2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) | 978-1-6654-5661-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASONAM55673.2022.10068640

Social Network Analysis of Popular YouTube Videos via Vertical Quantitative Mining

Adam G.M. Pazdor, Carson K. Leung(✉), Thomas J. Czubryt, Junyi Lu, Denys Popov, Sanskar Raval
Department of Computer Science, *University of Manitoba*
Winnipeg, MB, Canada
Carson.Leung@UManitoba.ca

***Abstract*—Frequent itemset (or frequent pattern) mining is a technique used in big data mining to discover frequently occurring sets of items (such as popular co-purchased merchandise) and has numerous applications in the field of databases. Traditional frequent pattern mining algorithms only look at Boolean mining; that is, considering only the presence or absence of an item in an itemset. In this paper, we present an algorithm for mining interesting quantitative frequent patterns. Our qEclat (or Q-Eclat) algorithm extends the common Eclat algorithm to be able to vertically mine quantitative patterns. When compared with the existing MQA-M algorithm (which was built for quantitative horizontal frequent pattern mining), our evaluation results show that qEclat mines quantitative frequent patterns faster.**

***Index Terms*—Database engineered application, Frequent pattern mining, Quantitative data mining, Vertical pattern mining, Eclat, Transaction ID, Set**

I. Introduction

In modern times, big data [1, 2] can be found everywhere. With advances in technology, high volumes of a wide variety of data (which may be of different levels of varsity) are generated and collected at a high velocity for numerous real-life applications and services. Embedded in these big data is implicit, previously unknown and potentially useful information and knowledge. This calls for big data management [3], big data mining [4], big data analytics [5, 6], as well as big data visualization and analytics [7, 8].

Association rule mining and *Frequent pattern mining* [9, 10] are popular techniques for big data mining. They seek to discover frequent patterns which exist in a database and then to use those patterns to find interesting association rules, which state that whenever a certain set of items occurs in a transaction, another set of items tends to occur in that transaction. These techniques form the basis of many real-life applications such as marketing in business, discovering biological patterns, studying human populations, and many database engineered applications. Frequent pattern mining has been extended to the mining of other patterns such as network and graph mining [11, 12], stream mining [13, 14], uncertain pattern mining [15, 16], and utility pattern mining [17].

Frequent patterns can be discovered horizontally by transaction-centric mining algorithms or vertically by item-centric mining algorithms [18–20]. The Eclat (Equivalence CLAss Transformation) algorithm [19] is an example of a vertical item-centric frequent pattern mining algorithms. Each domain item is represented by an Equivalence Class, according to its prefix, and the corresponding transaction ID set for an item captures which transactions contain the specific item. One advantage of this method of representation is that the size of the sets are proportional to the density of the data, making them highly efficient for sparse data.

Traditional mining algorithms (both horizontal and vertical) aim to discover Boolean frequent patterns; that is, patterns which only capture the presence or absence of an item. This is a significant limitation, as it does not allow us to capture the case where a transaction can contain more than one instance of an item. For example, in a grocery store, we can only determine if a person has bought apples, not how many. Or, in a social network context, we can determine that a person commented on a video, but not how many times. To address this shortcoming, the notion of quantitative association rule mining or quantitative frequent pattern mining [21, 22] was introduced, which is an extension of frequent pattern mining to allow transactions to contain an item more than once. The MQA-M algorithm [21] extends the Apriori algorithm for mining quantitative association rules with multiple comparison operators—i.e., mining *quantitative frequent patterns* (aka sets of item expressions, *itemexpsets* for short)—horizontally.

We present in this paper an equivalence class-based algorithm to mine quantitative frequent patterns (i.e., itemexpsets) vertically. Our qEclat algorithm represents the data as a collection of bitmaps; each transaction set captures the IDs of transactions containing a given item, as well as the quantity of that item in each transaction. Our algorithm then vertically mines the quantitative frequent patterns. When compared with the existing MQA-M algorithm (which was built for quantitative frequent pattern mining), evaluation results show that our algorithm requires shorter execution time to mine frequent patterns. Our key contributions in this paper include our qEclat algorithm and its pruning rules.

We organize the remainder of this paper as follows. In Section II, we discuss the MQA-M algorithm, which we are using as a comparison. We formally introduce our qEclat algorithm in Section III, including an example using YouTube videos. Section IV contains analysis of the algorithm and evaluation to compare qEclat with MQA-M. Finally, we conclude in Section V.

IEEE/ACM ASONAM 2022, November 10-13, 2022
978-1-6654-5661-6/22/$31.00 © 2022 IEEE

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on December 11,2024 at 08:46:18 UTC from IEEE Xplore. Restrictions apply.

II. Background and Related Works

Here, we discuss MQA-M, the algorithm to which we are comparing our qEclat algorithm. Discussions of other mining algorithms have been omitted to conserve space.

A. The MQA-M Algorithm for Quantitative Mining

The MQA-M (Mining Quantitative Association rules with Multiple comparison operators) [21] is similar to the Apriori algorithm except that it is generalized to handle quantitative transaction databases. For any $k \in \mathbb{Z}^+$, let C_k be the set of candidate itemexpsets containing k itemexps, and let L_k be the set of frequent itemexpsets containing k itemexps. Like in the Apriori algorithm, $L_k \subseteq C_k$. The MQA-M algorithm starts by generating C_1. Suppose that $itemmax[p]$ represents the maximum number of times an item p appears in a transaction.

Example 1. *If a quantitative database consists of transactions* $t_1 = \{(a, 1)\}$ *and* $t_2 = \{(a, 3)\}$*, then* $itemmax[a] = 3$ *because* a *appears in a transaction at most 3 times.*

Afterwards, for each item p appearing in the quantitative transaction database, add every itemexpset of the form $\{(p, \theta, q)\}$ to C_1, where $\theta \in \{=, \geq, \leq\}$ and $q \in \{1, \ldots, itemmax[p]\}$. The algorithm computes the support of each itemexpset in C_1 by iterating through the transactions and incrementing the support of an itemexpset in C_1 if the transaction satisfies the itemexpset. Let $k = 1$. Then, L_1 becomes the set of all itemexpsets with a support $\geq minsup$. The algorithm removes some itemexpsets from L_1 by using two pruning rules [21]:

1) Suppose X contains an itemexp of the form $(z \leq r)$, where z is an item and $r \in \mathbb{Z}^+$. If itemexpset $Y \in L_k$ has the same support as X except $(z \leq r)$ is replaced by $(z \leq r + 1)$, then Y can be pruned from L_k.
2) Suppose X contains an itemexp of the form $(z \geq r)$, where z is an item and $r \in \mathbb{Z}^+$. If itemexpset $Y \in L_k$ has the same support as X except $(z \geq r)$ is replaced by $(z \geq r - 1)$, then Y can be pruned from L_k.

Like the Apriori algorithm, the MQA-M has a main loop, beginning with $k = 2$. The loop body begins with generating C_k from L_{k-1}. C_k is initially generated using a self-join on L_{k-1}. If two itemexpsets in L_{k-1} have the same first $(k - 2)$ itemexps, then it generates an itemexpset in C_k consisting of those (k-2) itemexps and the last itemexp in the two itemexpsets in L_{k-1}. However, it imposes an additional restriction that it does not create an itemexpset in C_k where there are two itemexps referring to the same item. After the join step, it prunes itemexpsets from C_k with a subset containing $(k-1)$ itemexps where that subset is not in L_{k-1}. It gets L_k from C_k using the same procedure that was used to obtain L_1. It uses the two aforementioned pruning rules to removes some itemsets from L_k. At the end of the loop body, it increments k and repeats the previous steps (if necessary). The loop terminates when L_{k-1} is empty. Afterwards, it returns $\bigcup_k L_k$, which contains all the interesting frequent itemexpsets.

Example 2. *Although* L_1 *contains* $\{(a = 1)\}$ *and* $\{(a \geq 2)\}$*, MQA-M does not form* $\{(a = 1), (a \geq 2)\} \in C_2$.

B. Vertical Representation of Quantitative Data

To represent quantitative transaction databases in a vertical format, for each item that occurs in the transaction database, we store it as a set of pairs. Each pair contains a transaction ID associated with that item and the number of occurrences of the item in the transaction. Since we are storing a pair, we can call these sets "pairsets".

Example 3. *A horizontal database containing two transactions* $t_1 = \{(a, 1)\}$ *and* $t_2 = \{(a, 3)\}$ *can be represented vertically using* $pairset(a) = \{(t_1, 1), (t_2, 3)\}$.

For quantitative association rule mining, we define $tidset(X)$ of any itemexpset X to be the set of transaction IDs corresponding to transactions which satisfy X. When X is an itemexpset containing at least two itemexps, we can break down $X = W \cup \{y\} \cup \{z\}$ where (a) W is an itemexpset with two fewer elements than X and (b) y and z are itemexps. We have the recursive equation: $tidset(X) = tidset(W \cup \{y\}) \cap tidset(W \cup \{z\})$. We will use this equation to generate tidsets for itemexpsets containing at least two elements when running our Q-Eclat algorithm. The support of an itemexpset X can be computed by counting the number of elements in its tidset, i.e., $sup(X) = |tidset(X)|$.

III. Vertical Quantitative Frequent Pattern Mining with Our Q-Eclat Algorithm

In this section, we describe our novel qEclat algorithm and provide an illustrative example using YouTube videos.

A. Q-Eclat Algorithm

For any integer $k \geq 1$, define C_k to be the set of candidate k-itemexpsets and L_k to be the set of frequent k-itemexpsets. First, we convert the quantitative transaction database into a vertical format if it is in its horizontal format. The next step of our algorithm is to compute all candidate 1-itemexpsets in C_1. Each of those itemexpsets consists of a single itemexp of the form $(item, operation, quantity)$ where *item* is an item in the transaction database, *operator* $\theta \in \{=, \geq, \leq\}$, and *quantity* $q \in \{1, \ldots, itemmax[item]\}$. We compute *itemmax*[*item*] as the maximum number of times an item appears in a transaction over all transactions in the transaction database.

After computing C_1, we compute the tidsets associated with each candidate 1-itemexpset. The tidsets can be computed from the vertical representation of the quantitative transaction database. We then compute the support of each candidate 1-itemexpset by counting the elements in its corresponding tidset. The frequent 1-itemexpsets are candidate 1-itemexpsets having a support $\geq$ *minsup*. Finally, we remove some itemexpsets from L_1 based on our two *new pruning rules*, which will be described in Section III-B.

Then, we set $k = 2$ and begin executing the main loop. The first step in the main loop body is to generate C_k using L_{k-1}. We initially create C_k by performing a self-join on L_{k-1}. If

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on December 11,2024 at 08:46:18 UTC from IEEE Xplore. Restrictions apply.

there are two frequent $(k-1)$-itemexpsets in L_{k-1} where their first $(k-2)$-itemexps are the same and their last itemexp refer to different items, then we add to C_k a candidate k-itemexpset that consists of the first $(k-2)$-itemexps and the last itemexp of both itemexpsets. Afterwards, we prune any candidate k-itemexpset from C_k that contains a sub-itemexpset with $(k-1)$-itemexps that do not belong to L_{k-1}. The next step is to create tidsets corresponding to every candidate k-itemexpset in C_k. This can be done using the recursive definition for tidsets:

$$tidset(X) = tidset(W \cup \{y\}) \cap tidset(W \cup \{z\}) \quad (1)$$

After computing the tidsets, we compute the support of each candidate k-itemexpset in C_k. Any candidate k-itemexpset having a support $\geq minsup$ is added to L_k. Using the two pruning rules, we remove some uninteresting itemexpsets from L_k if necessary. After pruning L_k, we have reached the end of the loop body. Hence, we increment k and repeat the main steps again if necessary. The main loop stops running once L_k is empty. Our Q-Eclat algorithm returns $\bigcup_k L_k$, which contains all interesting frequent itemexpsets.

B. Our New Pruning Rules for Q-Eclat Algorithm

The MQA-M algorithm contains two pruning rules to remove unnecessary itemexpsets from L_k where integer $k \geq 1$. We present two new pruning rules that are more general, and thus remove additional itemexpsets. Our pruning rules are described as follows (where X and Y are itemexpsets in L_k, z is an item, and r and $s \in \mathbb{Z}^+$):

1) Suppose X contains an itemexp of the form $(z \leq r)$. If Y has the same support as X except $(z \leq r)$ is replaced by $(z \leq r+s)$, then Y can be pruned from L_k.
2) Suppose X contains an itemexp of the form $(z \geq r)$,If Y has the same support as X except $(z \geq r)$ is replaced by $(z \geq r-s)$, then Y can be pruned from L_k.

A key difference between the original pruning rules and our new pruning rules is that the new rules can handle differences in quantity greater than 1. Instead of considering itemexpsets of the form $(z \leq r+1)$ or $(z \geq r-1)$, we consider the more general cases of $(z \leq r+s)$ or $(z \geq r-s)$ for some positive integer s. As a result, these rules eliminate at least as many itemexpsets from L_k as the original pruning rules. As observed from Example 4, our improved pruning rules are more powerful in removing redundant frequent itemexpsets.

Example 4. *Suppose that L_2 contains two frequent 2-itemexpsets $\{(a \geq 2), (b = 1)\}$ and $\{(a \geq 4), (b = 1)\}$ before pruning and they have the same support. Using the original pruning rules used in MQA-M, neither itemexpset would be pruned. In contrast, when using our new pruning rules, $\{(a \geq 2), (b = 1)\}$ would be pruned from L_2.*

C. Youtube Video Example

Consider the example of a database tracking comments on popular YouTube videos. It is useful to know not just that people are leaving comments but how many comments a particular person is leaving, as multiple comments left on a single video is a sign of higher engagement.

Here we will represent each video as a transaction (e.g., t_1) and each commenter as a letter (e.g., a), where the quantity is the number of comments left on a video by that person.

Suppose we set *minsup*=1 and we have three transactions in a quantitative transaction database: $t_1 = \{a:2\}$, $t_2 = \{a:4, b:1\}$, and $t_3 = \{b:1\}$. Then, we observe the occurrences of each domain item in the transaction (that is, the number of comments on each video) and compute its itemmax:

- $itemmax[a] = 4$ because the highest number of occurrences of a in a transaction is 4 (in transaction t_2), and
- $itemmax[b] = 1$ because the highest number of occurrences of b in a transaction is 1 (in both t_2 and t_3).

To generate C_1, we must generate every combination of an item (that is, commenter), comparison operation, and quantity. There are two commenters (i.e., a and b) and three operators (i.e., $=, \geq$ and $\leq$). For commenter a, there are four quantity values (i.e., from 1 to $itemmax[a] = 4$). This leads to a total of $1 \times 3 \times 4 = 12$ candidate itemexpsets from commenter a. Similarly, the one quantity value (due to $itemmax[b] = 1$) leads to a total of $1 \times 3 \times 1 = 3$ candidate itemexpsets from commenter b. Consequently, this leads to a total of $12+3 = 15$ candidate itemexpsets as shown in the first column of Table I.

For each itemexpset in C_1, we compute its corresponding tidsets. The support of those itemexpsets is equal to the number of elements (i.e., transaction IDs) in the tidset. We present the itemexpsets X in C_1, their tidsets, and their supports in Table I.

TABLE I
CANDIDATE AND FREQUENT 1-ITEMEXPSETS

$X \in C_1$	$tidset(X)$	$sup(X)$	$\geq$ minsup	interesting
$\{(a=1)\}$	$\emptyset$	0		
$\{(a=2)\}$	$\{t_1\}$	1	$\surd$	$\surd$
$\{(a=3)\}$	$\emptyset$	0		
$\{(a=4)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a \geq 1)\}$	$\{t_1, t_2\}$	2	$\surd$	
$\{(a \geq 2)\}$	$\{t_1, t_2\}$	2	$\surd$	$\surd$
$\{(a \geq 3)\}$	$\{t_2\}$	1	$\surd$	
$\{(a \geq 4)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a \leq 1)\}$	$\emptyset$	0		
$\{(a \leq 2)\}$	$\{t_1\}$	1	$\surd$	$\surd$
$\{(a \leq 3)\}$	$\{t_1\}$	1	$\surd$	
$\{(a \leq 4)\}$	$\{t_1, t_2\}$	2	$\surd$	$\surd$
$\{(b=1)\}$	$\{t_2, t_3\}$	2	$\surd$	$\surd$
$\{(b \geq 1)\}$	$\{t_2, t_3\}$	2	$\surd$	$\surd$
$\{(b \leq 1)\}$	$\{t_2, t_3\}$	2	$\surd$	$\surd$

Among these 15 candidate 1-itemexpsets, only 12 of them satisfy $minsup = 1$. We obtain initial L_1 by only keeping these 12 candidate 1-itemexpsets having support $\geq minsup$. They are listed on the fourth column of Table I.

Then, by using our pruning rules described in Section III-B, we further prune away three more redundant 1-itemexpsets:

- Prune $\{(a \geq 1)\}$ by Pruning Rule (2) because $\{(a \geq 1)\} \in L_1$ and $\{(a \geq 2)\}$ have the same support;
- Prune $\{(a \geq 3)\}$ by Pruning Rule (2) because $\{(a \geq 3)\} \in L_1$ and $\{(a \geq 4)\}$ have the same support; and

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on December 11,2024 at 08:46:18 UTC from IEEE Xplore. Restrictions apply.

TABLE II
CANDIDATE AND FREQUENT 2-ITEMEXPSETS

$X \in C_2$	$tidset(X)$	$sup(X)$	$\geq$ minsup	interesting
$\{(a=2),(b=1)\}$	$\emptyset$	0		
$\{(a=2),(b\geq 1)\}$	$\emptyset$	0		
$\{(a=2),(b\leq 1)\}$	$\emptyset$	0		
$\{(a=4),(b=1)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a=4),(b\geq 1)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a=4),(b\leq 1)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a\geq 2),(b=1)\}$	$\{t_2\}$	1	$\surd$	
$\{(a\geq 2),(b\geq 1)\}$	$\{t_2\}$	1	$\surd$	
$\{(a\geq 2),(b\leq 1)\}$	$\{t_2\}$	1	$\surd$	
$\{(a\geq 4),(b=1)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a\geq 4),(b\geq 1)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a\geq 4),(b\leq 1)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a\leq 2),(b=1)\}$	$\emptyset$	0		
$\{(a\leq 2),(b\geq 1)\}$	$\emptyset$	0		
$\{(a\leq 2),(b\leq 1)\}$	$\emptyset$	0		
$\{(a\leq 4),(b=1)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a\leq 4),(b\geq 1)\}$	$\{t_2\}$	1	$\surd$	$\surd$
$\{(a\leq 4),(b\leq 1)\}$	$\{t_2\}$	1	$\surd$	$\surd$

- Prune $\{(a \leq 3)\}$ by Pruning Rule (1) because $\{(a \leq 2)\} \in L_1$ and $\{(a \leq 3)\}$ have the same support.

Hence, the final L_1 ends up with only $12-3=9$ frequent but not redundant 1-itemexpsets: $\{(a=2)\}$, $\{(a=4)\}$, $\{(a \geq 2)\}$, $\{(a \geq 4)\}$, $\{(a \leq 2)\}$, $\{(a \leq 4)\}$, $\{(b=1)\}$, $\{(b \geq 1)\}$,$\{(b \leq 1)\}$.

Afterwards, the main loop is executed with $k=2$. We begin with the generation of C_2. The first step in generating C_2 is to perform a self-join on L_1. In this scenario, this means getting pairs of itemexps where the itemexps refer to different items. This yields $6 \times 3 = 18$ different candidate 2-itemexpsets in C_2 as shown in Table II.

Among these 18 candidate 2-itemexpsets, only 12 of them satisfy $minsup = 1$. We obtain the initial L_2 by only keeping these 12 candidate 2-itemexpsets having support $\geq minsup$. They are listed on the fourth column of Table II.

Note that none of these 12 itemexpsets in the initial L_2 can be pruned by the *original* pruning rules used in the MQA-M algorithm. Recall from Example 4, although both $\{(a \geq 2),(b=1)\} \in L_2$ and $\{(a \geq 4),(b=1)\} \in L_2$ have the same support, $\{(a \geq 2),(b=1)\}$ would not be pruned. However, by using our pruning rules described in Section III-B, we can further prune away three more redundant 2-itemexpsets:

- Prune 2-itemexpset $\{(a \geq 2),(b=1)\}$ by our new Pruning Rule (2) because both $\{(a \geq 2),(b=1)\} \in L_2$ and $\{(a \geq 4),(b=1)\} \in L_2$ have the same support;
- Prune 1-itemexpset $\{(a \geq 2),(b \geq 1)\}$ by our new Pruning Rule (2) because both $\{(a \geq 2),(b \geq 1)\} \in L_2$ and $\{(a \geq 4),(b \geq 1)\} \in L_2$ have the same support; and
- Prune 1-itemexpset $\{(a \geq 2),(b \leq 1)\}$ by our new Pruning Rule (2) because both $\{(a \geq 2),(b \leq 1)\} \in L_2$ and $\{(a \geq 4),(b \leq 1)\} \in L_2$ have the same support.

Hence, the final L_2 ends up with only $12-3=9$ frequent but not redundant 2-itemexpsets: $\{(a=4),(b=1)\}$, $\{(a=4),(b \geq 1)\}$, $\{(a=4),(b \leq 1)\}$, $\{(a \geq 4),(b=1)\}$, $\{(a \geq 4),(b \geq 1)\}$, $\{(a \geq 4),(b \leq 1)\}$, $\{(a \leq 4),(b=1)\}$, $\{(a \leq 4),(b \geq 1)\}$, $\{(a \leq 4),(b \leq 1)\}$.

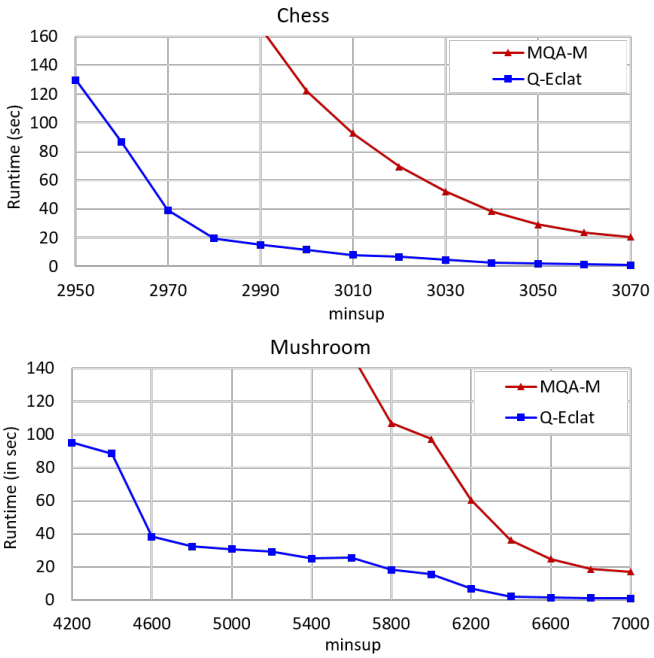

Fig. 1. Runtimes of the existing MQA-M algorithm and our Q-Eclat algorithm for quantitative frequent pattern mining with various *minsup* values on *real-life* datasets: chess and mushroom.

With only two items a and b, no candidate 3-itemexpsets can be formed. Consequently, our Q-Eclat returns the nine frequent but not redundant 1-itemexpsets and the nine other frequent but not redundant 2-itemexpsets, for a total of 18 itemexpsets as the output.

IV. EVALUATION

To evaluate our Q-Eclat algorithm, we compared it with the existing MQA-M algorithm [21]. The performance of the algorithms is assessed using four different quantitative transaction databases:

- two synthetic datasets: We assume that there are n transactions and $|I|$ different items. Each item has a probability $prob$ of occurring in a particular transaction, where $0 \leq prob \leq 1$. If the item appears in the transaction, then the number of occurrences of that item follows a Poisson(λ) distribution plus 1. We set $n = 1000$, $|I| = 50$, and $\lambda = 1$. The values of *prob* for these two quantitative transaction databases are 0.2 and 0.8, making them sparse and dense, respectively.
- two real-life datasets from UCI ML Repository [23]. We modified the chess and mushroom datasets to make them quantitative transaction databases. Whenever an item occurs in a transaction, instead of it only occurring once, its number of occurrences follows a Poisson(λ) distribution plus 1.

The two algorithms for quantitative frequent itemset mining (i.e., MQA-M [21] and our Q-Eclat) have been implemented in the Python language. The algorithms were run on a Windows 10 Nitro AN515-55 laptop using an Intel Core i5-10300H CPU

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on December 11,2024 at 08:46:18 UTC from IEEE Xplore. Restrictions apply.

at 2.50 GHz and 8.00 GB RAM. To keep the comparisons between the algorithms fair, we used many of the same functions between the algorithms. These include generation of candidate itemexpsets, discovery of frequent itemexpsets, and application of our pruning rules on the frequent itemexpsets. When we implemented the MQA-M algorithm, we use our improved pruning rules used in Q-Eclat rather than the pruning rules originally used with MQA-M. This allows the simulations to emphasize the differences between the algorithms.

We ran the main code for each of the aforementioned quantitative transaction datasets. For each combination of a quantitative transaction database and each of several values for *minsup*, the MQA-M and Q-Eclat algorithms were run and timed. Reported runtimes were average of multiple runs.

Fig. 1 shows the runtimes of each of the two algorithms for a variety of values of *minsup* for each of the real-life quantitative transaction datasets (results from the synthetic datasets have been omitted to conserve space, but are similar). In all cases, our Q-Eclat had a shorter runtime than the existing MQA-M algorithm to return the same collections of itemsexpsets.

V. Conclusions

In this paper, we presented our vertical quantitative frequent itemset mining called Q-Eclat. This Q-Eclat algorithm first represents the big data as a collection of sets of transaction IDs (i.e., tidsets). Each item-centric tidset captures the IDs of transactions containing the item, as well as the quantity of that item in each transaction. With this representation, our algorithm then vertically mines quantitative frequent patterns. During the mining process, our new pruning rules reduce the mining space, and thus shorten the runtime. When compared to the existing MQA-M algorithm (which was built for quantitative frequent pattern mining), evaluation results show that our quantitative vertical Q-Eclat algorithm takes shorter runtime to mine quantitative frequent patterns. As *ongoing and future work*, we explore ways to further enhance the mining of quantitative frequent patterns and to extend this work to mine other quantitative patterns.

Acknowledgements

This work is partially supported by the (a) Natural Sciences and Engineering Research Council of Canada (NSERC), as well as the (b) University of Manitoba.

References

[1] S. Sahri, R. Moussa. 2021. Customized eager-lazy data cleansing for satisfactory big data veracity. In IDEAS 2021, pp. 157-165.

[2] Y. Zhao, et al. 2021. A zone-based data lake architecture for IoT, small and big data. In IDEAS 2021, pp. 94-102.

[3] C.K. Leung. 2021. Data science for big data applications and services: data lake management, data analytics and visualization. In Big Data Analyses, Services, and Smart Data. AISC, vol. 899, pp. 28-44.

[4] R. Froese, et al. 2022. The border k-means clustering algorithm for one dimensional data. In IEEE BigComp 2022, pp. 35-42.

[5] C.K. Leung, et al. 2021. Explainable data analytics for disease and healthcare informatics. In IDEAS 2021, pp. 65-74.

[6] S.P. Singh, et al. 2020. Analytics of similar-sounding names from the web with phonetic based clustering. In IEEE/WIC/ACM WI-IAT 2020, pp. 580-585.

[7] C.S.H. Hoi, et al. 2022. Data, information and knowledge visualization for frequent patterns. In IV 2022, pp. 227-232.

[8] C.K. Leung, et al. 2011. Visual analytics of social networks: mining and visualizing co-authorship networks. In HCII-FAC 2011. LNCS (LNAI), vol. 6780, pp. 335-345.

[9] M.T. Alam, et al. 2021. Mining frequent patterns from hypergraph databases. In PAKDD 2021, Part II. LNCS (LNAI), vol. 12713, pp. 3-15.

[10] C.K. Leung. 2019. Pattern mining for knowledge discovery. In IDEAS 2019, pp. 34:1-34:5.

[11] M.T. Alam, et al. 2022. UGMINE: utility-based graph mining. Applied Intelligence. DOI: 10.1007/s10489-022-03385-8

[12] M.E.S. Chowdhury, et al. 2022. A new approach for mining correlated frequent subgraphs. ACM Transactions on Management Information Systems (TMIS) 13(1), pp. 9:1-9:28.

[13] S.D. Bernhard, et al. 2016. Clickstream prediction using sequential stream mining techniques with Markov chains. In IDEAS 2016, pp. 24-33.

[14] A. El Ouassouli, et al. 2019. Mining complex temporal dependencies from heterogeneous sensor data streams. In IDEAS 2019, pp. 23:1-23:10.

[15] M.S. Islam, et al. 2022. Discovering probabilistically weighted sequential patterns in uncertain databases. Applied Intelligence. DOI: 10.1007/s10489-022-03699-7

[16] C.K. Leung, et al. 2022. Visualization and visual knowledge discovery from big uncertain data. In IV 2022, pp. 336-341

[17] S. Dawar, V. Goyal, 2015. UP-Hist tree: an efficient data structure for mining high utility patterns from transaction databases. In IDEAS 2015, pp. 56-61.

[18] P. Shenoy, et al. 2000. Turbo-charging vertical mining of large databases. In ACM SIGMOD 2000, pp. 22-33.

[19] M.J. Zaki. 2000. Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering 12(3), pp. 372-390.

[20] M.J. Zaki, K. Gouda. 2003. Fast vertical mining using diffsets. In ACM KDD 2003, pp. 326-335.

[21] P.Y. Hsu, et al. 2004. Algorithms for mining association rules in bag databases. Information Sciences 166(1-4), pp. 31-47.

[22] R. Srikant, R. Agrawal. 1996. Mining quantitative association rules in large relational tables. In ACM SIGMOD 1996, pp. 1-12.

[23] D. Dua, C. Graff. 2019. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on December 11,2024 at 08:46:18 UTC from IEEE Xplore. Restrictions apply.