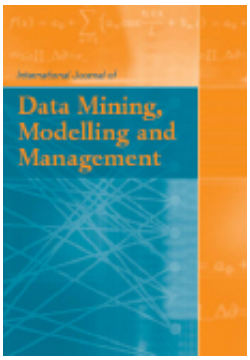


International Journal of Data Mining, Modelling and Management

ISSN online: 1759-1171 - ISSN print: 1759-1163
https://www.inderscience.com/ijdmmm

HARUIM: high average recent utility itemset mining

Mathe John Kenny Kumar, Dipti Rana

DOI: 10.1504/IJDMMM.2024.10055782

Article History:

Received:	24 November 2022
Last revised:	17 March 2023
Accepted:	19 March 2023
Published online:	22 January 2024

HARUIM: high average recent utility itemset mining

Mathe John Kenny Kumar* and Dipti Rana

Department of Computer Engineering,
Sardar Vallabhbhai National Institute of Technology,
Surat, Gujarat, India
Email: ds18co001@coed.svnit.ac.in
Email: dpr@coed.svnit.ac.in
*Corresponding author

Abstract: High utility itemset mining (HUIM) discovers itemsets that are profitable in nature. Previously, the recency of an itemset was determined by adding the recency of each transaction of an itemset. A major disadvantage of this method is that some transactions of an itemset which are very recent can cause the whole itemset to be recent. To overcome this limitation, we present a novel measure called *average recency* to mine recent and high utility itemsets. Average recency upper-bound (arub) and estimated recency co-occurrence structure (ERCS) are proposed to prune unpromising itemsets. A variation of list structure known as average recent utility list (ARUL) has been created to hold data regarding utility and recency of itemsets. Through a series of comprehensive experimentation carried out on both real as well as synthetic datasets, it has been demonstrated that the proposed system surpasses the baseline algorithm in runtime, memory utilisation, and candidate generation.

Keywords: data mining; high utility itemset mining; HUIM; recency; average recency; list structure; pattern mining; EUCS; knowledge engineering; candidate generation.

Reference to this paper should be made as follows: Kumar, M.J.K. and Rana, D. (2024) 'HARUIM: high average recent utility itemset mining', *Int. J. Data Mining, Modelling and Management*, Vol. 16, No. 1, pp.66–100.

Biographical notes: Mathe John Kenny Kumar is an Assistant Professor at the Universal College of Engineering, Vasai, India. He is pursuing his PhD from the Sardar Vallabhbhai National Institute of Technology, Surat, Gujarat, India. His research interests include data mining, frequent pattern mining, and high utility itemset mining. He is and author of a great deal of research studies published at national and international journals.

Dipti Rana received her PhD from the Sardar Vallabhbhai National Institute of Technology, Surat, Gujarat, India. Her research interests include data mining, machine learning, and natural language processing. She has published research papers at national and international journals, conference proceedings as well as chapters of books.

1 Introduction

The topic of data mining has been researched effectively for many years (Chen et al., 1996). The process of extraction of itemsets which are frequent in nature is called frequent pattern mining (FPM) (Gan et al., 2017; Han et al., 2000; Shanthi et al., 2016). Rules that are extracted from frequent itemsets are called association rules (Agrawal et al., 1994). FPM has large number of applications in real world such as marketing of certain products to increase the profitability of an organisation. As FPM extracts itemsets that are only frequent in nature, these itemsets are sometimes not useful as they do not mine itemsets that are profitable to the organisation. Unlike FPM which only computes itemsets that are frequent in nature, HUIM (Ahmed et al., 2009; Gan et al., 2018b; Liu and Qu, 2012; Liu et al., 2005b) is used to compute itemsets that are profitable in nature. When the usefulness of an itemset exceeds a particular minimum threshold value set by the user, it is considered to have high utility. HUIM has its advantages over FPM. For example, an itemset containing milk and cereals which appears together ten times in the database are less profitable than an itemset containing a camera and a laptop which appear only once in the database. Unlike FPM, HUIM is not downward closed, therefore exponential candidate itemsets are generated. The field of HUIM has been significantly researched and investigated. As far as our knowledge, RUP (Gan et al., 2019) is the only algorithm which has used *recency* and *utility* as constraints to mine recent as well as high utility itemsets (HUIs). The proposed system uses a novel measure called *average recency* to mine recent as well as HUIs.

1.1 Motivation

The proposed system is driven by the following motivation.

1 The proposed HARUIM mines itemsets that are recent as well as high utility, therefore trending itemsets can be extracted.

2 The itemsets that are trending can be marketed aggressively for greater profits.

3 As HARUIM algorithm mines trending itemsets the store owner or the manager can order large amount of stock without worrying about the stock being unsold.

4 Average recency presents a fair representation of the recency of the itemsets.

1.2 Contributions

This research has the following contributions.

- High average recent utility itemset mining (HARUIM) is the first work to propose *average recency* as a measure to mine recent as well as HUIs.
- A modified average recent utility list (ARUL) structure stores details about recency and utility of itemsets.
- Two novel pruning techniques, $arub$ and $ERCS$ are used to eliminate unpromising itemsets.

- Comprehensive experimentation performed on various datasets showcase the significance of HARUIM algorithm.

1.3 Organisation of the paper

Section 2 describes various algorithms related to HUIM and Recent-HUIM. Section 3 lists various definitions used throughout this research. Section 4 lists three proposed algorithms for extraction of high average recent utility itemsets (HARUIs). Section 5 is used to test the HARUIM algorithm using various real and synthetic datasets. Section 6 explains in brief about the work done and potential applications of the proposed system.

2 Related work

The process of extracting itemsets with a utility value surpassing the user-defined threshold is accomplished through the use of HUIM. There are various real-time applications of HUIM such as market basket analysis, web mining, stream processing, bio-medicine (Erwin et al., 2008; Li et al., 2018; Lin et al., 2016; Gan et al., 2018a; Ahmed et al., 2011, 2009; Shie et al., 2011, 2013; Golab and Özsu, 2003; Chi et al., 2004; Liu et al., 2013). Extensive amount of work has been done in the field of HUIM. A $two-phase$ algorithm (Liu et al., 2005b), used an apriori-based approach to mine HUI's in two phases. During the initial stage, candidate itemsets were extracted using twu as an upper-bound. In the second stage, a thorough scan of the database was needed to extract all the HUI's. $UP-growth$, (Tseng et al., 2010) algorithm was proposed to mine HUI's using a tree-based data structure. In the first phase various strategies such as discarding local unpromising (DLU) were used to eliminate unpromising itemsets. A final database scan was required to mine HUI's from the candidates which were generated from the utility-pattern (UP-growth) tree structure.

$HUI-miner$ (Liu and Qu, 2012) was introduced to extract the HUI's in just one phase. A list data structure was used to store information about itemsets. A recursive approach was used to mine HUI's in just two database scans. FHM (Fournier-Viger et al., 2014a) algorithm proposed a new matrix-based structure called $EUCS$ to prune unpromising two-itemsets. Co-related HUIs were extracted by using a bond measure (Fournier-Viger et al., 2016a). Minimum co-relation was used as an constraint to mine co-related HUIs. In some instances, a store might sell a particular item at a loss so that a profit can be made on the overall itemsets bought by the customer. FHN (Fournier-Viger, 2014) algorithm was developed to include items that are sold for a loss. The items that are sold for a loss are represented using a negative utility. In order to mine HUI's in a more efficient manner $EFIM$ (Zida et al., 2015) algorithm was introduced. This algorithm employs a technique to project and merge transactions to lower the expense of database scans. Several algorithms have been put forth to mine concise representations of HUI's. $CHUI-miner$ (Dam et al., 2019) used three pruning strategies to prune itemsets that are not promising. An algorithm to mine Top-K utility itemsets was proposed (Tseng et al., 2015). PHM (Fournier-Viger et al., 2016b) was introduced for the purpose of mining HUI's that are periodically purchased by customers. An algorithm to mine HUI's with on-shelf time periods of items was proposed with items having negative or positive profit (Fournier-Viger and Zida, 2015). $FHM+$ (Fournier-Viger et al., 2016c) algorithm mines HUI's with length constraints.

It uses length upper-bound reduction as well as two upper-bounds to remove itemsets that are not promising.

Table 1 HUIM

Algorithm	*Explanation*	*No. of phases*	*Data structure*
$Two-phase$ (Liu et al., 2005b)	Uses two phases to mine HUI's	Two	Apriori-based
$UP-growth$ (Tseng et al., 2010)	Uses a tree structure to mine HUI's	Two	Tree
$HUI-miner$ (Liu and Qu, 2012)	Algorithm uses utility list structure to mine HUI's	One	List
FHM (Fournier-Viger et al., 2014a)	A novel EUCS structure was used to prune unpromising 2-itemsets	One	List
$FCHM$ (Fournier-Viger et al., 2016a)	Algorithm mines co-related HUI's using a bond measure	One	List
FHN (Fournier-Viger, 2014)	Algorithm mines HUI's with both positive and negative utilities	One	List
$EFIM$ (Zida et al., 2015)	Algorithm proposes techniques to merge and project transactions	One	Horizontal
$CHUI-miner$ (Dam et al., 2019)	Efficient algorithm to mine closed HUI's using novel pruning strategies	One	List
TKO (Tseng et al., 2015)	Algorithm is used to mine top-K HUI's with using a minimum utility threshold	One	List
PHM (Fournier-Viger et al., 2016b)	Algorithm is used to mine periodic HUI's	One	List
$FOSHU$ (Fournier-Viger and Zida, 2015)	Algorithm is used to mine HUI's with on-shelf time periods of items	One	List
$FHM+$ (Fournier-Viger et al., 2016c)	Algorithm efficiently mines HUI's with length constraints	One	List
$TPAU$ (Hong et al., 2009)	Algorithm is used to mine high average utility itemsets (HAUI's)	Two	Tree
$EHAUPM$ (Lin et al., 2017)	Efficient algorithm to mine HAUI's using novel pruning techniques	One	List

High average utility itemset mining (HAUIM) (Kumar and Rana, 2021) gives an accurate representation of the utility of an itemset. $TPAU$ (Hong et al., 2009) used a breadth first approach to mine high average utility itemsets (HAUI's) in two phases. $EHAUPM$ (Lin et al., 2017) algorithm used a depth first approach. It used a utility list structure to mine all HAUI's in one phase. The algorithms that are detailed in Section 2 are tabulated in Table 1.

As for as our knowledge some amount of research has been done in the field of *recent-HUIM*. $UDHUP-apriori$ and $UDHUP-list$ (Lin et al., 2015) algorithms

mine up-to-date patterns. Up-to-date HUIs represent itemsets that are not only of high utility but also recent in nature. RUP (Gan et al., 2019) algorithm mines recent HUIs using a recent utility list data structure. Table 2 represents all algorithms which mine recent HUI's.

Table 2 Recent-HUIM

Algorithm	*Explanation*	*No. of phases*	*Data structure*
$UDHUP-apriori$ (Lin et al., 2015)	Algorithm mines up-to-date HUI's using an level-based approach	Two	Apriori
$UDHUP-list$ (Lin et al., 2015)	Algorithm uses a utility list structure to mine up-to-date HUI's	One	List
RUP (Gan et al., 2019)	Algorithm mines recent HUI's using recency as an additional measure	One	List

3 Definitions and problem statement

This sections presents a list of preliminaries that pertain to HARUIM. Transaction database Table 3 and external utility Table 4 are used to define and illustrate various notations related to HARUIM.

Table 3 Transaction database

Tr_d	*Timestamps*	*Items with quantities*
Tr_1	08/07/2021 07:15	a:1, b:1, e:2
Tr_2	08/07/2021 08:45	b:2, e:1
Tr_3	08/07/2021 09:55	c:1, d:1
Tr_4	08/07/2021 10:25	a:2, d:1
Tr_5	08/07/2021 12:52	a:1, c:1, d:3
Tr_6	08/07/2021 13:20	b:2, d:1
Tr_7	08/07/2021 14:33	b:3, c:1
Tr_8	08/07/2021 16:24	a:1, b:3
Tr_9	08/07/2021 17:50	a:2, b:2, d:1
Tr_{10}	08/07/2021 20:10	d:2, e:2

3.1 Definitions

Consider $IT = \{it_1, it_2, ..., it_m\}$, be m unique items and let DB be a quantitative database, where $DB = \{Tr_1, Tr_2, ..., Tr_n\}$. Every transaction, $Tr_d \in DB$ and $1 \leq d \leq n$, where d is a unique identifier for each transaction called Tr_d. Each item $it_p (1 \leq p \leq m)$, has the quantity purchased which is represented by internal utility. It is denoted by $iu(it_p, Tr_d)$. External utility of an item is denoted by $eu(it_p)$.

Item utility for transaction Tr_d, can be defined as, $ut(it_p, Tr_d) = iu(it_p, Tr_d) \times eu(it_p)$. For example, from Tables 3 and 4, $iu(a, Tr_1) = 1$, $eu(a) = 2$, $ut(a, Tr_1) =$

$1 \times 2 = 2$. A set of k unique items $X = \{it_1, it_2, ..., it_k\}$, where $X \subseteq IT$, is called a k-itemset, where k refers to the length of the itemset X, $X \subseteq Tr_d$.

Table 4 External utilities

Item	*Profit*
a	2
b	3
c	1
d	3
e	2

Definition 3.1 (Yao et al., 2004): Utility of an itemset X, in a transaction Tr_d, is denoted by $ut(X, Tr_d)$, and is defined by,

$$ut(X, Tr_d) = \sum_{it_p \in X \wedge X \subseteq Tr_d} ut(it_p, Tr_d). \tag{1}$$

Using Tables 3 and 4, $ut(ab, Tr_1) = 1 \times 2 + 1 \times 3 = 5$.

Definition 3.2 (Yao et al., 2004): Utility of an itemset X, in a transactional database DB, is denoted as $ut(X)$, and is defined as,

$$ut(X) = \sum_{X \subseteq Tr_d \wedge Tr_d \in DB} ut(X, Tr_d). \tag{2}$$

Using Tables 3 and 4, $ut(ab) = ut(ab, Tr_1) + ut(ab, Tr_8) + ut(ab, Tr_9) = 5 + 11 + 10 = 26$.

Definition 3.3 (Yao et al., 2004): Transaction utility tu, of a transaction Tr_d, is denoted as $tu(Tr_d)$, and is defined as,

$$tu(Tr_d) = \sum_{it_p \in Tr_d \wedge Tr_d \in DB} ut(it_p, Tr_d). \tag{3}$$

From Tables 3 and 4, $tu(Tr_1) = ut(a, Tr_1) + ut(b, Tr_1) + ut(e, Tr_1) = 2 + 3 + 4 = 9$.

Definition 3.4 (Yao et al., 2004): The sum of transaction utilities of all the transactions in a database DB, where $Tr_d \in DB$, is denoted as $ut(DB)$, and is defined as,

$$ut(DB) = \sum_{Tr_d \in DB} ut(Tr_d). \tag{4}$$

From Tables 3 and 4, $ut(DB) = 93$.

Definition 3.5 (Yao et al., 2004): Given a minimum utility threshold by the user δ, minimum utility, i.e., min_util, is defined as,

$$min\_util = \delta \times ut(DB). \tag{5}$$

Given minimum utility threshold $\delta = 0.1$, from Tables 3 and 4, $min\_util = 0.1 \times 93 = 9.3$.

Definition 3.6 (Liu et al., 2005a): The transaction weighted utilisation (twu) of an itemset X, in a database DB, is defined by the sum of utilities of all the transactions containing X in DB. Transaction weighted utilisation is denoted by $twu(X)$, and is defined as,

$$twu(X) = \sum_{X \subseteq Tr_d \wedge Tr_d \in DB} tu(Tr_d). \tag{6}$$

$twu(ab) = tu(Tr_1) + tu(Tr_8) + tu(Tr_9) = 33.$

Definition 3.7: EUCS is defined as a set of triplets of the form $(X, Y, Z) \in IT \times IT \times R$. A triplet (X, Y, Z) is denoted by, $twu(X, Y) = Z$, where X, Y are itemsets and Z represents the twu value of (X, Y) (Fournier-Viger et al., 2014a).

Theorem 3.1: A full search space to mine HARUI's is depicted using a set-enumeration tree (Rymon, 1992) where items (nodes) are sorted in $\prec$ order of their twu values.

Proof: From the set-enumeration tree (Rymon, 1992) and RUP-tree (Gan et al., 2019), $2^m - 1$ itemsets (nodes) can be formed from the items (nodes) in IT. Let m be the number of distinct items (nodes) in IT. Using a depth first search approach, all the supersets of items (nodes) in IT can be enumerated according to their transaction weighted utilisation (twu). This representation is complete and correct as all the supersets of the root node are explored. Items arranged according to twu values are, $c \prec e \prec a \prec d \prec b$. The number of nodes in the search space are $2^5 - 1 = 31$. The complete search space is represented using Figure 4.

Definition 3.8 (Liu et al., 2005a): An itemset X, is called a high transaction-weighted utilisation itemset (HTWUI), if $twu(X) \geq min\_util$. As $twu(ab) = tu(Tr_1) + tu(Tr_8) + tu(Tr_9) = 33 > 9.3(min\_util)$, therefore $\{ab\}$ is a HTWUI.

Theorem 3.2 (Liu et al., 2005a): Let X^k, be a k-itemset and X^{k-1}, be a $k-1$ itemset such that $X^{k-1} \subset X^k$, if X^k is a HTWUI, X^{k-1} is also a HTWUI.

Proof: Let Tr_{X^k}, be a set of transactions containing itemset X^k and $Tr_{X^{k-1}}$ be a set of transactions containing itemset X^{k-1}. As $X^{k-1} \subset X^k$, $Tr_{X^{k-1}}$ is a superset of Tr_{X^k}.

$$\begin{aligned} twu(X^{k-1}) &= \sum_{X^{k-1} \subseteq Tr_d \in DB} tu(Tr_d) \\ &\geq \sum_{X^k \subseteq Tr_c \in DB} tu(Tr_c) \end{aligned}$$

$$= twu(X^k)$$
$$\geq min\_util.$$

Definition 3.9 (Yao et al., 2004): An itemset X, in a transactional database DB, is called a HUI, if its utility $ut(X)$, is not less than min_util.

$$HUI \leftarrow \{X | ut(X) \geq min\_util\}. \tag{7}$$

where $X \subseteq Tr_d \wedge X \in DB$. $ut(ab) = 26 > 9.3$, therefore itemset $\{ab\}$ is a HUI.

Theorem 3.3 (Liu et al., 2005a): Let $HTWUI$ be all the HTWUI's in a transactional database DB, and let HUI represent all the HUIs in a database DB, then $HUI \subseteq HTWUI$.

Proof: For all $X \in HUI$, if X is a HUI, then,

$$\begin{aligned} min\_util \leq ut(X) &= \sum_{X \subseteq Tr_d} ut(X, Tr_d) \\ &= \sum_{X \subseteq Tr_d} \sum_{it_p \in X} ut(it_p, Tr_d) \\ &\leq \sum_{X \subseteq Tr_d} \sum_{it_p \in Tr_d} ut(it_p, Tr_d) \\ &= \sum_{X \subseteq Tr_d} tu(X, Tr_d) \\ &= twu(X). \end{aligned}$$

Therefore an itemset, $X \in HUI$, is also a $HTWUI$.

Definition 3.10: Recency of an itemset X, in a transaction Tr_d, is denoted as $rec(X, Tr_d)$, and is defined as,

$$rec(X, Tr_d) = rec(Tr_d) = Tr_d/n. \tag{8}$$

where n represents the number of transactions in a database DB, Tr_d represents a transaction id. From Tables 3 and 4, $rec(abd, Tr_9) = rec(Tr_9) = 9/10 = 0.9$, $rec(a, Tr_9) = rec(Tr_9) = 9/10 = 0.9$, $rec(be, Tr_2) = rec(Tr_2) = 2/10 = 0.2$.

Definition 3.11: Recency of an itemset X, in a database DB, is denoted by $rec(X)$, and is defined as,

$$rec(X) = \sum_{X \subseteq Tr_d \wedge Tr_d \in DB} rec(X, Tr_d). \tag{9}$$

From Tables 3 and 4, $rec(abd) = rec(abd, Tr_9) = 0.9$, $rec(a) = rec(a, Tr_1) + rec(a, Tr_4) + rec(a, Tr_5) + rec(a, Tr_8) + rec(a, Tr_9) = 0.1 + 0.4 + 0.5 + 0.8 + 0.9 = 2.7$, $rec(be) = rec(be, Tr_1) + rec(be, Tr_2) = 0.1 + 0.2 = 0.3$.

Unlike the *RUP* (Gan et al., 2019) algorithm the proposed HARUIM algorithm does not use the 'decay factor' to calculate the recency of an itemset. Recency measure has been inspired by RUP (Gan et al., 2019) algorithm.

Definition 3.12 (Agrawal et al., 1994): Support count is the number of times an itemset X, occurs in a transactional database DB. It is denoted by $supportcount(X)$ and is defined as:

$$supportcount(X) = Count(X, DB) \tag{10}$$

where $X \subseteq Tr_d \wedge Tr_d \in DB$. From Tables 3 and 4, $supportcount(abd) = 1$, $supportcount(a) = 5$, $supportcount(be) = 2$.

Definition 3.13: Average recency of an itemset X, in a transactional database DB, is denoted by $avgrec(X)$, and is defined as:

$$avgrec(X) = rec(X)/supportcount(X). \tag{11}$$

where $X \subseteq Tr_d \wedge Tr_d \in DB$. $avgrec(abd) = 0.9/1 = 0.9$, where recency of $rec(abd) = 0.9$ and $supportcount(abd) = 1$, $avgrec(a) = 2.7/5 = 0.54$, where recency of $rec(a) = 2.7$ and $supportcount(a) = 5$, $avgrec(be) = 0.3/2 = 0.15$, where recency of $rec(be) = 0.3$ and $supportcount(be) = 2$.

Property 1: The average recency of an itemset X in a transactional database DB is:

1 Not less than the average recency of its subsets (or).

2 Not greater than the average recency of its subsets.

Definition 3.14: The average recency upper bound (arub) of an itemset X, in a transactional database DB, is denoted by $arub(X)$, and is defined as,

$$arub(X) = \max(rec(X, Tr_d)). \tag{12}$$

where $X \subseteq Tr_d \wedge Tr_d \in DB$.

$$arub(c) = \max\{rec(Tr_3, Tr_5, Tr_7)\} = \max\{0.3, 0.5, 0.7\} = 0.7.$$

Definition 3.15: User defined minimum recency threshold is denoted by min_rec, and is defined as,

$$0 \leq min\_rec \leq 1 \tag{13}$$

For example, we consider $min\_rec = 0.8$.

Definition 3.16: ERCS is defined as a set of triplets of the form $(X, Y, Z) \in IT \times IT \times R$. A triplet (X, Y, Z) is denoted by, $arub(X, Y) = Z$, where X, Y are itemsets and Z represents the $arub$ value of (X, Y) (Fournier-Viger et al., 2014a).

Definition 3.17: An itemset X is called a high recency weighted itemset (HRWI), if the average recency upper bound of X is greater than minimum recency threshold. It is denoted as $HRWI(X)$, and is defined as,

$$HRWI(X) \leftarrow \{X | arub(X) \geq min\_rec\}. \tag{14}$$

where $X \subseteq Tr_d \wedge Tr_d \in DB$.

$$\begin{aligned} arub(a) &= \max\{rec(Tr_1, Tr_4, Tr_5, Tr_8, Tr_9)\} \\ &= \max\{0.1, 0.4, 0.5, 0.8, 0.9\} = 0.9. \end{aligned}$$

As $0.9 > 0.8(min\_rec)$, $\{a\}$ is a HRWI.

Theorem 3.4: Let X^k, be a k-itemset and X^{k-1}, be a $k-1$ itemset such that $X^{k-1} \subset X^k$, if X^k is high recency-weighted itemset, X^{k-1} is also a high recency-weighted itemset.

Proof: Let Tr_{X^k}, be a set of transactions containing itemset X^k and $Tr_{X^{k-1}}$ be a set of transactions containing itemset X^{k-1}. As $X^{k-1} \subset X^k$, $Tr_{X^{k-1}}$ is a superset of Tr_{X^k}.

$$\begin{aligned} arub(X^{k-1}) &= \{X^{k-1}, X^{k-1} \subseteq Tr_d \in DB \mid max(rec(X^{k-1}, Tr_d))\} \\ &\geq \{X^k, X^k \subseteq Tr_d \in DB \mid max(rec(X^k, Tr_d))\} \\ &= arub(X^k) \\ &\geq min\_rec. \end{aligned}$$

Corollary 3.4.1: The $arub$ of an itemset X is not less than its average recency, i.e $arub(X) \geq avgrec(X)$.

Corollary 3.4.2: Anti-monotonic property is satisfied by $arub$. Let X and Y be two itemsets, if $X \subset Y$, then $arub(X) \geq arub(Y)$.

Corollary 3.4.3: If $arub(X) < min\_rec$, where X is an itemset, then the itemset X is a low average recent itemset along with all its supersets.

Definition 3.18: An itemset X, is said to be high average recent itemset (HARI) if the average recency $avgrec(X)$, of the itemset X, is not less than the user defined minimum recency threshold min_rec.

$$HARI \leftarrow \{X | avgrec(X) \geq min\_rec\}. \tag{15}$$

where $X \subseteq Tr_d \wedge Tr_d \in DB$. $avgrec(abd) = 0.9/1 = 0.9 > 0.8(min\_rec)$, where $rec(abd) = 0.9$, $supportcount(abd) = 1$ therefore itemset $\{abd\}$ is a HARI. $avgrec(a) = 2.7/5 = 0.54 < 0.8(min\_rec)$, where $rec(a) = 2.7$, $supportcount(a) = 5$ therefore itemset $\{a\}$ is not a HARI. $avgrec(be) = 0.3/2 = 0.15 < 0.8(min\_rec)$, where $rec(be) = 0.3$, $supportcount(be) = 2$, therefore itemset $\{be\}$ is not a HARI.

Theorem 3.5: Let $HRWI$ be all the high recency-weighted utilisation itemsets in a database DB, and let $HARI$ represent all the HARIs in a database DB, then $HARI \subseteq HRWI$.

Proof: For each itemset $X \in HARI$, X is also a $HRWI$.

$$\begin{aligned} min\_rec \leq avgrec(X) &= \sum_{X \subseteq Tr_d} rec(X, Tr_d)/supportcount(X) \\ &\leq \{X, X \subseteq Tr_d \mid max(rec(X, Tr_d)\} \\ &= arub(X). \end{aligned}$$

Therefore an itemset, $X \in HARI$, is also a $HRWI$.

Property 2: Using Theorems 3.2, 3.3, 3.4, 3.5, for any itemset X, if $twu(X) < min\_util \vee arub(X) < min\_rec$, then the itemset X can be:

1 A low utility itemset along with all its supersets (or)

2 A low average recency itemset along with all its supersets (or)

3 A low utility and a low average recency itemset along with all its supersets.

Definition 3.19: An itemset X, is called a high average recent utility upper bound itemset (HARUUI), if transaction weighted utilisation $twu(X)$, is not less than min_util, and the average recency upper-bound $arub(X)$, is not less than min_rec. It is denoted as $HARUUI(X)$ and is defined as,

$$HARUUI(X) \leftarrow \{X | twu(X) \geq min\_util \wedge arub(X) \geq min\_rec\} \tag{16}$$

where $X \subseteq Tr_d \wedge Tr_d \in DB$. Consider min_util = 9.3 and min_rec = 0.8, $twu(bd)$ = 22, $arub(bd)$ = max{$rec(Tr_6)$, $rec(Tr_9)$} = max{0.6, 0.9} = 0.9. As $twu(bd)$ = 22 > 9.3(min_util) and $arub(bd)$ = 0.9 > 0.8(min_rec), itemset {bd} is a HARUUI.

Definition 3.20: An itemset X is said to be a HARUI, if its utility $ut(X)$, is not less than min_util, and its average recency $avgrec(X)$, is not less than min_rec.

$$HARUI(X) \leftarrow \{X | ut(X) \geq min\_util \wedge avgrec(X) \geq min\_rec\}. \tag{17}$$

where $X \subseteq Tr_d \wedge Tr_d \in DB$.

$ut(de) = 10 > 9.3(min\_util)$, $avgrec(de) = 1/1 = 1 > 0.8(min\_rec)$, where $rec(de) = 1$ and $supportcount(de) = 1$, therefore itemset {de} is a HARUI.

Definition 3.21: The ARUL of an itemset X, in a transactional database DB, is a set of rows, such that each row contains $(Tr_d, iutil, rutil, rec)$ for each transaction Tr_d, containing X. The element $iutil$ (Liu and Qu, 2012) of a row represents the utility of an itemset X in Tr_d, and is denoted by $ut(X, Tr_d)$. The element $rutil$ (Liu and Qu, 2012) of a row for a particular transaction Tr_d, is defined as, $\sum_{it_p \in Tr_d \wedge it_p \notin X} ut(it_p, Tr_d)$. A separate row of ARUL for an itemset X is denoted by $(arub)$, and is defined as, $X.aurb = max(rec(X, Tr_d))$.

The ARUL of $\{a\}$ is $\{(Tr_1, 2, 7, 0.1)\ (Tr_4, 4, 3, 0.4)\ (Tr_5, 2, 10, 0.5)\ (Tr_8, 2, 9, 0.8)\ (Tr_9, 4, 9, 0.9)\ (0.9)\}$. ARUL of an itemset $\{ab\}$ is $\{(Tr_1, 5, 4, 0.1)\ (Tr_8, 11, 0, 0.8)\ (Tr_9, 10, 3, 0.9)\ (0.9)\}$.

Definition 3.22 (Liu and Qu, 2012): Given ARUL, the sum of utilities of an itemset X is denoted by $X.IU$, and is defined as,

$$X.IU = \sum_{X \subseteq Tr_d \wedge Tr_d \in DB} X.iutil. \tag{18}$$

$a.IU = 2 + 4 + 2 + 2 + 4 = 14$ and $ab.IU = 5 + 11 + 10 = 26$.

Definition 3.23 (Liu and Qu, 2012): Given ARUL of an itemset X, the sum of remaining utility of X is defined as,

$$X.RU = \sum_{X \subseteq Tr_d \wedge Tr_d \in DB} X.rutil. \tag{19}$$

$a.RU = 3 + 3 + 9 + 9 + 9 = 33$ and $ab.RU = 0 + 0 + 0 = 0$.

Definition 3.24: Given ARUL of an itemset X, the sum of recency is denoted by $X.SUMREC$ and is defined as,

$$X.SUMREC = \sum_{X \subseteq Tr_d \wedge Tr_d \in DB} X.rec \tag{20}$$

Given ARUL of an itemset X, the average recency is denoted by $X.AVGREC$, and is defined as,

$$X.AVGREC = \{X|X.SUMREC/X.supportcount\} \tag{21}$$

where $X \subseteq Tr_d \wedge Tr_d \in DB$ and $X.supportcount$ is the count of an itemset X occurs in a transactional database DB.

$a.AVGREC = (0.1 + 0.4 + 0.5 + 0.8 + 0.9)/5 = 0.54$ and $ab.AVGREC = (0.1 + 0.8 + 0.9)/3 = 0.6$.

4 Proposed HARUIM algorithm

The current section presents a novel algorithm called HARUIM for extracting all HARUIs. The ARULs of all extensions of each item $it_p \in IT$, are processed recursively using a method that follows a depth-first approach. A modified ARUL structure is used to store utility, recency, $arub$ values. Additionally, two new strategies for pruning are employed to remove unpromising itemsets. Algorithms 1, 2 and 3 are explained with a complete and comprehensive running example.

Step 1 Scan the transactional database DB, to compute the twu (Liu et al., 2005b) and $arub$ of all items $it_p \in IT$, Definitions 3.6 and Definition 3.14 [Algorithm 1, line 3].

Algorithm 1 HARUIM-algorithm

Input:DB, min_util, min_rec
Output:Set of all $HARUIs$
Scan DB to compute twu and $arub$ of each item $it_p \in IT$;
Find $IT(twu(it_p) \geq min\_util \wedge arub(it_p) \geq min\_rec)$;
Arrange the items in IT in $\prec$ order of twu values;
Revise DB according to $\prec$ order of twu values;
Scan DB to build $X.ARUL$ of each item $it_p \in IT$ and construct $EUCS$ and $ERCS$ structures;
search-HARUIM $(\phi, IT, min\_util, min\_rec, EUCS, ERCS)$;
return HARUIs.

Algorithm 2 Search-HARUIM

Input:$X, extensionsOfX, min\_util, min\_rec, EUCS, ERCS$
Output:Set of all HARUIs
for itemset $X_a \in extensionsOfX$ **do**
    calculate the values of $X_a.AVGREC$, $X_a.IU$, $X_a.RU$ from $X_a.ARUL$;
    if $(X_a.IU \geq min\_util) \wedge (X_a.AVGREC \geq min\_rec)$ **then**
        $HARUI \leftarrow HARUI \cup X_a$;
    end if
    if $(X_a.IU + X_a.RU \geq min\_util) \wedge (X_a.arub \geq min\_rec)$ **then**
        $extensionsOfX_a \leftarrow \emptyset$;
        for $X_b \in extensionsOfX$ such that $a \prec b$ **do**
            if $\exists arub(ab) \in ERCS \wedge arub(ab) \geq min\_rec$ **then**
                if $\exists twu(ab) \in EUCS \wedge twu(ab) \geq min\_util$ **then**
                    $X_{ab}.ARUL \leftarrow construct(X, X_a, X_b)$;
                end if
            end if
        end for
        Search-HARUIM$(X_a, extensionsOfX_a, min_util, min_rec, EUCS, ERCS)$
    end if
end for

Algorithm 3 Construct procedure-ARUL

Input: X: An itemset, X_a: Extension of X with an item a, X_b: Extension of X with an item b
Output: $X_{ab}.ARUL$: ARUL of an itemset X_{ab}
set $X_{ab}.ARUL \leftarrow \emptyset$;
for $element\ E_a \in X_a.ARUL$ **do**
    if $\exists E_a \in X_b.ARUL \wedge E_a.tid = E_b.tid$ **then**
        if $X.ARUL \neq 0$ **then**
            Search $E \in X.ARUL, E.tid = E_a.tid$
            $E_{ab} \leftarrow (E_a.tid, E_a.iutl + E_b.iutil - E.iutil, E_b.rutil, E_b.rec)$;
        else
            $E_{ab} \leftarrow (E_a.tid, E_a.iutl + E_b.iutil, E_b.rutil, E_b.rec)$;
        end if
        $X_{ab}.ARUL \leftarrow X_{ab}.ARUL \cup E_{ab}$;
    end if
end for

For example, using Tables 3 and 4, the twu of items are, $\{a = 52, b = 60, c = 26, d = 55, e = 27\}$. *For example*, The $arub$ of items are, $\{a = 0.9, b = 0.9, c = 0.7, d = 1.0, e = 1.0\}$.

Step 2 *Pruning strategy (twu):* If the twu (Liu et al., 2005b) of an item $it_p \in IT$, is not greater than min_util, the item is pruned and none of its supersets are explored, Definition 3.6, Theorem 3.2 [Algorithm 1, line 4].

Though out the running example min_util, is considered as 9.3.

For example, from Step 1, it can be observed that the twu of all the items are not less than the user defined min_util of 9.3. Therefore, no item is pruned and all the items are high transaction weighted utilisation itemsets (HTWUIs), Definition 3.8.

Step 3 *Novel pruning strategy (arub):* If $arub$ of an item $it_p \in IT$, is less than min_rec, the item it_p is pruned and no other supersets of it_p are explored, Theorem 3.4, Corollary 3.4.3 [Algorithm 1, line 4].

For example, from Step 1, it can be observed that $arub(c) = 0.7 < 0.8(min\_rec)$. Therefore item $\{c\}$ is pruned and no supersets of $\{c\}$ are explored.

Though out the illustrative example min_rec is considered as 0.8.

Step 4 The *twu* values are sorted in $\prec$ order where each item $it_p \in IT$, Theorem 3.1 [Algorithm 1, line 5].

For example, items are stored in the order, $e \prec a \prec d \prec b$.

Step 5 The transactional database DB, is revised for all items $it_p \in IT$, according to $\prec$ of twu values [Algorithm 1, line 6].

The transaction database DB is revised as Table 5.

Table 5 Revised transaction database

Tr_d	*Timestamps*	*Items with quantities*
Tr_1	08/07/2021 07:15	e:2, a:1, b:1
Tr_2	08/07/2021 08:45	e:1, b:2
Tr_3	08/07/2021 09:55	d:1
Tr_4	08/07/2021 10:25	a:2, d:1
Tr_5	08/07/2021 12:52	a:1, d:3
Tr_6	08/07/2021 13:20	d:1, b:2
Tr_7	08/07/2021 14:33	b:3
Tr_8	08/07/2021 16:24	a:1, b:3
Tr_9	08/07/2021 17:50	a:2, d:1, b:2
Tr_{10}	08/07/2021 20:10	e:2, d:2

Step 6 The transactional database DB is scanned again to build the ARULs of all items, $it_p \in IT$, Definition 3.21 [Algorithm 1, line 7].

For example, using Table 5, Table 4, ARULs of $\{e\}$, $\{a\}$, $\{d\}$, $\{b\}$ are constructed.

Figure 1 ARUL of $\{e\}$, $\{a\}$, $\{d\}$, $\{b\}$ (see online version for colours)

{e}

arub=1.0			
tr_d	iutil	rutil	rec
1	4	5	0.1
2	2	6	0.2
10	4	6	1.0

{a}

arub=0.9			
tr_d	iutil	rutil	rec
1	2	3	0.1
4	4	3	0.4
5	2	9	0.5
8	2	9	0.8
9	4	9	0.9

{d}

arub=1.0			
tr_d	iutil	rutil	rec
3	3	0	0.3
4	3	0	0.4
5	9	0	0.5
6	3	6	0.6
9	3	6	0.9
10	6	0	1.0

{b}

arub=0.9			
tr_d	iutil	rutil	rec
1	3	0	0.1
2	6	0	0.2
6	6	0	0.6
7	9	0	0.7
8	9	0	0.8
9	6	0	0.9

Step 7 Estimated utility co-occurrence ($EUCS$), $ERCS$ structures are constructed, Definitions 3.7 and 3.16 [Algorithm 1, line 7].

For example, using Table 5, Table 4, $EUCS$ and $ERCS$ structures are constructed as shown in Figures 2 and 3.

Figure 2 EUCS structure

	a	b	d	e
b	33	-	-	-
d	31	22	-	-
e	9	17	10	-

Figure 3 ERCS structure

	a	b	d	e
b	0.9	-	-	-
d	0.9	0.9	-	-
e	0.1	0.2	1.0	-

Step 8 The search Algorithm 2 uses a prefix-based recursive approach to mine all the HARUIs using a set-enumeration (Rymon, 1992) tree. A total of $2^m - 1$ non-empty itemsets can be formed by items in IT containing m elements. The algorithm employs a depth-first approach to extract all the HARUIs.

Step 9 In ARUL, if $X.IU$ and $X.AVGREC$ are no less than min_util and min_rec respectively, the item is added to the set of HARUIs, Definitions 3.21, 3.24, 3.22, 3.20, Figure 1 [Algorithm 2, lines 5 and 6].

For example, Consider ARUL for item $\{e\}$, $e.IU = 10 > 9.3(min\_util)$, $e.AVGREC = 1.3/3 = 0.43 < 0.8(min\_rec)$ where $e.SUMREC = 1.3$, $e.SUPPORTCOUNT = 3$. As utility of $\{e\}$ is not less than min_util, but average recency of $\{e\}$ is less than min_rec, item $\{e\}$ is not a HARUI, Figure 1.

Step 10 Consider ARUL for an item X, if the sum of utilities $X.IU$, remaining utilities $X.RU$, is not less than min_util, and $arub$ of X is not less than min_rec, the item cannot be pruned and its supersets are explored, Property 2, Definition 3.14, 3.22, 3.23 [Algorithm 2, line 8].

For example, Consider ARUL for item $\{e\}$, $e.IU + e.RU = 27 > 9.3(min\_util)$, $e.arub = 1.0 > 0.8(min\_rec)$, therefore $\{e\}$ is not pruned and its supersets are explored, Figure 1.

Step 11 If an item $it_p \in IT$, is a $HARUUI$, its superset is explored, Definition 3.19 [Algorithm 2, line 10].

For example, item $\{e\}$, is a $HARUUI$, therefore its superset $\{ea\}$ is explored with item $\{e\}$ as a prefix, where $e \prec a$.

Step 12 *Pruning strategy (ERCS):* For itemset X, containing two items, the constructed $ERCS$ structure is checked whether $arub(X) < min\_rec$. If it is true the itemset X is pruned and no supersets of X are explored. $ERCS$ structure avoids construction of itemsets containing two items, thus avoiding expensive join operations, Definition 3.16 [Algorithm 2, line 11].

For example, $arub(ea) = 0.1 < 0.8(min\_rec)$, therefore $\{ea\}$ is pruned and no supersets of $\{ea\}$ are explored, Figure 3.

Step 13 *Pruning strategy (EUCS):* For itemset X, containing two items, the constructed $EUCS$ structure is checked whether $twu(X) < min\_util$. If it is true the itemset X is pruned and no supersets of X are explored. $EUCS$ structure avoids construction of itemsets containing two items, thus avoiding expensive join operations, Definition 3.7 [Algorithm 2, line 12].

For example: As itemset $\{ea\}$, is already pruned, $EUCS$ structure is not checked for $twu(ea)$.

Step 14 Itemset $\{ed\}$, is considered with $\{e\}$ as prefix. From the ERCS structure, $arub(ed) = 1.0 > 0.8(min\_rec)$, therefore itemset $\{ed\}$, is not pruned, Definition 3.16, Figure 3.

From the EUCS structure $twu(ed) = 10 > 9.3(min\_util)$, therefore itemset $\{ed\}$, is not pruned, Definition 3.7, Figure 2.

Itemset $\{eb\}$ is considered with $\{e\}$ as prefix. From the ERCS structure $arub(eb) = 0.2 < 0.8(min\_rec)$, therefore itemset $\{eb\}$, is pruned, Definition 3.16, Figure 3.

As itemset $\{eb\}$, is already pruned, $EUCS$ structure is not checked for $twu(eb)$, Definition 3.7.

Step 15 Construct ARUL of itemset $\{ed\}$. As $ed.IU = 10 > 9.3(min\_util)$ and $ed.AVGREC = 1.0/1 > 0.8(min\_rec)$, where $ed.SUMREC = 1.0$, $ed.SUPPORTCOUNT = 1$, therefore itemset $\{ed\}$ is a HARUI, Definitions 3.20, 3.22, 3.24, Figure 5.

Step 16 As no supersets of $\{ed\}$ exist in the search space, consider ARUL of item $\{a\}$. As $a.IU = 14 > 9.3(min\_util)$, $a.AVGREC = 2.7/5 = 0.54 < 0.8(min\_rec)$, where $a.SUMREC = 2.7$, $a.SUPPORTCOUNT = 5$, therefore item $\{a\}$ is not a HARUI, Definitions 3.20, 3.22, 3.24, Figure 1.

As $a.IU + a.RU = 47 > 9.3(min\_util)$, $a.arub = 0.9 > 0.8(min\_rec)$, therefore item $\{a\}$ is a HARUUI and its supersets are explored, Definitions 3.19, 3.21, 3.22, 3.23, Figure 1.

Step 17 Consider itemset $\{ad\}$, from the ERCS structure $arub(ad) = 0.9 > 0.8(min\_rec)$, therefore itemset $\{ad\}$, is not pruned, Definition 3.16, Figure 3.

From the EUCS structure $twu(ad) = 31 > 9.3(min\_util)$, therefore itemset $\{ad\}$ is not pruned. ARUL of itemset $\{ad\}$ is constructed, Definition 3.7, Figure 2.

Now, for itemset $\{ab\}$, from the ERCS structure, $arub(ab) = 0.9 > 0.8(min\_rec)$, therefore itemset $\{ab\}$ is not pruned, Definition 3.16, Figure 3.

From the EUCS structure $twu(ab) = 33 > 9.3(min\_util)$, therefore itemset $\{ab\}$ is not pruned. ARUL of itemset $\{ab\}$ is constructed, Definition 3.7, Figure 2.

Step 18 Construct ARUL of $\{ad\}$. As $ad.IU = 25 > 9.3(min\_util)$ but $ad.AVGREC = 1.8/3 = 0.6 < 0.8(min\_rec)$, where $ad.rec = 1.8$, $ad.supportcount = 3$, therefore itemset $\{ad\}$ is not a HARUI, Definitions 3.20, 3.22, 3.24, Figure 5.

As $ad.IU + ad.RU = 31 > 9.3(min\_util)$, and $ad.arub = 0.9 > 0.8(min\_rec)$, itemset $\{ad\}$ is a HARUUI and superset of $\{ad\}$ can be explored, Definitions 3.19, 3.21, 3.22, 3.23, Figure 5.

Step 19 For itemset $\{adb\}$, using ERCS structure for itemset $\{db\}$, $arub(db) = 0.9 > 0.8(min\_rec)$, therefore itemset $\{adb\}$ is not pruned, Definition 3.16, Figure 3.

Using EUCS structure for itemset $\{db\}$, $twu(db) = 22 > 9.3(min\_util)$, therefore itemset $\{db\}$ is not pruned and ARUL of itemset $\{abd\}$ is constructed, Definition 3.7, Figure 2.

Step 20 Construct ARUL of $\{adb\}$. As $adb.IU = 13 > 9.3(min\_util)$ and $adb.AVGREC = 0.9/1 = 0.9 > 0.8(min\_rec)$, where $adb.SUMREC = 0.9$, $adb.SUPPORTCOUNT = 1$, therefore itemset $\{adb\}$ is a HARUI, Definitions 3.20, 3.22, 3.24, Figure 5.

Step 21 Consider item $\{d\}$ in ARUL, as $d.IU = 27 > 9.3(min\_util)$ but $d.AVGREC = 3.7/6 = 0.61 < 0.8(min\_rec)$ where $d.SUMREC = 3.7$,

$d.SUPPORTCOUNT = 6$, therefore item $\{d\}$ is not a HARUI, Definitions 3.20, 3.22, 3.24, Figure 1.

Figure 4 Pruned search-space (see online version for colours)

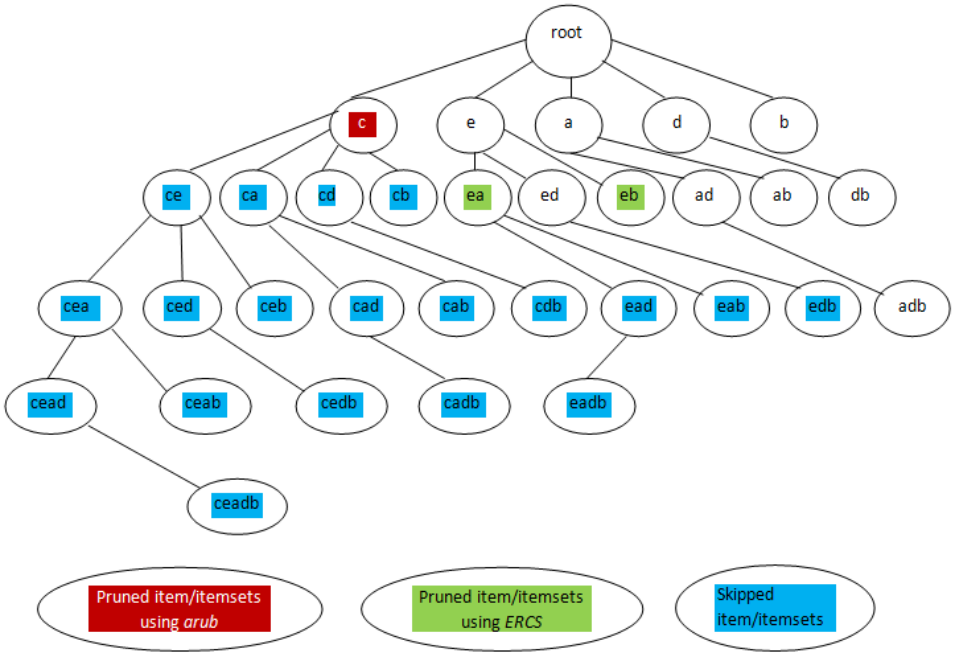


Figure 5 ARUL of $\{ed\}$, $\{ad\}$, $\{ab\}$, $\{db\}$, $\{adb\}$ (see online version for colours)

{ed}

arub=1.0

tr_d	iutil	rutil	rec
10	10	0	1.0

{ad}

arub=0.9

tr_d	iutil	rutil	rec
4	7	0	0.4
5	11	0	0.5
9	7	6	0.9

{ab}

arub =0.9

tr_d	iutil	rutil	rec
1	5	0	0.1
8	11	0	0.8
9	10	0	0.9

{db}

arub=0.9

tr_d	iutil	rutil	rec
6	9	0	0.6
9	9	0	0.9

{adb}

arub =0.9

tr_d	iutil	rutil	rec
9	13	0	0.9

As $d.IU + d.RU = 39 > 9.3(min\_util)$ and $d.arub = 1.0 > 0.8(min\_rec)$, therefore item $\{d\}$ is a HARUUI and supersets of $\{d\}$ can be explored, Definitions 3.19, 3.21, 3.22, 3.23, Figure 1.

Step 22 Construct ARUL of $\{db\}$. As $db.IU = 18 > 9.3(min\_util)$ and $db.AVGREC = 1.5/2 = 0.75 < 0.8(min\_rec)$, where $db.SUMREC = 1.5$, $db.SUPPORTCOUNT = 2$, therefore itemset $\{db\}$ is not a HARUI, Definitions 3.20, 3.22, 3.24, Figure 5.

Step 23 Consider ARUL of $\{ab\}$. As $ab.IU = 26 > 9.3(min\_util)$ and $ab.AVGREC = 1.8/3 = 0.6 < 0.8(min\_rec)$, where $ab.SUMREC = 1.8$, $db.SUPPORTCOUNT = 3$, therefore itemset $\{ab\}$ is not a HARUI, Definitions 3.20, 3.22, 3.24, Figure 5.

Figure 4 represents the pruned search space for $2^5 - 1 = 31$ itemsets. It can be observed that out of 31 itemsets, pruning strategies $arub$ and $ERCS$ avoid construction of 22 ARULs. By using transaction database Table 3, and external utility Table 4, the proposed $HARUIM$ algorithm mines itemset $\{d, e\}$ and $\{a, d, b\}$ as HARUIs.

4.1 Complexity analysis

Suppose there are m distinct items and n transactions in the database DB. The database is scanned first to compute the twu, rec and $arub$ values of all items $it_p \in IT$. It takes $O(m \times n)$ time in the worst case. Sorting each item it_p in ascending order of twu takes $O(mlogm)$ time. Scanning the database DB and constructing ARUL for each item in IT also requires $O(m \times n)$ time. In the worst case, calculating the $arub$ value of each item or an itemset X requires $O((2^m - 1) \times n)$ time. Traversing the search space to mine all HARUIs takes $O(2^m - 1)$ time in the worst case. Therefore, the time complexity of HARUIM in the worst case is $O((2^m - 1) \times n)$.

5 Experimental results

This section details the results of HARUIM algorithm on three real datasets (foodmart, retail, and mushroom) and two synthetic datasets (t20i6d100k and t25i10d10k). The minimum utility and recency thresholds, denoted by min_util and min_rec respectively, have been applied as constraints in the experiments.

The efficiency of the $HARUIM$ algorithm is compared with several other algorithms including:

1 FHM (Fournier-Viger et al., 2014a): This algorithm only uses *utility* to mine HUI's.

2 RUP (Gan et al., 2019): This algorithm uses *recency* as a measure along with decay factor to mine recent high utility utemsets (RHUIs).

3 $HARUIM_{baseline}$: This *baseline algorithm* uses *average recency* as a measure without using the pruning techniques used in the proposed $HARUIM$ algorithm.

4 $HARUIM$: The *proposed algorithm* uses *average recency* as a measure along with two novel pruning strategies (arub, $ERCS$).

5.1 Execution steps

1 The experiments were conducted on a system equipped with a Dell Intel Core i3 processor with a clock speed of 1.70 GHz, 8 GB of RAM, and a 64-bit Windows 10 operating system.

2 Pattern analysis was performed on patterns derived using all the four algorithms FHM (Fournier-Viger et al., 2014a), RUP (Gan et al., 2019), baseline, proposed algorithms.

3 Efficiency evaluation in terms runtime was performed by comparing the baseline, proposed algorithms.

4 Efficiency evaluation in terms of memory utilisation was performed by comparing the baseline and the proposed algorithms.

5 In all the experiments min_util threshold was increased while keeping the min_rec as constant.

6 Effect of pruning strategies, $arub$ and $ERCS$ was shown through the number of candidates generated by FHM (Fournier-Viger et al., 2014a), RUP (Gan et al., 2019), baseline, proposed algorithms.

5.2 Dataset characteristics

1 *Foodmart:* The foodmart dataset utilised in the experiments has 4,141 transactions and 1,559 unique items. The average length of a transaction is 4.42 items.

2 *Retail:* The retail dataset used in the experiments has 88,162 transactions and 16,470 unique items. The average length of a transaction is 10.30 items.

3 *Mushroom:* The mushroom dataset used in the experiments consists of 8,124 transactions with 119 unique items. The average length of a transaction is 23 items.

4 *t25i10d10k:* This dataset comprises of 9,976 transactions and 929 unique items. The average length of a transaction is 24.77 items.

5 *t20i6d100k:* This dataset comprises 99,922 transactions and 893 unique items. The average length of a transaction is 19.90 items.

5.3 Pattern analysis

This section aims to analyse the patterns generated by the $HARUIM$ algorithm and compare them with those generated by two other algorithms, namely RUP (Gan et al., 2019) and FHM (Fournier-Viger et al., 2014a). The $HARUIM$ algorithm utilises a novel measure known as *average recency* and two pruning strategies, $arub$ and $ERCS$, to extract concise and interesting patterns from the dataset. To illustrate, we will consider the patterns obtained from the dataset shown in Table 6.

1 When foodmart dataset was used at min_util = 1,500 and min_rec= 0.7, $HARUIM$ algorithm mines 62,539 patterns compared to 191,173 patterns mined by FHM (Fournier-Viger et al., 2014a) algorithm. As the min_util increases from 1,500 to 3,500, the number of patterns decrease from 62,539 to 21,752.

2 When retail dataset was used at min_util = 5,500 and min_rec = 0.4, $HARUIM$ algorithm derives 2,350 patterns when compared to the FHM algorithm which mines 2,598 patterns. The number of patterns decrease from 2,350 to 920 as min_util is increased from 5,500 to 9,500.

3 When synthetic dataset $t25i10d10k$ was used at min_util = 2,000 and min_rec = 0.7, $HARUIM$ algorithm derives 401,219 patterns when compared to RUP (Gan et al., 2019) which mines 1,046,525 patterns. The number of patterns decrease from 401,219 to 539 as min_util is increased form 2,000 to 6,000.

Table 6 Patterns generated

Dataset	*Algorithms*	*Number of patterns*				
		Test1	*Test2*	*Test3*	*Test4*	*Test5*
Foodmart (min_rec = 0.7)	min_util	1,500	2,000	2,500	3,000	3,500
	FHM	191,173	154,670	117,592	85,034	59,351
	RUP	191,173	154,670	117,592	85,034	59,351
	$HARUIM_{baseline}$	62,539	51,464	39,993	29,874	21,752
	$HARUIM$	62,539	51,464	39,993	29,874	21,752
Retail (min_rec = 0.4)	min_util	5,500	6,500	7,500	8,500	9,500
	FHM	2,598	1,967	1,526	1,208	992
	RUP	2,598	1,967	1,526	1,208	992
	$HARUIM_{baseline}$	2,350	1,776	1,392	1,114	920
	$HARUIM$	2,350	1,776	1,392	1,114	920
Mushroom (min_rec = 0.7)	min_util	250,000	260,000	270,000	280,000	290,000
	FHM	31,706	27,186	23,590	20,945	18,772
	RUP	31,706	27,186	23,590	20,945	18,772
	$HARUIM_{baseline}$	1,692	1,230	877	608	402
	$HARUIM$	1,692	1,230	877	608	402
t25i10d10k (min_rec = 0.7)	min_util	2,000	3,000	4,000	5,000	6,000
	FHM	1,046,525	410,545	114,125	18,547	1,157
	RUP	1,046,525	410,545	114,125	18,547	1,157
	$HARUIM_{baseline}$	401,219	260,797	87,218	15,611	539
	$HARUIM$	401,219	260,797	184,135	15,611	539
t20i6d100k (min_rec = 0.5)	min_util	5,000	15,000	25,000	35,000	45,000
	FHM	336,490	78,299	13,476	1,923	188
	RUP	336,490	78,299	13,476	1,923	188
	$HARUIM_{baseline}$	121,070	20,528	5,507	407	48
	$HARUIM$	121,070	20,528	5,507	407	48

4 It can be observed that $HARUIM$ and $HARUIM_{baseline}$ algorithms mine same number of patterns but the number of derived candidates might vary due to the novel pruning techniques ($arub$, $ERCS$) used in the proposed system. For

example, using foodmart dataset at min_util = 1,500 and min_rec = 0.7, the number of patterns derived by $HARUIM$ and $HARUIM_{baseline}$ algorithms are 62,539, whereas it can be observed from Table 9 that the number of candidates (nodes) visited by $HARUIM$ and $HARUIM_{baseline}$ are 150,716 and 498,082 respectively.

Figure 6 Patterns-real datasets (see online version for colours)

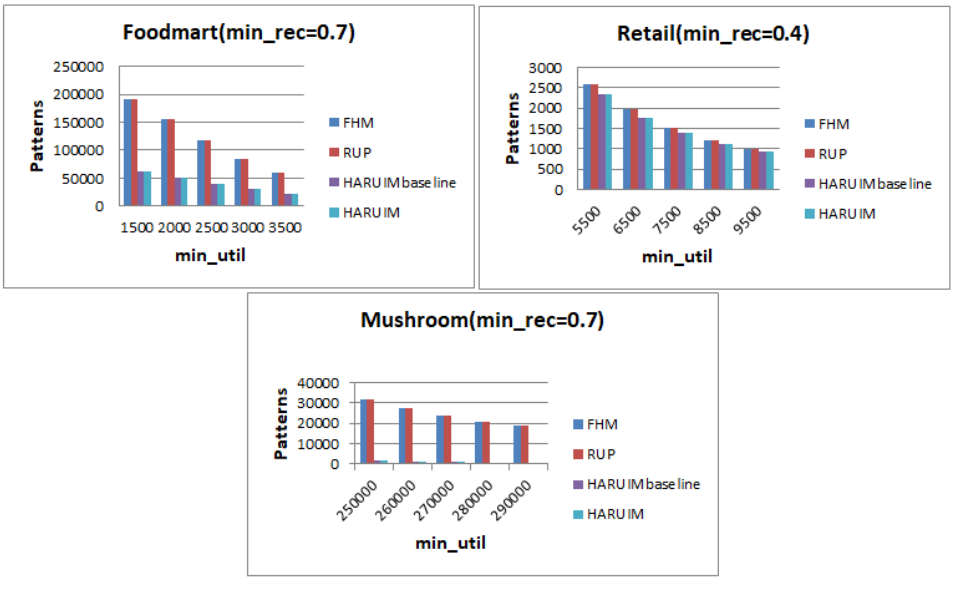
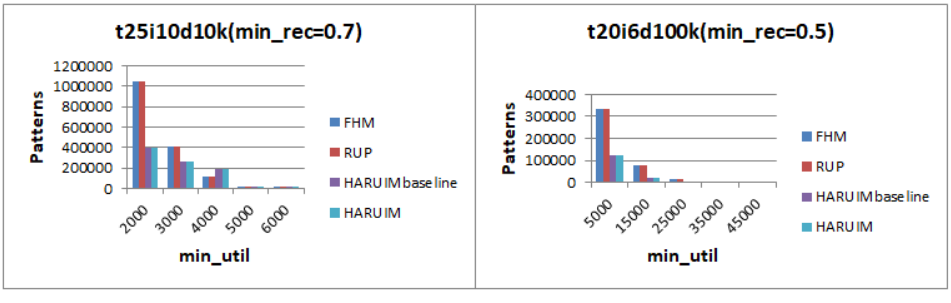


Figure 7 Patterns-synthetic datasets (see online version for colours)



Using Table 6, Figures 6 and 7 the following observations can be made:

1 $HARUIM$ algorithm mines concise as well as high quality patterns.

2 In $HARUIM$, as min_util increases, the number derived patterns decrease.

3 Although the $HARUIM_{baseline}$ and $HARUIM$ algorithms generate similar patterns, the number of candidates generated by each algorithm may differ due to the application of novel pruning techniques in $HARUIM$. While the $HARUIM_{baseline}$ algorithm utilises a traditional approach without pruning

techniques, the $HARUIM$ algorithm applies the $arub$ and $ERCS$ strategies to effectively decrease the search space.

Table 7 Runtime analysis

Dataset	*Algorithms*	*Runtimes (ms)*				
		Test1	*Test2*	*Test3*	*Test4*	*Test5*
Foodmart (min_rec = 0.7)	min_util	1,500	2,000	2,500	3,000	3,500
	FHM	1,451	1,178	1,150	1,100	1,019
	RUP	1,746	1,479	1,423	1,366	1,291
	$HARUIM_{baseline}$	1,799	1,558	1,326	1,202	1,150
	$HARUIM$	1,334	1,123	1,094	996	931
Retail (min_rec = 0.4)	min_util	5,500	6,500	7,500	8,500	9,500
	FHM	11,355	10,592	10,065	9,244	8,600
	RUP	10,881	10,556	10,291	9,058	8,949
	$HARUIM_{baseline}$	17,449	15,796	15,545	15,290	14,827
	$HARUIM$	16,726	15,513	14,985	14,338	13,893
Mushroom (min_rec = 0.7)	min_util	250,000	260,000	270,000	280,000	290,000
	FHM	19,808	17,814	15,881	14,450	13,045
	RUP	20,552	17,780	16,561	14,822	14,727
	$HARUIM_{baseline}$	20,266	17,951	16,158	15,119	13,501
	$HARUIM$	11,479	10,821	9,408	8,565	7,778
t25i10d10k (min_rec = 0.7)	min_util	2,000	3,000	4,000	5,000	6,000
	FHM	19,219	12,448	8,540	6,685	4,906
	RUP	28,146	14,326	9,288	7,381	5,906
	$HARUIM_{baseline}$	25,191	13,582	10,058	8,205	6,818
	$HARUIM$	20,915	13,412	10,056	8,185	6,819
t20i6d100k (min_rec = 0.5)	min_util	5,000	15,000	25,000	35,000	45,000
	FHM	54,000	53,417	37,016	25,975	21,203
	RUP	242,321	56,560	34,843	26,315	20,229
	$HARUIM_{baseline}$	136,278	62,820	41,650	32,093	26,279
	$HARUIM$	137,104	71,940	41,571	32,325	26,909

5.4 Runtime evaluation

This section analyses the effect of pruning strategies on the runtime of the $HARUIM$ algorithm. Specifically, runtime performance of $HARUIM$ is compared with two other algorithms, namely $HARUIM_{baseline}$ and RUP (Gan et al., 2019). By doing so, we aim to assess the efficiency of the $HARUIM$ algorithm in terms of runtime.

1 From Table 9, when foodmart dataset was used at min_util = 1,500 and min_rec = 0.7, $HARUIM$ algorithm prunes 347,366 more candidates than the $HARUIM_{baseline}$ algorithm.

2 The proposed system generates 69.74% less candidates (nodes) in comparison to the $HARUIM_{baseline}$ algorithm. As exponential number of candidates (nodes) have been pruned the efficiency of the proposed system with respect to runtime

increases by almost half a second. As foodmart is a sparse dataset, min_rec prunes more number of candidates (nodes) when compared to a dense dataset.

Figure 8 Runtime-real datasets (see online version for colours)

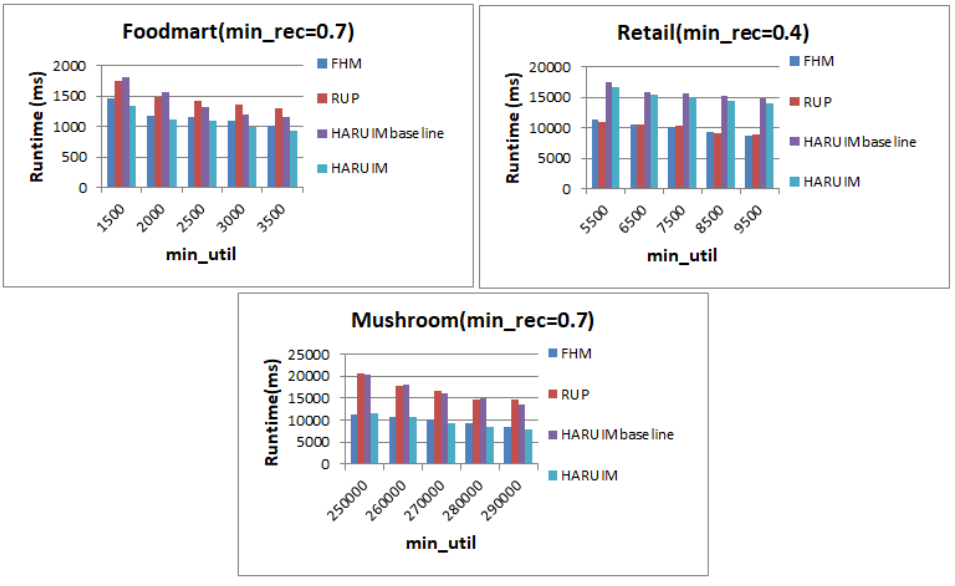
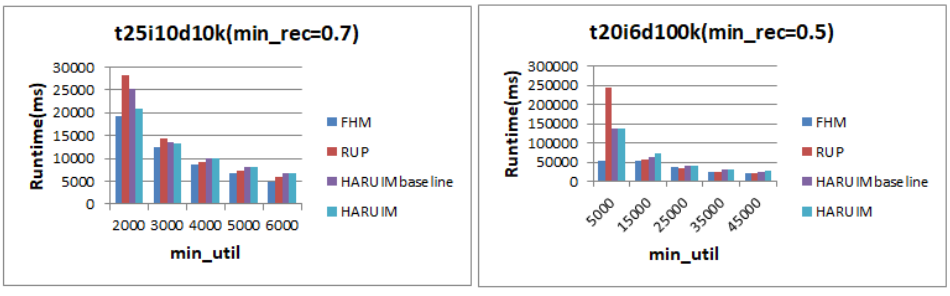


Figure 9 Runtime-synthetic datasets (see online version for colours)



3 Similarly Table 7 shows that the proposed system also outperforms the RUP (Gan et al., 2019) algorithm by 412 ms. The efficiency of $HARUIM$ algorithm cannot be compared with FHM (Fournier-Viger et al., 2014a) as the proposed system uses two constraints namely min_util, min_rec when compared to the FHM (Fournier-Viger et al., 2014a) algorithm which uses a single constraint called min_util to mine high utility patterns.

4 From Table 7, when Mushroom dataset (dense) was used at min_util = 250,000, min_rec = 0.7 the proposed system outperforms the $HARUIM_{baseline}$ algorithm by 8.7 seconds. Similarly $HARUIM$ algorithm outperforms RUP (Gan et al., 2019) algorithm by 9.07 seconds.

5 When synthetic dataset t25i10d10k (dense dataset) with min_util = 2,000, min_rec = 0.7 was used, the proposed $HARUIM$ algorithm outperforms the $HARUIM_{baseline}$ algorithm by 4.2 seconds, Table 7.

Table 8 Memory utilised

Dataset	*Algorithms*	*Memory*				
		Test1	*Test2*	*Test3*	*Test4*	*Test5*
Foodmart (min_rec = 0.7)	min_util	1,500	2,000	2,500	3,000	3,500
	FHM	77.64	76.25	65.35	52.94	41.94
	RUP	77.93	77.93	77.93	64.43	50.02
	$HARUIM_{baseline}$	54.32	50.11	49.34	49.30	40.14
	$HARUIM$	39.62	37.13	31.64	26.94	22.16
Retail (min_rec = 0.4)	min_util	5,500	6,500	7,500	8,500	9,500
	FHM	379.4	362.4	349.3	336.8	328.5
	RUP	384.6	355.9	354.6	344.7	335.7
	$HARUIM_{baseline}$	564.7	545.1	522	508.8	490
	$HARUIM$	551.02	529.25	514.34	494	475.2
Mushroom (min_rec = 0.7)	min_util	250,000	260,000	270,000	280,000	290,000
	FHM	123.16	122.55	121.71	121.55	120.96
	RUP	124.81	123.82	122.21	121.89	119.88
	$HARUIM_{baseline}$	123.24	122.05	121.95	121.35	119.88
	$HARUIM$	120.53	120.51	119.93	116.91	115.73
t25i10d10k (min_rec = 0.7)	min_util	2,000	3,000	4,000	5,000	6,000
	FHM	550.79	522.95	510.31	484.15	257.95
	RUP	652	611.49	584.38	551.06	279.72
	$HARUIM_{baseline}$	855.53	848.89	855.58	626.53	283.94
	$HARUIM$	837.69	832.91	824.44	608.82	275.42
t20i6d100k (min_rec = 0.5)	min_util	5,000	15,000	25,000	35,000	45,000
	FHM	1,131.08	1,097.11	1,077.57	842.71	687.18
	RUP	1,140.13	1,130.38	1,112.63	1,009.86	683.79
	$HARUIM_{baseline}$	1,274.65	1,250.99	1,249.14	1,240.52	859.38
	$HARUIM$	1,276.69	1,267.57	1,261.59	1,244.70	853.72

Using Table 7, Figures 8 and 9 the following observations can be made:

1 The $HARUIM$ algorithm employs two pruning strategies, namely $arub$ and $ERCS$, to decrease the search space.

2 As min_util increases, the runtime of the $HARUIM$ algorithm decreases.

3 The execution time of the $HARUIM$ algorithm is lower than that of the $HARUIM_{baseline}$ algorithm. This is because the $HARUIM$ algorithm uses two pruning techniques, $arub$ and $ERCS$, which help to eliminate unpromising itemsets. In contrast, the $HARUIM_{baseline}$ algorithm requires the construction of ARULs, which increases the computational complexity and hence the execution time.

Figure 10 Memory-real datasets (see online version for colours)

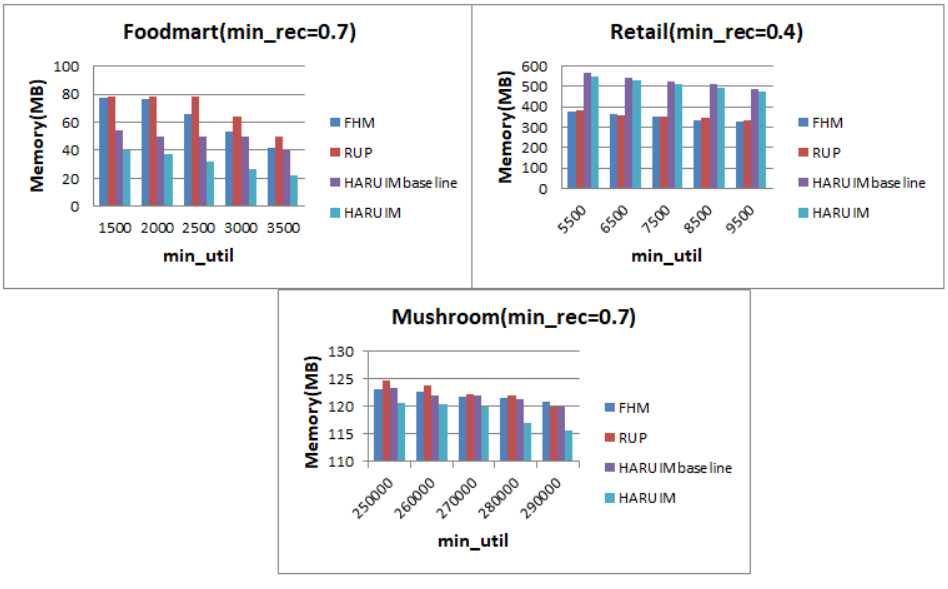
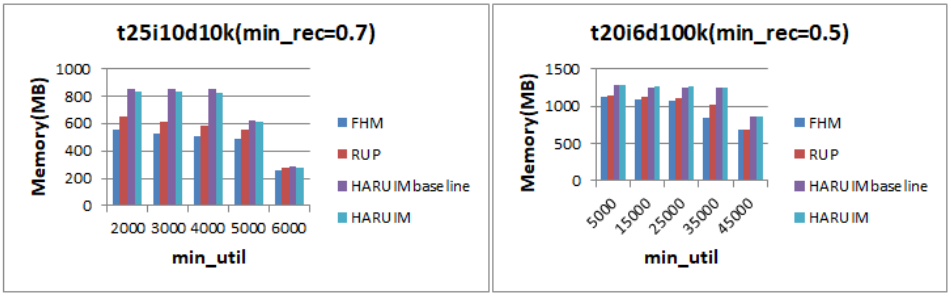


Figure 11 Memory-synthetic datasets (see online version for colours)



5.5 *Memory utilisation*

This sub section focuses on comparing the memory utilisation of the proposed $HARUIM$ algorithm with the $HARUIM_{baseline}$ and RUP (Gan et al., 2019) algorithms.

1 As $arub$, $ERCS$ structures prune unpromising itemsets (nodes) less number of ARULs are constructed thereby resulting in less amount of memory being utilised. $HARUIM$ algorithm works well with sparse datasets namely foodmart and retail.

2 For example, from Table 8 when Foodmart dataset was used at min_util = 1,500, min_rec = 0.7, $HARUIM_{baseline}$, RUP (Gan et al., 2019) algorithms require 14.7 MB, 38.31 MB of memory more than the proposed HARUIM algorithm.

3 When mushroom (dense) dataset was used at min_util = 250,000, min_rec = 0.7, $HARUIM_{baseline}$, RUP (Gan et al., 2019) algorithms require 2.71 MB, 4.28 MB of memory more than the proposed $HARUIM$ algorithm.

4 When synthetic dataset t25i10d10k, was used at min_util = 2,000, min_rec = 0.7, $HARUIM_{baseline}$ algorithm requires 17.84 MB more than the proposed $HARUIM$ algorithm.

Table 9 Number of candidates

Dataset	*Algorithms*	Candidates				
		Test1	*Test2*	*Test3*	*Test4*	*Test5*
Foodmart	min_util	1,500	2,000	2,500	3,000	3,500
(min_rec = 0.7)	FHM	249,041	230,854	206,141	174,882	141,406
	RUP	498,082	461,708	412,282	349,764	282,812
	$HARUIM_{baseline}$	498,082	461,708	412,282	349,764	282,812
	$HARUIM$	150,716	143,912	132,144	115,344	95,576
Retail	min_util	5,500	6,500	7,500	8,500	9,500
(min_rec = 0.4)	FHM	40,753	30,482	23,846	19,353	16,249
	RUP	81,506	60,964	47,692	38,706	32,498
	$HARUIM_{baseline}$	81,506	60,964	47,692	38,706	32,498
	$HARUIM$	79,530	59,702	46,746	37,994	31,956
Mushroom	min_util	250,000	260,000	270,000	280,000	290,000
(min_rec = 0.7)	FHM	61,514	17,814	15,881	14,450	13,045
	RUP	123,028	102,708	86,842	75,762	66,974
	$HARUIM_{baseline}$	123,028	102,708	86,842	75,762	66,974
	$HARUIM$	59,644	51,122	43,542	38,330	33,736
t25i10d10k	min_util	2,000	3,000	4,000	5,000	6,000
(min_rec = 0.7)	FHM	5,767,024	1,935,227	863,859	403,981	188,705
	RUP	11,534,048	3,870,454	1,727,718	807,962	377,410
	$HARUIM_{baseline}$	11,534,048	3,870,454	1,727,718	807,962	377,410
	$HARUIM$	11,491,010	3,865,496	1,727,508	807,944	377,410
t20i6d100k	min_util	5,000	15,000	25,000	35,000	45,000
(min_rec = 0.5)	FHM	10,618,457	776,959	220,981	89,033	43,746
	RUP	21,236,914	1,553,918	441,962	178,066	87,492
	$HARUIM_{baseline}$	21,236,914	1,553,918	441,962	178,066	87,492
	$HARUIM$	21,236,914	1,553,918	441,962	178,066	87,492

5.6 Effect of pruning techniques

In this subsection, we examine how the number of candidates (nodes) generated is affected by pruning strategies, specifically $arub$ and $ERCS$.

1 The efficiency of $HARUIM$ algorithm can be observed in terms of the percentage of candidates or nodes pruned. Sparse datasets such as foodmart and retail are more efficiently pruned by the proposed system than dense datasets such as t25i10d10k and t20i6d100k. This suggests that the effectiveness of the pruning

strategy is dependent on the dataset's sparsity. The number of candidates (nodes) pruned is directly proportional to the runtime and memory utilisation of the proposed system.

2 From Table 9 it can be observed that $HARUIM$ algorithm visits less number of nodes (candidates) than its baseline algorithm. The real efficiency can be found by comparing the proposed system to the $HARUIM_{baseline}$ algorithm.

3 When foodmart dataset was used at min_util at 2,000 and min_rec at 0.7, $HARUIM$ algorithm visits just 143,912 candidates (nodes) where as the $HARUIM_{baseline}$ algorithm visits 461,708 candidates (nodes) in the search space. The number of candidates (nodes) visited by $HARUIM$ algorithm are 68.83% less than the $HARUIM_{baseline}$ algorithm.

4 Similarly, retail dataset was used to test the proposed system. When min_util was set to 6,500 and min_rec to 0.4, the number of candidates (nodes) visited by $HARUIM$ algorithm are 2.07% less than the $HARUIM_{baseline}$ algorithm. As min_rec value was set to a lower value of 0.4, less number of candidates (nodes) are pruned by the proposed system.

5 When mushroom dataset was used to test the system at min_util = 260,000, min_rec = 0.7, the proposed algorithm generates 51,586 candidates less than the $HARUIM_{baseline}$ algorithm.

6 When synthetic dataset t25i10d10k was used at min_util = 3,000, min_rec = 0.7, the $HARUIM$ algorithm generates 4,958 candidates less than the $HARUIM_{baseline}$ algorithm. $HARUIM$ which uses *average recency* as a novel measure works well with sparse datasets.

Using Table 9, Figures 12 and 13, the following observations can be made:

1 The efficiency of the $HARUIM$ algorithm is improved by pruning the search space, resulting in fewer nodes, and the algorithm performs faster as min_util increases.

2 As $HARUIM$ prunes large amount of search space, smaller search space is processed much faster than the complete search space.

3 A smaller search space due to pruning strategies results in faster runtime and lesser utilisation of memory.

5.7 Scalability

The efficiency of the proposed system remains stable even as the database size is increased. Using synthetic dataset t20i6d100k, when the size of the database is increased by 20,000, the runtime and memory allocation increase. For example, with min_util = 5,000, min_rec = 0.5, when database size is increased from 20,000 to 40,000, the runtime increases by 18.57 seconds and memory utilisation increases by 567.01 MB. When the database size is increased from 20,000 to 60,000, the runtime increases by 58 seconds and memory utilisation increases by 632 MB. Therefore the proposed $HARUIM$ algorithm scales well with respect to runtime and memory allocation, Table 10.

Figure 12 Candidates-real datasets (see online version for colours)

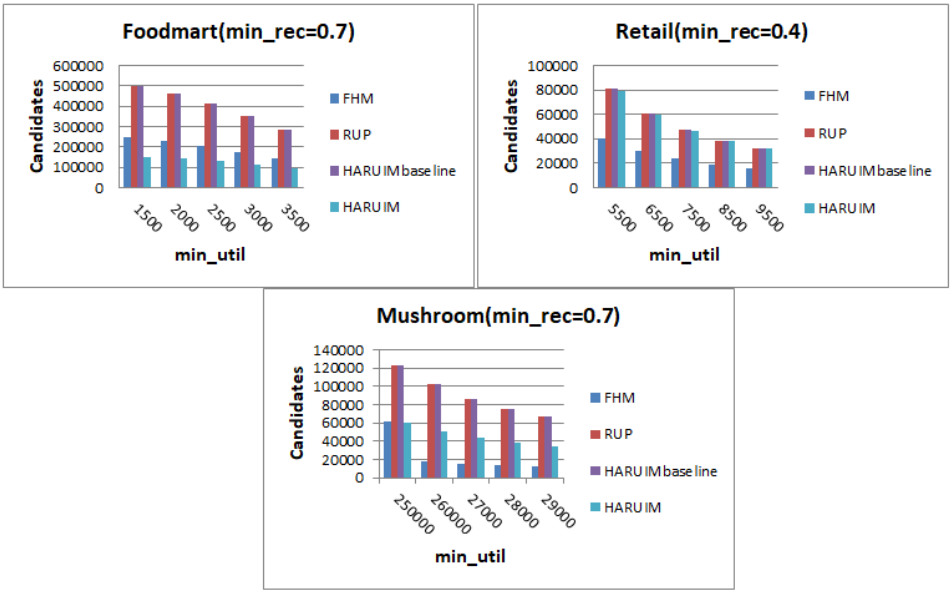


Figure 13 Candidates-synthetic datasets (see online version for colours)

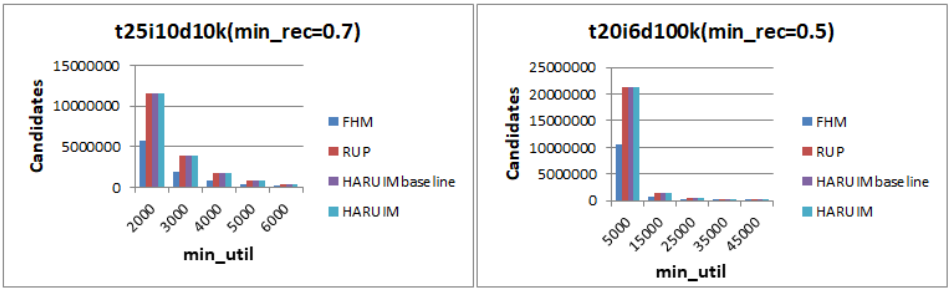


Table 10 Scalability

Dataset		*Database size*				
t20i6d100k		*20,000*	*40,000*	*60,000*	*80,000*	*99,922*
min_util = 5,000	Patterns	8,466	56,233	97,884	117,935	121,070
min_rec = 0.5	Candidates	276,418	1,588,844	5,856,010	12,620,444	21,236,914
	Runtime (ms)	7,702	26,278	65,231	97,168	137,104
	Memory (MB)	436.32	1,003.33	1,068.32	1,176.75	1,276.69

5.8 *Generalisation of the proposed HARUIM algorithm*

The HARUIM algorithm has the flexibility to be applied to any item that is connected with a weight or profit value. HARUIM can not only be used in context to retail and

e-commerce but can also be used in other domains such as finance and healthcare. An illustration of using HARUIM in the domain of finance is the use of the *NIFTY-50* stock dataset. We refer to a set of stocks as 'stocksets'. Since each stock has an associated profit or loss value, we transform the raw stock data from a period of one hundred trading days into SPMF format (Fournier-Viger et al., 2014b) to mine *high average recent stocksets*. These stocksets can be used by investors and traders to recognise *recent trending stocksets*. To collect historical data for all stocks in the *NIFTY-50*, we utilised Yahoo Finance (https://finance.yahoo.com/) and obtained a set of data containing various columns for each stock. The dataset's focus was on the percentage difference between the previous day's closing price and the current day's closing price for each stock on a given day.

$$Utility = (todays\ close\ value - previous\ day\ close\ value) \\ /\ (previous\ day\ close\ value)$$

Figure 14 Scalability-varied minimum utility (see online version for colours)

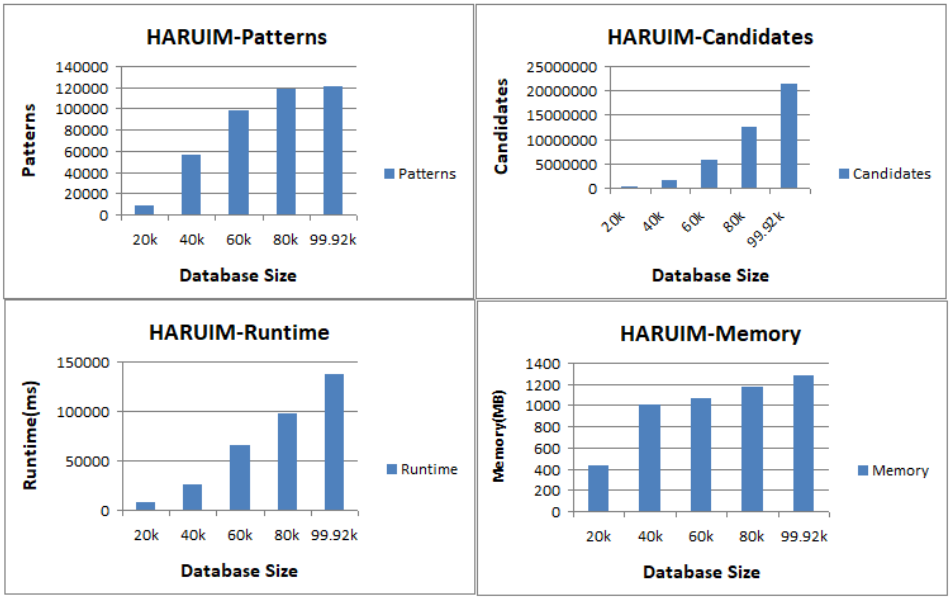
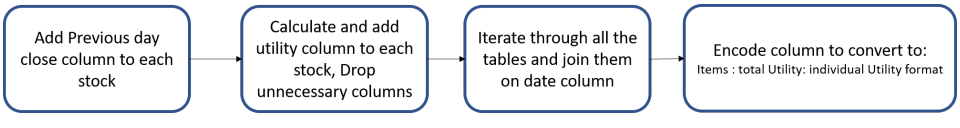


Figure 15 Data pre-processing (see online version for colours)



5.8.1 Steps to convert raw data into spmf (Fournier-Viger et al., 2014b) format

1 Add closing value of previous day to each row of each file of stock, Figure 16.

Figure 16 Previous close (see online version for colours)

Date	Open	High	Low	Close	Adj Close	Volume	Prev Close
1995-12-26	72.819626	72.819626	72.819626	72.819626	53.472786	0.0	72.819626
1995-12-27	72.819626	72.819626	72.819626	72.819626	53.472786	0.0	72.819626
1995-12-28	72.819626	72.819626	72.819626	72.819626	53.472786	0.0	72.819626
1995-12-29	72.819626	72.819626	72.819626	72.819626	53.472786	0.0	72.819626
1996-01-01	72.436363	72.819626	72.340553	72.819626	53.472786	40703.0	72.819626
1996-01-02	72.819626	73.566986	72.053108	72.398041	53.163197	100975.0	72.819626

2 Add utility column, Figure 17.

Figure 17 Utility column (see online version for colours)

Date	Utility
2002-07-02	0.3454
2002-07-03	-0.0257
2002-07-04	1.0225
2002-07-05	0.0658
2002-07-08	-0.9857

3 Join utility columns based on date, Figure 18.

Figure 18 Stock utility columns (see online version for colours)

Date	TATASTEEL_Utility_percent	ADANIPORTS_Utility_percent	APOLLOHOSP_Utility_percent	ASIANPAINT_Utility_percent	AXISBANK_Utility_percent	BAJAJ-AUTO_Utility_percent
2018-04-27	-2.0062	-1.3473	0.4179	-2.2411	-9.2051	-2.156
2018-04-30	-0.8732	-1.2051	-1.7344	-1.7011	4.0616	-0.1849
2018-05-02	3.4373	1.5222	1.0037	-1.7142	-1.2469	-0.4521
2018-05-03	-1.906	1.0222	0.9953	1.2722	-1.8425	-0.3119
2018-05-04	0.8028	-2.8338	-3.002	1.3507	2.4559	2.909
2018-05-07	-2.7034	-1.0165	1.8613	-2.0833	-3.0367	-0.9226
2018-05-08	0.0168	-0.097	0.8604	0.4402	-1.1472	0.7907
2018-05-09	-1.1403	-0.1332	2.4356	-1.0455	-1.1526	0.7001
2018-05-10	1.4922	0.6169	2.0421	0.3967	0.155	1.5494
2018-05-11	-2.2048	-0.4625	-2.0261	-5.6459	-1.0317	0.4262
2018-05-14	-0.5105	0.6663	2.0289	-0.8473	0.253	-0.5861
2018-05-15	-2.5559	0.1952	-1.7681	-1.3335	0.3986	0.2931
2018-05-16	0.7509	1.4787	1.1855	-0.6884	-0.0546	0.4161
2018-05-17	1.7626	1.29	1.1317	1.0765	1.9182	-0.1263
2018-05-18	3.0968	2.1111	0.113	0.084	1.2142	0.9927

4 Encode and convert to .txt format which is used as an input to the proposed HARUIM aglorithm. For example, snapshot of *NIFTY-50* stocks converted into spmf (Fournier-Viger et al., 2014b) format is shown in Figure 19. As trending stocksets are mined, we only considered stock sets with a positive increase in percentage of stock price while mining for trending stocks. Each transaction has the following format:

$$Items : transaction\ utility : individual\ utility$$

Figure 19 NIFTY-50-SPMF-format

```
1 13 15 26 41 45 50:2.8925:0.503 0.8676 0.175 0.0816 0.1956 0.6095 0.4602
5 12 15 22 38 40 45 46:6.5501:0.7777 1.369 0.4963 0.8885 1.1325 1.043 0.6018 0.2413
```

The proposed HARUIM algrorithm applied on *NIFTY-50* dataset yeilds the following results (Table 11).

Table 11 HARUIM-NIFTY-50

Dataset	*Algorithm*	*HARUIM-NIFTY-50*				
		Test1	*Test2*	*Test3*	*Test4*	*Test5*
NIFTY-50	min_util	235	245	255	265	275
(min_rec = 0.5)	Stocksets	50,646	11,034	2,027	301	36
	Candidates	84,298,210	49,905,160	31,067,408	20,229,282	13,710,988
	Runtime (ms)	35,511	18,493	12,815	9,032	6,313
	Memory (MB)	133.45	133.42	133.29	133.34	133.38

When *NIFTY-50* dataset was used at min_util = 275 and min_rec = 0.5, the proposed $HARUIM$ algorithm mines 36 trending stocksets. One example of mined stockset is:

$$ut(ADANIENT, INDUSINDBK, INDUSINDBK, HDFCBANK, UPL, TATACONSUM, TATASTEEL, ONGC, ULTRACEMCO, ICICIBANK, BHARTIARTL, SBIN, HINDALCO, BHARTIARTL, GRASIM) = 277.45, \text{ from Definition 3.2.}$$

$$avgrec(ADANIENT, INDUSINDBK, INDUSINDBK, HDFCBANK, UPL, TATACONSUM, TATASTEEL, ONGC, ULTRACEMCO, ICICIBANK, BHARTIARTL, SBIN, HINDALCO, BHARTIARTL, GRASIM) = 0.564, \text{ from Definition 3.13.}$$

5.9 Guidelines for policy makers based on HARUIM

Based on the findings we enumerate various guidelines that policymakers can follow:

1 Identification of HUI's: The proposed HARUIM algorithm is used to identify the most profitable itemsets purchased by the customers. These itemsets can help to identify customer preferences, market trends and demand patterns.

2 Analysing average recency measure: The average recency measure used in the proposed HARUIM algorithm helps the policy makers to design marketing strategies and promotions based on recently purchased itemsets.

3 Based on the findings of the proposed system, the policy makers can adjust the prices of certain items/itemsets according to their average recency values.

4 Items/itemsets can be recommended to customers based on their past purchases and the average recency measure.

6 Conclusions and future work

HARUIM is a novel method to mine HARUI'S. Large number of HUIM algorithms only consider utility as a measure to mine HUI's, which might result in outdated patterns.

To overcome this limitation, we introduced a novel measure, namely average recency, which computes the average recency of the transactions to mine HARUIs. Unlike the previous approach, this measure considers both recent and past transactions and avoids the limitation of extremely recent transactions. To further enhance the performance of HARUIM algorithm, we developed novel pruning techniques to discard unpromising itemsets. The experimental results obtained using real and synthetic datasets demonstrate that $HARUIM$ outperforms the baseline algorithm with respect to runtime and memory utilisation. Our future work includes extending HARUIM with various databases like uncertain and dynamic databases.

References

Agrawal, R., Srikant, R. et al. (1994) 'Fast algorithms for mining association rules', *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, Vol. 1215, pp.487–499.

Ahmed, C., Tanbeer, S., Jeong, B. and Lee, Y. (2009) '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.

Ahmed, C., Tanbeer, S. and Jeong, B. (2011) 'A framework for mining high utility web access sequences', *IETE Technical Review*, Vol. 28, No. 1, pp.3–16.

Chen, M., Han, J. and Yu, P. (1996) 'Data mining: an overview from a database perspective', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp.866–883.

Chi, Y., Wang, H., Yu, P. and Muntz, R. (2004) 'Moment: maintaining closed frequent itemsets over a stream sliding window', *Fourth IEEE International Conference On Data Mining (ICDM'04)*, pp.59–66.

Dam, T., Li, K., Fournier-Viger, P. and Duong, Q. (2019) 'CLS-miner: efficient and effective closed high-utility itemset mining', *Frontiers of Computer Science*, Vol. 13, No. 3, pp.357–381.

Erwin, A., Gopalan, R. and Achuthan, N. (2008) 'Efficient mining of high utility itemsets from large datasets', *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp.554–561.

Fournier-Viger, P. (2014) 'FHN: efficient mining of high-utility itemsets with negative unit profits', *International Conference on Advanced Data Mining and Applications*, Vol. 8502, pp.16–29.

Fournier-Viger, P. and Zida, S. (2015) 'FOSHU: faster on-shelf high utility itemset mining – with or without negative unit profit', *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, No. 1, pp.857–864.

Fournier-Viger, P., Wu, C., Zida, S. and Tseng, V. (2014a) 'FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning', *International Symposium on Methodologies for Intelligent Systems*, pp.83–92.

Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C., Tseng, V. et al. (2014b) 'SPMF: a Java open-source pattern mining library', *J. Mach. Learn. Res.*, Vol. 15, pp.3389–3393.

Fournier-Viger, P., Lin, J., Dinh, T. and Le, H. (2016a) 'Mining correlated high-utility itemsets using the bond measure', *International Conference on Hybrid Artificial Intelligence Systems*, pp.53–65.

Fournier-Viger, P., Lin, J., Duong, Q. and Dam, T. (2016b) 'PHM: mining periodic high-utility itemsets', *Industrial Conference on Data Mining*, pp.64–79.

Fournier-Viger, P., Lin, J., Duong, Q. and Dam, T. (2016c) 'FHM: faster high-utility itemset mining using length upper-bound reduction', *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pp.115–127.

Gan, W., Lin, J., Fournier-Viger, P., Chao, H. and Zhan, J. (2017) 'Mining of frequent patterns with multiple minimum supports', *Engineering Applications of Artificial Intelligence*, April, Vol. 60, pp.83–96.

Gan, W., Lin, J., Fournier-Viger, P., Chao, H. and Fujita, H. (2018a) 'Extracting non-redundant correlated purchase behaviors by utility measure', *Knowledge-Based Systems*, Vol. 143, No. 1, pp.30–41.

Gan, W., Lin, J., Fournier-Viger, P., Chao, H., Tseng, V. and Yu, P. (2018b) *A Survey of Utility-Oriented Pattern Mining*, ArXiv Preprint ArXiv:1805.10511.

Gan, W., Lin, J., Chao, H., Fournier-Viger, P., Wang, X. and Yu, P. (2019) *Utility-Driven Mining of Trend Information for Intelligent System*, ArXiv Preprint ArXiv:1912.11666.

Golab, L. and Özsu, M. (2003) 'Issues in data stream management', *ACM Sigmod Record*, Vol. 32, No. 2, pp.5–14.

Han, J., Pei, J. and Yin, Y. (2000) 'Mining frequent patterns without candidate generation', *ACM Sigmod Record*, Vol. 29, No. 2, pp.1–12.

Hong, T., Lee, C. and Wang, S. (2009) 'Mining high average-utility itemsets', *2009 IEEE International Conference on Systems, Man and Cybernetics*, pp.2526–2530.

Kumar, M. and Rana, D. (2021) 'High average utility itemset mining: a survey', *Proceedings of International Conference on Computational Intelligence and Data Engineering*, pp.347–374.

Li, Y., Yeh, J. and Chang, C. (2008) 'Isolated items discarding strategy for discovering high utility itemsets', *Data and Knowledge Engineering*, Vol. 64, No. 1, pp.198–217.

Lin, J., Ren, S., Fournier-Viger, P. and Hong, T. (2017) 'EHAUPM: efficient high average-utility pattern mining with tighter upper bounds', *IEEE Access*, Vol. 5, pp.12927–12940 (accessed 20 June 2017).

Lin, J., Gan, W., Fournier-Viger, P., Hong, T. and Tseng, V. (2016) 'Fast algorithms for mining high-utility itemsets with various discount strategies', *Advanced Engineering Informatics*, Vol. 30, No. 2, pp.109–126.

Lin, J., Gan, W., Hong, T. and Tseng, V. (2015) 'Efficient algorithms for mining up-to-date high-utility patterns', *Advanced Engineering Informatics*, Vol. 29, No. 3, pp.648–661.

Liu, M. and Qu, J. (2012) 'Mining high utility itemsets without candidate generation', *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pp.55–64.

Liu, Y., Liao, W. and Choudhary, A. (2005a) 'A fast high utility itemsets mining algorithm', *Proceedings of the 1st International Workshop on Utility-based Data Mining*, pp.90–99.

Liu, Y., Liao, W. and Choudhary, A. (2005b) 'A two-phase algorithm for fast discovery of high utility itemsets', *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp.689–695.

Liu, Y., Cheng, C. and Tseng, V. (2013) 'Mining differential top-k co-expression patterns from time course comparative gene expression datasets', *BMC Bioinformatics*, 21 July, Vol. 14, No. 230, pp.1–13.

Rymon, R. (1992) *Search through Systematic Set Enumeration*, Technical Report No. MS-CIS-92-66, Proceedings KR-92, Department of Computer and Information Science, University of Pennsylvania.

Shanthi, K., Akilandeswari, J. and Jothi, G. (2016) 'Mining frequent patterns without tree generation', *International Journal of Data Mining, Modelling and Management*, Vol. 8, No. 3, pp.265–278.

Shie, B., Hsiao, H., Tseng, V. and Philip, S. (2011) 'Mining high utility mobile sequential patterns in mobile commerce environments', *International Conference on Database Systems for Advanced Applications*, pp.224–238.

Shie, B., Hsiao, H. and Tseng, V. (2013) 'Efficient algorithms for discovering high utility user behavior patterns in mobile commerce environments', *Knowledge and Information Systems*, Vol. 37, No. 2, pp.363–387.

Tseng, V., Wu, C., Shie, B. and Yu, P. (2010) 'UP-growth: an efficient algorithm for high utility itemset mining', *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.253–262.

Tseng, V., Wu, C., Fournier-Viger, P. and Philip, S. (2015) 'Efficient algorithms for mining top-k high utility itemsets', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 1, pp.54–67.

Yao, H., Hamilton, H. and Butz, C. (2004) 'A foundational approach to mining itemset utilities from databases', *Proceedings of the 2004 SIAM International Conference on Data Mining*, pp.482–486.

Zida, S., Fournier-Viger, P., Lin, J., Wu, C. and Tseng, V. (2015) 'EFIM: a highly efficient algorithm for high-utility itemset mining', *Mexican International Conference on Artificial Intelligence*, pp.530–546.