stores an itemset, a pointer to its parent node, its support, an upper-bound of its utility (like transaction-weighted utility (TWU)), and pointers to its child nodes which are extensions of the itemset by one more item. As the algorithm recursively visits a node, local trees are created and used for recursively exploring its child nodes. An itemset X is unpromising in a transaction database if $TWU(X)$ is less than the minimum utility threshold θ. Unpromising itemsets are removed, thereby pruning the search space, during both the global and local tree creation; removing unpromising items have an added advantage of deriving better estimates during exploration.

The general idea can be extended to SM utility functions. The unpromising items can be removed during a global tree creation with an additional step to recompute the utility of a transaction after removing the unpromising items (this is easy for Sum). However, removing unpromising items during local tree creation may not give correct upper-bound estimates for an arbitrary function, even though it works for the Sum function. Imagine a tree-based algorithm at an intermediate stage and let Y denote the path to a node in the tree. Further, consider the case where an item A appears on the path to Y and, at that intermediate stage, A was found to be unpromising. The tree-based algorithms at this point remove A from the local tree and re-adjust the utility upper bounds of all the nodes on the path to Y. To update the utility upper bound of some node, say Z, the algorithm simply subtracts from it $\sum_T u(\{A\}, T)$ summed over all transactions that contain A. This happens to be correct when $u = Sum$ since the $Sum(Z \setminus \{A\}, T) = Sum(Z, T) - Sum(\{A\}, T)$, but may not necessarily hold for other utility functions. If we nevertheless use the same method and update $u(Z \setminus \{A\}, T) = u(Z, T) - u(\{A\}, T)$, we may incorrectly prune itemsets with high-utility; this is since subadditivity only guarantees that $u(Z \setminus \{A\}, T) > u(Z, T) - u(\{A\}, T)$ which clearly shows that the updated value, which appears on the right-hand side, could be lesser than the actual utility value of the left.

Let $T = \{(A_1 : q_1), \ldots (A_n : q_n)\}$ be some transaction, X denote the itemset $X = \{A_1\}$ and Y denote the itemset $\{A_2, \ldots, A_n\}$ with $n - 1$ items. Suppose it happens that the item A_1 is unpromising, i.e., $TSMWU(A_1) < \theta$; thus, A_1 cannot be a part of any high-utility itemset. It can be easily verified that $u(X, T) + u(Y, T) = u(X \cup Y, T)$ when u$(\cdot)$ is the $Sum(\cdot)$ defined for HUIM. Therefore, $u(Y, T) = u(X \cup Y, T) - u(X, T)$ gives a correct bound for HUIM. However, computing tighter utility estimates by removing unpromising items can result in incorrect upper bound estimate i.e. resulting in false negatives for those functions where $f(X, T) + f(Y, T) > f(X \cup Y, T)$. We found that the tree-based algorithms for HUIM can be adapted for arbitrary subadditive monotone function without any change but for the removal of the unpromising items during local tree creation.

We identified an additional optimization to speed things up. The transaction utility (TU) of a transaction is the *sum* of the utilities of its items. TWU of an itemset X is the sum of TU for transactions that contain X. The *transaction-weighted downward closure* property of TWU ensures that if $TWU(X)$ is less than the threshold, then X and its super-sets cannot be high-utility itemsets [15]. We define related terms for SM functions.

Definition 2 (TSMU and TSMWU). *The transaction subadditive monotone utility of a transaction T is defined as $TSMU(T) = u(T, T)$. The transaction subadditive monotone weighted utility (TSMWU) of an itemset X is the sum of $TSMU(T)$ for all the transactions containing X.*

For example, consider transaction T_1 from Figure 1. TU for T_1 (using *ucov* for utility) is equal to $ucov(\{A\}, T_1)+ucov(\{C\}, T_1)+ucov(\{D\}, T_1)$ which evaluates to $20 + 30 + 8 = 58$. However, $TSMU(T_1) = ucov(\{ACD\}, T_1) = 37$.

Observe that $TSMU(T) = TU(T)$ and $TSMWU = TWU$ when $Sum()$ is used as the utility function (as in HUIM). Like TWU, we were able to show that $TSMWU$ satisfies the downward-closure property; further, it turns out be a tighter bound compared to TWU for an arbitrary $u(\cdot, \cdot)$.

Lemma 1. *If $TSMWU(X)$ is less than the threshold, then X and its supersets cannot be high-utility itemsets. Further, $TSMWU(X)$ is a tighter upper-bound compared to $TWU(X)$.*

5.3 SM-Miner algorithm

The list-based HUIM algorithms [14,6] use the exact-utility (EU) and remaining-utility (RU) bounds that are tighter compared to the TU bound explained earlier. These algorithms process items according to some fixed order, and the items in each transaction are sorted accordingly. Let X be some itemset that is being processed, T be some transaction containing X and T/X be the *items appearing after* X in T (note that $X \cup (T/X)$ is not T, e.g., items appearing in T *before* X in the processing order are not in T/X). The exact utility $EU(X, T)$ is the *sum* of the utilities of the items in X in T and the remaining utility $RU(X, T)$ is defined as the *sum* of the utilities of the items in T/X. It can be shown that X and its extensions (according to the processing order) cannot be high utility if $EU\_RU(X) = \sum_{T \supseteq X} EU(X, T) + RU(X, T)$ is less than the threshold [14]; this fact is used in HUIM algorithms to decide whether to examine extensions of the currently explored itemset X. Towards this purpose, they maintain a utility-list consisting of triples $\langle$Transaction id (TID), Exact-utility ($EU(X, T)$), Remaining-utility ($RU(X, T)$)$\rangle$ with each itemset X. They scan the transaction database to construct the utility-list for promising items, i.e., items with TWU no less than the minimum utility threshold. The utility-list for a $\{k\}$-itemset (an itemset consisting of k items) is constructed by intersecting lists of two $\{k-1\}$ itemsets with the same prefix. The algorithms also store two mappings $sumEU(X)$ and $sumRU(X)$ that store the sum of $EU(X, T)$ and $RU(X, T)$ for all transactions that contain an itemset X.

Table 1: Illustration explaining Algorithm 1 on the dataset in Figure 1.

SMI list of $\{A\}$

TID	CWI	RWI
1	$\{(A:5)\}$	$\{(C:10),(D:2)\}$
2	$\{(A:10)\}$	$\{(C:6),(E:6),(G:5)\}$
3	$\{(A:10)\}$	$\{(B:4),(D:12),(E:6),(F:5)\}$
4	$\{(A:5)\}$	$\{(B:2),(C:3),(D:2),(H:2)\}$

SMI-list of $\{B\}$

TID	CWI	RWI
3	$\{(B:4)\}$	$\{(D:12),(E:6),(F:5)\}$
4	$\{(B:2)\}$	$\{(C:3),(D:2),(H:2)\}$
5	$\{(B:8)\}$	$\{(C:13),(D:6),(E:3)\}$
6	$\{(B:4)\}$	$\{(C:4),(E:3),(G:2)\}$

SMI-list of $\{A,B\}$

TID	CWI	RWI
3	$\{(A:10),(B:4)\}$	$\{(D:12),(E:6),(F:5)\}$
4	$\{(A:5),(B:2)\}$	$\{(C:3),(D:2),(H:2)\}$

We show how to define a tighter bound for utility functions that are arbitrary, not necessarily *Sum*, but subadditive and monotone.

Definition 3 (Combined utility (CU)). *Given an itemset X and a transaction T with $X \subseteq T$, the combined utility of X is defined as $CU(X,T) = u(X \cup T/X, T)$ and $CU(X) = \sum_{T \supseteq X} CU(X,T)$.*

Note that $CU(X) = EU\_RU(X)$ when *Sum* is used as the utility function (as in HUIM). We were able to prove the following properties which show that we can use $TSMWU$ in place of TWU and CU in place of EU_RU to design our SM-Miner algorithm. The proof of the lemma is included in the full version of the paper.

Lemma 2. *If $CU(X)$ is less than a threshold, any extension X' of X in the processing order of items is not a high-utility itemset. Further, $CU(X)$ is a tighter upper-bound compared to $EU\_RU(X)$.*

Our algorithm uses a data structure called SMI-list to store a list of triples of the form ⟨Transaction Id (TID), Current Weighted Itemset (CWI), Remaining Weighted Itemset (RWI)⟩ with each itemset X. Here CWI stores the item-quantity information for all items in X present in the transaction given by TID and RWI stores the items with their quantities which appear *after* X in a transaction. We associate with each item X two variables $SumEU(X)$ and $SumCU(X)$ that accumulate the EU and CU values during the construction of the inverted-list for X. A variable combined-utility (CU) stores $CU(X)$ with the utility-list for every itemset X. The construction of an SMI-list is described in Algorithm 1 in which we do not scan the SMI-list of the prefix while constructing the list for a k-itemset (unlike HUIM algorithms like HUI-Miner and FHM [14,6]) since an SMI-list for X stores the quantity information for all items in X as well as extensions of X in a transaction. We illustrate these bounds in our example transaction database presented in Figure 1 for *ucov*.

Algorithm 1 Construct-SMI-List of a $\{k\}$-itemset from SMI-Lists of two $\{k-1\}$-itemsets with common prefix

Input: L_{Ix}: SMI-List of a $\{k-1\}$-itemset Ix
Input: L_{Iy}: SMI-List of a $\{k-1\}$-itemset Iy
Global: Tables $SumEU$ and $SumCU$
Output: L_{Ixy}: SMI-List of $Ixy = Ix \cup Iy$

$Ixy = [\,]$
for each triple Ex in L_{Ix} **do**
  if $\exists Ey \in Iy$ such that $Ex.Tid = Ey.Tid$ **then**
    $Exy = \langle Ex.TID, Ex.CWI \cup Ey.CWI, Ey.RWI \rangle$
    Append Exy to L_{Ixy}
    $sumEU(Ixy) \mathrel{+}= u(Ex.CWI \cup Ey.CWI, Ex.TID)$
    $sumCU(Ixy) \mathrel{+}= u(Ex.CWI \cup Ey.CWI \cup Ey.RWI, Ex.TID)$
  end if
end for
Return L_{Ixy}

Algorithm 2 SM-Miner with threshold θ

Input I**:** Prefix itemset I
Input $\mathcal{L}_{I+}$**:** Set of SMI-Lists of Ix for all items x
Output: the set of high-utility itemsets with I as prefix

for SMI-List L_{Ix} in $\mathcal{L}_{I+}$ **do** ▷ Ix denotes itemset I and item x
  if $sumEU(Ix) \geq \theta$ **then**
    Output Ix as a high-utility itemset
  end if
  if $sumCU(Ix) \geq \theta$ **then** ▷ Recursively explore extensions of Ix
    $L_{Ix+} = \{\,\}$
    for SMI-List L_{Iy} after L_{Ix} in $\mathcal{L}_{I+}$ **do**
      $L_{Ixy} = \text{Construct} - \text{SMI} - \text{List}(L_{Ix}, L_{Iy})$ ▷ Algorithm 1
      $\mathcal{L}_{Ix+} = \mathcal{L}_{Ix+} \cup L_{Ixy}$
    end for
    $SM - Miner(Ix, \mathcal{L}_{Ix+}, \theta))$
  end if
end for

SM-Miner has an initialization phase in which the database is scanned to compute the TSMWU of the items; TSMWU is computed since it is a tighter bound compared to TWU (Lemma 1). Another database scan is then performed to remove from the database the items with TSMWU less than θ (Lemma 1), and alongside, construct the SMI-lists of individual items that remain in the database. SM-Miner is then called on each of the items, along with the corresponding list, and its procedure is described in Algorithm 2. When called on an itemset I, it explores the search space in a depth-first manner to identify the complete set of high-utility itemsets with prefix I; alongside it constructs the promising extensions of I in the form of their SMI-Lists using Algorithm 1. Pruning of an itemset Ix happens if $SumCU(Ix)$ is less than the utility threshold θ since Ix and its supersets/extensions cannot be high-utility as per Lemma 2; we use $SumCU$ instead of EU_RU since it gives a tighter bound (Lemma 2).

5.4 Empirical observations

The objective of our experiments was to identify if there is any change in the performance trend in SMIM compared to HUIM for a complex utility function. In the absence of any existing algorithm for SMIM, we report the performance of our tree-based, projection-based and list-based algorithms for SMIM (listed

Table 2: Implementations of SMIM

SMIM algorithm	Adaptation of (HUIM algorithm)	Type
SM-Miner	(this work)	list-based
UPG+SM	UP-Growth+ [21]	tree-based
EFIMSM	EFIM [25]	projection-based
D2HUPSM	D2HUP [13]	projection-based

in Table 2) using *ucov*, on two sparse and two dense datasets, namely Chainstore, Kosarak, Mushroom, Accidents [8]. Due to the lack of real-life networks on the corresponding items, we synthetically constructed graphs (for computing coverage) by taking the items present in a database as vertices and linking them by edges randomly such that the average degree of a vertex in a graph is four. The detailed results are available in the full version of this paper.

The most prominent observation is that the performance trend of the SMIM algorithms with the *ucov* utility function is completely different from the HUIM scenario. Projection-based HUIM algorithms are known to perform an order of magnitude better than the other algorithms on dense and sparse datasets; for example, D2HUP performs the best on the sparse datasets and EFIM performs the best on the dense datasets. However, we observe that for SMIM, the tree-based and the list-based algorithms perform better compared to the projection-based algorithms on the sparse datasets. Further, SM-Miner competes with EFIMSM for the best performance on the dense datasets. Our investigations revealed that much of this behaviour is attributed to the number of utility function calls, and the tightness of the various bounds discussed earlier. In fact, the total execution times of the algorithms appear to be more correlated with the number of utility function calls than to the number of candidates generated, unlike HUIM.

Acknowledgements This work was supported in part by Infosys Centre for Artificial Intelligence, Indraprastha Institute of Information Technology, Delhi (IIIT-Delhi), and Visvesvaraya Ph.D. scheme for Electronics and IT.

References

1. Ahmed, C.F., Tanbeer, S.K., Jeong, B., Lee, Y.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Transactions on Knowledge and Data Engineering **21**(12) (2009)
2. Bansal, R., Dawar, S., Goyal, V.: An efficient algorithm for mining high-utility itemsets with discount notion. In: International Conference on Big Data Analytics (2015)
3. Cerf, L., Meira, W.: Complete discovery of high-quality patterns in large numerical tensors. In: 2014 IEEE 30th International Conference on Data Engineering (2014)
4. Coussat, A., Nadisic, N., Cerf, L.: Mining high-utility patterns in uncertain tensors. Procedia Computer Science **126** (2018)
5. Dawar, S., Goyal, V., Bera, D.: A hybrid framework for mining high-utility itemsets in a sparse transaction database. Applied Intelligence **47**(3) (2017)
6. Fournier-Viger, P., Wu, C.W., Zida, S., Tseng, V.S.: FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Foundations of Intelligent Systems (2014)

7. Fournier-Viger, P., Chun-Wei Lin, J., Truong-Chi, T., Nkambou, R.: A Survey of High Utility Itemset Mining (2019)
8. Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The SPMF open-source data mining library version 2. In: Machine Learning and Knowledge Discovery in Databases. Cham (2016)
9. Guns, T., Dries, A., Nijssen, S., Tack, G., Raedt, L.D.: Miningzinc: A declarative framework for constraint-based mining. Artificial Intelligence **244** (2017)
10. Jaysawal, B.P., Huang, J.W.: DMHUPS: Discovering multiple high utility patterns simultaneously. Knowledge and Information Systems **59**(2) (2019)
11. Krause, A., Golovin, D.: Submodular function maximization. In: Tractability: Practical Approaches to Hard Problems (2014)
12. Lin, J.C., Gan, W., Fournier-Viger, P., Hong, T., Tseng, V.S.: Mining high-utility itemsets with various discount strategies. In: 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA) (2015)
13. Liu, J., Wang, K., Fung, B.C.M.: Mining high utility patterns in one phase without generating candidates. IEEE Transactions on Knowledge and Data Engineering **28**(5) (2016)
14. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proceedings of the 21^{st} ACM International Conference on Information and Knowledge Management (2012)
15. Liu, Y., Liao, W.k., Choudhary, A.: A fast high utility itemsets mining algorithm. In: Proceedings of the 1st International Workshop on Utility-based Data Mining (2005)
16. Liu, Y., Liao, W.k., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Advances in Knowledge Discovery and Data Mining (2005)
17. Lou, T., Tang, J., Hopcroft, J., Fang, Z., Ding, X.: Learning to predict reciprocity and triadic closure in social networks. ACM Trans. Knowl. Discov. Data **7**(2) (Aug 2013)
18. Silva, A., Antunes, C.: Constrained pattern mining in the new era. Knowledge and Information Systems **47**(3) (2016)
19. Sohrabi, M.K.: An efficient projection-based method for high utility itemset mining using a novel pruning approach on the utility matrix. Knowledge and Information Systems (2020)
20. Tschiatschek, S., Singla, A., Krause, A.: Selecting sequences of items via submodular maximization. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (2017)
21. Tseng, V.S., Shie, B., Wu, C., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Transactions on Knowledge and Data Engineering **25**(8) (2013)
22. Tseng, V.S., Wu, C.W., Shie, B.E., Yu, P.S.: UP-growth: An efficient algorithm for high utility itemset mining. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2010)
23. Yao, H., Hamilton, H., Geng, L.: A unified framework for utility based measures for mining itemsets. 2nd International Workshop on Utility-Based Data Mining (2006)
24. Zhang, C., Han, M., Sun, R., Du, S., Shen, M.: A survey of key technologies for high utility patterns mining. IEEE Access **8**, 55798–55814 (2020). https://doi.org/10.1109/ACCESS.2020.2981962
25. Zida, S., Fournier-Viger, P., Lin, J.C.W., Wu, C.W., Tseng, V.S.: EFIM: a fast and memory efficient algorithm for high-utility itemset mining. Knowledge and Information Systems **51**(2) (2017)

A Analysis of a Twitter dataset

To demonstrate the effectiveness of *ucov* over *sumcov*, we used them to analyse a Twitter dataset curated from the Twitter-Dynamic-Net dataset on Aminer [17]. Following the procedure of constructing a follower-followee graph that is given above, we construct a directed graph with 14766 followees and 20423 followers with 46164 edges between them. The average degree of a followee is 3, and the maximum degree is 48. We ensured that every user in the constructed graph has at least one follower by removing nodes that do not meet this criteria. Even though we discussed coverage on an undirected graph earlier, it is straightforward to adapt it for directed graphs and we do the same. The transaction database had 2631 transactions with 14766 items, with average and maximum lengths as 87 and 381, respectively.

First, it can be shown that the patterns generated by *ucov* will always be a filtered subset of the patterns generated by *sumcov* for the same threshold; in fact, *sumcov* generated 10,29,921 patterns while *ucov* generated only 681 patterns when the minimum utility threshold was set to 10,000.

Theorem 1. *For a given threshold θ, the set of high-utility patterns generated by ucov$(\cdot)$ will always be a subset of the patterns generated by sumcov$(\cdot)$.*

Next, we generate the top 100 patterns using both *ucov* and *sumcov*. Table 3 compares them on various dimensions, and we see that *ucov* may be suitable for analysing our dataset.

Table 3: Comparison of the top-100 patterns using *ucov* and *sumcov*.

	Length				Frequency				Coverage				# tweets			
	min	max	avg.	median	min	max	avg.	med.	min	max	avg.	med.	min	max	avg.	med.
ucov	1	5	2.06	2	15	462	115	73	16	79	38.9	38	376	1605	808	742
sumcov	1	7	2.5	2	8	462	99.6	67.5	16	105	42.7	38	376	1605	832	778.5

B *ucov vs. sumcov*

Proof of Theorem 1. Consider any transaction T and let X be an itemset of T that is represented as $X = [(A_1 : q_1), \ldots (A_n : q_n)]$ where $q_1 \leq \cdots \leq q_n$. We can express $sumcov(X, T)$ as $\sum_{i=1}^{n} q_i \times Co(A_i)$. For the proof refer to Table 4 where we compare the expressions of $ucov(X, T)$ and $sumcov(X, T)$ term-by-term.

The table shows that the utility value computed by $sumcov(\cdot)$ is at least that by $ucov(\cdot)$. In other words, all patterns with $ucov(\cdot)$ no less than θ will also have $sumcov(\cdot)$ no less than θ and this proves the theorem; in fact, $sumcov(\cdot)$ can overestimate the coverage of a pattern if the sets of vertices covered by individual items present in a pattern have lots of common neighbors. □

For example, $ucov(ACD, T_1)$ in the database in Figure 1 is 37 and $sumcov(\{ACD\}, T_1) = 5 \times Co(\{A\}) + 10 \times Co(\{C\}) + 2 \times Co(\{D\}) = 5 \times 4 + 10 \times 3 + 2 \times 4 = 58$.

Table 4: Comparison of $ucov(X,T)$ with $sumcov(X,T)$ for proving Theorem 1

$\mathbf{ucov(X,T)} =$	$\mathbf{sumcov(X,T)} =$
$q_1 \times \text{Co}(\{A_1 \cdots A_n\})$	$\leq q_1 \times$ ($\text{Co}(\{A_1\}) + \text{Co}(\{A_2\}) + \ldots + \text{Co}(\{A_n\})$)
$(q_2 - q_1) \times \text{Co}(\{A_2 \cdots A_n\})$	$\leq (q_2 - q_1) \times$ ($\text{Co}(\{A_2\}) + \text{Co}(\{A_3\}) + \ldots + \text{Co}(\{A_n\})$)
$\vdots$	$\vdots \; \vdots$
$(q_n - q_{n-1}) \times \text{Co}(\{A_n\})$	$= (q_n - q_{n-1}) \times \text{Co}(\{A_n\})$
	$= q_1 \times Co(A_1) + q_2 \times Co(A_2) + \cdots q_n \times Co(A_n)$

C Proofs relating to *ucov*

In this subsection we first prove that $Co()$ is a subadditive and monotone function, and then we prove that $ucov(\cdot, T)$ is subadditive for every transaction T. The proof of monotonicity of *ucov* has been presented earlier in Section C.2.

C.1 *Co*() is subadditive and monotone

A set function $f()$ is submodular if $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for any two sets A, B. It is known that submodular functions are also subadditive. Krause et al. [11] proved that coverage of a set of nodes is submodular but they proved it for the traditional definition of coverage that only considers the neighbors and not the set itself. Therefore, we need to explicitly show that our $Co()$ function is submodular.

Theorem 2. *$Co()$ is monotone and submodular, hence, subadditive.*

Proof. The fact that $Co()$ is monotone is straightforward. Let X be a set of nodes (items), and let there be k nodes altogether in X or in the neighborhood of some node in X — call this set of nodes as N_X. Observe that $Co(X) = k$. Now let Y be a superset of X. All nodes in N_X are either already in Y or neighbor of some node in Y; thus, $Co(Y) \geq k$ which establishes the monotonicity of $Co()$.

For submodularity we will prove a stronger property, that Co exhibits diminishing returns. $f()$ has the property of diminishing returns if for any $A \subseteq B$ and any $z \notin B$, $f(A \cup \{z\}) - f(A) \geq f(B \cup \{z\}) - f(B)$. It is known that submodularity is equivalent to possession of the property of diminishing returns. Below we describe how the diminishing returns property is satisfied by $Co()$.

Consider any two sets of nodes (items) X, Y such that $X \subseteq B$, and consider any z not in these sets. Consider these three sets of nodes.

- N_X containing X and other nodes that are neighbors of some node in X
- N_Y containing Y and other nodes that are neighbors of some node in Y (note that N_Y contains N_X)
- N_z containing z and neighbors of z

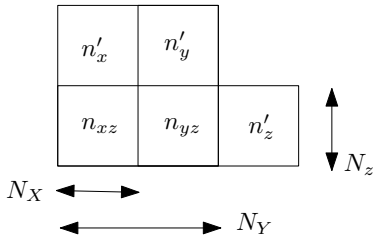


Fig. 2: Partitioning the neighborhood of X and Y for proving submodularity of Co

N_X can be partitioned into $N_X \setminus N_z$ and $N_X \cap N_z$. Let n'_x and n_{xz} denote the cardinalities of these, respectively. Similarly, divide N_Y into $N_Y \setminus N_z$ and $N_Y \cap N_z$ with cardinalities denoted n'_y and n_{yz}, respectively. Last, denote the cardinality of $N_z \setminus N_X \setminus N_Y$ by n'_z. A pictorial representation of these subsets and their cardinalities is given in Figure 2 from which we can readily observe these coverage values.

- $Co(X) = n'_x + n_{xz}$
- $Co(Y) = n'_x + n'_y + n_{xz} + n_{yz}$
- $Co(X \cup \{z\}) = n'_x + n_{xz} + n_{yz} + n'_z$
- $Co(Y \cup \{z\}) = n'_x + n'_y + n_{xz} + n_{yz} + n'_z$

For these values it is straightforward to verify that $Co(X \cup \{z\}) - Co(X) \geq Co(Y \cup \{z\}) - Co(Y)$. This concludes the proof of the theorem. □

Table 5: Comparing the summands in $ucov(Z,T), ucov(X,T), ucov(Y,T)$

$ucov(Z,T) =$		$ucov(X,T) =$	$ucov(Y,T) =$
1 $MCo([B_2, A_1, C_1, B_1, A_2])$	$\leq$		1 $MCo([B_2, C_1, B_1])$
2 $MCo([A_1, C_1, B_1, A_2])$	$\leq$	2 $MCo([A_1, C_1, A_2])$	
3 $MCo([C_1, B_1, A_2])$	$\leq$	3 $MCo([C_1, A_2])$	3 $MCo([C_1, B_1])$
3 $MCo([B_1, A_2])$	$\leq$		3 $MCo([B_1])$
7 $MCo([A_2])$	$\leq$	7 $MCo([A_2])$	

C.2 *ucov*() is monotonic

We use the following notation. Let X be any itemset which is represented as $[x_1, x_2, \ldots, x_k]$ when ordered in the non-decreasing order. We use X^i to denote the suffix of X starting at x_i, i.e., $X^i = [x_i, x_{i+1} \ldots x_n]$. Further, we drop braces inside $Co()$ to simplify readability, e.g., $Co(X^i)$ denotes $Co(x_i, x_{i+1}, \ldots x_k)$.

For monotonicity, without loss of generality, let $X = [x_1, \ldots x_n]$ be an itemset in T ordered in the specified manner. Further, let z be an item from T not in X, call $Z = X \cup \{z\}$; note that Z can be ordered as $[x_1, x_2, \ldots, x_j, z, x_{j+1}, \ldots x_n]$ according to the quantity of z (denoted q_z), where j is some value from $\{0, \ldots, n\}$.

Lemma 3. *$ucov(X) \leq ucov(Z)$ for X and Z defined above.*

Table 6: Comparing expressions for $ucov(X,T)$ and $ucov(Z,T)$ from Lemma 3.

	$\mathbf{ucov(X,T)}$	Comparison	$\mathbf{ucov(Z,T)}$	
(a)	$= q_1 Co(X^1)$	$\leq$	$= q_1 Co(Z^1)$	by Eqn. 1
	$+(q_2 - q_1)Co(X^2)$	$\leq$	$+(q_2 - q_1)Co(Z^2)$	by Eqn. 1
	$\vdots$	$\vdots$	$\vdots$	
(b)	$+(q_j - q_{j-1})Co(X^j)$	$\leq$	$+(q_j - q_{j-1})Co(Z^j)$	by Eqn. 1
(c)	$+(q_{j+1} - q_j)Co(X^{j+1})$	$=$	$+(q_z - q_j)Co(z \cup Z^{j+1})$ $+(q_{j+1} - q_z)Co(Z^{j+1})$	
(d)	$+(q_{j+2} - q_{j+1})Co(X^{j+2})$	$=$	$+(q_{j+2} - q_{j+1})Co(Z^{j+2})$	by Eqn. 2
	$\vdots$	$\vdots$	$\vdots$	
(e)	$+(q_n - q_{n-1})Co(X^n)$	$=$	$= (q_n - q_{n-1})Co(Z^n)$	by Eqn. 2

Proof. The proof is best understood from Table 6 where we expand $ucov(X,T)$ and $ucov(Z,T)$ using their suffixes and compare the terms in the expansions.

$$\text{For } i = 1 \text{ to } j,\ Co(X^i) \leq Co(Z^i) \text{ since, } X^i \subset Z^i \tag{1}$$

$$\text{For } i = j+1 \text{ to } n,\ Co(X^i) = Co(Z^i) \text{ since, } X^i = Z^i \tag{2}$$

It can be seen in the table that all the terms from (a)–(b) and (d)–(e) of $ucov(X,T)$ are less than or equal to those in $ucov(Z,T)$. We only need to establish the equality of (c) that we prove below.

$$\begin{aligned}
&(q_{j+1} - q_j)Co(X^{j+1}) \\
=&(q_z - q_j)Co(z \cup Z^{j+1}) + (q_{j+1} - q_z)Co(Z^{j+1}) \\
=&(q_z - q_j)Co(z \cup X^{j+1}) + (q_{j+1} - q_z)Co(X^{j+1}) \\
&\text{(which is true due to monotonicity of } Co()) \\
=&(q_z - q_j)Co(X^{j+1}) + (q_{j+1} - q_z)Co(X^{j+1}) \\
\leq&(q_z - q_j)Co(z \cup X^{j+1}) + (q_{j+1} - q_z)Co(X^{j+1})
\end{aligned}$$

□

The above lemma allows us to prove that *ucov* is a monotone function.

Theorem 3. *The utility function $ucov(\cdot, T)$ is monotone for any fixed T.*

Proof. Take any two subsets $X \subset Y \subseteq T$. It is always possible to construct several more subsets $Y_0, Y_1, Y_2, \ldots, Y_k$ such that $X = Y_0 \subset Y_1 \subset Y_2 \ldots Y_{k-1} \subset Y_k = Y$ and each Y_i contains exactly one new element in addition to those from Y_{i-1}. Then using Lemma 3 repeatedly, we get that $ucov(X_{i-1}) \leq ucov(X_i)$ for $i = 1 \ldots n$. Combining all of them proves that $ucov(X) \leq ucov(Y)$ which proves that *ucov* is a monotone set function when T is fixed. □

C.3 *ucov*() is subadditive

Next we prove subadditivity of *ucov*. Let $X = \{x_1, x_2, \ldots, x_k\}$ be any set of items in which x_1 denotes the items with the smallest quantity (ties broken arbitrarily). Define marginal coverage of X as

$$MCo(X) = Co(\{x_1, x_2, \ldots, x_k\}) - Co(\{x_2, \ldots, x_k\})$$

which is the change in coverage due to the item in X with the smallest quantity on the rest of X. For the sake of completeness, define the coverage of an empty set to be 0. It is easy to show that $Co(X) \leq Co(Y)$ if $X \subseteq Y$. Thus it follows that $MCo()$ is a non-negative function.

Lemma 4. *Let Y be a subset of items from a transaction. Let the items in Y when ordered in non-decreasing quantities be $[x_1, x_2, \ldots]$. Further, let X be a subsequence of Y starting with x_1; so, $X = [x_1, \ldots] \subseteq Y$. Then $MCo(X) \geq MCo(Y)$.*

Proof. Clearly, x is the item in X with the smallest quantity, and also the item in Y with the smallest quantity. Let $X' = X \setminus \{x\}$ and $Y' = Y \setminus \{x\}$. Submodular functions are known to have lower diminishing returns; so, $Co(X' \cup \{x\}) - Co(X') \geq Co(Y' \cup \{x\}) - Co(Y')$. However, the former was defined as $MCo(X)$ and the latter as $MCo(Y)$, thus proving the lemma. □

We now re-define *ucov* using MCo by simply re-arranging and collating terms. To simplify expressions, $MCo([x_1, x_2, \ldots x_k])$ will mean $MCo(\{x_1, x_2, \ldots, x_k\})$ and the items in the sequence appearing in non-decreasing quantities; thus $MCo([x_1, \ldots])$ will mean the coverage of the first item x_1 in the sequence. Suppose X denotes $[x_1, x_2, \ldots, x_k]$ where the items are ordered in a non-decreasing manner. Then *ucov* can be re-written as

$$ucov(X, T) = \sum_{j=1}^{n} q_{i_j} \times MCo([x_j, x_{j+1}, \ldots, x_k]). \tag{3}$$

Lemma 5. *Let X, Y denote any two itemsets of a transaction T and let Z denotes $X \cup Y$. Then, $ucov(Z, T) \leq ucov(X, T) + ucov(Y, T)$.*

Proof. Suppose the items of Z when ordered in non-decreasing quantities are denoted $z_1, z_2, \ldots, z_n$ where any z_i belongs to exactly one of X or Y; further, suppose the quantity of z_i be denoted q_i. Note that when ordered, the sequences of the items in X and the items in Y are (possibly overlapping) subsequences of Z. Using the alternative definition of *ucov* given in 3, we get the following.

$$ucov(Z, T) = \sum_{i=1}^{n} q_i \times MCo(Z^i)$$

$$ucov(X, T) = \sum_{\substack{i=1: \\ z_i \in X}}^{n} q_i \times MCo(X^i)$$

$$ucov(Y, T) = \sum_{\substack{i=1: \\ z_i \in Y}}^{n} q_i \times MCo(Y^i)$$

We will analyse the terms in the expression of $ucov(Z,T)$. Take any $i \in \{1,\ldots,n\}$; the i-th summand is $q_i \times MCo(Z^i)$. Consider the case where $z_i \in X$. X^i is the subsequence of X starting with z_i; in fact, $X^i \subseteq Z^i$ and both the sequences start with z_i. Therefore, by Lemma 4 $MCo(X^i) \geq MCo(Z^i)$. For the other case of $z_i \in Y \setminus X$ we similarly obtain that $MCo(Y^i) \geq MCo(Z^i)$; note that any $z_i \in Z$ either belongs to X or belongs to $Y \setminus X$.

Combining both of these, and summing over all $i = 1 \ldots n$ (and using the fact that $MCo()$ is a positive function due to which the additional summands in $ucov(Y,T)$ only make the RHS larger), we obtain the claim in the theorem.

$$\sum_{i=1}^{n} q_i \times MCo(Z^i) \leq \sum_{\substack{i=1: \\ z_i \in X}}^{n} q_i \times MCo(X^i) + \sum_{\substack{i=1: \\ z_i \in Y}}^{n} q_i \times MCo(Y^i)$$

This proves the theorem. For an intuitive explanation, consider a transaction $T = \{(A_1 : 2), (A_2 : 7), (A_3 : 4), (C_1 : 3), (B_1 : 3), (B_2 : 1)(B_3 : 2)\}$, itemset $X = \{A_1, A_2, C_1\}$ and itemset $Y = \{B_1, B_2, C_1\}$. Then, written in order, $Z = [B_2, A_1, C_1, B_1, A_2]$, $X = [A_1, C_1, A_2]$ and $Y = [B_2, C_1, B_1]$. The above inequalities can be applied to these subsets which can be seen in Table 5. It can be seen that $ucov(Z,T) \leq ucov(X,T) + ucov(Y,T)$.

□

D Proof of claims in Section 5

Lemma 1. *If $TSMWU(X)$ is less than the threshold, then X and its supersets cannot be high-utility itemsets. Further, $TSMWU(X)$ is a tighter upper-bound compared to $TWU(X)$.*

Proof. We first prove that if $TSMWU(X)$ is less than the threshold, then X and its supersets cannot be high-utility itemsets. For any transaction T and any $X \subseteq X' \subseteq T$, $u(X',T) \leq u(T,T)$ (due to monotonicity) which equal $TSMU(T)$. Therefore, $u(X') \leq TSMWU(X)$ and thus, $TSMWU(X) < \theta$ implies that $u(X') < \theta$ as well.

Now we prove that $TSMWU(X)$ is a tighter upper-bound compared to $TWU(X)$. Observe that $TU(T) = \sum_{x \in T} u(x,T) \geq u(T,T)$ due to subadditivity. Further the right hand side equals $TSMU(T)$. Therefore, $TSMWU(X) = \sum_{T:X \subseteq T} TSMU(T) \leq \sum_{T:X \subseteq T} TU(T) = TWU(X)$. □

Lemma 2. *If $CU(X)$ is less than a threshold, any extension X' of X in the processing order of items is not a high-utility itemset. Further, $CU(X)$ is a tighter upper-bound compared to $EU\_RU(X)$.*

Proof. We first prove that if $CU(X)$ is less than a threshold, any extension X' of X in the processing order of items is not a high-utility itemset. Let X' be some extension of X in the processing order of items; since transactions are also (implicitly) stored in the same order, $X' \subseteq X \cup T/X$. The proof of the lemma follows easily since the set of transactions containing X' is always a subset of

the set of transactions containing X and $u(\cdot, T)$ is a monotone function which implies that $u(X', T) \leq u(X \cup T/X, T) = CU(X, T)$.

Then we prove that $CU(X)$ is a tighter upper-bound compared to $EU\_RU(X)$. From the subadditivity property we get that $EU(X, T) = \sum_{x \in X} u(x, T) \geq u(X, T)$. Further $RU(X, T) \geq u(T/X, T)$. We immediately get $EU(X, T) + RU(X, T) \geq u(X, T) + u(T/X, T) \geq u(X \cup T/X, T) = CU(X, T)$ (second inequality again uses subadditivity). Therefore it follows that $EU\_RU(X) \geq CU(X)$. □

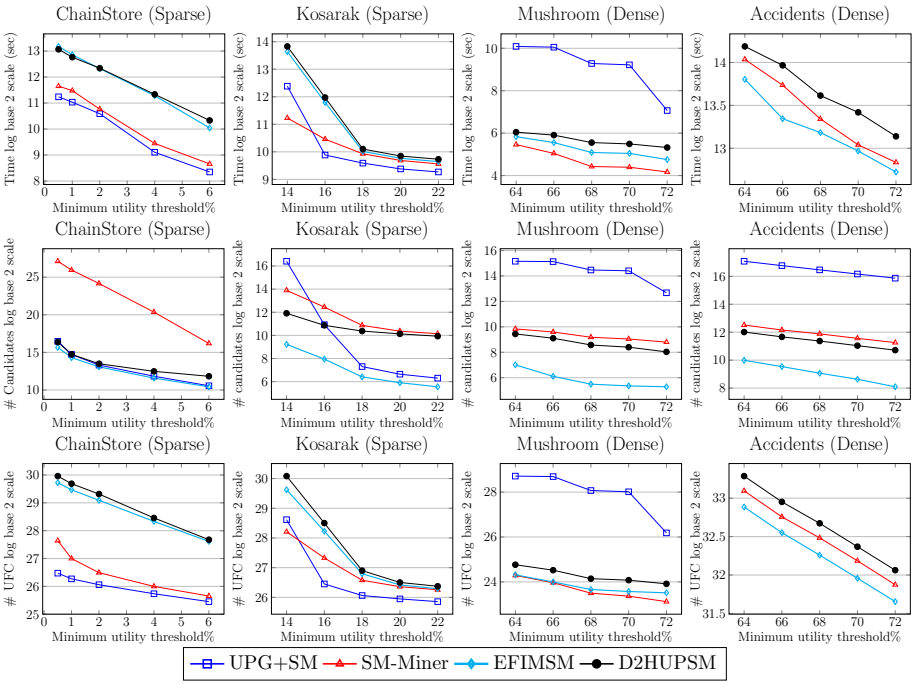


Fig. 3: Performance evaluation of SM-Miner (our), and SMIM implementations of EFIM, D2HUP, UP-Growth+ (using *ucov*). We report the average values of five executions, wherever appropriate. The utility values are expressed in terms of the percentage of the utility threshold at which at least one candidate itemset is reported.

Table 7: Transaction datasets used in our experiments [8]. The networks among the items were generated in a synthetic manner.

Dataset	**#Tx**	**Avg. length**	**#Items**	**Type**
Chainstore	11,12,949	7.2	46,086	Sparse
Kosarak	9,90,002	8.1	41,270	Sparse
Mushroom	8,124	23	119	Dense
Accidents	3,40,183	33.8	468	Dense