IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

A Novel Approach for Mining Top-K High Utility Itemset

Mr. Patil Lalit[1], Miss. Kadlag Utkarsha[2], Miss. More Pranjal[3], Miss. Walke Naina[4], Prof. R. N. Devikar[5]
Students, Department of Information Technology[1,2,3,4]
Professor, Department of Information Technology[5]
Amrutvahini College of Engineering, Sangamner, Maharashtra, India
lalitpatil0501@gmail.com[1], uttukadlag0812@gmail.com[2], morepranjal2000@gmail.com[3]
nayanawalke62@gmail.com[4]

Abstract: *Top-k high utility itemset mining refers to the discovery of top-k patterns using a user-specified value k by considering the utility of items in a transactional database. Since existing top-k high utility itemset mining algorithms are based on the pattern-growth method, they search the patterns in two steps. Therefore, the generation of many candidates and additional database scan for calculating exact utilities are unavoidable. In this paper, we propose a new algorithm, TKUL-Miner, to mine top-k high utility itemsets efficiently. It utilizes a new utility-list structure which stores necessary information at each node on the search tree for mining the itemsets. The proposed algorithm has a strategy using search order for specific region to raise the border minimum utility threshold rapidly. Moreover, two additional strategies for calculating smaller overestimated utilities are suggested to prune unpromising itemsets effectively. Experimental results on various datasets showed that the TKUL-Miner outperforms other recent algorithms both in runtime and memory efficiency.*

Keywords: High Utility Itemset, Top-K Pattern Mining, Utility-List Structure, Data Mining

I. INTRODUCTION

Itemset mining is a useful pattern search technique to find correlations among items as in association rule mining. Frequent itemset mining (FIM) [1, 2] is studied in the early stage of the itemset mining which finds the patterns that have no less existence-based support than a user specified minimum support. However, it is very difficult for users to set an appropriate minimum support because it highly depends on data types. If it is set too high, no result patterns are found while too small value makes an enormous number of result patterns which cause inefficiencies in terms of computation time and memory usage. Thus, it requires multiple trials for users to find an appropriate minimum support value, which costs a lot[3-5].

To address this issue, top-k frequent itemset mining [3-5] has been proposed. Top-k FIM mines the most frequent k itemsets without using the minimum support value from the user. Instead, the user inputs the desired number of useful patterns k which is much intuitive and comfortable. In Top-k FIM, the anti- monotone property of support is utilized to prune the mining search space.

The research of the FIM has been developed into the weighted frequent pattern mining [6, 7] and progressed to the high utility itemset mining (HUIM) [8-13]. HUIM reflects more realistic situations by adopting the utility concept which considers both quantity and profit of items. Similarly, the research of the top-k FIM has been progressed to top-k HUIM. The top-k HUIM algorithms are proposed in various types such as data stream top-k HUIM [14, 15], sequential top-k HUIM [16], and transactional databases top-k HUIM [17, 18].

The top-k high utility itemset mining on the transactional database discovers the desired number of high utility itemsets by the user specified value k. In top-k HUIM, the border minimum utility which is initially zero increases gradually as itemset mining is processed. Since the utility of an item is the product of the quantity and the profit of the item, it does not have the anti- monotone property which makes search space pruning difficult in HUIM. Likewise, this property causes highly limited situations for pruning unpromising itemsets in top-k HUIM. The undecided

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

minimum utility of top-k HUIM may generate more candidates and cost even more in memory consumption [17]. In this paper, we propose a new efficient top-K high utility itemset mining algorithm, TKUL-Miner. The contribution of this paper is as follows:

- A new framework of the utility-list structure based top-k high utility itemset mining is proposed.
- A new utility-list structure and several strategies are proposed to raise the border minimum utility threshold rapidly.

The performance improvement of the proposed algorithm compared to other top-k HUIM algorithms [17, 18] is experimentally demonstrated, especially when the datasets have a dense distribution and contain many long transactions.

The remainder of this paper is organized as follows. In Section II, we present the problem statement and define some relevant terms. Also, several related works are reviewed in the same section. In Section III, a newly designed data structure and mechanism of the proposed algorithm are explained. The experimental performance comparison of the proposed.

II. Related Work:

Then, it searches next frequent itemsets by increasing their length by one in the „level-wise" manner and repeats this process until it finds frequent itemsets. Another well-known FIM algorithm is the FP-Growth [2]. This method mines patterns without generating candidate itemsets by constructing the conditional-trees recursively after building a global FP-tree. Both algorithms utilize the anti-monotone property to prune the search space of the FIM. However, they have a problem in deciding the proper minimum support value.

Itemset-Loop and Itemset-iLoop [3] algorithms were proposed to address the minimum support problem in the FIM. In these algorithms, user inputs the maximum length of itemsets as well as the desired number of patterns, N. Using these values, they applies a backtracking method to the Apriori technique to mine N most frequent patterns whose lengths are no longer than the given maximum length. Another top-k FIM algorithm, TFP [4], constructs the FP-Tree and finds the top-k frequent itemsets whose lengths are no shorter than the given minimum length. An advanced version of TFP, TF2P-Growth [5], was introduced to obtain the frequent patterns sequentially in response to the user without any threshold.

The high utility itemset mining (HUIM) has been proposed to reflect real situations by considering both the quantity and the weight factor of each item. Two-Phase [8] algorithm approaches HUI in two-steps using the „level-wise" method. In phase I, it initially mines all patterns whose TWUs are no less than the minimum utility. In phase II, it calculates the exact utilities of candidates by scanning the database again. IHUP [9] algorithm applies the Two-Phase technique to the pattern growth method. UP-Growth and UP-Growth$^+$ [10] algorithms which followed IHUP algorithm decrease the number of candidate itemsets by additional pruning strategies. After that, HUI-Miner [11] algorithm is proposed to avoid the database scanning in the phase II by using a utility-list structure which maintains the real utilities of itemsets. FHM [12] algorithm adds the EUCS structure which stores 2-itemsets TWU to prune the search space of the utility pattern mining. Recently, a utility-list based HUIM algorithm, TUL-Miner [13], was introduced which uses transaction utilities of itemsets for effective search space pruning. Also it utilizes common utilities to reduce calculation speed in join operation. However, the HUIM algorithms have a problem in determining the minimum utility threshold.

T-HUDS [14] and TOPK-SW [15] algorithms are introduced to find top-k high utility patterns over sliding windows of a data stream. Both of them utilize tree structures with the pattern growth method. The TUS [16] algorithm finds the top-k high utility sequential patterns by the itemset and sequence concatenation process.

TKU [17] algorithm based on the UP-Growth was proposed to search top-k patterns in high utility itemsets. Since the minimum utility is not given in this problem, the algorithm applied several strategies to raise the border minimum utility threshold quickly to minimize the candidate itemsets. REPT [18] algorithm improved the performance of TKU by applying additional strategies that raise the border minimum utility rapidly.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

III. PROPOSED METHOD

3.1 Itemset Node and Utility-List Structure

An itemset node of the search tree in the TKUL-Miner is newly designed to contain more information than that used in other utility-list based algorithms [11, 13]. Fig. 1 is an example of the itemset node used in the TKUL-Miner algorithm. It consists of its node name, TWU, Sum_iutils, Sum_rutils, Sum_cutils and Sum_zrutils which can be calculated from the transaction database and the profit information. For an itemset, the TWU indicates sum of transaction utilities of itself in the database. The Sum_iutils, Sum_rutils, Sum_cutils are sum of iutil, rutil and cutil of each transaction in the utility-list of the itemset [13]. The Sum_zrutils is the sum of iutils whose transaction has zero rutil in the utility-list.

The utility-list structure of every itemset in TKUL-Miner is shown in Fig. 1. It has an attribute of cutil (common utility) which proposed in UTU-List [13]. The cutil is a value used for storing the iutil of its parent node which is necessary for calculating the itemset utility in utility-list join operation. The (k+1)-itemset stores iutils of the parent itemset node, the k- itemset, as cutils. Therefore the cutils of itemset Pxy is the relevant iutils of itemset Px. Note that all cutil values in utility- lists of 1-itemsets are initially zero. Thus, the cutils of itemset {dc} are the iutils of T_2, T_3, T_4, T_6 of {d} which are 8, 4, 8, 4 in Fig. 1.

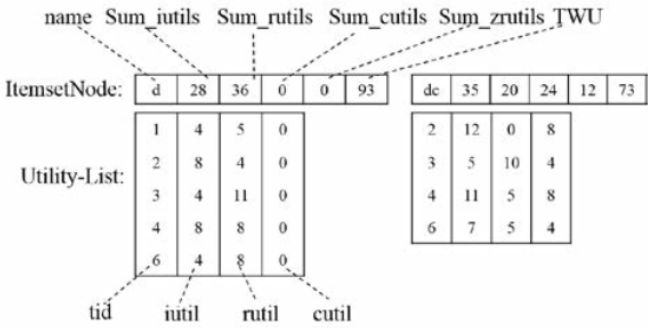


Figure 1: Itemset nodes and utility-list structures

3.2 TKUL-Miner A lgorithm

The transaction database with the profit table and user specified value k are given before the mining process starts. The border minimum utility threshold called minutil which is initially set to zero gradually increases during the mining process of the top-k HUI. The proposed algorithm starts with a pre- processing as described in Fig. 2. It scans the database twice to construct the data structures including the utility-lists and the EUCS [12] which are used throughout the whole mining process. After the pre-processing, the TKUL-Miner steps into the main top-k HUI mining. It extends searching from an item to each itemset by performing join operation of the utility-lists to find top-k high utility itemsets. Whenever an itemset contains no less utility than the minutil, it is added to the minimum heap. If the minimum heap is full and the smallest utility of the itemsets in the heap is bigger than the minutil, the minutil is updated to the smallest utility. This process ends when there are no more itemsets to generate. Moreover, the TKUL-Miner algorithm adopts several strategies called FSD, RUZ, and FCU to raise the minutil rapidly and prune the search space effectively, which contributes a lot to reduce execution time.

Algorithm: TKUL-Miner Algorithm
input : *D*, a transaction database; *minuti;, k.*
output: the top-*k* high-utility itemsets.
1 Scan *D* to calculate the TWU of 1-itemsets
2 *I** ← each item i such that TWU(i) ≥ *minutil*
3 Sort items in TWU ascending values on *I**
4 Scan *D* to build the initial utility-list of each item i ∈ *I** and build the EUCS structure
5 TUL-FirstLevelSearch(∅, *I**, *minutil*, EUCS, *k*)

Fig. 2. TKUL-Miner algorithm

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

```
Algorithm: TKUL-FirstLevelSearch Algorithm
input : P, an itemset; ExtensionsOfP, a set of extensions of P;
        minutil; EUCS; k.
output: minHeap, the top-k high-utility itemsets.
1   for i ← length(ExtensionsOfP)-1 to 0 do
2       Px ← ExtensionsOfP[i]
3       if Px.Sum_iutils ≥ minutil then
4           add Px to minHeap(k)
5       if Px.Sum_iutils + Px.Sum_rutils - Px.Sum_zrutils ≥ minutil then
6           ExtensionsOfPx ← ∅
7           for j ← i+1 to length(ExtensionsOfP)-1 do
8               Py←ExtensionsOfP[j] D yafterx
9               if c ≥ minutil such that (x, y, c) ∈ EUCS then
10                  Pxy ← Utility-List Join (Px, Py)
11                  if Pxy.Sum_itus ≥ minutil then
12                      ExtensionsOfPx ← ExtensionsOfPx ∪ Pxy
13          TKUL-Search(Px, ExtensionsOfPx, minutil, k)
```

Fig. 3. TKUL-FirstLevelSearch algorithm

In detail, rearrangement of 1-itemsets is performed in the pre-processing to build data structures efficiently. First, the algorithm scans the database to calculate the TWU of each 1- itemset. Then, it rearranges them in ascending order according to the TWU. For the aforementioned example, 1-itemsets are rearranged as b < a < d < c < e. After the rearrangement process, the utility-list and the EUCS structure are constructed while it scans the database again.

Two main algorithms of the TKUL-Miner are TKUL- FirstLevelSearch and TKUL-Search. After the pre-processing, the main parts of TKUL-Miner starts with the TKUL- FirstLevelSearch which has FSD and RUZ strategies.

A. FSD (First-level Search in TWU Decreasing-order) Strategy

The FSD strategy refers to a searching method of the TKUL- FirstLevelSearch algorithm, which searches 1-itemsets of the first level by TWU decreasing-order in Line 1 of Fig. 3 instead of the increasing-order in HUI-Miner [11]. The TKUL- FirstLevelSearch algorithm only searches 1-itemsets and generates 2-itemsets, whereas the rest will be searched and generated by the TKUL-Search algorithm. The TKUL- FirstLevelSearch algorithm discovers itemsets from the right side whose TWU is bigger than that in the left. This order is very efficient in raising the border minimum utility because itemsets with bigger TWU are more likely to generate children itemsets with higher utilities. Hence, the border minimum utility increases rapidly and it improves the pruning efficiency.

The first selected itemset Px is the right most node of 1- itemsets. The Px is an extension itemset of P with an item x. If the exact utility of Px is not less than the minutil, the Px belongs to a candidate of the top-k high utility itemset. Line 4 in Fig. 3 shows that it inserts the itemset into the minimum heap and updates the border minimum utility with the smallest utility if the minimum heap is full.

B. RUZ (Reducing the Overestimated Utility by Sum_zrutils) Strategy

In the existing algorithms [11, 12], the overestimated utility of each itemset in the utility-list is used to check if Px is promising. If we reduce the overestimated utility safely without losing any top-k itemset, then the mining process will be very efficient. To improve the search space pruning, the RUZ strategy minimizes the overestimated utility by subtracting Sum_zrutils from the overestimated utility which is the total value of Sum_iutils and Sum_rutils, in Line 5 of Fig. 3. This idea is based on the fact that the transactions of Px which have zero rutil will not be included to children of Px. It means that subtracting iutils of those transactions still maintains the overestimated utility of Px‟s descendants. The itemset Px is figured out as unpromising and pruned if the minimized overestimated utility is less than the minutil.

When itemset Px is promising, select Py where y is an item whose TWU is greater than that of x, and then check whether the value of EUCS is no less than minutil [12] or not. If the EUCS of x and y is promising, the Utility-List Join function is called as in Line 10 in Fig. 3. It generates the itemset Pxy by joining the utility-lists of Px and Py. This function uses DTJ (Decreasing TWU while Joining) strategy [13] that halts the joining process. To use TU of the

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

utility-list, the TKUL-Miner uses transaction utilities in Table 3 to subtract TU of the excluded transactions from the respective TWU of Px and Py.

The TKUL-Search algorithm starts after all children nodes of each Px are generated. Like the TKUL-FirstLevelSearch algorithm, the TKUL-Search adopts RUZ and EUCS strategies with the same utility-list join operation. However, the TKUL- Search has two different characteristics from the TKUL- FirstLevelSearch to enhance the pruning efficiency. One is that it searches the itemsets in TWU increasing order and the other is that the TKUL-Search has an additional strategy called FCU using Sum_cutils.

An example of set-enumeration tree is described in Fig. 4. The number above each node represents the construction order and the number below the node in the bracket represents the search order of the TKUL-Miner algorithm. Here, all 2-itemsets from {ce} to {ba} are generated by the TKUL-FirstLevelSearch algorithm while the rest itemsets from {dce} to {badce} are generated by the TKUL-Search algorithm.

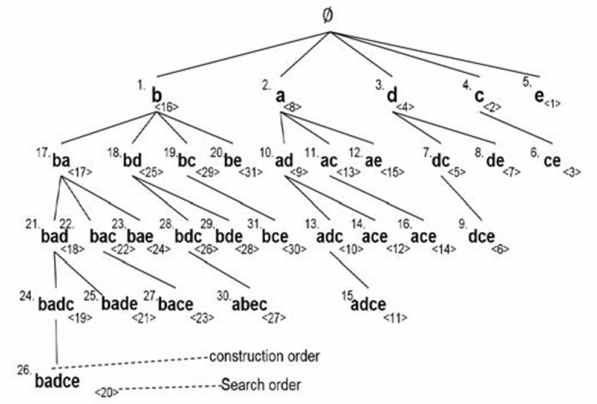


Fig. 4. Set-enumeration tree with the construction and search order

In the TKUL-Search, the Px is selected from the left most child of P. The Sum_iutils for validating a top-k high utility itemset and the RUZ strategy for determining a promising itemset are used. The FCU strategy in the TKUL-Search is used when Px has no child. Otherwise, it uses the EUCS strategy as same with the previous case.

C. FCU (First Child Pruning by using Sum_Cutil) Strategy

The FHM algorithm checks whether the overestimated utility of (x, y) in the EUCS is not less than the minutil before the algorithm generates the child node Pxy from the parent nodes Px and Py. We will reduce the overestimated utility of (x, y) in the EUCS by using the attribute of Sum_cutils in the utility-list, which is applied only to the first child of Px. The FCU strategy checks whether the result of the formula, Px.Sum_iutils + Py.Sum_iutils + Py.Sum_rutils – Px.Sum_cutils, is not less than the minutil when Px has no child. This formula calculates the overestimated value of Pxy which is less than the value of EUCS. The Sum_iutils of Pxy is obtained by adding iutils of transactions and subtracting the cutils during the utility-list join operation. The iutil of Px is always bigger than its cutil and the maximum value for Sum_rutils of Pxy is the Sum_rutils of Py. Thus, if the overestimated value of Pxy is less than minutil, the join operation of Px and Py will be meaningless.

The reason why this strategy is only available to the first child is that the first child does not affect the others, whereas the nodes from the second child can affect the children of their left- side sibling nodes. If the FCU strategy applies to the second child onwards, some promising children may not be generated.

The TKUL-Search will let the itemsets Px and Py generate the extended itemset Pxy when they are turned out as promising by the aforementioned process. The algorithm performs all join operations of Px with every possible extension item y and calls the search algorithm recursively. The TKUL-Miner algorithm ends the process when no more promising itemsets left. The desired number of high utility itemsets will be stored in the minimum heap after the process ends with the fixed value of the border minimum utility.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

IV. EXPERIMENTAL RESULTS

Experiments were conducted to compare performance of the proposed algorithm TKUL-Miner with other state-of-the-art algorithms TKU [17], REPT [18], and UP-Growth+ [10]. For the REPT algorithm, the value of N which is needed to build the structure RSD is set according to [18]. Also, we executed the UP-Growth+ algorithm with optimal minimum utilities which are obtained by a top-k HUIM algorithm.

The real dataset Chain and Chess are obtained from NU- MineBench 2.0 [19] and FIM Repository [20], respectively. The synthetic datasets T10I4D100K and T40I10D100K are generated by IBM Quest Data Generator [1]. For the datasets, the quantity is randomly generated in the range of 1 to 10 and the profit is generated under the log-normal distribution from 1 to 1000. The real datasets are either dense or sparse according to the distribution of length of each transaction as shown in Table V. On the other hand, the synthetic datasets are almost evenly distributed because they are generated randomly.

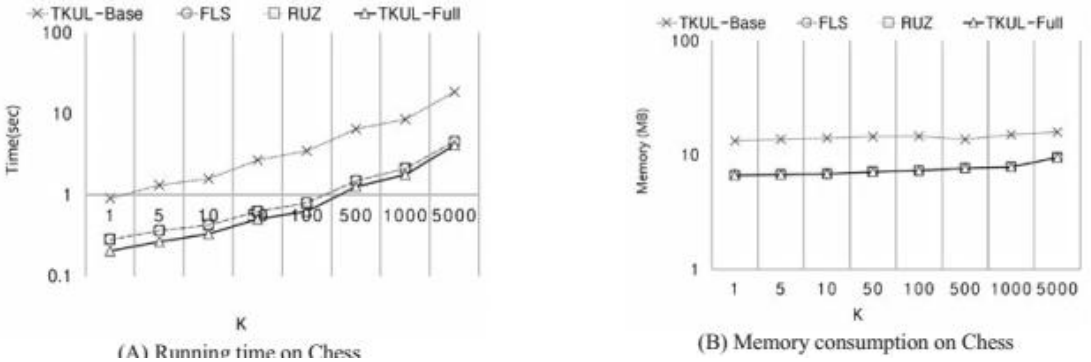
The experiments were performed under the environment of Windows 64bit OS with Intel i7-4770 CPU 3.40GHz and 32GB memory. The results are measured from the algorithms that are implemented in MS Visual Studio 2010 C++.

Database	Size (KB)	#Trnx	#items	Avg Len	Max Len	type
Chain	63573	1112949	46086	7.3	170	Sparse
Chess	591	3196	75	37	37	Dense
T10I4D100K	6268	98424	1000	10.1	30	-
T40I10D100K	24905	100000	1000	39.6	78	-

Table V: Characteristics of Datasets

4.1 Performance Comparison of Different Strategies

Fig. 5 shows the running time and the memory consumption of difference strategies in the TKUL-Miner algorithm on dense dataset Chess. In the graph, the notations TKUL-base, FLS, RUZ, TKUL-full are the TKUL-Miner algorithm without any strategy, TKUL-base with FLS strategy, TKUL-base with FLS and RUZ strategy and the full version of TKUL-Miner algorithm including FSD strategy respectively. According to Fig. 5(A), TKUL-base takes the worst time and FLS have improved TKUL-base over 70%. The runtime performance of RUZ is also improved but not very significantly. The TKUL-Full with FSD strategy improved almost 20%. Fig. 5(B) shows the strategy of TKUL-Miner algorithm improved the usage of memory on Chess. Likewise, the improvement of FLS strategy is noticeable than the others. Since increasing the minutil as fast as possible in the top-k high utility pattern mining affects its performance significantly, the FLS strategy works well in the proposed algorithm.



(A) Running time on Chess (B) Memory consumption on Chess

Fig. 5. Performance comparison

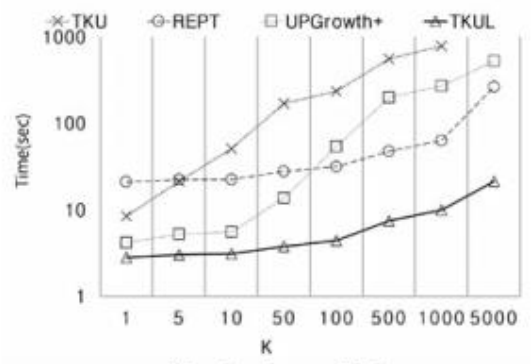
4.2 Running Time Comparison on Different Datasets

Running time of the TKUL-Miner, TKU, REPT, and optimal UP-Growth+ algorithms on each datasets is shown in Fig. 6 in a logarithmic scale. The experiment of the algorithm was terminated if the running time exceeds 10,000 seconds.

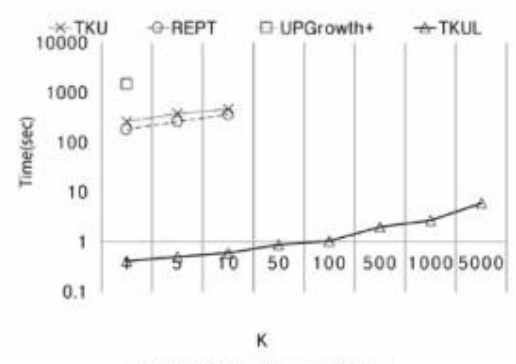
IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

Fig. 6(A) shows the running time of the algorithms for the sparse dataset Chain. Excluding the TKUL-Miner, the optimal UP-Growth+ takes the shortest running time when k is smaller than or equal to 50. From the 100 of the k value, the REPT (N = 1000) becomes the fastest one. However, the TKUL-Miner takes shorter execution time than the other three algorithms for all k. For the case where k is 5000, the running time of the TKUL-Miner becomes 21 seconds whereas REPT takes 266 seconds. It is more than 10 times improvements.

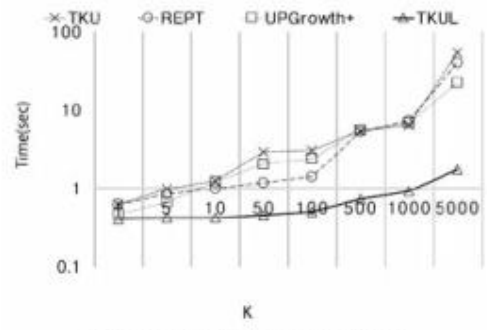
The experimental result on the dense dataset Chess is shown in Fig. 6(B). For the dense dataset, the gap between the TKUL- Miner algorithm and other algorithms is bigger than that with the dataset Chain and those tree based algorithms exceed 10000 seconds even k is not very large. The running time of the TKUL- Miner algorithm never exceeds 2 seconds for any k. Comparing to the performance of REPT(N = 100), it improved the running time by 100 times when k is 1 and 200 times when k is 10. Figs. 6(C) and 6(D) show the experimental results on the synthetic datasets T10I4D100K and T40I10D100K, respectively. For both cases, the TKUL-Miner outperformed the other algorithms for all k



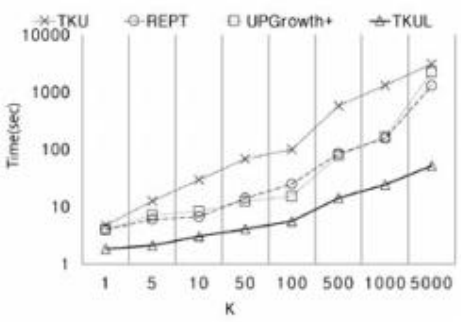
(A) Running time on Chain



(B) Running time on Chess



(C) Running time on T10I4D100K



(D) Running time on T40I10D100K

Fig. 6. Running time

Fig. 6(C) shows that the execution time of the other algorithms increases steeply, whereas it grows gradually for the proposed algorithm. Also, it does not exceed 2 seconds even when k is 5000. In Fig. 6(D), running time of the TKUL-Miner is 10 times improved in the best case compared to TKU algorithm. The improvement of the TKUL-Miner algorithm is larger than that of Fig. 6(C) when k varies from 1 to 100.

The experimental results show that the TKUL-Miner takes the shortest time over the state-of-the-art algorithms by more than multiple times in many cases. Also, the TKUL-Miner algorithm tends to be much faster when the datasets are dense and contain longer average lengths.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

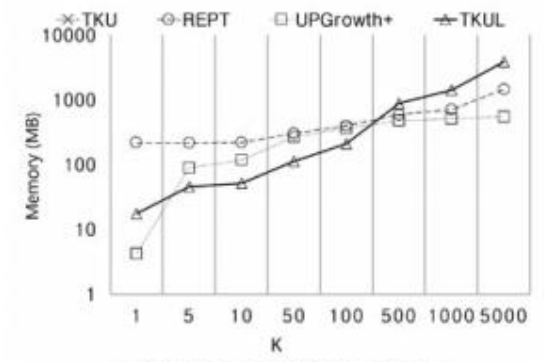
4.3 Memory Usage Comparison on Different Datasets

Fig. 7 shows the peak memory consumption of the TKUL- Miner, TKU, REPT, and optimal UP-Growth+ algorithms on the datasets in a logarithmic scale. The experiment was terminated when the algorithm uses over 20000MB of memory.

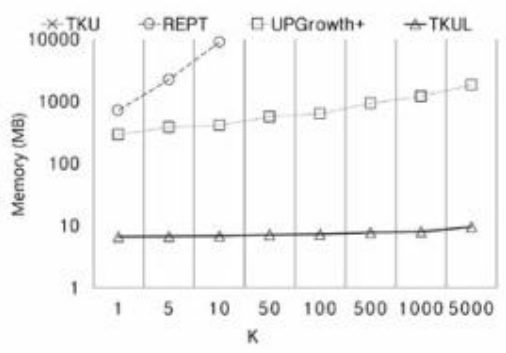
Fig. 7(A) is the result on the sparse dataset Chain. In this graph, the TKUL-Miner algorithm uses the least memory when k is less than 100. The memory consumption of the TKUL- Miner grows larger than others when k is more than 500 because it does not make a compressed structure like the utility pattern tree. However, it is applicable since k is usually not larger than 100 in real applications.

In Fig. 7(B), the proposed algorithm outperforms over other algorithms on the dense dataset Chess. The TKU algorithm terminates when k is 1. The REPT algorithm exceeds 20000MB when k is 50. The UP-Growth+ uses large memory throughout the mining process. Memory efficiency of the proposed algorithm, which is based on the utility-list structure, improved the performance approximately 98% in the dense dataset.

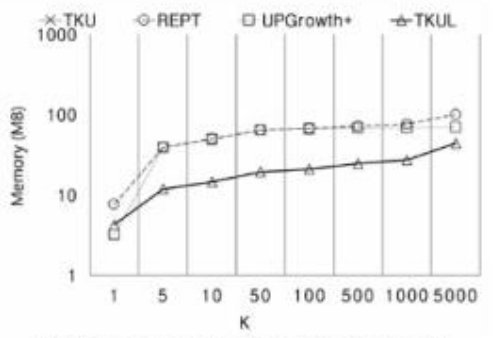
According to Figs. 7(C) and 7(D), the TKU algorithm uses the largest memory because it generates the most number of candidate itemsets on synthetic datasets T10I4D100K and T40I10D100K. The REPT and the optimal UP-Growth+ algorithm consume almost the same memory. On the other hand, the TKUL-Miner consumes the least memory. Since the TKUL- Miner is a utility-list based algorithm which does not maintain the many candidates, the memory usage is reduced by approximately 60% in Fig. 7(C) and 80 % in Fig. 7(D) compared to REPT (N = 100 and N = 1000). Although the consumed memory of the TKUL-Miner algorithm varies depending on the number of transactions in a database, reduction of memory consumption becomes even larger when the experiments are taken on the dense datasets.



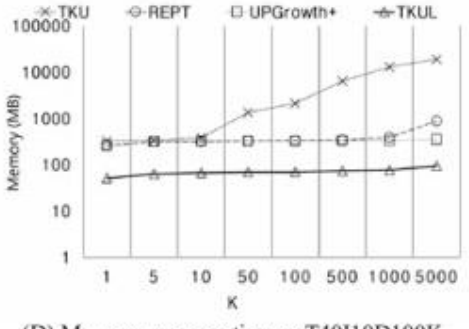
(A) Memory consumption on Chain



(B) Memory consumption on Chess



(C) Memory consumption on T10I4D100K



(D) Memory consumption on T40I10D100K

Fig. 7. Memory consumption

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

V. CONCLUSION

This paper proposed the TKUL-Miner algorithm to mine top-k high utility itemsets efficiently. The TKUL-Miner algorithm uses the utility-list to avoid the additional scanning of database which is a necessary step for the existing top-k HUIM algorithms. To increase the border minimum utility rapidly, the proposed algorithm takes the TWU descending order for searching first-level itemsets. It utilizes the sum of common utilities and the sum of itemset utilities that have zero remaining utilities in order to prune the search space of the mining effectively. The TKUL-Miner is outperformed the state-of-the- art top-k high utility itemset mining algorithms in all demonstrated experiments on real and synthetic datasets. Both running time and memory consumption of the algorithms are decreased in all cases of the experiments.

VI. ACKNOWLEDGMENT

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1012) supervised by the IITP (Institute for Information and communications Technology Promotion)

REFERENCES

[1]. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," In Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Vol. 1215, pp. 487-499, 1994.
[2]. J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, pp. 1-12, 2000.
[3]. A.W.-C. Fu, R.W.-W. Kwong, and J. Tang, "Mining n-most interesting itemsets," In Proceeding of International Symposium on Methodologies for Intelligent Systems (ISMIS), Charlotte, Vol. 1932, pp. 59-67, 2000.
[4]. J. Han, J. Wang, Y. Lu, and P. Tzvetkov, "Mining top-k frequent closed patterns without minimum support," In Proceedings of IEEE International Conference on Data Mining(ICDM), Maebashi, pp. 211- 218, 2002.
[5]. Y. Hirate, E. Iwahashi, and H. Yamana, "TF2P-Growth: an efficient algorithm for mining frequent patterns without any thresholds", In Proceedings of IEEE ICDM 2004 Workshop on Alternative Techniques for Data Mining and Knowledge Discovery, Brighton, 2004.
[6]. W. Wang, J. Yang, and P. Yu, "Efficient mining of weighted association rules (WAR)," In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, pp. 270- 274, 2000.
[7]. F. Tao, F. Murtagh, and M. Farid, "Weighted association rule mining using weighted support and significance framework," In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, pp. 661-666, 2003.
[8]. Y. Liu, W. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," In Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, Vol. 3518, pp. 689-695, 2005.
[9]. C.F. Ahmed, S.K. Tanbeer, B.S. Jeong, and Y.K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No. 12, pp. 1708-1721, 2009.
[10]. V.S. Tseng, B.E. Shie, C.W. Wu, and P.S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," IEEE Transactions on Knowledge and Data Engineering, Vol. 25, No. 8, pp. 1772-1786, 2013.
[11]. M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Maui, pp. 55-64, 2012.
[12]. P. Fournier-Viger, C.W. Wu, S. Zida, and V.S. Tseng, "FHM: Faster high- utility itemset mining using estimated utility co-occurrence pruning," Foundations of Intelligent Systems, Springer, pp. 83-92, 2014.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, February 2022
Impact Factor: 6.252

[13]. S. Lee and J.S. Park, "High utility itemset mining using transaction utility of itemsets," KIPS Transactions on Software and Data Engineering, Vol. 4, No. 11, pp. 499-508, 2015.
[14]. M. Zihayat and A. An, "Mining top-k high utility patterns over data streams," Information Sciences, Vol. 285, pp. 138-161, 2014.
[15]. T. Lu, Y. Liu, and L. Wang, "An algorithm of top-k high utility itemsets mining over data stream," Journal of Software, Vol. 9, No. 9, pp. 2342- 2347, 2014.
[16]. J. Yin, Z. Zheng, L. Cao, Y. Song, and W. Wei, "Efficiently mining top- k high utility sequential patterns," IEEE 13th International Conference on Data Mining(ICDM), Dallas, pp. 1259-1264, 2013.
[17]. C. Wu, B. Shie, V.S. Tseng, and P.S. Yu, "Mining top-k high utility itemsets," In Proceedings of ACM SIGKDD 18th International Conferemce on Knowledge discovery and data mining, New York, pp. 78- 86, 2012.
[18]. H. Ryang and U. Yun. "Top-k high utility pattern mining with effective threshold raising strategies," Knowledge-Based Systems, Vol. 76, pp. 109- 126, 2015.
[19]. J. Pisharath, Y. Liu, W.K. Liao, A. Choudhary, G. Memik, and J. Parhi, (2005). Numinebench version 2.0 dataset and technical report. Available at <http://cucis.ece.northwestern.edu/projects/DMS/MineBench.html>. Accessed on June 2015.
[20]. FIMI, (2003). Fimi: The frequent itemset mining dataset repository. Accessed on August 2015. <http://fimi.ua.ac.be/data/>.