

An Effective Correlated High Utility Itemset Mining Algorithm

Priscilla Owiredo Okai¹, Vincent Mwintieru Nofong², Selasi Kwashie³, and Michael Bewong³

¹ Deloitte Ghana, Accra, Ghana
owiredupriscillaokai@gmail.com

² University of Mines and Technology, Tarkwa, Ghana
vnofong@umat.edu.gh

³ Charles Sturt University {skwashie,mbewong}@csu.edu.au

Abstract. High-Utility Itemset Mining (HUIM) is a non-trivial task for analyzing databases such as, customer transactions, to reveal all itemsets that are of high importance. Over the past years, a number of traditional algorithms have been developed to mine High Utility Itemsets (HUIs). However, as a significant drawback, these algorithms often report a large number of HUIs, the majority of which lack any inherent correlations between the items in the HUIs. Such reported HUIs without inherent item correlation often are unreliable in decision making as they are usually of high utility by random chance of occurrence. To address this issue, this paper proposes the Correlated High Utility Itemset Miner (CHUIM). CHUIM employs the concept of productivity (in both normal and exclusive domains) to prune HUIs that occur by random chance without inherent item relationships. Experimental analysis on benchmark databases show that, CHUIM is efficient and can prune the set of HUIs without inherent item correlations.

Keywords: High-utility itemsets · Correlation · Productivity measure · Correlated high-utility itemsets

1 Introduction

Frequent Itemset Mining (FIM), introduced by [1] for market basket analysis, seeks to identify items that frequently co-occur in a database based on a user-defined “frequency” threshold. This process is valuable for understanding events in a database, such as discovering that many customers purchase *bread* and *jam* together (i.e., $\{bread, jam\}$), providing insights for marketing decisions, like shelf placement. Over the years, various techniques (such as [10,14]) have been proposed to enhance the discovery of categories of frequent itemsets.

Frequent itemset mining identifies co-occurring patterns based on “frequency,” but its universal applicability is limited due to its exclusive reliance on this criterion. In market basket analysis, decision-makers prioritize high-profit products over all identified patterns through FIM, recognizing that not all frequently co-occurring patterns ensure profitability.

To address frequent itemset mining’s limitations, High Utility Itemset Mining (HUIM) emerged as a refined approach, emphasizing the identification of high-importance co-occurring patterns, like customer transactions with substantial profits. Techniques and approaches proposed in recent works, such as [4,5,6,12,13], facilitate the discovery of various categories of High Utility Itemsets (HUIs).

Discovering sets of correlated HUIs in databases is challenging due to issues with user-defined thresholds in past approaches ([4,5,6,13]). Achieving an optimal balance proves difficult, as low thresholds result in numerous uncorrelated HUIs, while high thresholds may eliminate some relevant correlated HUIs.

To this end, this paper presents CHUIM (Correlated High Utility Itemset Miner) as a solution to challenges in existing techniques for Correlated HUIM. CHUIM addresses the issues mentioned above by using productivity, a positive correlation test. CHUIM determines productivity based on inherent item relationships within high utility itemsets, eliminating the need for users to set productivity thresholds.

This paper presents the following contributions:

- Introduction of an effective Correlated High Utility Itemset Miner (CHUIM) for extracting correlated high utility itemsets from databases.
- Utilization of positive correlation, grounded in inherent itemset relationships, to ensure the exclusive reporting of correlated high utility itemsets.
- Conducting extensive experiments on benchmark datasets, demonstrating the efficiency and effectiveness of CHUIM.

2 Related Works

The concept of mining high utility itemsets (HUIs) originated in [3] as a solution to challenges in frequent pattern mining algorithms, later, [17] mathematically formalized this concept. Various techniques for discovering HUIs have since been proposed, categorized into traditional approaches (e.g., [2,11]) and application-specific extensions.

While traditional techniques are effective, their application limitations led to proposed extensions addressing diverse decision-making scenarios, including concise representations ([7,9]), top-k HUIs ([15]), and target HUIs ([12,13]).

Despite numerous proposed techniques for mining high utility itemsets, a persistent challenge is the failure to exclusively report correlated high utility itemsets. For instance, top-k or concise representations may lack inherent item relationships, making them unreliable for decision-making, as the reported HUIs may result from random chance rather than meaningful item connections.

The emergence of correlated HUIM aimed to ensure reliable HUIs for decision-making are reported. Despite efforts with bond and/or all-confidence measures in works like [4,5], subjectivity arises, with extremely low values reporting weak correlations and high values eliminating useful ones. Setting suitable thresholds in existing techniques is challenging, rendering them difficult to use. Even with moderate thresholds, some relevant correlated itemsets may be pruned by techniques using bond and all-confidence measures (see Examples 10 and 11).

This paper introduces CHUIM (Correlated High Utility Itemset Miner), mitigating limitations in existing correlated HUIM techniques by utilizing productivity to prune uncorrelated HUIs. Unlike other methods, CHUIM's productivity measure relies on inherent item relationships, eliminating challenges associated with user-dependent bond and all-confidence thresholds.

3 Preliminaries

3.1 Frequent Itemset Mining

The notations associated with frequent itemset mining are as follows.

Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of literals, called items. A set $X_1 = \{i_a, \dots, i_n\} \subseteq I \mid a \leq n \wedge a, n \in [1, m]$, is called a pattern (an itemset). A transaction database is a set of transactions $D = \{T_1, T_2, T_3, \dots, T_k\}$ such that $\forall T_a \in D, T_a \subseteq I$ and T_a has a unique identifier ' a ' called its transaction ID (TID).

Example 1. Let Table 1 be a store's transaction database, with $I = \{b, c, e, p, t, ts\}$ as the set of items (where; $b = book$, $c = calculator$, $e = extension board$, $p = pen$, $t = television$, $ts = television stand$). Transaction T_2 in Table 1 which has a transaction ID of 2 and two items (that is, $\{c\}$ and $\{e\}$) is a length-2 itemset.

Table 1: Sample Customer Transactions

TID	T_1	T_2	T_3	T_4	T_5	T_6
Transaction	$\{p, b, c\}$	$\{c, e\}$	$\{p, t, e\}$	$\{p, b, c, ts\}$	$\{t, e, ts\}$	$\{p, b, c, t\}$

The *cover set* ($cov(S)$) of an itemset, S , in a database, D is defined as $cov(S) = \{TID \mid \forall T_{TID} \in D \wedge S \subseteq T_{TID}\}$.

Example 2. In Table 1, given, $S = \{p, b\}$, then $cov(S) = \{1, 4, 6\}$ as the itemset $\{p, b\}$ was bought by customers in transactions T_1, T_4 and T_6 .

The *support count* (*count*) of S in D (denoted as $|cov(S)|$) is defined as the number of times S appears in the database. The *support* (also known as *absolute support*) of S (denoted as $sup(S)$) is defined as follows: $sup(S) = \frac{|cov(S)|}{|D|}$ where $|D|$ is the size of the database in which S occurs.

Example 3. In Table 1, given $S = \{p, b\}$, then *count* of S become 3 as $\{p, b\}$ occur in three transactions. The support of S , ($sup(S)$) is evaluated as $sup(S) = \frac{3}{6} = 0.5$, since $|cov(S)| = |\{1, 4, 6\}| = 3$ and the database size, that is, $|D| = 6$.

A frequent itemset is defined by [1] as follows:

Definition 1. (Frequent Itemset [1]) Given a user defined minimum support threshold of frequency, *minsup* (such that, $0 < minsup \leq 1$) and a database D , an itemset S , is a frequent itemset if the support of S is greater than or equal to the minimum support threshold of frequency, that is, $sup(S) \geq minsup$.

The problem of Frequent Itemset Mining is defined as follows:

Definition 2. (Frequent itemset mining [1]). *Frequent Itemset Mining involves discovering all frequent itemsets in a transaction database D , given a set of items I and a user-defined minimum support threshold (minsup).*

Example 4. Given a minimum support count of 3 ($\text{minsup} = 0.5$), and Table 1, the frequent itemsets and their supports will be: $\{b\} : 0.50$, $\{c\} : 0.67$, $\{e\} : 0.50$, $\{p\} : 0.67$, $\{t\} : 0.50$, $\{b, c\} : 0.50$, $\{b, p\} : 0.50$, $\{c, p\} : 0.50$, and $\{b, c, p\} : 0.50$.

3.2 High Utility Itemset Mining

The notations for high utility itemset mining in databases are the same as those of frequent pattern mining with the following additions.

Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of literals representing items. A *quantitative transaction database*, D , consists of transactions $\{T_1, T_2, \dots, T_k\}$, where each transaction T_c is a subset of items from I , denoted as $T_c \subseteq I$, with a unique identifier c called its Transaction Identifier (TID). Each item $i \in I$ has an *external utility*, $eu(i)$, denoting its unit profit, and an *internal utility*, $n(i, T_c)$, representing its purchase quantity in transaction T_c .

Example 5. In Table 2, representing customer transactions of a store, the set of items is $I = \{b, c, e, p, t, ts\}$. Table 3 shows the external utility (profit) of each item in Table 2. For instance, transaction $T_1 = b\ p:6:4\ 2$ indicates the purchase of 4 units of b and 2 units of p , with 6 as the total number of items bought. The external utility values in Table 3 denote the profit gained for selling one unit of each item, such as 2 for b and 10 for t .

Table 2: Sample Transactions

TID	Transaction	TID	Transaction
T_1	$b\ p:6:4\ 2$	T_7	$b\ c\ p:7:1\ 2\ 4$
T_2	$c\ e:2:1\ 1$	T_8	$p\ t:3:2\ 1$
T_3	$e\ t\ ts:3:1\ 1\ 1$	T_9	$p\ t\ ts:5:3\ 1\ 1$
T_4	$b\ p:4:1\ 3$	T_{10}	$b\ c\ p:7:1\ 2\ 4$
T_5	$t:1:1$	T_{11}	$e\ ts:2:1\ 1$
T_6	$e\ p\ t:3:1\ 1\ 1$		

Table 3: External Utilities

Item	External Utility
b	2
c	2
e	4
p	2
t	10
ts	10

A utility function computes the relevance of each itemset in a database, specifically measuring the profit generated by items purchased together. The utility of an item is formally defined as:

Definition 3. (Utility of an item) *The utility of an item i in a transaction T_c (denoted as $u(i, T_c)$), is the product of its external and internal utilities.*

$$u(i, T_c) = eu(i) \times n(i, T_c) \quad (1)$$

Example 6. For the item $\{p\}$ purchased in transaction T_4 , its utility is calculated using Equation 1 as $u(p, T_4) = 2 \times 3 = 6$, referencing Tables 2 and 3.

Definition 4. (Utility of an itemset) *The utility of an itemset S in a transaction T_c , denoted as $u(S, T_c)$, is the sum of the products of the external and internal utilities of the items in S for transaction T_c , formally expressed as:*

$$u(S, T_c) = \sum_{i \in S} u(i, T_c) \quad (2)$$

Example 7. Given $S = \{b, p\}$ purchased in T_4 , its utility will be evaluated with reference to Tables 2 and 3 using Equation 2 as:

$$u(b, p, T_4) = u(b, T_4) + u(p, T_4) = (1 \times 2) + (3 \times 2) = 8$$

Definition 5. (Total Utility of an itemset) *The total utility of an itemset S in a database, denoted as $u(S, D)$, is the sum of the products of external and internal utilities of items in S for all transactions where S appears, formally expressed as:*

$$u(S, D) = \sum_{T_c \in p(S)} u(S, T_c) \quad (3)$$

where $p(S)$ is the set of transactions in the database that contain the itemset S .

For brevity, we use $u(S)$ as the total utility for the rest of the paper.

Example 8. Given the itemset $S = \{b, p\}$ in Table 2, its total utility will be computed using Equation 3 as:

$$\begin{aligned} u(\{b, p\}) &= \sum_{T_c \in p(\{b, p\})} u(\{b, p\}, T_c) \\ &= u(\{b, p\}, T_1) + u(\{b, p\}, T_4) + u(\{b, p\}, T_7) + u(\{b, p\}, T_{10}) \\ &= 12 + 8 + 10 + 10, \text{ hence, } u(b, p) = 40 \end{aligned}$$

Formally, a high utility itemset is defined as:

Definition 6. (High utility itemset) [5] *An itemset S is a high-utility itemset if its utility $u(S)$ is no less than a user-specified minimum utility threshold (denoted as minutil) set by the user (i.e., $u(S) \geq \text{minutil}$). Otherwise, S is a low-utility itemset.*

Example 9. Applying $\text{minutil} = 10$ to Table 2 and Table 3 would return 18 HUIs as: $\{b\} : 14$, $\{c\} : 10$, $\{e\} : 16$, $\{p\} : 38$, $\{t\} : 50$, $\{ts\} : 30$, $\{b, c\} : 12$, $\{b, p\} : 40$, $\{c, p\} : 24$, $\{e, t\} : 28$, $\{e, ts\} : 28$, $\{p, t\} : 42$, $\{p, ts\} : 16$, $\{t, ts\} : 40$, $\{b, c, p\} : 28$, $\{e, p, t\} : 16$, $\{e, t, ts\} : 24$, and $\{p, t, ts\} : 26$.

Recent works [4,5] aimed to reduce the number of reported high-utility itemsets, utilizing criteria like frequency affinity, bond measure, and all-confidence (refer to [4,5] for more details of these measures).

Example 10. Applying the *bond* measure (from [4]) with a *minimum bond* of 0.0 reports all HUIs in Example 9 as correlated. With a *minimum bond* of 0.4, only specific HUIs $\{b\}$, $\{e\}$, $\{p\}$, $\{t\}$ and $\{b, p\}$ are reported as correlated, while further increasing the *minimum bond* to 0.6 narrows down the correlated HUIs to be $\{p\}$ and $\{t\}$, with the rest pruned as uncorrelated.

Example 11. Applying the *all-confidence* measure (from [5]) with a *minimum all-confidence* threshold of 0.0 returns all HUIs in Example 9 as correlated. Setting the threshold to 0.4 returns 12 HUIs, and further raising it to 0.6 returns the correlated HUIs to $\{b\}$, $\{c\}$, $\{e\}$, $\{p\}$, $\{t\}$ and $\{ts\}$, with the rest pruned as uncorrelated.

In Examples 10 and 11, *bond* and *all-confidence* thresholds yield different outcomes for the same values. For instance, with both thresholds set to 0.6, the *bond* approach prunes $\{b\}$, $\{c\}$ and $\{e\}$ as uncorrelated, while the *all-confidence* approach reports them as correlated. However, of the 18 HUIs in Example 9, only 10 ($\{c\}$, $\{e\}$, $\{p\}$, $\{t\}$, $\{b, p\}$, $\{e, t\}$, $\{e, ts\}$, $\{t, ts\}$, $\{b, c, p\}$ and $\{e, t, ts\}$) have inherent item relations (see Section 4 for details), and despite being strongly correlated, the chosen user-defined thresholds may prune these while reporting HUIs without inherent item relations.

This paper introduces the Correlated High Utility Itemset Miner (CHUIM) to ensure the reporting of HUIs with inherent item relationships for decision-making. CHUIM employs a productivity test for correlation, eliminating the need for user-defined thresholds, unlike existing techniques using *bond* and *all-confidence* measures.

4 Mining Correlated HUIs

To mine correlated high utility itemsets, we adapt the definition of high utility itemsets from [5]. While Definition 6 identifies high utility itemsets, it may include itemsets that lack correlation, especially when expensive items that generate high profits when sold together. This can lead to erroneous business decisions, such as co-promotion or proximity placement on display shelves.

Specifically, we define a Correlated High-Utility Itemset as follows:

Definition 7. (Correlated high utility itemset). *Given a utility transaction database, D , of items and their external utilities, a minimum utility, minutil , an itemset S is a correlated high-utility itemset if it is a high utility itemset and is productive in both the normal and exclusive domain.*

where, the *productivity* of S in the normal domain is defined as:

Definition 8. (Productive itemset in normal domain) [16]. *An itemset S is productive in the normal domain if and only if, for all S_1, S_2 such that, $S_1 \subset S$; $S_2 \subset S$; $S_1 \cup S_2 = S$, and $S_1 \cap S_2 = \emptyset$, then, $\text{sup}(S) > \text{sup}(S_1) \times \text{sup}(S_2)$.*

While the productivity of S in the exclusive domain is defined as:

Definition 9. (Productive itemset in exclusive domain) [16]. An itemset S is productive in the exclusive domain if and only if, it is productive in its exclusive domain, $edom(S)$, where the exclusive domain is defined as:

$$edom(S) = \{1, \dots, |D|\} \setminus \bigcup_{R \supset S} cov(R \setminus S) \mid R \rightarrow \text{productive}$$

The condition specified in Definition 8 (i.e., for all S_1, S_2 such that, $S_1 \subset S$; $S_2 \subset S$; $S_1 \cup S_2 = S$, and $S_1 \cap S_2 = \emptyset$, then, $sup(S) > sup(S_1) \times sup(S_2)$) serves as a positive correlation test. It implies that the itemset's frequency must exceed what is expected under any assumption of independence between any partition of the pattern S into two independent itemsets. This means that for a given itemset, it must be productive in conjunction with every rule that can be formed from it. By using this productivity measure, the set of HUIs that appear in the dataset by random chance can be eliminated.

Example 12. Take the HUIs in Example 9, with the productiveness test on the itemset $S = \{p, t\}$ (that is, $\{pen, tv\}$), it means the support of $\{p, t\}$ together must be greater than the product of their independent supports if there exists a positive correlation between the items p and t . However, from Table 2, the $sup(p, t) = 0.2727$ while $sup(p) = 0.6364$ and $sup(t) = 0.5455$. Hence, we get $sup(p) \times sup(t) = 0.3471$. Since the support of their independent occurrences is greater than the support of their joint occurrence, it implies there is no positive correlation between p and t . As such, though the itemset $\{p, t\}$ is a high utility transaction, it cannot be relied on in decision making such as co-promoting $p(pens)$ with $t(tvs)$ or placing $p(pens)$ on same shelf $t(tvs)$ are placed.

The productivity measure in the *normal* domain aids in eliminating High Utility Itemsets lacking positive correlation, but it falls short in identifying truly correlated itemsets. To address this, correlation testing is necessary in the *exclusive* domain (defined in Definition 9). In the *exclusive* domain, obtained by removing transactions with supersets of S , the itemset S is tested for productivity in remaining transactions without its proper supersets. A truly correlated itemset is one productive in both the *normal* and *exclusive* domains, ensuring correlation is not due to correlated proper supersets.

Example 13. Consider the itemset $\{b, c\}$ (that is, $\{book, calculator\}$) in Example 9 to determine if its correlation is not caused by any of its proper supersets, all transactions containing its proper supersets that are productive (which in this case are T_7 and T_{10} since they contain $\{book, calculator\}$) will be taken out of the database (that is Table 2) resulting the *exclusive* domain of $\{b, c\}$ as shown in Table 4. The productiveness of $\{b, c\}$ in its *exclusive* domain (Table 4) is then evaluated. From Table 4, $sup(b, c)$ in the *exclusive* domain is 0.0 while the $sup(b)$ and $sup(c)$ are 0.2222 and 0.1111 respectively, resulting in $sup(b) \times sup(c) = 0.0247$. Since expected support of $\{b, c\}$, that is 0.0247, is greater than actual support of $\{b, c\}$, that is 0.0, it implies people are rather buying $\{b, c, p\}$ (that is, $\{book, calculator, pen\}$) together and not $\{b, c\}$ (that is,

$\{book, calculator\}$ alone. As such, $\{book, calculator\}$ alone cannot be used in co-promotion or shelf-placement since their correlation is rather caused by their proper superset $\{book, calculator, pen\}$.

Table 4: Exclusive Domain of $\{b, c\}$ from Table 2

TID	Transaction	TID	Transaction	TID	Transaction
T_1	$b \ p:6:4 \ 2$	T_4	$b \ p:4:1 \ 3$	T_8	$p \ t:3:2 \ 1$
T_2	$c \ e:2:1 \ 1$	T_5	$t:1:1$	T_9	$p \ t \ ts:5:3 \ 1 \ 1$
T_3	$e \ t \ ts:3:1 \ 1 \ 1$	T_6	$e \ p \ t:3:1 \ 1 \ 1$	T_{11}	$e \ ts:2:1 \ 1$

Using the productivity measure for mining correlated HUIs has several advantages. It guarantees that reported HUIs result from inherent item relationships, minimizing occurrences due to random chance. Additionally, it aids in reducing the number of recorded HUIs and eliminates the challenges associated with users setting correlation thresholds. Ultimately, the reported correlated HUIs become more dependable for making informed business and marketing decisions.

4.1 The Correlated High Utility Itemset Miner (CHUIM)

Algorithm 1 depicts the proposed Correlated High Utility Itemset Miner (CHUIM) implemented for discovering the set of correlated HUIs based on Definition 7.

Algorithm 1: CHUIM($D, eD, minUtil$)

Input: $D, eD, minUtil$
Output: Set of correlated HUIs: $corHUI$

- 1 Create: $iCov, eUtil, iUtil, iList, HUI, corHUI$
- 2 Data Preprocessing ($D, eD, minUtil, iCov, eUtil, iUtil, iList, HUI$)
- 3 **while** $|iList| > 1$ **do**
- 4 Create $tempList$
- 5 **for each** $item, S_1$ **in** $iList$ **do**
- 6 **for each** $item, S_2$ **in** $iList$ **do**
- 7 **if** S_1 and S_2 satisfy candidate generation property **then**
- 8 Get $cov(S_1)$ and $cov(S_2)$ from $iCov$
- 9 Let $S = S_1 \cup S_2$ and $cov(S) = cov(S_1) \cap cov(S_2)$
- 10 **if** $|cov(S)| > 0$ and S is productive **then**
- 11 Update $iCov[S] = [cov(S)]$ and $tempList \leftarrow \{S\}$
- 12 Compute total utility of S , $u(S)$
- 13 **if** $u(S) \geq minUtil$ **then**
- 14 $HUI \leftarrow \{S : u(S)\}$
- 15 Clear content in $iList$ and copy content in $tempList$ into $iList$
- 16 **return** HUI
- 17 PruneHUIs($HUI, corHUI$)
- 18 **return** $corHUI$

CHUIM which takes as inputs: (a) a quantitative transaction database (D), (b) an external utility database (eD), and (c) a minimum utility threshold ($minUtil$); returns the set of correlated high utility itemsets based on Definition 7. The set of correlated HUIs are mined with CHUIM as follows.

For the given inputs, Line 1 (of Algorithm 1) creates the following data structures: (a) $iCov$, which stores unique itemsets and their coversets from D , (b) $eUtil$, which stores unique length-1 items and their external utilities from eD , (c) $iUtil$, which stores unique length-1 items and their internal utilities from D , (d) $iList$, which contains candidate high utility itemsets, (e) HUI , which stores productive HUIs, and (f) $corHUI$, which stores correlated HUIs.

Next, in Line 2, the data is preprocessed into the created data structures. For the running example, executing Line 2 on Tables 2 and 3 with $minUtil = 10$, $iCov$, $eUtil$, $iUtil$, $iList$ and HUI will be returned with the following content:

- (a) $iCov \rightarrow \{ b : [1, 4, 7, 10]; c : [2, 7, 10]; e : [2, 3, 6, 11]; p : [1, 4, 6, 7, 8, 9, 10]; t : [3, 5, 6, 8, 9]; ts : [3, 9, 11] \}$
- (b) $eUtil \rightarrow \{ b : 2; c : 2; e : 4; p : 2; t : 10; ts : 10 \}$
- (c) $iUtil \rightarrow \{ b : [4, 1, 1, 1]; c : [1, 2, 2]; e : [1, 1, 1, 1]; p : [2, 3, 1, 4, 2, 3, 4]; t : [1, 1, 1, 1, 1]; ts : [1, 1, 1] \}$
- (d) $iList \rightarrow \{ b, c, e, p, t, ts \}$
- (e) $HUI \rightarrow \{ b : 14; c : 10; e : 16; p : 38; t : 50; ts : 30 \}$

Note that all length-1 items in the running example are productive in the normal domain since their supports are less than 1 (productivity test for length-1 items).

Lines 3 to 16 repeatedly mines the set of productive HUIs as follows. While there are more than one items in $iList$ (that is, $|iList| > 1$), a temporary list called $tempList$ is created in Line 4. For any two sets of itemsets (S_1 and S_2) in $iList$ that satisfy the Apriori candidate generation property, their coversets ($cov(S_1)$ and $cov(S_2)$) respectively, are obtained from $iCov$ in Line 8. In Line 9, a candidate itemset (S) is created as the union of S_1 and S_2 , while the coverset of S ($cov(S)$) is obtained as the intersection of $cov(S_1)$ and $cov(S_2)$. If $|cov(S)| > 0$ and S is productive in the normal domain, $iCov$ is updated with S and its coverset ($cov(S)$) in Line 11, and at the same time, S is added to $tempList$. This ensures that only productive itemsets are stored as candidate HUIs. The total utility of S , $u(S)$, is then computed in Line 12 (see Example 14). If $u(S)$ is greater than $minUtil$, the set of productive high utility itemsets is updated with S and $u(S)$ in Line 14.

After completing the nested for-loops, $iList$ is cleared in Line 15. Subsequently, the content of $tempList$ is copied into $iList$. The length of $iList$ is then rechecked, and if the loop-continuation condition is met, the while-loop is iterated again. The process continues until $|iList|$ is no longer greater than 1, leading to the termination of the productive HUIM. The set of productive HUIs is then returned in Line 16. This process is demonstrated using the running example, as shown in Example 14.

Example 14. Given $iList$ with $iCov$ and $iUtil$ as previously described, in Lines 5 and 6, S_1 will initially be $\{b\}$ while S_2 will be $\{c\}$ respectively. As they satisfy the Apriori candidate generation property, their coversets will be obtained from $iCov$ as $cov(b) = \{1, 4, 7, 10\}$ and $cov(c) = \{2, 7, 10\}$ in Line 8. In Line 9, S will be generated as $\{b, c\}$ with $cov(b, c) = \{7, 10\}$. Since $|cov(b, c)| > 0$ and $\{b, c\}$ is productive (that is, $sup(b, c) > sup(b) \times sup(c)$), $iCov$ will be updated in Line

11 with $iCov[b, c] = [7, 10]$ while $\{b, c\}$ will be added to $tempList$. The total utility of $\{b, c\}$ will be calculated in transactions 7 and 10 as $cov(b, c) = \{7, 10\}$. This will result in $u(b, c) = ((1 \times 2) + (1 \times 2)) + ((2 \times 2) + (2 \times 2)) = 12$. Since $u(b, c) > 10$, the key-value pair $\{b, c\} : 12$ will be added to HUI in Line 14. While $S_1 = \{b\}$ for the outer for-loop, the inner for-loop iterates to $S_2 = \{e\}$ and similar process is repeated. However, for $S_2 = \{e\}$, since $cov(b, e) = \emptyset$ and $\{b, e\}$ is not also productive, its total utility will not be computed in Line 12 and neither will $iCov$ or $tempList$ be updated in Line 11. The inner for-loop thus iterates to the next item $\{p\}$. Upon the completion of the first nested for-loops, the content of $iList$ changes from $iList = \{b, c, e, p, t, ts\}$ to $iList = \{\{b, c\}, \{b, p\}, \{c, p\}, \{e, t\}, \{e, ts\}, \{t, ts\}\}$ after executing Line 15. The candidate generation process repeats until $|iList| \leq 1$. For our running example, the set of productive high utility itemsets returned in Line 16 given $minUtil = 10$, will be $HUI = \{(\{b\} : 14), (\{c\} : 10), (\{e\} : 16), (\{p\} : 38), (\{t\} : 50), (\{ts\} : 30), (\{b, c\} : 12), (\{b, p\} : 40), (\{c, p\} : 24), (\{e, t\} : 28), (\{e, ts\} : 28), (\{t, ts\} : 40), (\{b, c, p\} : 28), (\{e, t, ts\} : 24)\}$.

While these are the productive HUIs reported in Example 14, some might derive productivity from their productive proper supersets. Therefore, there is a need to prune such HUIs and report only the set of truly correlated HUIs.

For each itemset i in the set of productive high utility itemset dictionary, HUI , its exclusive domain is obtained and the itemset is tested for productivity in the exclusive domain. If the itemset i is productive in the *exclusive domain*, it is added to the set of correlated high utility itemset dictionary $corHUI$ in Line 17. When all itemsets have been tested, the set of correlated high utility itemsets are returned in Line 18 and the correlated HUIM terminates. It is worth noting that the correlation test in this case aims to eliminate itemsets that are productive due to their proper productive supersets that are of high utility.

For our running example, given HUI in Example 14, itemset $\{b\}$ will be first tested for productivity in the exclusive domain (see Example 13 on how this is done). Though $\{b\}$ is productive in the normal domain, its productivity is actually caused by its proper productive superset $\{b, p\}$. That is, $\{b\}$ is only productive because people are buying $\{b, p\}$. Hence, the high utility of $\{b\}$ alone cannot be relied on in decision making, but the high utility of $\{b, p\}$ together can be relied on since $\{b, p\}$ is correlated in both normal and exclusive domain. When all itemsets in the set of productive high utility itemsets are tested for productivity in the exclusive domain, the resulting set of correlated high utility itemsets returned in Line 18 will be $corHUI = \{(\{c\} : 10), (\{e\} : 16), (\{p\} : 38), (\{t\} : 50), (\{b, p\} : 40), (\{e, t\} : 28), (\{e, ts\} : 28), (\{t, ts\} : 40), (\{b, c, p\} : 28), (\{e, t, ts\} : 24)\}$.

5 Experimental Analysis

To demonstrate the effectiveness of CHUIM, due to space constraints, we performed experiments on the following two datasets (obtained from [8]): (a) retail (88,162 total transactions with 16,470 unique items - partly dense in nature); and (b) Kosarak10k (10,000 total transactions with 10,094 items - sparse in nature).

We compare our proposed CHUIM (implemented in Python) with D2HUP [11], FCHMbond [4] and FCHMallconf [5] (all implemented in [8] using Java).

Kosarak10K Dataset									
	Reported High Utility Itemset			Runtime in Seconds			Memory Used (MB)		
	18K	18.4K	18.8K	18K	18.4K	18.8K	18K	18.4K	18.8K
D2HUP	54	49	42	2663.10	111.26	1.23	2738.40	1542.60	981.60
FCHMallconf 0.2	12	12	12	2843.10	14.10	0.87	2368.30	2355.60	2151.20
FCHMallconf 0.8	5	5	5	2160.00	7.89	0.75	2366.10	2313.70	1675.50
FCHMbond 0.2	9	9	9	10639.10	55.60	0.95	2365.50	1911.70	1789.40
FCHMbond 0.8	5	5	5	4085.70	22.44	0.83	2310.60	1834.50	1662.30
CHUIM	30	28	26	1.15	1.07	0.78	19.00	18.86	18.80

Retail Dataset									
	Reported High Utility Itemset			Runtime in Seconds			Memory Used (MB)		
	20K	30K	40K	20K	30K	40K	20K	30K	40K
D2HUP	259	114	66	0.94	0.73	0.41	1775.90	1479.30	1301.50
FCHMallconf 0.2	67	31	20	1.84	1.50	1.13	2423.40	1647.60	1358.30
FCHMallconf 0.8	59	24	14	1.71	1.39	1.04	1627.50	1522.40	1325.20
FCHMbond 0.2	63	27	16	4.86	1.23	0.89	2344.60	1654.60	1261.30
FCHMbond 0.8	59	24	14	4.45	1.20	0.73	1559.30	1476.20	1103.70
CHUIM	233	105	58	14.09	13.61	13.41	279.40	198.86	175.57

Fig. 1: Reported HUIs, Runtime and Memory Usage Analysis

As shown in Figure 1, for the given minimum utility thresholds of 18K, 18.4K and 18.8k in the Kosarak10K dataset, D2HUP reports all HUIs while FCHMallconf and FCHMbond report a much smaller number depending on the *minimum confidence* or *minimum bond* threshold. CHUIM, though reporting a smaller number than D2HUP reports slightly higher number of correlated HUIs. However, when the *minimum confidence* or *bond* thresholds are set to 0.0 both FCHMallconf and FCHMbond report same number of correlated HUIs as reported D2HUP. With regards to runtime and memory usage it can be observed that the Kosarak10K dataset, CHUIM is more efficient.

Similarly, in the Retail dataset, for the given minimum utility thresholds, FCHMallconf and FCHMbond miss some correlated HUIs which are reported in CHUIM. Again if the *minimum confidence* or *bond* thresholds are set to 0.0, FCHMallconf and FCHMbond will report same number of HUIs as D2HUP which includes uncorrelated HUIs. For the Retail dataset, though CHUIM has longer runtime than the compared algorithms, it is worth noting, it used less memory during the discovery process.

6 Conclusion

In conclusion, our CHUIM (Correlated High Utility Itemset Miner) algorithm effectively discovers highly correlated high utility itemsets in transaction databases by employing productivity measures in both normal and exclusive domains. This approach filters out weakly correlated itemsets, resulting in a more concise and decision-relevant set. Extensive experiments on benchmark datasets confirm CHUIM's effectiveness in mining and reporting a smaller set of itemsets with inherent relationships. Future work involves extending CHUIM to handle dynamic and complex data, and exploring the application of correlation and HUIM concepts in diverse domains like social networks, IoT, and sensor networks.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207-216, (1993).
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.-S., Lee, Y.-K.: Efficient Tree Structures for High-utility Pattern Mining in Incremental Databases. *IEEE Trans. Knowl. Data Eng.* 21(12), 1708–1721 (2009).
3. Chan, R., Yang, Q., Shen, Y. D.: Mining High Utility Itemsets. In *Third IEEE International Conference on Data Mining* (pp. 19-19). IEEE Computer Society, (2003).
4. Fournier-Viger, P., Lin, J. C. W., Dinh, T., Le, H. B.: Mining Correlated High-Utility Itemsets using the Bond Measure. In: *11th International Conference on Hybrid Artificial Intelligent Systems*, Springer, pp. 53-65 (2016).
5. Fournier-Viger, P., Zhang, Y., Lin, J. C. W., Dinh, D. T., Bac Le, H. Mining Correlated High-Utility Itemsets using Various Measures. *Logic Journal of the IGPL*, 28(1), 19-32, (2020).
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: *21st International Symposium on Foundations of Intelligent Systems*, pp. 83-92, (2014).
7. Fournier-Viger, P., Zida, S. Lin, C.W., Wu, C.-W., Tseng, V. S.: EFIM-Closed: Fast and Memory Efficient Discovery of Closed High-Utility Itemsets. In: *Proceedings 12th International Conference on Machine Learning and Data Mining*, pp. 199–213. Springer (2016).
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 *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases, Part III*, pp. 36-40. Springer (2016).
9. Fournier-Viger, P., Wu, C.W., Tseng, V.S.: Novel Concise Representations of High Utility Itemsets using Generator Patterns. In: *Proceedings of 10th International Conference on Advanced Data Mining and Applications*, pp. 30–43. Springer (2014).
10. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns without Candidate Generation. *ACM Sigmod Rec.* 29(2), 1-12, (2000).
11. Liu, J., Wang, K., Fung, B.: Direct Discovery of High Utility Itemsets without Candidate Generation, In: *Proc. 12th IEEE International Conference on Data Mining*, pp. 984–989. IEEE (2012).
12. Miao, J., Wan, S., Gan, W., Sun, J., Chen, J.: Targeted High-Utility Itemset Querying. *IEEE Transactions on Artificial Intelligence* (2022).
13. Nofong, V. M., Okai, P. O., Abdel-Fatao, H., Kwashie, S., Bewong, M., Wondoh, J.: Towards Efficient Discovery of Target High Utility Itemsets. In *2022 IEEE International Conference on Data Mining Workshops*, pp. 517-526, IEEE (2022).
14. Shenoy, P., Haritsa, J. R., Sudarshan, S., Bhalotia, G., Bawa, M., Shah, D.: Turboharging Vertical Mining of Large Databases. *ACM Sigmod Rec.* 29(2), 22-33, (2000).
15. Tseng, V., Wu, C., Fournier-Viger, P., Yu, P.S.: Efficient Algorithms for Mining Top-K High Utility Itemsets. *IEEE Trans. Knowl. Data Eng.* 28(1), 54–67 (2016).
16. Webb, G. I.: Self-sufficient itemsets: An Approach to Screening Potentially Interesting Associations between Items. *ACM Transactions on Knowledge Discovery from Data* , 4(1), 1-20, (2010).
17. Yao, H., Hamilton, H. J., Butz, C. J.: A Foundational Approach to Mining Itemset Utilities from Databases. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pp. 482-486, SIAM, (2004).