A General Method for mining high-Utility itemsets with correlated measures

Nguyen Manh Hung\textsuperscript{a,b,c}, Tung NT\textsuperscript{a} and Bay Vo\textsuperscript{a}

\textsuperscript{a}Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Vietnam; \textsuperscript{b}University of Information Technology, Ho Chi Minh City, Vietnam; \textsuperscript{c}Vietnam National University, Ho Chi Minh City, Vietnam

ABSTRACT

Discovering high-utility itemsets from a transaction database is one of the important tasks in High-Utility Itemset Mining (HUIM). The discovered high-utility itemsets (HUIs) must meet a user-defined given minimum utility threshold. Several methods have been proposed to solve the problem efficiently. However, they focused on exploring and discovering the set of HUIs. This research proposes a more generalized approach to mine HUIs using any user-specified correlated measure, named the General Method for Correlated High-utility itemset Mining (GMCHM). This proposed approach has the ability to discover HUIs that are highly correlated, based on the all-confidence and bond measures (and 38 other correlated measures). Evaluations were carried out on the standard datasets for HUIM, such as Accidents, BMS_utility and Connect. The results proved the high effectiveness of GMCHM in terms of running time, memory usage and the number of scanned candidates.

ARTICLE HISTORY

Received 9 May 2021
Accepted 28 May 2021

KEYWORDS

high-utility itemset; high-correlated itemset; general method

Introduction

The task of High-Utility Itemset Mining aims to discover itemsets from customer transaction databases, for which the itemsets must satisfy a user-specified threshold. Several existing algorithms have been proposed to perform the task in many efficient ways, such as Two-Phase (Liu et al., 2005)), HUI-Miner (Liu & Qu, 2012), EFIM (Zida et al., 2017), Hminer (Krishnamoorthy, 2017), MEFIM, iMEFIM (Nguyen et al., 2019a), EFIM-Closed (Nguyen et al., 2019b) and HMiner-Closed (Nguyen et al., 2019b). However, these algorithms only consider the utility measure of the patterns and ignore the correlation among the high-utility patterns.

Recently, Fournier-Viger et al. proposed an approach for mining HUIs considering the patterns’ correlation. The authors developed the FCHM algorithm (Fournier-Viger et al., 2020) to explore HUIs to find correlated measures that satisfy the downward-closure property (DCP). Fast measure calculation techniques and pruning strategies on each measure are also presented (the authors proposals are based on the all-confidence and bond
measurements). These DCP strategies along with their conditions allow the FCHM algorithm (Fournier-Viger et al., 2020) to gain better performance compared with those of the FHM algorithm (Fournier-Viger et al., 2014).

**Motivation:** Fournier-Viger et al. pointed out the importance of HUIM in considering the itemset’s correlation. Thus, the authors developed a solution to solve the problem. However, this approach still has the following drawbacks:

1. Correlation measures must satisfy the DCP. Considering the correlation measure \( m_C \), then \( m_C \) must meet the following condition: \( m_C(X) \geq m_C(Y) \), \( \forall X \subset Y \). To apply a correlation measure into the mining process, it is necessary to prove that DCP is satisfied, and thus several correlation measures are not applicable.
2. Each measure requires different strategies to calculate and prune to enhance the performance of the corresponding mining task.

To address the above limitations, this work proposes a more generalized approach to mine HUIs using any correlated measures. Based on previous works on correlation measures (Vo & Le, 2011; Geng & Hamilton, 2006; Huynh et al., 2007), we present an algorithm capable of fast and efficient calculation of the measures by storing all the required information. This definitely improves the effectiveness of mining HUIs using correlated measures. Our major contributions are as follows.

1. The proposed approach is a HUIM-class algorithm, namely GMCHM, which considers the high-correlated property among itemsets within customer transaction databases.
2. Previous algorithms must use a different strategy for each respective measure. Our work proposes an approach to store all necessary information to compute the given measure in only one database scan.
3. Evaluation studies on standard databases illustrate that the new GMCHM algorithm is more efficient than the existing algorithms, in terms of execution time, memory consumption, and the candidates checked. This will be discussed in Experimental Study section of this paper.

The rest of this research is structured as follows. The Related Work section surveys the related works on the field of HUIM with correlated measures. The Preliminaries section presents the preliminaries and definitions. The next section, GMCHM Algorithm, proposes the GMCHM algorithm, along with a new data structure to store the required information while mining HUI using a specific correlation measure. Evaluation studies are carried out and discussed in the Experimental Study section. Finally, the Conclusion and Future Work section draws out the conclusions and discusses future improvements of the work.

**Related Work**

The HUIM problem was initially investigated in 2004 by Yao et al. (Liu et al., 2005)). Then, Liu et al. introduced the Two-Phase algorithm (Liu et al., 2005)) to mine HUIs in two stages, as its name implies. Based on the pattern growth approach, Han et al. extended the FP-Growth (Han et al., 2004) algorithm to develop the UP-Growth algorithm (Tseng et al., 2010) using the UP-Tree structure. Later, the authors improved that algorithm and
introduced the UP-Growth+ algorithm (Srinivasa Rao and Krishna Prasad 2012). Then Liu and Qu (2012) proposed the HUI-Miner algorithm to mine HUIs using only one phase. The authors also introduced a novel and efficient structure called a utility-list. In this list, each element is a tuple containing three fields, $TID$, $iutil$, $rutil$. Fournier-Viger et al. proposed the FHM algorithm (Fournier-Viger et al., 2014) with a new structure named EUCS (Estimated Utility Co-occurrence Structure) and a new pruning strategy based on this newly introduced structure, called EUCP (Estimated Utility Co-occurrence Pruning). Krishna-moorthy presented the Hmler algorithm (Krishnamoorthy, 2017) using five different pruning strategies to improve the performance of the HUI mining process. The EFIM algorithm (Zida et al., 2017) proposed by Zida et al. incorporates two new techniques to reduce the cost of database scans and memory consumption, named high-utility database projection (HDP) and high-utility transaction merging (HTM). By applying these two new strategies, EFIM (Zida et al., 2017) is much more effective in mining HUIs from dense databases. However, sparse databases contain several short or unidentical transactions, and HTM faces a major performance issue in terms of long execution time. To overcome this issue, Nguyen et al. presented a new extension of the EFIM algorithm, named iMEFIM (Nguyen et al., 2019a). This relies on a P-set novel structure to significantly reduce the cost of database scans. Furthermore, the authors also proposed a novel utility framework enabling previous HUIM algorithms to extract HUIs from databases having dynamic profit.

Mining patterns based on correlated measures are used to find itemsets that satisfy a user-specified correlated threshold, such as association rules, a set of highly correlated itemsets within a database. The use of database’s correlated measures was first proposed by Agrawal et al. to mine association rules (ARs) (Agrawal et al., 1993) based on the itemset’s occurrence frequency measure (or support). Later on, several works on association rule mining based on other measures were introduced, for instance support, confidence, lift, chi-square, cosine, gini-index, Laplacian, and so (up to 40 different measures). Vo and Le (2011) employed lattice and hash tables to mine ARs using a correlated measure from large databases. Baena-García and Morales-Bueno proposed an algorithm for mining frequent sequences with interestingness measures (Baena-García & Morales-Bueno, 2012). Barsky et al. proposed the Flipper algorithm to mine a new type of pattern, named flipping patterns, based on the contrasting of correlations among items from different levels of abstractions (Barsky et al., 2011). Fournier-Viger et al. developed the FCHM algorithm to mine HUIs using two correlated measures: $all\_confidence$ and $bond$ measure (Fournier-Viger et al., 2020).

**Preliminaries**

The mining high-correlated itemsets problem is to discover all itemsets that satisfy the following conditions: they must be high-utility itemsets and highly-correlated based on a correlated measure specified by the user. This section states the problem and preliminary concepts with regard to the correlated HUI mining task.

**Definition 1 (Transaction database).** A database containing a set of transactions \( \{T_1, T_2, \ldots, T_n\} \) is called a transaction database, denoted as \( D \). The sample data and structure are described in Table 1. Let \( l = \{i_1, i_2, \ldots, i_n\} \) be the set of all distinct items in \( D \), \( T_q \) be a transaction in \( D \) with a unique identifier, and \( q \) be the set of all items in
that transaction, \( T_q \subseteq I \). Each item \( i \) in \( T_q \) is assigned with a non-negative integer value, called the quantity, denoted as \( q(i, T_q) \). Furthermore, item \( i \) also has another associated value, also a positive integer, called the unit profit and denoted as \( p(i) \) (Liu et al., 2005; Zida et al., 2017). Table 2 presents the unit of every item from \( D \) given in Table 1.

**Definition 2 (Utility of an item, an itemset).**
- Utility of an item \( i \) in a transaction \( T_q \), denoted as \( u(i, T_q) \), is defined as the product between its quantity in \( T_q \) and unit profit, \( u(i, T_q) = p(i) \times q(i, T_q) \) (Liu et al., 2005).
- Utility of an itemset \( X \) in transaction \( T_q \), denoted as \( u(X, T_q) \), determines the sum of utility values of every item \( i \) in \( T_q \), and calculated as \( u(X, T_q) = \sum_{i \in X \subseteq T_q} u(i, T_q) \) (Liu et al., 2005; Zida et al., 2015).
- Utility of an itemset \( X \) within the whole database \( D \), denoted as \( u(X) \), is specified as the sum of utility of \( X \) from all the transactions containing \( X \), and is calculated as \( u(X) = \sum_{T_q \in D \wedge X \subseteq T_q} u(X, T_q) \) (Liu et al., 2005).

**Definition 3 (High-utility itemset mining).** The task of mining high-utility itemset is to discover the complete set of itemsets with a utility that satisfies a prespecified minimum utility threshold: \( u(X) \geq \text{minutil} \). The discovered itemsets are called high-utility itemsets (Liu et al., 2005; Fournier-Viger et al., 2020; Liu et al., 2012; Tseng et al., 2013; Song et al., 2014; Fournier-Viger et al., 2018).

**Definition 4 (Support, support-bin).**
- Support measure of an itemset \( X \), denoted as \( \text{supp}(X) \), is the occurrence frequency of that itemset within the database \( D \), and is defined as \( \text{supp}(X) = |\{T_q \mid T_q \in D \wedge X \subseteq T_q\}| \) (Fournier-Viger et al., 2020).
- \( I \) is defined as the set of all distinct items in \( D \), then the array storing the support of every item \( z \in I \) is denoted as \( \text{supp}[z] \) and has the length equal to \( |I| \). Each element of the array is called a support-bin and stores the support value of a single item \( z \in I \). The support-bin array allows direct access to the support of every item \( z \in I \) with the cost of \( O(1) \).

**Calculation of the support of an itemset.** Initialize the support-bin \( \text{supp} \) array to zeros. For each transaction \( T_q \) in the database \( D \), update the value for support-bin \( \text{supp}[z] \) for each \( z \in T_q \) as follows.

**Table 1.** An example transaction database.

| TID | Transaction | Quantity |
|-----|-------------|----------|
| \( T_1 \) | \{a, c, d, f\} | \{4, 3, 3, 1\} |
| \( T_2 \) | \{a, b, c, d, e, f\} | \{1, 2, 1, 6, 2, 5\} |
| \( T_3 \) | \{a, c, d\} | \{1, 1, 1\} |
| \( T_4 \) | \{b, c, f, g\} | \{2, 1, 1, 2\} |
| \( T_5 \) | \{b, c, d, f, g\} | \{2, 1, 6, 6, 6\} |

**Table 2.** Items’ unit profits.

| Item | a | b | c | d | e | f | g |
|------|---|---|---|---|---|---|---|
| Profit | 6 | 3 | 1 | 2 | 1 | 3 | 1 |
• If transactions are not merged, \( \text{supp}[z] = \text{supp}[z] + 1 \).
• If transactions are merged, \( \text{supp}[z] = \text{supp}[z] + \text{mergeCount} \), whereas \( \text{mergeCount} \) denotes the number of transactions merged.

**Definition 5 (Measures).** Let \( X \) and \( Y \) be the two itemsets in database \( D \); \( n \) be the number of transactions in \( D \), \( n = |D| \); \( n_X, n_Y, n_{XY} \) be the occurrence frequencies of \( X \), \( Y \) and \( XY \) in \( D \), respectively; \( n_{\overline{X}}, n_{\overline{Y}}, n_{\overline{XY}}, n_{\overline{XY}} \) be non-occurrence count of \( X \), \( Y \), \( XY \), \( X \) and not \( Y \), \( Y \) and not \( X \), respectively; \( \text{dissup}(XY) \) is the occurrence count of \( X \), \( Y \) or \( XY \) (disjunctive support), in other words, \( \text{dissup}(XY) \) will increase by one if a transaction contains \( X \) or \( Y \) or both \( XY \) (Vo & Le, 2011; Geng & Hamilton, 2006; Huynh et al., 2007).

We then have the following equations.

\[
\begin{align*}
  n_{\overline{X}} &= n - n_X \\
  n_{\overline{XY}} &= n_X - n_{XY} \\
  n_{\overline{Y}} &= n - n_Y \\
  n_{\overline{XY}} &= n_Y - n_{XY} \\
  \text{dissup}(XY) &= |\{T \in D \mid XY \cap T \neq \emptyset\}|
\end{align*}
\]

**Definition 6 (TID-set).** Let \( i \) be an item and \( i \in I \). \( \text{TID} - \text{set}[i] \) denotes the set containing the transaction-ID of all transactions containing \( i \) in \( D \).

**Definition 7 (Correlated high-utility itemset).** An itemset is called a correlated high-utility itemset if and only if it’s a HUI and its interestingness measure \( \geq \) minimum correlated threshold, with the minimum correlated threshold is specified by the user prior to mining.

**Definition 8 (Correlated high-utility itemset mining).** With all the definitions and preliminaries presented, the task of Correlated High-utility Itemset Mining (CHIM) using an interestingness measure is to discover the complete set of high-utility itemsets and highly correlated itemsets based on the user-specified interestingness measure.

Example: with \( \text{minutil} = 65 \) and the correlated threshold varying in \{0.0, 0.5, 0.7\} using two interestingness measures called all confidence and bond. Table 3 below shows the obtained set of correlated high-utility itemsets discovered from database \( D \) in Tables 1 and 2 above.

The current manuscript also uses several known definitions in the task of high-correlated itemset mining, interestingness measures, high-utility itemset mining, TWU pruning strategies (Liu et al., 2005), local-utility (Zida et al., 2015), sub-tree utility (Zida et al., 2015), primary and secondary sets (Zida et al., 2015) to eliminate unpromising candidates and reduce the search space. Other than that, transaction merging (Zida et al., 2015) and database projection techniques (Zida et al., 2015) are also used to further enhance the performance of the mining task.

**Table 3.** Correlated high-utility itemset for each correlated measure.

| Measure   | 0.0  | 0.5  | 0.7  |
|-----------|------|------|------|
| all confidence | adf, adfc, bdf, bdfc, df, dfc | adf, bdf, bd, bdc | bd |
| bond      | df, dfc | dfc | \( \emptyset \) |
GMCHM Algorithm

In general, all the correlation measures are about values such as the support of the currently considered itemset (support of \( X \)), support of the extension itemset (support of \( Y \)), support of \( XY \), disjunctive support of \( XY \) and the total number of transactions \( n \) in database \( D \). The remaining values can be easily obtained via Definition 5.

To quickly identify the correlation of each itemset in the list of HUIs, we extend the structure to store the HUIs as follows.

- The TID-set is constructed to record the occurrence of each item in the transactions during the first database scan. The constructed TID-set would speed up the computation of the disjunctive support of every itemset significantly after the itemset is identified as a HUI.
- The support-bin is to store the support measure of every single item \( i \in I \), and the support calculation of any item in \( D \) would thus be faster and more convenient.

Based on the above structure, we modified the EFIM algorithm (Zida et al., 2017) to discover HUIs combined with all the techniques to calculate the correlation measure into a new algorithm, called the General Method for Correlated High-utility itemset Mining (GMCHM). The GMCHM algorithm first discovers all HUIs, then computes the correlation measure using the discovered set of HUIs.

Algorithm 1: The GMCHM algorithm

\begin{algorithm}
\begin{small}
\textbf{Input:} \( D \): a transaction database, \textbf{minutil}: user-defined min utility threshold, \textbf{mincore}: user-defined correlate threshold
\textbf{Output:} The list of high-utility and correlated itemsets.
1. \( X=\emptyset \);
2. Scan \( D \) to build TID-set, loop each item \( i \in I \), calculate \( lu(X,i) \) and \( sup(i) \);
3. \( \text{Secondary}(X)=\{i \mid i \in I \land lu(X,i) \geq \text{minutil}\} \);
4. Sort \( \text{Secondary}(X) \) in ascending order of \( lu(X,i) \);
5. Scan \( D \) to eliminate any item \( i \notin \text{Secondary}(X) \) from transactions, and pull out the empty transactions;
6. Sort transaction in \( D \) by the item order in ascending order according to \( \text{Secondary}(X) \) during reading backward;
7. \( \text{su}(X,i)=\text{utility of the sub-tree of each item } i \in \text{Secondary}(X) \) by scanning \( D \) and using a utility-bin array;
8. \( \text{Primary}(X)=\{i \mid i \in \text{Secondary}(X) \land \text{su}(X,i) \geq \text{minutil}\} \);
9. \text{RETURN} Search \((X, D, \text{Primary}(X), \text{Secondary}(X)), \text{minutil}, \text{mincore}) \);
\end{small}
\end{algorithm}

Procedure: Search

\begin{algorithm}
\begin{small}
\textbf{Input:} \( X \): an itemset, \( DX \): the projected database of \( X \), \( \text{Primary}(X) \): the primary items of \( X \), \( \text{Secondary}(X) \): the secondary items of \( X \), \text{minutil} and \text{mincore} threshold
\textbf{Output:} the set of correlated high-utility itemsets that are extensions of \( X \).
1. \text{FOR EACH} item \( i \in \text{Primary}(X) \) \text{do}
2. \( \beta=X\cup\{i\} \);
3. Scan \( DX \) to calculate \( u(\beta) \) and create \( D\beta \);
4. \text{IF} \( u(\beta) \geq \text{minutil} \text{ AND } \text{measure}(\beta) \geq \text{mincore} \) THEN RETURN \( \beta \);
5. Calculate \( \text{su}(\beta, z) \) and \( lu(\beta, z) \) for all items \( z \in \text{Secondary}(X) \) by scanning \( D_\alpha \) once, using two utility-bin arrays;
6. \( \text{Primary}(\beta)=\{z \in \text{Secondary}(X) \mid \text{su}(\beta,z) \geq \text{minutil}\} \);
7. \( \text{Secondary}(\beta)=\{z \in \text{Secondary}(X) \mid \text{lu}(\beta,z) \geq \text{minutil}\} \);
8. Search \((\beta, D_\beta, \text{Primary}(\beta), \text{Secondary}(\beta), \text{minutil}, \text{mincore}) \);
9. \text{END FOR}
\end{small}
\end{algorithm}
The input parameters for the algorithm are the transaction database \( D \), minimum utility threshold \( \text{minutil} \), and minimum correlate threshold \( \text{mincore} \) (all confidence, bond, etc.). The two thresholds are user-specified. The major modification of the algorithm comes from line #2 of the GMCHM algorithm (Algorithm 1) and line #4 in the Search procedure. Line #2 of the GMCHM algorithm performs construction of the TID-set and the support-bin. Line #4 of the Search procedure calculates the correlated measure on an itemset besides its utility. Line #3 of the GMCHM algorithm uses the local utility of every single item \( i \in I \) to construct set \( \text{Secondary} \) set, at this step \( X = \emptyset \). Line #4 then sorts the Secondary set based on the ascending order of items’ TWU values. Line #5 removes all items \( i \in I \) which do not exist in the Secondary set and pulls out any empty transactions. Line #6 sorts all transactions in \( D \) in the ascending order of items in Secondary when reading backward. The block of line #4 to line #6 aims to reduce memory consumption and database scan costs. The algorithm GMCHM then constructs the \( \text{Primary} \) set using the items from \( \text{Secondary} \) having their sub-tree utility no less than \( \text{minutil} \). The algorithm then passes all the gathered information into the recursive Search procedure in order to start the depth-first search process on the search space.

The recursive Search procedure performs the main correlated HUI mining task using the information passed from the main GMCHM algorithm. Its input parameters are the itemset to be extended \( X \), the projected database on \( X \) (named as \( DX \)), the \( \text{Primary} \) and \( \text{Secondary} \) sets of \( X \), minimum utility threshold \( \text{minutil} \) and minimum correlated measure threshold \( \text{mincore} \). Let \( \beta \) be the extended itemset \( X \) using a single item \( i \in \text{Primary}(X) \) (Line #2), \( \beta = X \cup \{i\} \). Line #3 scans through the projected database \( DX \) to calculate the utility value of \( \beta \) and construct the projected database of \( \beta \), denoted as \( D_\beta \). This step also performs transaction merging if necessary. Line #4 determines whether the itemset \( \beta \) is a correlated high-utility itemset or not by comparing its utility value \( u(\beta) \) and \( \text{measure}(\beta) \) against the specified thresholds \( \text{minutil} \) and \( \text{mincore} \), respectively. Whereas \( \text{measure}(\beta) \) is the user-defined function that computes the correlated measure of item \( \beta \). If the itemset \( \beta \) co-satisfies the two conditions, it will be a correlated high-utility itemset and will be returned.

Experimental Study

To validate the effectiveness of the GMCHM algorithm, several experiments were carried out on a machine with the following configuration: Intel® Core™ i5-10400F processor @ 2.90 GHz, 8GB of RAM, and running on Windows 10 Pro 64-bit operating system. Java was selected as the primary language to implement the algorithm, using JDK8. The common standard testing databases from the SPMF, an Open-Source Data Mining Library\(^1\) are used, and their characteristics are shown in Table 4.

Besides recording the execution time and memory usage in each test case with a different threshold, the number of generated candidates is also recorded. The recorded

| Dataset | #Transactions | #Items | Average transaction length |
|---------|---------------|--------|--------------------------|
| BMS     | 77,512        | 3,340  | 4.62                     |
| Accident| 340,183       | 468    | 33.8                     |
| Connect | 67,557        | 129    | 43                       |

Table 4. Test databases characteristics.
data is then compared against that of the FCHM algorithm using \textit{all-confidence} and \textit{bond}. The source code of the FCHM algorithm is loaded from the SPMF\textsuperscript{2}, which is Java-based source code, for a fair comparison. Throughout the experiments, the \textit{minutil} threshold value was varied from 0.04\% to 28\% and the \textit{mincore} value was varied in \{0.0, 0.1, 0.3, 0.5\}. The Figures 1–6 present the results of the comparison of mining HUIs with \textit{mincore} = 0.1 on different datasets.

\textbf{Runtime:} In each testing dataset, the obtained results are shown from Figures 1–3. It is recognized that GMHCM dominates the execution time test. The speed of GMHCM is up to 7.5 times faster than that of the FCHM algorithm using the \textit{bond} measure and up to 2,600 times faster when the \textit{all-confidence} measure is 0.1. Furthermore, the same results are also observed on 38 other measures. The performance of GMCHM is mainly due to the transaction merging and database projection strategies, which significantly reduce the database scan costs. As seen in Figure 1, the runtime of GMCHM is up to 2–170 times faster than FHCM by with respect to the \textit{all-confidence} and \textit{bond} measures. The same speed-up factor is also shown in Figure 2 on the BMS database, and when using \textit{bond} it is from 1.1 to 2 times faster. For the \textit{all-confidence} measure, the improved runtime is from 90 to 170 times faster than with the FCHM algorithm. In the Connect database, which is shown in Figure 3, our proposed algorithm has a runtime that is from 3.5 to 7.5 times faster on the \textit{bond} measure. Especially when using the \textit{all-confidence} measure, the algorithm is observed to have a significant speed boost, from 700 to 2,600 times faster. In Table 6, when changing the correlation threshold, the execution time of the GMCHM algorithm does not change significantly and is lower than that of the FCHM algorithm in most cases and thus the algorithm performed very much better than the original FCHM algorithm.

\centering
\includegraphics[width=0.8\textwidth]{accident.png}
\caption{Runtime comparison on the Accident database.}
When the value of the minimum correlated threshold varies, the GMCHM has its runtime slightly decreased while FCHM tends to see it fall by a large amount. This is because FCHM uses the pruning strategy that relies on the downward-closure property.
of the measures `all_confidence` and `bond`. For the GMCHM algorithm with the `bond` measure, the runtime is higher than those of the `all_confidence` and the remaining 38 measures, since it needs to perform intersections on the itemsets’ TID-sets.

**Figure 4.** Memory usage comparison on the Accident database.

**Figure 5.** Memory usage comparison on the BMS database.
Memory usage: The results shown in Figures 4–6 show that on all the tested minimum utility and correlated thresholds the memory usage of the GMCHM algorithm is much better than FCHM using both all confidence and bond measures, better by from 2 to 10 times. The high memory consumption of the FCHM algorithm is due to the EUCS structure (Fournier-Viger et al., 2020), which is actually a two-dimensional array to store the utility and support information of item pairs. Other than that, the FCHM algorithm adopts the bit vector approach to calculate the bond measure (Fournier-Viger et al., 2020), and the OR operator to compute the disjunctive support of an itemset, which requires $m \times n$

![Figure 6. Memory usage comparison on the Connect database.](image)

| Table 5. Generated candidates across thresholds. |
|-----------------------------------------------|
| **Data Measure** | **GMCHM** | **FCHM** |
| | All_conf & 38 other measures | Bond | All_conf | Bond |
| Accident 0.1 | 1.900E+7 | 26006 | 26006 | 75134 | 75173 |
| | 1.850E+7 | 32790 | 32790 | 89712 | 89754 |
| | 1.800E+7 | 41119 | 41119 | 106621 | 106669 |
| | 1.750E+7 | 51883 | 51883 | 128087 | 128135 |
| | 1.700E+7 | 65067 | 65067 | 153037 | 153106 |
| BMS 0.1 | 2.275E+6 | 274 | 274 | 4034885 | 23341 |
| | 2.250E+6 | 309 | 309 | 63511589 | 42487 |
| | 2.225E+6 | 346 | 346 | 591727604 | 68263 |
| | 2.220E+6 | 411 | 411 | 3266929403 | 99935 |
| | 2.175E+6 | 463 | 463 | 11857403881 | 142586 |
| Connect 0.1 | 1.525E+8 | 35938 | 35938 | 160932 | 160932 |
| | 1.500E+8 | 59734 | 59734 | 238329 | 238329 |
| | 1.475E+8 | 94871 | 94871 | 344828 | 344828 |
| | 1.450E+8 | 147254 | 147254 | 495728 | 495728 |
| | 1.425E+8 | 223064 | 223064 | 703495 | 703495 |
complexity, in which \( m \) is the length of the currently considered transaction and \( n \) is the number of transactions to be scanned. This would thus increase the memory consumption of the algorithm in the case of long transactions, as shown in Figures 4 and 6.

**Candidate generation:** Using the adopted several pruning strategies in the GMCHM algorithm leads to a greater reduction in search space than with the FCHM algorithm, and the results are described in detail in Table 5. The FCHM algorithm only uses three pruning strategies based on TWU, utility list, and interestingness measure to prune the search space. The GMCHM algorithm also utilizes more efficient approaches like High-utility Database Projected (HDP), High-utility-Transaction Merging (HTM), and tighter upper-bounds, such as local utility and sub-tree utility, to prune the candidates. The proposed algorithm thus significantly improves the performance of mining HUIs.

**Conclusion and Future Work**

Mining correlated high-utility itemsets represents an essential task in the science of data mining. In this work, we developed the GMCHM algorithm to adopt several strategies to efficiently calculate the interestingness correlated measures combined with the task of high-utility itemset mining. These strategies reduce the cost of database scans and improve the candidate pruning efficiency to achieve better performance with regard to execution time and memory usage.

As observed in the experiments, the bond measure and others provide speed gains to the mining phase, but the improvement is not as great as with the all confidence measure. Thus, better pruning strategies on such measures are needed to address this drawback. Mining correlated high-utility itemsets from dynamic profit databases also should be considered, since these reflect the nature of the real-world databases. Moreover, it could be worth modifying and deploying the proposed algorithm using a parallel or distributed approach to further utilize the available power of computer systems.

**Notes**

1. [http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php](http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php)
2. [http://www.philippe-fournier-viger.com/spmf/](http://www.philippe-fournier-viger.com/spmf/)
Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2), 207–216. https://doi.org/10.1145/170036.170072

Baena-García, M., & Morales-Bueno, R. (2012). Mining interestingness measures for string pattern mining. *Knowledge-Based Systems*, 25(1), 45–50. https://doi.org/10.1016/j.knosys.2011.01.013

Barsky, M., Kim, S., Weninger, T., & Han, J. (2011). Mining flipping correlations from large datasets with taxonomies. *Proceedings of the VLDB Endowment*, 5(4), 370–381. https://doi.org/10.14778/2095686.2095695

Fournier-Viger, P., Wu, C. W., Zida, S., & Tseng, V. S. (2014). FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in bioinformatics)*, 8502, 83–92. https://doi.org/10.1007/978-3-319-08326-1_9

Fournier-Viger, P., Zhang, Y., Lin, J. C. W., Dinh, D. T., & Le, H. B. (2020). Mining correlated high-utility itemsets using various measures. *Logic Journal of the IGPL*, 28(1), 19–32. https://doi.org/10.1093/jigpal/jzz068

Fournier-Viger, P., Zhang, Y., Lin, J. C. W., Fujita, H., & Koh, Y. S. (2018). Mining local high utility itemsets. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in bioinformatics)*, 11030, 450–460. https://doi.org/10.1007/978-3-319-98812-2_41

Geng, L., & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3), https://doi.org/10.1145/1132960.1132963

Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1), 53–87. https://doi.org/10.1023/B:DAMI.0000005258.31418.83

Huynh, X. H., Guillet, F., Blanchard, J., Kuntz, P., Briand, H., & Gras, R. (2007). A graph-based clustering approach to evaluate interestingness measures: A tool and a comparative study. *Stud. Comput. Intell.*, 43, 25–50, https://doi.org/10.1007/978-3-540-44918-8_2

Krishnamoorthy, S. (2017). HMiner: Efficiently mining high utility itemsets. *Expert Systems with Applications*, 90, 168–183. https://doi.org/10.1016/j.eswa.2017.08.028

Liu, Y., Liao, W. K., & Choudhary, A. (2005). A two-phase algorithm for fast discovery of high utility itemsets. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in bioinformatics)*, vol. 3518, 689–695. https://doi.org/10.1007/11430919_79

Liu, M., & Qu, J. (2012). Mining high utility itemsets without candidate generation categories. *Proc. 21st ACM Int. Conf. Inf. Knowl. Manag.*, 55-64, https://doi.org/10.1145/2396761.2396773

Liu, J., Wang, K., & Fung, B. C. (2012). Direct discovery of high utility itemsets without candidate generation, in IEEE International Conference on Data Mining (ICDM).

Nguyen, L. T. T., Nguyen, P., Nguyen, T. D. D., Vo, B., Fournier-Viger, P., & Tseng, V. S. (2019a). Mining high-utility itemsets in dynamic profit databases. *Knowledge-Based Syst.*, 175, 130–144. https://doi.org/10.1016/j.knosys.2019.03.022

Nguyen, L. T., Vu, V. V., Lam, M. T., Duong, T. T., Manh, L. T., Nguyen, T. T., Vo, B., & Fujita, H. (2019b). An efficient method for mining high utility closed itemsets. *Information Sciences*, 495, 78–99. https://doi.org/10.1016/j.ins.2019.05.006

Song, W., Liu, Y., & Li, J. (2014). BAHUI. *International Journal of Data Warehousing and Mining*, 10(1), 1–15. https://doi.org/10.4018/ijdwm.2014010101

Srinivasa Rao, A. B. O., & Krishna Prasad, M. (2012). An improved UP-Growth high utility itemset mining. *IJA ICA - International Journal of Computer Applications*, 58(2), 25–28. https://doi.org/10.5120/9255-3424
Tseng, V. S., Shie, B. E., Wu, C. W., & Yu, P. S. (2013). Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Transactions on Knowledge and Data Engineering, 25*(8), 1772–1786. https://doi.org/10.1109/TKDE.2012.59

Tseng, S. V., Wu, C. W., Shie, B. E., & Yu, P. S. (2010). UP-Growth: an efficient algorithm for high utility itemset mining. In: The 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 253–262. https://doi.org/10.1145/1835804.1835839

Vo, B., & Le, B. (2011). Interestingness measures for association rules: Combination between lattice and hash tables. *Expert Systems with Applications, 38*(9), 11630–11640. https://doi.org/10.1016/j.eswa.2011.03.042

Zida, S., Fournier-Viger, P., Lin, J. C. W., Wu, C. W., & Tseng, V. S. (2015). EFIM: A highly efficient algorithm for high-utility itemset mining. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in bioinformatics), 9413*, 530–546. https://doi.org/10.1007/978-3-319-27060-9_44

Zida, S., Fournier-Viger, P., Lin, J. C. W., Wu, C. W., & Tseng, V. S. (2017). EFIM: A fast and memory efficient algorithm for high-utility itemset mining. *Knowledge and Information Systems, 51*(2), 595–625. https://doi.org/10.1007/s10115-016-0986-0