ZIBELINE INTERNATIONAL™ PUBLISHING

ISSN: 2773-6202 (Online)
CODEN: JTIOCC

Journal of Technology & Innovation (JTIN)

DOI: http://doi.org/10.26480/jtin.02.2021.44.47

RESEARCH ARTICLE

MINING OF HIGH-UTILITY ITEMSETS WITH NEGATIVE UTILITY

Tung N.T*, Nguyen Le Van, Trinh Cong Nhut, Tran Van Sang

Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Vietnam
Corresponding Author Email: nt.tung@hutech.edu.vn

ARTICLE DETAILS

Article History:

Received 23 September 2021
Accepted 26 October 2021
Available online 03 November 2021

ABSTRACT

The goal of the high-utility itemset mining task is to discover combinations of items that yield high profits from transactional databases. HUIM is a useful tool for retail stores to analyze customer behaviors. However, in the real world, items are found with both positive and negative utility values. To address this issue, we propose an algorithm named Modified Efficient High-utility Itemsets mining with Negative utility (MEHIN) to find all HUIs with negative utility. This algorithm is an improved version of the EHIN algorithm. MEHIN utilizes 2 new upper bounds for pruning, named revised subtree and revised local utility. To reduce dataset scans, the proposed algorithm uses transaction merging and dataset projection techniques. An array-based utility-counting technique is also utilized to calculate upper-bound efficiently. The MEHIN employs a novel structure called P-set to reduce the number of transaction scans and to speed up the mining process. Experimental results show that the proposed algorithms considerably outperform the state-of-the-art HUI-mining algorithms on negative utility in retail databases in terms of runtime.

KEYWORDS

high-utility itemset mining, negative utility, pattern mining, utility mining

1. INTRODUCTION

To discover the associations and the relations among the items within a transactional database, frequent itemset mining (FIM) methods were applied (Agrawal and Srikant, 1994). Companies have incorporated FIM into their available databases to boost executive performance. The analysis of the transactions helps put forth effective strategies in their business, such as catalog designing, marketing, customer behavior analysis, or basket analysis. FIM analyzes the customer's shopping habits, and then discovers the associations among the items that were selected by the customer. Retailers use the discovered knowledge to develop effective strategies to boost their sales.

Some algorithms to perform this task are Apriori, AprioriTid, Eclat, FP-Growth, etc (Agrawal and Srikant, 1994; Zaki, 2000; Han et al., 2004). Of them, the FP-Growth only requires two database scans to construct the FP-tree and directly extracts frequent itemsets from the tree. Thus, discovering the complete set of frequent itemsets (FIs). FIM only considers the existence of the items within transactions and treats all items equally. It completely ignores other important information such as the purchased quantity of items, item's profit, etc. In real-world applications, every item has its value (unit profit), and in most cases, frequent patterns might not be the ones that yield high profit, or usefulness to the users. Briefly, the generated profit of an item when purchased, called utility, is the product of the purchased quantity and its unit profit. Patterns or itemsets that generate high profit and satisfy a user-specified threshold are called high-utility patterns (HUPs) or itemsets (HUIs).

In the literature survey, we find that the profits of items have been assumed to be positive values, but the assumption of these profits to be positive is unrealistic for real-life transactional datasets. Most of the algorithms cannot handle datasets where items have negative utility. Such negative items often occur in many real-life transactional datasets. EHIN reduces the dataset scanning cost by merging the identical transactions (Singh et al., 2018). It uses projected dataset based transaction merging techniques to further reduce the dataset scanning cost. EHIN proposed two techniques to prune the search space named redefined subtree and redefined local utility. But EHIN needs to perform several database scans to calculate the local utility, sub-tree utility, and utility of each itemset. This process is costly because EHIN scans the transaction containing the targeted item.

The rest of the paper is organized as follows: Section 2 surveys related works on HUIs with negative utility. Section 3 describes definitions and proposes methods of high utility itemsets mining with negative utility. Experimental results will be shown in section 4. Finally, section 5 presents the conclusion and future improvements.

2. RELATED WORD

Extending from FIM, the task of mining the complete set of HUIs is called high-utility itemset mining (HUIM). The problem of mining HUIs was first proposed (Yao et al., 2004). The authors presented the concepts of utility and high-utility measures. HUIM is considered a challenging task since the utility measure does not satisfy the downward closure property (Agrawal and Srikant, 1994), which originally states in FIM that all subsets of a frequent itemset must also be frequent. In 2005, Liu et al. presented an algorithm to mine HUIs in two phases (Liu et al., 2005), namely Two-Phase. In the first phase, the algorithm computes the downward closure on the utility of itemsets in transactions to prune the search space. This novel upper-bounds on utility is called Transaction Weighted Utility (TWU). The Second phase scans the database again to calculate the exact utility value of itemsets to determine which one is HUI. However, TWU is not tight enough to effectively prune the search space, leaving a huge number of candidates for the algorithms to check. Thus, to reduce further

Quick Response Code	Access this article online	
	Website: www.jtin.com.my	DOI: 10.26480/jtin.02.2021.44.47

Cite The Article: Tung N.T, Nguyen Le Van, Trinh Cong Nhut, Tran Van Sang (2021). Mining Of High-Utility Itemsets With Negative Utility. *Journal of Technology & Innovation, 1(2): 44-47.*

the generated candidates and database scans, Lin et al. proposed a tree structure, called HUP-tree (Lin et al., 2011), to effectively mine HUIs. It first calculates the utility values for 1-itemsets and uses them to construct the HUP-tree. Then the algorithm recursively traverses the tree to extract HUIs based on a header table. The algorithm requires only two database scans to discover the complete set of HUIs in a database. In 2016, Zida et al proposed a single phase for effectively mining HUIs, namely EFIM (Zida et al., 2017). By proposing a new framework for calculating the dynamic utility value of items, the iMEFIM algorithm further enhances EFIM by proposing a new P-Set structure to significantly reduce the cost of database scans (Nguyen et al., 2019).

HUINIV-Mine is among the works that consider the concept of negative utility mining algorithms (Chu et al., 2009). HUINIV-Mine is an extension of Two-phase algorithm. Two-phase algorithm cannot deal with items that have negative utility. To find HUIs with negative utility, Two-phase algorithm loses some candidate itemsets. The Two-Phase algorithm focuses on positive item values and is not suited to items with negative utility values. Therefore, HUNIV-Mine addresses the problem of discovering HUIs with negative utility. It is a level-wise algorithm and needs to maintain plenty of itemsets in memory to find larger patterns. It needs multiple dataset scans to find HUIs. To overcome the limitations of these Two-phase based algorithms, Fournier-Viger et al proposed a one-phase algorithm named FHN (Lin et al., 2016). FHN reduces the dataset scan and takes the advantage of depth-first search. Hence, it consumes much less memory than HUINIV-Mine. FHN is a negative utility version of FHM algorithm. FHN can prune the search space efficiently because it utilizes the utility-list structure that is introduced in HUI-Miner (Liu and Qu, 2012). It discovers HUIs without generating candidates and introduces several strategies to handle items with negative utility efficiently. It mines HUIs in a single-phase and hence avoids the costly candidate generation.

In 2018, Singh et al. proposed an algorithm for mining HUIs with negative utility by using a patten-growth tree. EHIN reduces the dataset scanning cost by merging the identical transactions. It uses projected dataset based transaction merging techniques to further reduce the dataset scanning cost. EHIN proposed two techniques to prune the search space named redefined subtree and redefined local utility. The author demonstrated that EHIN is up to 28 times faster and consumes up to 10 times less memory than the state-of-the-art algorithm FHN. The author showed the relative runtime and relative memory comparison between EHIN and the state-of-the-art algorithm FHN.

3. Proposed Algorithm

This section presents several key definitions and formulates the task of mining HUIs from transactional databases with negative utility.

3.1 Preliminaries

Let there be a finite set of m distinct items, denoted $\mathcal{I}$. A transaction database, denoted $\mathcal{D} = \{T_1, T_2, \dots, T_n\}$, is a multiset of transaction T_q. In which, each transaction T_q has a unique identifier called TID and $T_q \subseteq \mathcal{I}$. Each item $i \in T_q$ has an associated positive value called its internal utility in transaction T_q, denoted $ui(i, T_q)$. Also, for each item $i \in \mathcal{I}$, it has an associated positive value called its external utility, denoted $ue(i)$. Table 1 present a sample transaction database and the unit price of these items is provided in Table 2. The utility value of item i in transaction T_q, denoted $u(i, T_q)$, is defined as the product of its external utility and internal utility in transaction T_q, $ue(i) \times ui(i, T_q)$. The utility of an itemset X (a set of items of arbitrary size, $X \subseteq \mathcal{I}$) in transaction T_q, denoted as $u(X, T_q)$, is defined as $u(X, T_q) = \sum_{i \in X} u(i, T_q)$. The utility value of an itemset X within the database $\mathcal{D}$, denoted $u(X)$, is defined as $u(X) = \sum_{T_i \in \mathcal{T}(X)} u(X, T_i)$, whereas $\mathcal{T}(X)$ is the set of all transactions containing itemset X.

Table 1: A transactional dataset

TID	Transaction
T_1	$(a,2)(b,2)(d,1)(e,3)$
T_2	$(b,1)(c,5)(e,1)$
T_3	$(b,2)(c,1)(d,3)(e,2)$
T_4	$(c,2)(d,1)(e,3)$
T_5	$(a,2)$
T_6	$(a,2)(b,1)(c,4)(d,2)(e,1)$
T_7	$(b,3)(c,2)(e,2)$

Table 2: External utility values of items

Item	a	b	c	d	e
External utility	2	−3	1	4	1

Let $\mathcal{D}$ be a transaction database and a user-specified minimum utility threshold μ. An itemset X is called a high-utility itemset (HUI) if and only if $u(X) \geq \mu$. The task of mining high-utility itemsets is to return the complete set of all HUIs.

3.2 Proposed algorithm

In this section, we propose an efficient algorithm for mining HUIs with negative utility items, namely, MEHIN. The MEHIN algorithm extends the EHIN algorithm (Singh et al., 2018). The EHIN algorithm needs to perform several database scans to calculate the local utility, sub-tree utility, and utility of each itemset. For an itemset X, calculating these values is necessary to determine the set of secondary items $Secondary(X)$ that should be considered to further extend X. This process is costly because EHIN scans the database D_X to calculate these values for any item $z \in Secondary(X)$ rather than by only scanning the transactions in $D_{X \cup z}$. The P-set structure is defined to resolve this problem.

Definition 1 (TIDs Projection of an Itemset on a Database) (Nguyen et al., 2019). *The TIDs projection of the itemset X on D*, also called *the P-set of X*, is defined as the set of identifiers of transactions containing X, that is, P-set$(X) = \{T.id | T \in D \wedge X \subseteq T\}$.

For example, P-set$(\{d\}) = \{1, 3, 4, 6\}$. Also, P-set$(\{a, b\}) = \{1, 6\}$.

Definition 2 (Expansion of the TIDs Projection of an Itemset with an Item i on a Database) (Nguyen et al., 2019). *The expansion of the TIDs projection of the itemset X on D*, denoted as Pex-set(X, i), is defined as Pex-set$(X, i) = \{T'.id | T' \in cD_X \wedge i \in T'\}$.

For example, Pex-set$(\{b, c\}, d) = \{3, 6\}$.

Definition 3 (The Construction of a P-set) (Nguyen et al., 2019). Given an itemset X, $Secondary(X)$, and database D, the P-set contains all transactions in D that contain items $i \in Secondary(X)$ with respect to item i. The *P-set structure is constructed* when scanning database D to calculate the upper bounds of itemsets that have $TWU \geq minutil$.

Property 1. The P-set of an itemset X such that $TWU(X) > minutil$ contains only transactions containing the itemset X. (Nguyen et al., 2019).

Property 2. Given an itemset X having $TWU(X) \geq minutil$, all transactions in D that do not contain X do not need to be scanned (Nguyen et al., 2019).

For example, Table 3 display TWU information of each item in database at Table 1:

Table 1: TWU of each item in database

Item	a	b	c	d	e
TWU	32	53	51	52	62

For $minutil = 50$, P-set(a) will not build. And the P-set(c) will contain the value {2,3,4,6,7} and transactions T_1 and T_5 will not be scanned.

The MEHIN algorithm uses 2 measures used in the EHIN algorithm including RLU and RSU. An item i is an item that promises with itemset X if the $RLU(X, i)$ is not less than $minutil$, they will be added to the list $Secondary(X)$. The $RSU(X, i)$ is compared with $minutil$ and items i with the $RSU(X, i)$ is greater than or equal to minutil are added to the $Primary(X)$, which contains items further extended from X. The P-set structure is used to avoid scanning numerous transactions. The pseudo-code of MEHIN is shown in Algorithm 1. The main modification in MEHIN is the calculation of Pex-set(X, z) in line 10 , the calculation of $RSU(X, z)$ in lines 4 and 7 of the Search_P procedure and the calculation of $RSU(X, z)$ in lines 4 and 6 of the Search_P procedure. Since the set Pex-set(X, z) contains all TIDs in cD_X , the complexity of calculating $RSU(X, z)$ does not change

with respect to the EHIN algorithm. Here, only the transactions having their identifiers in TIDs in Pex-set (X, z) are scanned instead of all transactions in cD_X

Algorithm 1. The MEHIN algorithm

Input: $\mathcal{D}$: transaction database; μ: user-specified threshold,

Output: the set of all discovered HUIs

1: $X \leftarrow \emptyset$;

2: $\rho \leftarrow$ set of positive utility items in D;

3: $\eta \leftarrow$ set of negative utility items in D;

4: Scan D, compute $RLU(X, z)$for all items $z \in \rho$, using $UA[z]$;

5: $Secondary(X) = \{z | z \in \rho \wedge RLU(X, z) \geq \mu\}$;

6: Let $\succ$ be the total order of RTWU increasing values on $Secondary(X)$.;

7: Scan D to remove each item $i \notin Secondary(X)$ from all transactions. Then remove empty transactions;

8: Sort each transaction T in ascending order according to $\succ_T$ with positive utility items followed by negative utility item;

9: Assign offset to each transaction in D;

10: Scan D to calculate $RSU(X, z)$ and Pex-set(X, z) for each item $z \in Secondary(X)$, using $UA[Z]$;

11 :$Primary(X) = \{z | z \in Secondary(X) \wedge RSU(X, z) \geq \mu\}$;

12: Search_P$(\eta, X, D, Primary(X), Secondary(X), \mu$, Pex-set$(X, z))$;

13: **RETURN** HUIs

Algorithm 2. The Search_P procedure

Input: η: set of negative items, X: an itemset,cD_X: transaction database,$Primary(X)$: the Primary items of X, $Secondary(X)$: the Secondary items of X , μ: user-specified threshold, Pex-set(X, z) the extension of the TIDs projection;

Output: the set of HUIs that are extensions of X with positive items.

1: Set HUIList= $\emptyset$;

2: **For** each item $z \in Primary(X)$ **do**

3: $\beta = X \cup \{z\}$;

4: Scan D_X in Pex-set(X, z) to calculate $u(\beta)$ and construct D_β;

5: **If** $u(\beta) \geq \mu$ **then** Output β;

6: **If** $u(\beta) > \mu$ **then** Search_N$(\eta, \beta, D_\beta, \mu$, Pex-set$(X, z))$;

7: Scan D_β to calculate $RSU(\beta, z)$, $RLU(\beta, z)$ and Pex-set(β, z) for all $i \in Secondary(X)$ after z;

8: $Primary(\beta) = \{i \in Secondary(X) | RSU(\beta, i) \geq \mu\}$;

9: $Secondary(\beta) = \{i \in Secondary(X) | RLU(\beta, i) \geq \mu\}$;

10: Search_P$(\eta, X, D_\beta, Primary(\beta), Secondary(\beta), \mu$, Pex-set$(\beta, z))$

11: **RETURN** HUIs

Algorithm 3. The Search_N procedure

Input: η: set of negative items, X: an itemset,cD_X: transaction database,$Primary(X)$: the Primary items of X , μ: user-specified threshold, Pex-set(X, z) the extension of the TIDs projection;

Output: the set of HUIs that are extensions of X with negative items.

1: Set HUIList= $\emptyset$;

2: **For** each item $z \in \eta$ **do**

3: . $\beta = X \cup \{z\}$;

4: Scan D_X in Pex-set(X, z) to calculate $u(\beta)$ and construct D_β;

5: **If** $u(\beta) \geq \mu$ **then** Output β;

6: Scan D_β to calculate $RSU(\beta, z)$, and Pex-set(β, z) for all $i \in Secondary(X)$ after z;

8: $Primary(\beta) = \{i \in \eta | RSU(\beta, i) \geq \mu\}$;

9: Search_N$(Primary(\beta), \beta, D_\beta, \mu$, Pex-set$(\beta, z))$;

10: **RETURN** HUIs

4. Evaluation Studies

To have a clearer view of how the proposed algorithm MEHIN performs, this section evaluates the performance of the MEHIN in terms of running time and peak memory usage. The experiments were conducted on a desktop computer equipped with an AMD Ryzen™ 5 3600 @ 3.6Ghz, using 24GB RAM and running Windows 10. The performance of MEHIN is compared against the EHIN algorithm to mine HUIs. All the tested algorithms were implemented using the Java programming language. The measured running time and peak memory usage were obtained by using standard Java API. Two datasets were used to evaluate the performance of the algorithms, namely *Retail*, *Kosarak*. Their characteristics are presented in; In which $|\mathcal{D}|$ denotes the number of transactions in each dataset, $|\mathcal{I}|$ denotes the number of distinct items and $Trans_{MAX}$ and $Trans_{AVG}$ denote the maximum transaction length and average transaction length, respectively.

Table 4: Dataset characteristics

Dataset	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	$Trans_{MAX}$	$Trans_{AVG}$
Retail	88,162	16,470	76	10.3
Kosarak	990,002	41,270	2498	8.1

The results are shown in Figure 1 to Figure 4. It can be seen from the results that the MEHIN algorithm dominates the EHIN algorithm in terms of execution time over all the tests. In every dataset, as the μ threshold decreased, the runtime of the two algorithms increased. The MEHIN can be up to more 10 times faster than EHIN. At the lowest $\mu = 5000$ in the Retail dataset, MEHIN took only 2 seconds to complete, about 10 times faster than EHIN, which took 25 seconds. The same performance gain can be observed on the Kosarak database, where MEHIN required only 7 seconds but EHIN took 31 seconds. MEHIN was thus 4.5 times faster

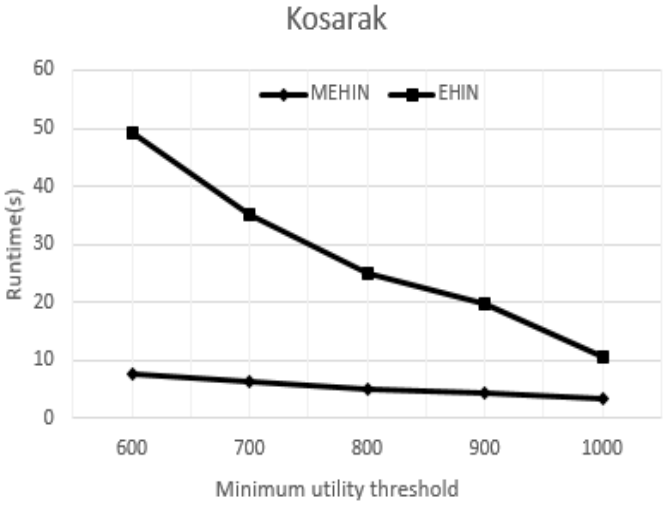

Figure 1: Runtime comparison for the Kosarak database

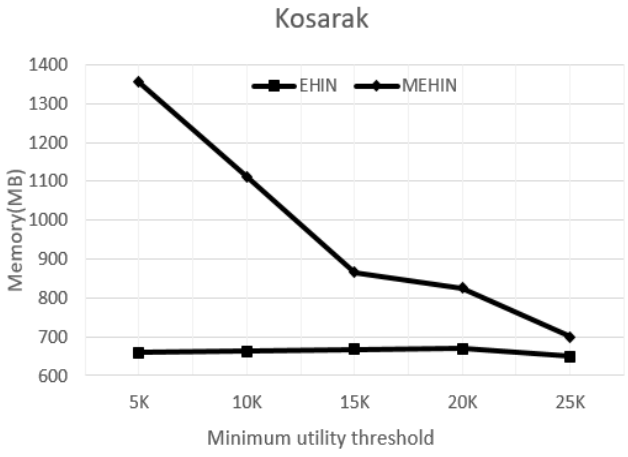

Figure 2: Memory comparison for the Kosarak database

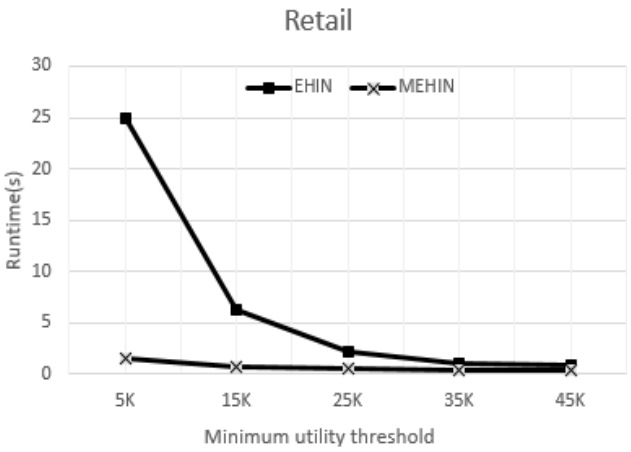

Figure 3: Runtime comparison for the Retail database

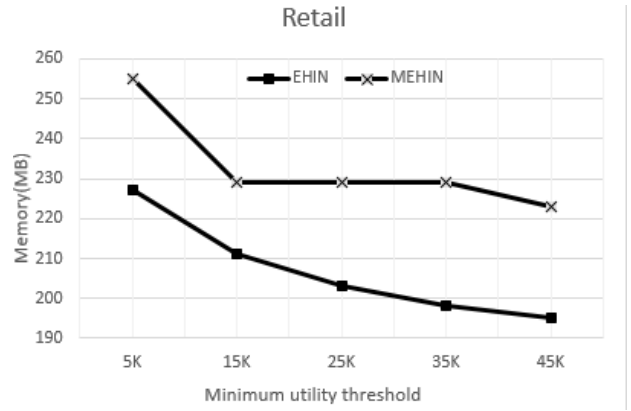

Figure 4: Memory comparison for the Retail database

To evaluate the impact of the μ threshold on memory consumption, the peak memory usage of each algorithm was recorded and averaged after several executions. The results show that EHIN has much lower memory consumption than MEHUI over all the tested μ thresholds for each database.

The difference in time and memory of EHIN and MEHIN algorithms is because the MEHIN algorithm uses a P-set structure to limit the time to scan the database in steps 4 of Algorithm 2 and step 4 of Algorithm 3. MEHIN algorithm only scans transaction su containing promise items and ignores other transactions to save time, However, this process consumes memory to store information of P-set, so MEHIN algorithm will consume more memory than EHIN.

5. Conclusion and Future Word

This work presented an efficient approach to mine high-utility itemsets with negative utility, introducing the MEHIN algorithm for mining HUIs with negative utility. MEHIN takes as input a quantitative transaction database with the negative utility of items. MEHIN relies on upper bounds on the utility, pruning strategies and P-set structure, which reduces the number of itemsets that MEHIN visits in the search space. MEHIN adopts these techniques to discover the complete set of high utility itemsets in quantitative transaction databases. Several experiments were conducted on real retail databases to compare the performance of EHIN with the MEHIN.

For future work, we aim at extending MEHIN for mining maximal, closed high utility itemsets on databases with negative utility and applying parallel computing frameworks to further boost its performance.

References

Agrawal, R. and Srikant, R. 1994. Fast Algorithms for Mining Association Rules in Large Databases, in Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499.

Chu, C.-J., Tseng, V. S. and Liang, T. 2009. An efficient algorithm for mining high utility itemsets with negative item values in large databases, Applied Mathematics and Computation, 215(2), pp. 767–778.

Han, J. et al. 2004. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, Data Mining and Knowledge Discovery, 8(1), pp. 53–87.

Lin, C.-W., Hong, T.-P. and Lu, W.-H. 2011. An effective tree structure for mining high utility itemsets, Expert Systems with Applications, 38(6), pp. 7419–7424.

Lin, J. C.-W., Fournier-Viger, P. and Gan, W. 2016. FHN: An efficient algorithm for mining high-utility itemsets with negative unit profits, Knowledge-Based Systems, 111, pp. 283–298.

Liu, M. and Qu, J. 2012. Mining high utility itemsets without candidate generation, in Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12. New York, New York, USA: ACM Press, p. 55.

Liu, Y., Liao, W. K. and Choudhary, A. 2005. A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets, in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer-Verlag (PAKDD'05), pp. 689–695.

Nguyen, L. T. T. et al. 2019. Mining high-utility itemsets in dynamic profit databases, Knowledge-Based Systems, 175, pp. 130–144.

Singh, K. et al. 2018. Mining of high-utility itemsets with negative utility, Expert Systems, 35(6), pp. 1–23.

Yao, H., Hamilton, H. J. and Butz, C. J. 2004. A Foundational Approach to Mining Itemset Utilities from Databases, in Proceedings of the 2004 SIAM International Conference on Data Mining. Philadelphia, PA: Society for Industrial and Applied Mathematics, pp. 482–486.

Zaki, M. 2000. Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge and Data Engineering, 12(3), pp. 372–390.

Zida, S. et al. 2017. EFIM: a fast and memory efficient algorithm for high-utility itemset mining, Knowledge and Information Systems, 51(2), pp. 595–625.