Efficient Pruning Strategy for Mining High Utility Quantitative Itemsets

Loan T. T. Nguyen[1,2], Anh N. H. Pham[1,2], Trinh D. D. Nguyen[3], Adrianna Kozierkiewicz[4], Bay Vo[5], and N. T. Tung[5(✉)]

[1] School of Computer Science and Engineering, International University, Ho Chi Minh City, Vietnam
nttloan@hcmiu.edu.vn, ITITIU18007@student.hcmiu.edu.vn
[2] Vietnam National University, Ho Chi Minh City, Vietnam
[3] Faculty of Information Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Vietnam
20126291.trinh@student.iuh.edu.vn
[4] Faculty of Computer Science and Management, Wroclaw University of Science and Technology, Wrocław, Poland
Adrianna.kozierkiewicz@pwr.edu.pl
[5] Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam
{vd.bay,nt.tung}@hutech.edu.vn

Abstract. High Utility Quantitative Itemset Mining (HUQIM), which is an extension of the original High Utility Itemset Mining (HUIM), has become an important research area that answers the ever-growing need for useful information from the copious pool of data in reality. Due to the nature of the HUQIM problem, its search space is huge and could heavily affect the execution time. Thus, the recently proposed FHUQI-Miner has overcome these limits with novel pruning strategies to narrow the space and outperform previously introduced algorithms. However, there are certain shortcomings that the algorithm still faces. One of the limitations is that the proposed strategies would not operate as efficiently on dense datasets as they would on sparse datasets, resulting from the similarity in structure of the transactions and thus increasing the number of join operations in progress. To address this limitation, this work introduces an enhanced version of the FHUQI-Miner algorithm with an improved TQCS structure to reduce mining time and the number of joins performed.

Keywords: High Utility Quantitative · Itemset Mining · High Utility Itemset Mining · Data mining

1 Introduction

In recent years, one of the most studied areas in the field of Data Mining is High Utility Itemset Mining (HUIM) [1]. The task reveals patterns with high importance associated with a utility function. For example, the discovered information can be used in market analysis to study customer behaviors [1] or the time user stays on webpages when

N. T. Nguyen et al. (Eds.): ICCCI 2023, CCIS 1864, pp. 326–338, 2023.
https://doi.org/10.1007/978-3-031-41774-0_26

applied in click-stream analysis [2, 3], etc. This mining task has variations to fit several application targets, such as high utility sequential itemsets, high utility periodic itemsets, etc. HUIM aims to discover a complete set of patterns that have their utility value satisfies a user-specified threshold (μ). Unlike FIM, which only considers the appearance of items in transactions, HUIM extends the inputs further. Each item now has its quantity per transaction instead of a binary occurrence. In addition, they can carry another value known as unit utility, representing their level of importance, such as profit, cost, time, etc. [1]. HUIM is extensively studied and has many applications, resulting in a series of algorithms introduced in recent years [4]. However, the outcomes of HUIM do not contain information about quantities of the discovered patterns. Thus, an extension to HUIM to address this critical drawback was introduced. The task is known as High Utility Quantitative Itemset Mining (HUQIM); its outputs are now containing information about item quantities that yield the utility, besides knowing they are high utility [5, 6]. For instance, a discovered high utility itemset containing the following items {*cheese*, *coffee*, *champagne*} could help retailers co-promote them to boost sales. However, the discovered pattern does not provide about the quantities often purchased. HUQIM addresses this limitation to return patterns containing quantities that yield high profit besides the items, such as {*cheese:2*, *coffee:5*, *champagne:8*}. Furthermore, quantity ranges are also revealed by HUQIM. For example, an itemset such as {*ham:4–8*, *lettuce:2–5*} suggests that the combinations of items with their purchased quantity range yield high profit. Useful information like this could help retailers adjust their marketing policies to fit user needs better. Thus, HUQIM is considered more useful than the traditional HUIM task. However, considering more information makes HUQIM more complex and more challenging than HUIM. An item with different quantities is considered a different quantitative item (*Q-item* for short). The same is applied to the set of quantitative items (*Q-itemset*). This expands the search space of the problem further than HUIM. A quantitative item/itemset is called a high utility quantitative item/itemset if its utility is greater or equal to the μ threshold, which is specified by the users.

This work addresses the limitations of the previously proposed algorithm, the FHUQI-Miner [6], to improve the mining performance in terms of execution time, and memory usage and reduce the number of joins taken.

The remainder of this paper is organized as follows. A literature review of recent work related to HUIM and HUQIM is presented in the next section. Section 3 describes the backgrounds of the HUQIM task. Section 4 introduces our proposed techniques to improve mining performance. Evaluations are carried out and discussed in Sect. 5. Conclusions are drawn in Sect. 6, as well as future research opportunities are also discussed.

2 Related Works

Frequent Itemset Mining (FIM) is fundamental to the world of data mining, whose goal is to determine all sets of items that occurred at least a certain number of times in a transaction database [7]. In a transaction database, items are either appear or absent in each transaction, and thus they are all considered equally in FIM. Many approaches were introduced to reveal the set of frequent items [6, 7]. These approaches utilize the

frequency measure's anti-monotonic property to improve the mining performance. Over time, the need for more important knowledge is raised besides the ones obtained from the discovered frequent patterns. Among the factors, the most focused is on the profit gained from the mined patterns. This is where HUIM took the focus of research interests [8].

HUIM extends FIM to address its limitations [8], removing the binary occurrence barrier. Each item appearing in the transaction now has a quantity linked. Besides, its level of importance (profit, weight, etc.) is also considered. For example, each pound of cherry is priced at \$2, thus selling 3 lb yield \$6 in profit. HUIM is targeting to reveal all combinations of items that provide profit no less than a user-specified threshold, known as minimum utility. Unlike FIM, the numeric function used in HUIM is neither anti-monotonic nor monotonic, as the frequency measure does. Thus, HUIM is considered more challenging than FIM. Several HUIM algorithms were introduced. The early algorithms followed the two-phase model. As its name implies, the two-phase methods reveal high utility itemsets (HUIs) in two phases. Some notable methods are Two-Phase [9], IHUP [10], TWU-Mining [11], etc. These algorithms rely on an upper bound known as Transaction Weighted Utilization (TWU), which satisfies the downward-closure property [9], to reduce the search space. Generating candidates is time and memory consuming. Thus, the one-phase method stepped in to resolve the limitations. Generated candidates are eliminated as soon as they are considered unpromising, saving further computation time and memory needed to check and store them. Several notable algorithms can be named, such as HUI-Miner [12], EFIM [13], iMEFIM [14], etc.

However, all these mentioned algorithms only discovered the high utility patterns to the user, no information, such as quantities, is returned. To address this drawback, the task of HUQIM is proposed to discover HUQIs. However, until recently, only a few approaches were proposed, as the task is more complex than HUIM. Some important algorithms are HUQA [15], HUQI-Miner [5] and FHUQI-Miner [6]. The most recent approach is the FHUQI-Miner. It comes with several optimizations, such as TQCS, and RQCPS, to overcome the limitations of previous approaches.

3 Preliminaries

This section introduces the foundation principles expected when mining HUQIs. As the study of HUQIM has been previously inspired by the abundance of knowledge from its predecessor HUIM, well-known definitions will not be presented here to keep the paper concise.

Let D be the input quantitative transaction database containing m transactions and is denoted as $D = \{T_1, T_2, \ldots, T_m\}$, $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n distinctive items in the database D. A transaction $T_q \in D$, $T_q \subseteq I (1 \leq q \leq m)$ have a unique *transaction identifier*, denoted as T_{ID}. Each item $i \in I$ appears in transaction T_q has an associated positive number $q(i, T_q)$, referred to as the *internal utility* or its quantity in each transaction T_q. Another value called *external utility*, denoted by $p(i)$, will be used to keep track of the unit profit that each item i possesses. Table 1 introduces an exemplary transaction database D containing five transactions. Each transaction contains a set of items and

their respective internal utilities. Considering transaction T_1, four items are currently being purchased, namely item a, c, h and i, with the respective quantities of 2, 7, 4, and 9. Moreover, Table 2 expresses the external utility values (profit) of the items $i \in I$. By reference, item a will always have a value of 20 profit units, and item c will always be worth 70 profit units. These external utility values are uniform across all transactions within the database D.

Table 1. An example of a Quantitative Transaction Database

TID	Items	Internal utilities
T_1	a, c, h, i	2, 7, 4, 9
T_2	a, c	3, 8
T_3	a, b, c, g, h	2, 1, 7, 7, 4
T_4	b, c, g, h	2, 9, 8, 5
T_5	a, d, e, f	2, 5, 1, 1

Table 2. The corresponding profit for each item in Table 1

Item	a	b	c	d	e	f	g	h	i
Profit	20	10	70	54	11	100	75	47	96

An *exact Q-item* x is a paired value in the form of (i, q) where $i \in I$ is the identifier for item x and q is its respective quantity. Furthermore, each transaction $T_q \in D$ can be viewed as a list of exact Q-items [6]. A *range Q-item* x considers the quantity of an item not as an exact value but rather a range of values. Therefore, a range Q-item will feature both a lower bound as well as a larger upper bound value, denoted as a tuple (i, l, u), where $i \in I$ is the identifier for that item x, l is the lower bound and u refers to the upper bound. The internal size of a *Q-item*, also known as the *Q-interval* can be calculated as following $Q - interval = (u - l + 1)$. A range *Q-item* among the mentioned would be able to represent all the different exact *Q-items* as an entirety [6]. It is important to remember that *range Q-items* do not inherently exist within a database, but rather a concept formed for the purpose of mining quantitative items. In other words, an item from a transaction can only have exactly one definite value, which is the exact *Q-item* mentioned above. Another characteristic that is often overlooked at this point is that exact *Q-items* of the form (i, q) can also be represented using the notion for *range Q-items* (i, l, u), where $q = l = u$.

Definition 1: Quantitative Itemset [6]
A set of *Q-items* that are collected together under a group will be referred to as a quantitative itemset X. A *k-Q-itemset* means a *Q-itemset* that consists of k distinctive *Q-items*, $X = [x_1, x_2, \ldots, x_k]$. If there is at least one *range Q-item* among the items in

the set, the itemset would be referred to as a *range Q-itemset*. Otherwise, if and only if all of the *Q-items* in the set are *exact Q-items*, then it would be called an *exact Q-itemset*.

Definition 2: Inclusion of Q-item [6]
With an exact Q-item $X = (i, q)$ and a range Q-item $y = (j, l, u)$, y includes x or x is included in y if $i = j$ and $l \leq q \leq u$. Similarly, with two range Q-items $x = (i, l, u)$ and $y = (j, l\prime, u\prime)$, *y includes x or x is included in y* if $i = j$, $l \geq l\prime$ and $u \leq u\prime$.

Definition 3: Occurrence of a Q-item, Q-itemset [6]
For any exact Q-item $x = (i, q)$, it will be considered to occur in a particular transaction T_d, if $x \in T_d$. A range Q-item $x = (i, l, u)$ occurred in T_d if at least one of the exact Q-items in x is covered by that range occurs in the transaction. A Q-itemset $X = [x_1, x_2, \ldots, x_k]$ occurs in a transaction T_d if $\forall x \in X$, x occurs in T_d.

Definition 4: Occurrence Set and Support Count of a Q-itemset [6]
The occurrence-set of a Q-itemset X, $OCC(X)$, is the set of transactions where X appears. Given a Q-itemset X, the support count of X, $SC(X)$, is defined as the number of transactions containing X, $SC(X) = |OCC(X)|$.

Definition 5: Utility of a Q-item, Q-itemset [6]

- The utility of an exact Q-item $x = (i, q)$ in transaction T_d, denote $u(x, T_q)$ is determined as $u(x, T_q) = q(i, T_q) \times p(i)$.
- The utility of a range Q-item $x = (i, l, u)$ in transaction T_q is defined as the sum of all utility of the Q-items included in x: $u(x, T_q) = \sum_{j=l}^{u} u((i, j), T_q)$.
- Considering a Q-itemset $X = [x_1, x_2, \ldots, x_k]$, the utility of a Q-itemset X in transaction T_q, denoted as $u(X, T_q)$, is defined as the sum of utilities of Q-items of X in T_q: $u(X, T_q) = \sum_{i=1}^{k} u(x_i, T_q)$.
- The utility of the Q-itemset X in database D, denoted as $u(X)$, is determined as the sum of utilities of X in all transactions containing X, $u(X) = \sum_{T_q \in OCC(X)} u(X, T_q)$.

Definition 6: Transaction and Database Utility [6]

- The utility of a transaction $T_q = \{x_1, x_2, \ldots, x_k\}$, $T_q \in D$, denoted as $TU(T_q)$, is defined as the sum of utilities of all Q-items contained in T_q: $TU(T_q) = \sum_{i=1}^{k} u(x_i, T_q)$.
- The total utility of a database D, denoted as $\sigma(D)$, is determined as the sum of utilities of all transactions in D: $\sigma(D) = \sum_{i=1 \wedge T_i \in D}^{m} TU(T_i)$.

Definition 7: High Utility Quantitative Itemset (HUQI) [6]
Let D be a quantitative transaction database, a user-specified minimum utility threshold and μ, a Q-itemset X. X is called a high utility quantitative itemset if and only if its utility is no less than μ.

As the task of HUIQM is inherited from HUIM, the utility measure does not satisfy the downward-closure property. Thus, a proper utility-based upper bound is required to adapt the efficient pruning strategies from HUIM. The Transaction Weighted Utilization (TWU) upper bound is used to reduce the search space [9]. The base definition of TWU and the extended TWU definitions to be applied in HUIQM were described in detail in [6]. In addition, our work extends the original FHUQI-Miner algorithm. Thus all the *combining strategies* (Combine All, Combine Min, Combine Max), and *Q-itemset utility list* used in FHQUI-Miner are also employed [6]. The related definitions are also given discussed in [6]. Thus, to keep the paper compact, they will not be presented in this section. The core contribution of our work is to improve the effectiveness of the FHUQI-Miner algorithm; the related definitions are presented as follows.

Definition 8: Promising Q-itemsets [6]
Let X be a Q-itemset, a minimum utility threshold μ, and a quantitative related coefficient qrc ($qrc > 0$). X is considered a promising Q-itemset if $TWU(X) \geq \frac{\mu}{qrc}$. Otherwise, X is an unpromising Q-itemset.

Property 1: (Pruning Using TWU).
Let X be a Q-itemset, if X is an unpromising Q-itemset then X and all of its extensions are low utility itemsets and can be safely pruned from the search space [6].

Definition 9: TQCS Structure [6]
The TCQS structure is comprised of tuples in the form of $\langle a, b, c\rangle$, whereas a and b are the two Q-items that co-occurred in the database D; c are the TWU of the 2-Q-itemset constructed from a, b, $c = TWU(\{ab\})$. The concept of TQCS is mainly based on the EUCS structure proposed in the work of the FHM algorithm [6]. Differing from EUCS, the TQCS only keeps track of all the 2-Q-itemsets that really co-occur in the database D.

Property 2: (Exact Q-items Pruning – EQCPS). Given two Q-items x, y, a quantitative related coefficient qrc ($qrc > 0$), if there does not exist a tuple $\langle a, b, c\rangle$ such that $x = a, y = b$ and $c > \frac{\mu}{qrc}$, then the Q-itemset $\{xy\}$ is not a high utility Q-itemset as well as all of its extensions [6].

Property 3: (Range Q-items Pruning – RQCPS). Let $x = (i, l, u)$ be a range Q-item, and y be an exact Q-item, consider $x_i = (i, q)$ to be an exact Q-item within the range of (l, u) and is extracted from x. Then, if $\sum_{i=l}^{u} c_i < \frac{\mu}{qrc}$, the Q-itemset $\{xy\}$ and all of its extensions are not high utility Q-itemset [6].

4 Proposed Approach

Our goal is to enhance the effectiveness of the FHUQI-Miner algorithm based on a thorough analysis of the TQCS structure [5]. This strategy is named the Improved TQCS Strategy. As presented in the previous section, the underestimated property points toward the fact that both exact Q-items and range Q-items could be represented in a similar fashion via the format (i, l, u), where i is the identifier of that item in a given transactional database, l and u sequentially acting as the lower and upper bounds to the range of quantities that item will exist across the transactions of that database. Re-using the example transactional database given in Table 1, any randomly picked exact Q-item in those transactions could be represented as a range Q-item. $(i, 9)$ and $(f, 1)$ are by exact nature Q-items, but they can also be viewed as range Q-items under the form of $(i, 9, 9)$ and $(f, 1, 1)$, as their lower and upper bounds are basically the same, and the Q-interval of the "range" is simply 0.

Algorithm 1. iFHUQI-Miner

Input: D – quantitative transaction database; μ – minimum utility threshold; CM – combining method; qrc – quantitative related coefficient

Output: Complete set of HUQIs.

1: Scans D to calculate the TWU of each item.
2: Creates set of promising Q-items P^*: $\forall x \in P^*: TWU(X) \geq \frac{\mu}{qrc}$
3: Scans D to sort transactions in the decreasing order of utility $\prec$; create the utility list of promising items $UL(P^*)$; build the TQCS.
4: **FOREACH** $x \in P^*$ **DO**
5: **IF** $UL(x).SumEutil \geq \mu$ **THEN**
6: $H = H \cup x$; Output x
7: **ENDIF**
8: **ELSE**
9: **IF** $UL(X).SumEutil + UL(X).SumRutil \geq \mu$ **THEN**
10: $E = E \cup x$
11: **ENDIF**
12: **IF** $\frac{\mu}{qrc} \leq UL(X).SumEutil \leq \mu$ THEN
13: $C = C \cup x$
14: **ENDIF**
15: **ENDELSE**
16: **ENDFOR**
17: Discover high utility range Q-itemset (HR) using CM and C
18: $QI \leftarrow sort(H \cup E \cup HR)$ using the $\prec$ order
19: **Improved_DFS_Mining**$(\emptyset, QI, UL(QI), P^*, qrc, CM, \mu)$

This change in the notation of any Q-item can easily make a difference through implementation, enabling the possibility of uniting the strategies that Nouioua et al. [5] initially and distinctively proposed. It is the default for every Q-item to contain both lower and upper bounds, and that exact Q-items to keep the same quantities for these variables. The detail of our proposal, the iFHUQI-Miner algorithm, is presented in Algorithm 1. The algorithm's pseudo-code is no different from the original FHUQI-Miner. It is shown here for the reader easier to catch up on the changes presented in Algorithm 2, which is the search space exploration phase.

Please note that in Algorithm 1, the set H contains all the HUQIs, the set C contains all the candidates Q-itemsets that can be combined to form high utility range Q-itemsets, the set E contains all Q-itemsets that should be explored further to discover possible HUQIs from the extensions, and the set HR contains all the high utility range Q-itemsets obtained by combining Q-items in C. The last line of Algorithm 1 invokes the recursive DFS-based function to scan the search space, the `Improved_DFS_Mining` function. This is where the *Improved TQCS Strategy* will be applied with respect to the previous assimilation of exact Q-items notation to range Q-items notation.

Comparing the original `Recursive_Mining_Search` in [6] versus the *Improved TQCS Strategy* presented here, the prominent distinction surfaces right at the extension traversal step when it scans through the set of promising Q-items. For the original FHUQI-Miner algorithm, it will explore all possible combination of Q-itemsets of the form $[Pxy]$, and prune away any combinations that are low-utility using the two pruning strategies. This approach, described from *line 9 to line 15* in the pseudo-code of the `Recursive_Mining_Search` [6], implies the different treatment between exact Q-items and range Q-items.

The strategy used in FHUQI-Miner can be summarized as follows.

- When both x and y are exact Q-items, the original FHUQI-Miner algorithm will compose the TQCS structure $\langle x, y, c\rangle$ and evaluate the TWU of between x and y. If the TWU does not satisfy the $\frac{\mu}{qrc}$ threshold, the next extension will be checked. This is the Exact Q-items Co-occurrence Pruning Strategy.
- When x is a range Q-item, the algorithm considers each exact Q-item x_i of $[Px]$ across all transactions and sums up all c values gathered under $\langle x_i, y, c\rangle$. If the sum is less than $\frac{\mu}{qrc}$, then the Q-itemset and its extensions will be dropped immediately. This is the Range Q-items Cooccurrence Pruning Strategy.

On the contrary, the proposed iFHUQI-Miner algorithm follows the principle of similar notation and shortens the validation stage to only *line 4* to *line 9*. In other words, this will treat range Q-items and exact Q-items in the same manner and unanimously apply the notation (i, l, u) to all items regardless of which side of the spectrum the current item falls under. In other words, the exact Q-items are now considered as a subset of the range Q-items, and this is done in line 4 of Algorithm 2. For example, using the transactional database D given in Table 1, any exact Q-item in D can be represented as a range Q-item. Such as $(a, 2)$ or $(e, 1)$ are exact Q-items by nature. But these exact Q-items can be represented as range Q-items of $(a, 2, 2)$ and $(e, 1, 1)$ with their ranges equal to zero.

Algorithm 2. Improved_DFS_Mining
Input: P — prefix Q-itemset, QI — Q-itemset list, $UL(QI)$ — utility list of Q-itemsets, P^* - promising Q-itemsets, qrc — quantitative related coefficient, CM — combining method, μ — predefined minimum utility threshold
Output: the set of HUQIs with respect to prefix P.

1: **FOREACH** $[Px]$ such that $x \in QI$ **DO**
2: $QI_s \leftarrow \emptyset$; $P_s^* \leftarrow \emptyset$
3: **FOREACH** $[Py]$ such that $y \in QI$ **AND** $x \prec y$ **DO**
4: **FOREACH** exact Q-item $x_i \in [Px]$ **DO**
5: $c \leftarrow \sum_{i=l}^{u} TQCS(x_i, y)$
6: **IF** $c == null$ **OR** $c \leq \frac{\mu}{qrc}$ **THEN**
7: next $[Py]$
8: **ENDIF**
9: **ENDFOR**
10: $Z \leftarrow [Pxy]$; $UL(Z) \leftarrow$ Construct(x, y, P)
11: **IF** $UL(Z) \neq null$ **AND** $TWU(Z) \geq \frac{\mu}{qrc}$ **THEN**
12: $P_s^* \leftarrow P_s^* \cup Z$
13: **IF** $UL(Z).SumEutil \geq \mu$ **THEN**
14: $H_s \leftarrow H_s \cup Z$; Output Z
15: **ENDIF**
16: **ELSE**
17: **IF** $UL(Z).SumEutil + UL(Z).SumRutil \geq \mu$ **THEN**
18: $E_s \leftarrow E_s \cup Z$
19: **ENDIF**
20: **IF** $\frac{\mu}{qrc} \leq UL(Z).SumEutil \leq \mu$ **THEN**
21: $C_s \leftarrow C_s \cup Z$
22: **ENDIF**
23: **ENDELSE**
24: **ENDIF**
25: **ENDFOR**
26: Discover high utility range Q-item (HR_s) using CM and C_s
27: $QI_s \leftarrow sort(H_s \cup E_s \cup HR_s)$ using $\prec$ order
28: **Improved_DFS_Mining**$(Px, QI_s, UL(QI_s), P_s^*, qrc, CM, \mu)$
29: **ENDFOR**

With this approach, the proposed algorithm will still be able to maintain the resources delegated to validate both exact Q-item and range Q-item as before and make the real implementation more manageable. In the case of exact Q-items because both boundaries will be referring to the same quantity, every item will still have only one tuple $\langle x, y, c \rangle$ created for it and validated with the same principles. As for the case of range Q-items, the procedure of retrieving the exact component Q-items within the range and validating their $\langle x_i, y, c \rangle$ tuples will also stay untouched. With this approach, the implementation of the first proposed algorithm will be improved compared to the original algorithm as it reduces the number of conditions required to validate before the mining process is

performed any further. Therefore, although the idea might appear very basic, it can still surprisingly lead the iFHUQI-Miner algorithm to better overall performance.

Adopting the *Improved TQCS Strategy* into the FHUQI-Miner does not reduce the theoretical complexity, especially in the worst case. The pruning phase is now a straightforward process with both exact Q-items and range Q-items considered using the same mechanism. However, the *Improved TQCS Strategy* does reduce the number of branching and joins that need to be carried out while pruning. Thus, the complexity of the iFHUQI-Miner remains the same as FHUQI-Miner [6]. The improvement in terms of execution time and join counts can be observed in the experimental evaluation section of this work.

5 Evaluation Studies

To assert the proposed algorithm's performance, a series of experiments on different datasets were conducted. All the datasets are acquired directly from the SPFM library [16]. These four datasets were selected as they are used in several HUIM kinds of literature to evaluate the mining performance. Besides, since the FHUQI-Miner miner algorithm was evaluated using these datasets, it would provide fair comparisons on how the iFHUQI-Miner improved the mining performance. The information regarding all the datasets is summarized in Table 3. All the experiments were carried out on a computer with a fifth generation, 64-bit Intel® Core® i5-5257U processor, running macOS® Monterey has 8.0GB of RAM. Both algorithms are implemented in Java. Thus, Java API is used to measure the execution time as well as memory usage of the evaluated algorithms.

Table 3. Datasets characteristics

Dataset	$\|D\|$	$\|I\|$	$Trans_{AVG}$	Density	Type
BMS1	59,061	497	2.42	0.51%	Sparse
BMS2	77,512	3,340	4.62	0.14%	Sparse
Retail	88,162	16,470	10.30	0.06%	Sparse
Connect	67,557	129	43	33.33%	Dense

In all experiments, the performance of the proposed iFHUQI-Miner (with Improved TQCS Strategy) was directly compared with the original FHUQI-Miner across the three combining methods. A varying set of minimum utility thresholds μ is also selected and applied to all the runs for all methods. The obtained results of runtime, memory usage and number of joins can be viewed in Fig. 1, Fig. 2, and Fig. 3, respectively.

It can be seen in all the evaluated results throughout all combination methods, the iFHUQI-Miner has a better performance compared to the original algorithm. On the sparse datasets such as Retail (Fig. 1a), BMS1 (Fig. 1c), BMS2 (Fig. 1d), the mining time of iFHUQI-Miner reduced by up to 37% those of FHUQI-Miner, thanks to the Improved TCQS Strategy. In addition, this strategy also helps cut down the number of

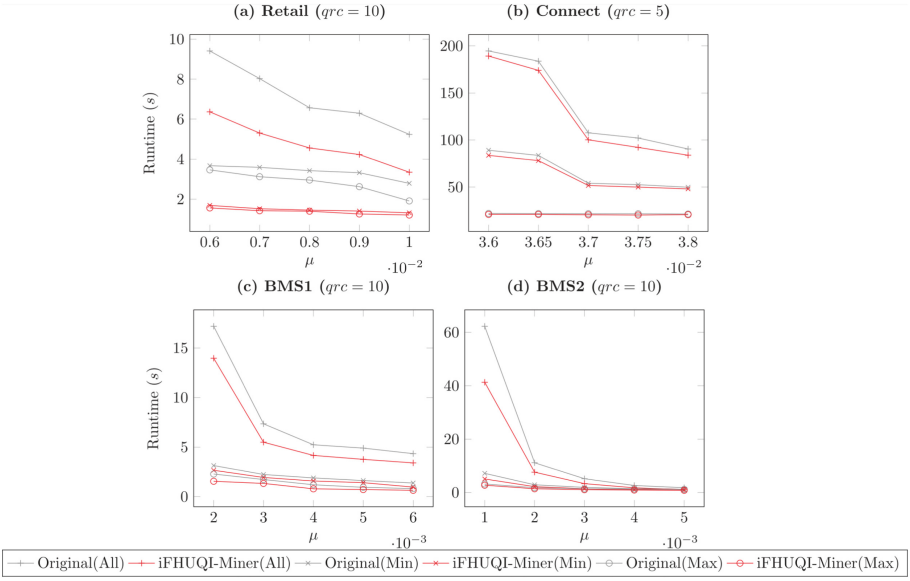

Fig. 1. Runtime comparisons of FHUQI-Miner and iFHUQI-Miner

joins operations needed (Fig. 3a, Fig. 3c and Fig. 3d) and thus lowers the memory usage (Fig. 2a, Fig. 2c and Fig. 2d). For the dense dataset Connect, the improved performance is also observed on all combination methods; however, it is not as high as in the sparse datasets. The execution time improvements are only up to 10% faster (Fig. 1b). As the Improve TQCS Strategy can only reduce a small number of join operations needed (Fig. 3b). This is also reflected in the memory usage of the Connect dataset (Fig. 2b).

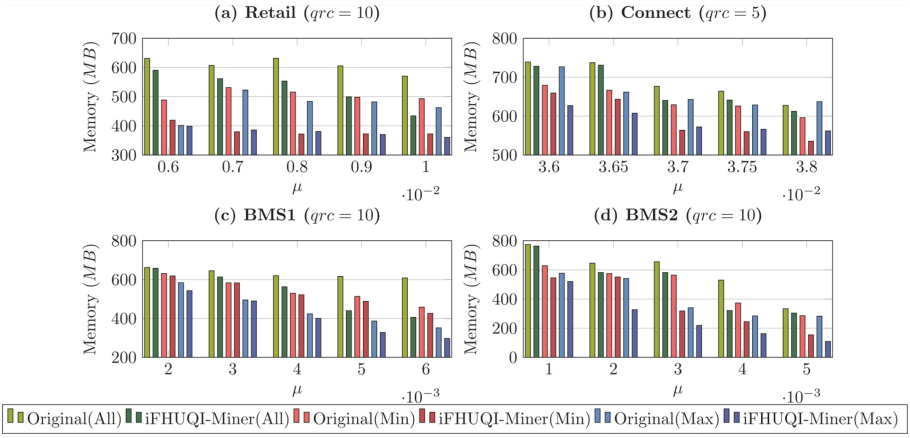

Fig. 2. Memory usage comparisons of FHUQI-Miner and iFHUQI-Miner

Generally, the iFHUQI-Miner algorithm has the fastest execution time among all of the experiments that were involved in this research. By overlooking the difference between exact Q-items and range Q-items as well as other restraints that were implemented in the original algorithm, with careful attention to a basic concept and an efficient execution of it in the real implementation of the algorithm, the first proposed algorithm was still able to apply the same evaluation method with TWU to all Q-items and eliminate low-utility Q-items beforehand, therefore also reducing the number of cases when the mining process is performed. All of the HUQIs generated by all algorithms were eventually compared with each other. This algorithm was eventually guaranteed to generate the same high utility itemsets but also eliminate a considerable amount of necessary join operations, as seen in the performance of the original algorithm.

6 Conclusion and Future Works

With respect to the works of Nouioua et al. on the FHUQI-Miner algorithm as the foundation, this work has presented an enhanced version of the algorithm, namely iFHUQI-Miner, equipped with the *Improved TCQS Strategy*, and has performed the necessary experiments to measure their performance against the original version. Results have shown that the iFHUQI-Miner, by considering exact Q-items and range Q-items as the same, is more efficient than the original FHUQI-Miner in pruning unpromising Q-items during the mining process, and thus reduces both the execution time and the number of join operations required for it to achieve the same output.

Besides the contribution of this work to enhance the mining performance of the quantitative high utility itemset, there are still is still for further improvements, such as: studying for a better upper bound could result in tighter search space and better execution time; adding new optimization to speed up the utility list construction and the combining methods; applying parallel computing methods onto the algorithm to make the algorithm scales better with the system it operates on.

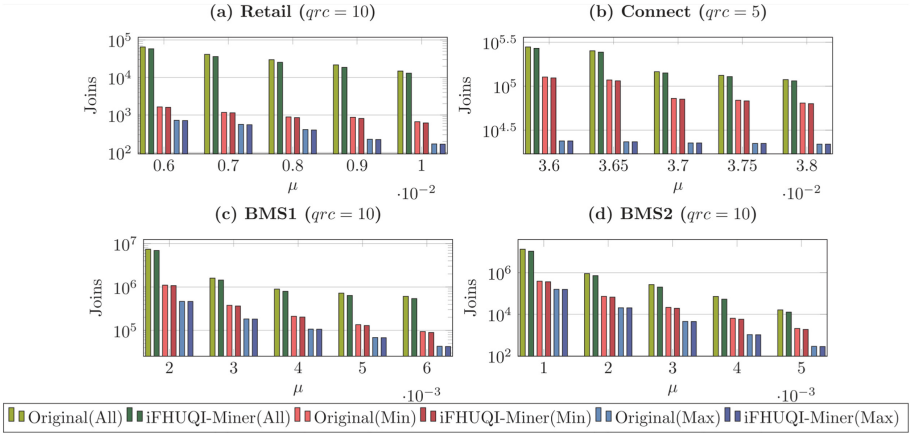

Fig. 3. Join counts comparisons of FHUQI-Miner and iFHUQI-Miner

References

1. Yao, H., Hamilton, H.J.: Mining itemset utilities from transaction databases. Data Knowl. Eng. **59**(3), 603–626 (2006)
2. 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)
3. Fournier-Viger, P., Chun-Wei Lin, J., Truong-Chi, T., Nkambou, R.: A survey of high utility itemset mining. In: Fournier-Viger, P., Lin, J.C.-W., Nkambou, R., Vo, B., Tseng, V.S. (eds.) High-Utility Pattern Mining. SBD, vol. 51, pp. 1–45. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-04921-8_1
4. Gan, W., Lin, J.C.-W., Fournier-Viger, P., Chao, H.-C., Tseng, V.S., Yu, P.S.: A survey of utility-oriented pattern mining. IEEE Trans. Knowl. Data Eng. **33**(4), 1306–1327 (2021)
5. Li, C.-H., Wu, C.-W., Huang, J., Tseng, V.S.: An efficient algorithm for mining high utility quantitative itemsets. In: 2019 International Conference on Data Mining Workshops (ICDMW), pp. 1005–1012 (2019)
6. Nouioua, M., Fournier-Viger, P., Wu, C.-W., Lin, J.C.-W., Gan, W.: FHUQI-miner: fast high utility quantitative itemset mining. Appl. Intell. (2021)
7. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: 20th International Conference on Very Large Data Bases (VLDB 1994), pp. 487–499 (1994)
8. Yao, H., Hamilton, H.J., Butz, G.J.: A foundational approach to mining itemset utilities from databases. In: SIAM International Conference on Data Mining, vol. 4, pp. 482–486 (2004)
9. Liu, Y., Liao, W.K., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, vol. 3518, pp. 689–695 (2005)
10. 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)
11. Le, B., Nguyen, H., Cao, T.A., Vo, B.: A novel algorithm for mining high utility itemsets. In: First Asian Conference on Intelligent Information and Database Systems, pp. 13–17 (2009)
12. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: ACM International Conference on Information and Knowledge Management, CIKM, pp. 55–64 (2012)
13. Zida, S., Fournier-Viger, P., Wu, C.-W., Lin, J.C.-W., Tseng, V.S.: Efficient mining of high-utility sequential rules. In: Machine Learning and Data Mining in Pattern Recognition, pp. 157–171 (2015)
14. Nguyen, L.T.T., Nguyen, P., Nguyen, T.D.D., Vo, B., Fournier-Viger, P., Tseng, V.S.: Mining high-utility itemsets in dynamic profit databases. Knowl.-Based Syst. **175**, 130–144 (2019)
15. Yen, S.-J., Lee, Y.-S.: Mining high utility quantitative association rules. In: Data Warehousing and Knowledge Discovery, pp. 283–292 (2007)
16. Fournier-Viger, P., et al.: The SPMF open-source data mining library version 2. In: Berendt, B., et al. (eds.) ECML PKDD 2016. LNCS (LNAI), vol. 9853, pp. 36–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46131-1_8