An Incremental Partial Periodic Mining Over Medical Infertility Dataset to Mine High Utility Patterns

Suvarna U, Srinivas Y

Abstract: ***Data mining has always been a source for researchers to explore mining techniques. The current curve of mining is tending towards the utility itemset mining algorithms where the most utility (profitable) items or patterns are discovered. The article aims at mining the dynamic databases over a period to find high utility patterns (we used medical infertility dataset to find the complex combination of anti oxidants and its dosage involved in success of treatment for male infertility or andrology).The proposed algorithm uses an incremental transaction database where we mine partial periodic patterns using a list structure in just one database scan, called the IPP_HUIM. The IPP_HUIM algorithm (Incremental Partial Periodic High Utility Itemset Mining) is implemented in three phases. In phase 1, the IHUIM performs the Construction of List structure for items, Transaction weighted utility & periodicity follows a reorder and a restructure of the reordered list. Phase 2 IPHUIM algorithm, finds the periodic high utility itemset mining for K-itemset and prunes with search strategies & novel periodic list for optimization and the Phase 3 obtaining partial periodic patterns resulting the best utility patterns which is the IPPHUIM algorithm. The experimental results show high performance and accuracy compared to previous periodic algorithms over incremental databases.***

Index Terms**: *Incremental, Perioidic, Partial periodic, High Utility itemset mining HUIM), Transactional database.***

I. INTRODUCTION

We are in a world where women are taken granted when it is infertility issue. The fact is, there is a scope of infertility in men which can also cause loss of pregnancy in women. Andrology is a subject on male reproductive & urological problems that is one of the factors for infertility from the male perspective. Our article uses this medical infertility dataset to solve the major problem in identifying the combination of anti oxidants supplement for the success rate using our 'Incremental Partial Periodic High utility Itemset Mining' IPPHUIM algorithm. Initially, we aim at finding the utility list structure with the data available incrementally and find the high utility itemsets by restructuring list using weighted transaction utilities which is called the IHUIM algorithm and next our algorithm identifies the periodic patterns that uses timestamp list, periodicity and utility threshold values to prune the periodic items which are not of any interest, namely IPHUIM algorithm and finally, the IPPHUIM algorithm identifies partial periodic items from periodic mining as a lot of interesting patterns are more likely in a subset combination which are lost in a full periodic pattern mining. The IPPHUIM algorithm requires one database scan and to speed up the process, effective pruning strategies are used.Our article is prepared as follows: section 2 discusses on the literature review about the HUIM. The next section 3, explains brief concepts of Andrology, the dataset, HUIM, Incremental Mining, Periodic & partial periodic mining. Section 4 involves in the problem statement and proposed model of IPPHUIM. The proposed algorithms of all phases are detailed in section 5. Example and experimental study can be seen in section 6 and the final section 7 concludes the paper.

Revised Manuscript Received on June 12, 2019.

Suvarna U, Department of Information Technology, Gitam University, vishakapatnam, AP, India.

Srinivas Y, Department of Information Technology, Gitam University, vishakapatnam, AP, India.

II. RELATED WORKS

A. High utility itemset mining: HUIM:

From the traditional frequent itemset mining (FIM) where the frequency of an itemset with its support is considered the most important factor to identify itemsets and assumes only one appearance for each item in one transaction [1] to find an itemset , the era of High utility itemset mining (HUIM) which considers the utility factor ,that each item can have a weight or unit price and that non binary purchase quantities should be allowed in transactions with the introduction of two utilities, internal utility and external utility values [2] emerged and efficiently mining the utility(profitable itemsets) maintaining the downward closure property of FIM, (which failed to work with initial utility mining) by a 2 phase algorithm with the concepts Total weighted utility (TWU) and upper bounds and applied overestimated concept to retain antimonote property by [3] and also applied effective strategy for pruning search space. The concepts of TWU , number of scans with the database and generating candidate itemsets were focussed in [4] and minimised the HUIM algorithms to just 2 scans and effective candidate pruning by using UP Growth algorithm using tree structure. [5] introduced a novel utlity list which performs only one database scan and no candidate generation and no tree structures to find the high utility itemsets. Tseng et al,[6][7] introduced various concepts of high utility itemset mining to filter the HUIs with Top K HUIs[9] ,maximal, minimal, closed itemsets, [10] found average utility list structure for economical pruning strategies over HUIs, which covers the average instead of max and min utilities for more efficient itemsets. While Viger et al [7] worked with FHM, perioidic HUIs, strict patterns, flexible patterns and filters the non perioidic itemsets and used an Estimated Utility Co-occurrence structure (EUCS) to find high utility

items,on the other hand T.P.hong et al [11] worked with incremental databases with HUIs , short periods[8]. The partial periodic mining was introduced in time series databases by J,Han et al [23] identifying that the partial periods which resulted in efficient patterns than the limited full perioidic patterns.

B. Incremental HUIM:

The incremental utility mining IUM, using an aprioiri method but numerous scans to find items was discovered by T.PHong et al(2008)[11],which uses a 2 phase algorithm with overestimation concept as discussed in HUIM , Ahmed et al (2009)[26] used a tree based approach over incremental databases, which performs two scans , one for global tree structure and another for overestimation, but this remained inefficient in pruning strategies. A canonical tree is also considered for incremental mining by Zhan et al [17] to improvise the tree search space and pruning effectively. A list based alogorithm for mining incremental HUIs without a candidate generation over dynamic databases using TWU values to sort the items and only 2 scans were found by Lin et al [13], Yun et al[14] worked on static and dynamic incremental databases, where high utility patterns in incremental databases growth were dicrimated with Tree and list structures. An efficient List based algorithm called Efficient Incremental HighUtility Itemset, by viger et al[19] uses utlity list and tree structure with two database scans to effectively find the most utility itemsetsover dynamic incremental databases. Yun et al[15] worked on a List Incremental High Utility Pattern where a single database scan model using list structure with an optimal sorting order which results in the most efficient utility itemsets. Yun worked further on List based on dynamic incremental database using an erasable pattern mining, also called LINE algorithm[16] which using more efficient Gain methods to erase the patterns which are less profitable

C. Periodic HUIM:

A pattern is said to be periodic if the pattern is repeating regularly over a given interval by the user. Ahmed et al[27], introduced Periodic Frequent pattern Tree(PF Tree),and enables the growth of the tree using periodicity and support thresholds. The periodic HUIM or PHM discovered by viger et al, [18], worked on periodic mining with average and minimal periodicity to filter a large number of non periodic itemsets. Ashish et al [21] worked on time series databases for periodic pattern mining and Ahmed et al [20], worked on a new framework for time series databases using weighted periodicity. Uday et al [22] worked on periodic mining over very large databases effectively.

D. Partial periodic HUIM:

Han et al [23] introduced partial periodic mining over time series databases but was not effective utility mining due to costs over the data structure and memory consumption, Aref et al [25], found the various kinds of mining with all possibilities of databases such as online mining, merge mining and partial perioidic mining and how they can be implemented over incremental databases but still suffers with costly scans and poor datastructures. The high utility mining for partial periodic mining was studied by T.PHong [24] used two phases ,where phase 1 uses periodic length and upper bound sequence mining for finding periodic itemsets and further in phase 2 they filter partial periodic itemsets.Our article aims at dynamic incremental partial periodic mining using a list structure to scan the database only once and work efficiently on large transactional databases. The proposed algorithm uses novel list structure to arrange the items scanned from the database dynamically in a List based structure and reordered based on TWU values and uses pruning strategy with support thresholds and gets all profitable HUIMs, In next phase the periodicity constraint is added to the support threshold to further prune the periodic itemsets and finally filter the partial periodic itemsets over a transactional database.

III. BACKGROUND

The dataset used in this article is Male Infertility dataset (Fig 2)and the proposed algorithm is a dynamic incremental database where data is loaded incrementally/dynamically into the database and high utility itemsets which are incremental, periodic and partial in nature are extracted effectively without missing the data due to dynamic insertions and changes in the utilities and also the list structure which is a challenge to the legacy algorithms with respect to execution time and memory consumption and consuming only one database scan without any candidate generations.

A. Male Infertility

Male infertility is defined as inability of men to cause pregnancy in a female who may be fertile, also called as andrology. Fig 1 explains that infertility is caused 30% due to male factor, 20% both and remaining due to female factors alone. The suggested treatments for Female were covered in the previous paper dataset suvarna et al (2019)[] ,the major cause is due to abnormlaities in semen analysis, treatments are suggested. But if the analysis is normal and still have male infertile issues, the unexplained infertility has to be treated which is a challenge. The reasons can be DNA damage, Reactive oxygen species(ROS), Oxidation stress, obesity, alcohol, smoking habits, leucocytes, etc.... to treat this kind of scenario, we need the combination of antioxidants dosage given to the patient.

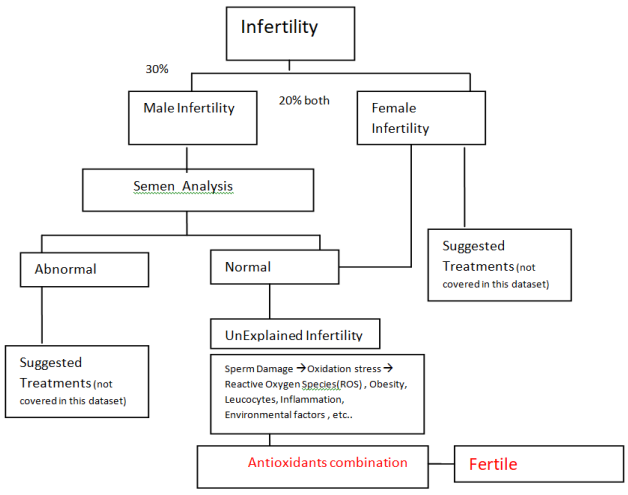


Fig 1: A chart on Male infertility causes and solution

The dataset MI_dataset has the following attributes and the description of each attribute is shown in the Fig2. The dataset

has 20 attributes, 2650 records collected from Krishna IVF clinic and is used to find the most important combination of antioxidants along with other major factors which are periodically done on patient for every attempt and determine when the treatment is successful. Our algorithm worked very effectively in flitering the most utility itemsets which are periodic and its partial periods that are more successful for the treatments accurately.

B. HUIM

High Utility itemset mining HUIM is the emerging mining activity in the area of data mining where transactional, timeseries databases can explore many utility measures in effectively extracting the interesting patterns. As discussed in section 2.1, frequent itemsets allows each item to appear once in each transaction and we determine it frequent with support measures, HUIM uses the internal utility (appearance of all items in transaction) and the external utility (cost incurred on each item) to find the utility measure and this HUIM suffers without able to generate antimonote property. For this reason, Total weighted utility is introduced and upperbounds to avoid overestimate concept and that if overestimation utility of an item is not smaller than utility threshold,we call those patterns promising and remaining are pruned. HUIM works on both static and dynamic databases. Let us take the database table Tab 1, utility table Tab 2 and the output table Tab 3 to show how a static database access the data in HUIM.

Attributes	Description
Patient_id	Patient_id (extension) from previous_IVF_dataset from Suvarna et al(2019) [14 attributes] [season, sex="male", age , child_disease, accidents, surgeries, fever,alcohol,smoke,sit_hours,condition="Malesubfertiltiy" or "Unexplained" , weight,attempts, diagnosis]
Impotence	Patient suffering with Impotence (Yes |No)
Inflammation	Inflammation in testicles (one | both | none)
Oxidation stress	Oxidation stress (protein damage | DNA damage | Lipid peroxidation |altered_cell | hypoxia | starvation | others | none)
Reactive oxygen species[ROS]	ROS stress affected in the structure (Oxygen | Superoxide Anion | Peroxide |Hydrogen Peroxide | Hydroxyl radical |Hydroxyl ion | others |none)
Leucocytes	Levels of blood cells (low | moderate |high)
X-Ray Radiation	Exposed to X-Ray radiation ever [never, once, few, regular]
Genetic_disorder	Have genetic disorders (yes | No)
Environment_impact	Did the environment impact on the mood swings (Yes | No)
Working_industry	Worked or working in an industry ever (never | few_years_back | working_now)
AO_Vit E	Antioxidant used for treatment, Vitamin E (yes | No)
AO_Vit C	Antioxidant used for treatment, Vitamin C(yes | No)
AO_Mg	Antioxidant used for treatment,Magnesium (yes | No)
AO_Folic	Antioxidant used for treatment, FolicAcid (yes | No)
AO_Zinc	Antioxidant used for treatment, Zinc (yes | No)
AO_N-acetyl	Antioxidant used for treatment, N-acetylcysteine (yes | No)
AO cartinites	Antioxidant used for treatment, Cartinites (yes | No)
AO_Coenzyme	Antioxidant used for treatment, CoenzymeQ10 (yes | No)
AO_selenium	Antioxidant used for treatment, Selenium (yes | No)
AO_DHA	Antioxidant used for treatment, DHA (yes | No)
Treatment_done	Treatment done and there was a (live birth | Pregancy | DNA_Damage fix | sperm_motility_ improved | sperm_concentration_improved | no change | others)

Fig 2: MI_dataset, Male infertility dataset with the attributes and their description

Tab 3 shows the overall result of the proposed algorithm which use TWU values and counts the frequency of the item to obtain frequency, noting the timestamps, periods and maxperiod length which will be discussed in periodic mining below section 3.4 for understanding.

Tab 1: static transactional database, Db with 10 records or transactions

TxId	Transaction	Count	Total Utility (TotUtil)
1	abcd	2,2,1,2	21
2	abdeg	4,1,3,1,2	31
3	abd	4,2,1	19
4	acefj	2,1,1,2,1	22
5	abdeh	5,1,3,2,1	36
6	cefgh	2,5,2,1,2	30
7	bdefi	3,5,4,2,1	56
8	acfj	1,5,3,2	31
9	abcfj	6,5,4,3,1	49
10	dgh	3,1,5	26

Tab 2: Utility Table

Item	A	b	C	D	E
Utility	2	3	1	5	3
Item	F	g	H	I	J
Utility	4	1	2	2	6

C. HUIM in incremental databases

Unlike Tab 3, a static database where the scan of database is done once and the values are updated statically and doesn't reflect any related parameters due to its static behaviour, In reality the databases are dynamic and static methods like above might get the result but not effective and cannot address dynamic issues. A dynamic or incremental database keeps updating the Original database (db) dymically, here every batch of transactions that are dymanically entered as called as increments (db1+, db2+, etc...) as shown in Tab 4. They require additional process in retaining the TWU values for every increments and ordering them to see the best utility items or patterns and avoid multiple scannings which is an expensive procedure.The previous section 2.2 explains various kinds of incremental mining algorithms. Our algorithm aims at finding the incremental HUIMs in the first phase with only one database scan.

Tab 4:Incremental transactional database, Db (original 1-6 rows) with two dynamic insertions Db1+ (7-8 rows) and Db2+ (9-10 rows)

TxId	Transaction	Count	Total Utility (TotUtil)
1	abcd	2,2,1,2	21
2	abdeg	4,1,3,1,2	31
3	abd	4,2,1	19
4	acefj	2,1,1,2,1	22
5	abdeh	5,1,3,2,1	36
6	cefgh	2,5,2,1,2	30
7	bdefi	3,5,4,2,1	56
8	acfj	1,5,3,2	31
9	abcfj	6,5,4,3,1	49
10	dgh	3,1,5	26

D. Perioidic mining

The periodic mining finds the repeated order of items over a period in a given instance. The occurrence of items (or a sequence of items) in a given period length and the frequentness of the pattern determines the periodic nature. The section 2.3 discusses on various periodic pattern mining algorithms and thei approaches of finding periodic patterns. The section 4.1 definitions of periodic frequent pattern explains how a pattern is periodic using the support ,utility and period lengths.

E. Partial periodic mining

When it comes to a periodic pattern, there are two scenarios, if a full period is high utility itemset and we have its subsets with even more high utilities and another scenario,where we prune the complete full period which is too large and we might loose valuable information that might yield a high profit. So, instead of discarding a full periodic itemset, we must provide flexibility by allowing the subsets which yield high profit in that period. For e.g 1: {abcd} is a periodic pattern, the subsets {ab},{cd} might be having high utility measure and meets the periodicity instead of complete {abcd} to yield profit and we try to discard them. Similarly, include subsets of high profit when a full period with high profit is considered. So, a partially periodic patterns are the subsets of the full periods which yield high profit.

IV PROBLEM STATEMENT AND PROPOSED MODEL

A. Problem statement

Let's consider the Incremental transactional database (Tab 4), and utility table (Tab 2) for the problem and apply the definitions and properties as follows:

Definition 1. Utility : A product of quantitative measure, 'internal utility' & profit measure, 'external utility' is said be an utility measure.

Utility measure: Util(i, Tx) = IntUtil(i,Tx) * ExtUtil(i,Tx) , where i is the item and Tx is the transaction of item i.e.g 2: The values that are counted in the transaction table (Tab 4) for items are considered to be the internal utility of an item & the values in the utility table (Tab 2) are the external utilities for items, it can be considered as the profit of the particular item. The Util(a, Tx1) = 2*2=4.

Definition 2. Total Utility (TotUtil): The total utility can be defined as the sum of all the utilities of all the items in the transaction Tx.

Total Utility: TotUtil(Tx) = $\sum_{i \in Tx}$ Util(i, Tx) where i is the items of that transaction.

e.g 3: From Tab 1 &2, TotUtil(Tx1) = $\sum_{abcd \in Tx1}$ Util(abcd, Tx1) = Util(a, 1) +Util(b, 1) + Util(c, 1) +Util(d, 1) =2*2 + 2*3 + 1*1 + 2*5 =21.

Similarly, TotUtil(Tx2)= $\sum_{abdeg \in Tx2}$ Util(abdeg, Tx2) = Util(a,2) +Util(b, 2) + Util(d, 2) +Util(e, 2) + Util(g, 2)=4*2 + 1*3 + 3*5 + 1*3 + 2*1 =31.

Definition 3. Transaction Weighted utility (TotWtUtil): The transaction weighted utility of an itemset can be defined as the sum of all transaction utilities in a database containing item i. TotWtUtil(i) = $\sum_{Tx \in Db\ ,\ i \in Tx}$ TotUtil(Tx) where 'i' is the itemset in all transactions Tx in database Db.

e.g 4: TotWtUtil(c) = TotUtil(Tx1) + TotUtil(Tx4) + TotUtil(Tx6) + TotUtil(Tx8) + TotUtil(Tx9) = 21+22+30+31+49= 153.

Property 1: Util threshold value: minutil: It is the value of total utility of the database D, Util(D) with the threshold value say γ. Then, minutil=Util(D) * γ.

e.g 5: The total utilty of database is 150 and threshold (γ) with 30% , then minutil=Util(D) * γ = 150*0.3=45.

Property 2: Overestimation utility property : if a total weighted utility of an item ov, TotWtUtil(ov)= $\sum$TotUtil(Tx) , where Tx€Db , ov€Tx then the over estimation of the value ov is not smaller than minutil. It is considered as a profitable pattern.

e.g 6: Say if minutil=45, Util (CE) =(1*1 + 1*3) +(2*1+5*3)= 21 < minutil, but TWU(CE)=TU(T4) +TU(T6) = 30 +21 = 52 > minutil. This concept is called overestimation. Now instead of neglecting these patterns, it can be used further to mine the items.

Property 3: upperbounds: The sum of the utility value and scaled utility which is the remaining in the list which is a previous utility of that item due to previous transactions.

Upperbound= Util(It) + ScaleUtil(It).

When the itemset doesn't meet the minutil, we use the upperbound value and allow the soft item.

Property 4: scaled utility list: It is a backup list of all the previous transaction values of the utility from high order items.

Defenition 4: List structure: A structure that consists of the set of lists for all the items It={It1, It2,…} individually with its first length and appended the List for every transaction Txid. The List structure is formed with three value records for each transaction of the item when scanned. [Txid , Util(It) ,ScaleUtil(It)] where Txid is the transaction ID, Util(It) is the utility of the itemset and the scaleutil (initially set to zero in creation) of the itemset as discussed in above property 4.

Property 5. Transaction Weighted utility List (TWUList): The List with the values [It1, v1],[It2, v2],……[Itn, vn] where It is the item and v is the total weighted utility value of ith item.Since the databases we deal are dynamic and as the new item gets inserted and if the item now is not more than minutil (an invalid item) can next become the value more than minutil which is a (valid item) and also viceversa. So we need to use property 5 for ordering the items as per the ascending order of items and the ScaleUtils have to be restructured.

Property 6. Restructure List: Since the databases are dynamic as discussed, we need to restructure the list with scaleutils which are initially set to zero while creating the List. Without any further scans, the restructure is done using the backup table which stores the previous value as discussed in property 4.

Property 7: Utility List of length K: The Lists are created for new itemset with length K after the first length and is repeated until no more lists can be created with new itemsets.

Definition 6. Period of an itemset : Let Db is the database with n transactions {Tx1, Tx2,…..Txn} and itemset X, then p(X) can be defined as { Tp(1) → Tp(2) → …Tp(n) } where period p(x) ranges between 1 and n.

Definition 7. Periodic Timestamp: The timestamp of a period of two transactions Tx and Ty is the difference between the transaction values, y-x.

Property 8: timestamp_list : It is a list that contains the itemset , timestamp periods, count, frequency and the maxperiod_value. [item, ts_period, count, freq, max_period]. Where Freq, The frequency of an item is the number of times the item appeared in all transactions.
Item, The itemset can be first length initially and extends to k-itemset as discussed.
Ts_period, The timestamp periods are defined in definition 6.
Count, The count of periods takes the initial transaction Tx value and summed up with the periodic timestamps defined in definition 7 until the last transaction. And max_period is the highest Ts_period in the list.
Property 9: K_itemset List: It is the list formed for every K-itemset formation where the list structure values are computed and stored with pruning strategies in the List for every itemset.
Property 10 : Pruning search space using util-lists. Let X be an itemset and ExX be its extensions, the extension itemsets formed by appending an oher item oi to X where oi ∈ X. If the sum of intUtil and extUtil values in Util(X) < utility threshold, minutil , then X and its extensions are low utility itemsets.
Property 11 : Pruning search space using util_threshold and period_length. Let X be an itemset and ExX be its extensions, the extension itemsets formed by appending an oher item oi to X where oi ∈ X. If the sum of intUtil and extUtil values in Util(X) < utility threshold and period > Max_period, minutil , then X and its extensions are low utility itemsets.

Definition 8 : partial_periodic itemset: It is defined as the subset of full periodic itemsets that are high utility itemsets.

B. Proposed model

The proposed model Fig 3 has three phases namely Incremental HUIM (IHUIM), Incremental Periodic HUIM (IPHUIM) and Incremental Partial Periodic HUIM (IPPHUIM). In the first phase, we find the incremental HUIM using List structures and pruning strategy with a utility threshold. Secondly, we aim at periodic itemsets by repeating the process for k-itemsets and further prune by max period length and utility threshold. The last phase filters the subsets of periodic itemsets which are already high utility itemsets.
During the first phase, we take MI_dataset ,here " Db" Tab 4 and and utility table Tab 2 as inputs and scan the database Db for every transaction dynamically and form a list structure for the items individually. The list is constructed when the item is coming for the first time, otherwise the details are appended in the list. we further work on constructing Total weighted utility List to reorder the list structure for every increment and Time stamp list to

Tab 3. static transactional database, with Total weighted values TWU, timestamps, period, frequency and maxperiod length

Item	TWU	Time stamp of Item	Count of Period (c)	Frequency (f)	Maximum Period length (MaxP)
A	209	Ts_a 1->2->3->4->5->8->9	1+1+1+1+1+3+1+1	7	3
B	212	Ts_b 1->2->3->5->7->9	1+1+1+2+2+2+1	6	2
C	153	Ts_c 1->4->6->8->9	1+3+2+2+1+1	5	3
D	189	Ts_d 1->2->3->5->7->10	1+1+1+2+2+3+0	6	3
E	175	Ts_e 2->4->5->6->7	2+2+1+1+1+3	5	3
F	188	Ts_f 4->6->7->8->9	4+2+1+1+1+1	5	4
G	87	Ts_g 2->6->10	2+4+4+0	3	4
H	92	Ts_h 5->6->10	5+1+4+0	3	5
I	56	Ts_i 7	7+3	1	7
J	102	Ts_j 4->8->9	4+4+1+1	3	4

Tab 5.1, a list structure for the items under scan

{a} [Tx,Util,Sutil]			{b}[Tx,Util,Sutil]			{c}[Tx,Util,Sutil]			{d}[Tx,Util,Sutil]		
1	4	0	1	6	0	1	1	0	1	10	0

Tab 5.2, a list structure after the second scan

{a}			{b}			{c}			{d}			{e}			{g}		
1	4	0	1	6	0	1	1	0	1	10	0	2	3	0	2	2	0
2	8	0	2	3	0				2	15	0						

Tab 5.3, final item utility list after the original database Db is scanned

{a}			{b}			{c}			{d}			{e}		
1	4	0	1	6	0	1	1	0	1	10	0	2	3	0
2	8	0	2	3	0	4	1	0	2	15	0	4	3	0
3	8	0	3	6	0	6	2	0	3	1	0	5	6	0
4	4	0	5	3	0				5	15	0	6	15	0
5	10	0												

{f}			{g}			{h}			{j}		
4	8	0	2	2	0	5	2	0	4	6	0
6	8	0	6	1	0	6	4	0			

.

Tab 5.4, Reordered utility List structure as per the TWU values in ascending order of their weights, j<f<g<h<c<b<d<e<a

{j}			{f}			{g}			{h}			{c}		
4	6	0	4	8	0	2	2	0	5	2	0	1	1	0
			6	8	0	6	1	0	6	4	0	4	1	0
												6	2	0

{b}			{d}			{e}			{a}		
1	6	0	1	10	0	2	3	0	1	4	0
2	3	0	2	15	0	4	3	0	2	8	0
3	6	0	3	1	0	5	6	0	3	8	0
5	3	0	5	15	0	6	15	0	4	4	0
									5	10	0

Tab 5.5, Restructured utility List structure using the backup table , Tab 8

{j}			{f}			{g}			{h}			{c}		
4	6	16	4	8	8	2	2	29	5	2	34	1	1	20
			6	8	22	6	1	21	6	4	17	4	1	7
												6	2	15

{b}			{d}			{e}			{a}		
1	6	14	1	10	4	2	3	8	1	4	0
2	3	26	2	15	11	4	3	4	2	8	0
3	6	9	3	1	8	5	6	10	3	8	0
5	3	31	5	15	16	6	15	0	4	4	0
									5	10	0

Tab 5.6, the final restructured utility list structure after db1 scan

{j}			{i}			{g}			{h}			{c}		
4	6	16	7	2	48	2	2	29	5	2	34	1	1	20
8	12	19				6	1	28	6	4	24	4	1	15
												6	2	23
												8	5	14

{f}			{a}			{b}			{d}			{e}		
4	8	7	1	4	16	1	6	10	1	10	0	2	3	0
6	8	5	2	8	21	2	3	18	2	15	3	4	3	0
7	8	40	3	8	7	3	6	1	3	1	0	5	6	0
8	12	2	4	4	3	5	3	21	5	15	6	6	15	0
			5	10	24	7	9	31	7	25	12	7	12	0
			8	2	0									

Tab 5.7 List structure after inserting the first increment database scan db1+

{j}			{f}			{g}			{h}			{c}		
4	6	16	4	8	8	2	2	29	5	2	34	1	1	20
8	12	0	6	8	22	6	1	21	6	4	17	4	1	7
			7	8	0							6	2	15
			8	12	0							8	5	0

{b}			{d}			{e}			{a}			{i}		
1	6	14	1	10	4	2	3	8	1	4	0	7	2	0
2	3	26	2	15	11	4	3	4	2	8	0			
3	6	9	3	1	8	5	6	10	3	8	0			
5	3	31	5	15	16	6	15	0	4	4	0			
7	9	0	7	25	0	7	12	0	5	10	0			
									8	2	0			

Tab 5.8, final Restructured utility list structure after db2+ scan

{i}			{g}			{h}			{j}			{c}		
7	2	54	2	2	29	5	2	34	4	6	16	1	1	20
			6	1	28	6	4	24	8	12	19	4	1	15
			10	1	25	10	10	15	9	6	43	6	2	23
												8	5	14
												9	4	39

{e}			{f}			{d}			{a}			{b}		
2	3	26	4	8	4	1	10	10	1	4	6	1	6	0
4	3	12	6	8	0	2	15	11	2	8	3	2	3	
5	6	28	7	8	34	3	1	14	3	8	6	3	6	0

6	15	8
7	12	42

8	12	2
9	12	27

5	15	13
7	25	9
10	15	0

4	4	0
5	10	3
8	2	0
9	12	15

5	3	0
7	9	0
9	15	0

Tab 7.4 , the final Ts-List after all incremental database scan is complete.

TimeStamp item	Time stamp [TS] ->	Frequency(f) =count(TS)	Period count (+) values	Max Period =(max value in the period count)
A	1->2->3->4->5->8->9	7	1+1+1+1+1+3+1+1	3
B	1->2->3->5->7->9	6	1+1+1+2+2+2+1	2
C	1->4->6->8->9	5	1+3+2+2+1+1	3
D	1->2->3->5->7->10	6	1+1+1+2+2+3+0	3
E	2->4->5->6->7	5	2+2+1+1+1+3	3
F	4->6->7->8->9	5	4+2+1+1+1+1	4
G	2->6->10	3	2+4+4+0	4
H	5->6->10	3	5+1+4+0	5
I	7	1	7+3	7
J	4->8->9	3	4+4+1+1	4

use it for the next phase. After the reordering of the items, we restructure the list using backup table and finally use the minutil or utility threshold to prune unwanted items and form the HUIMs which are incremental in nature, IHUIMs

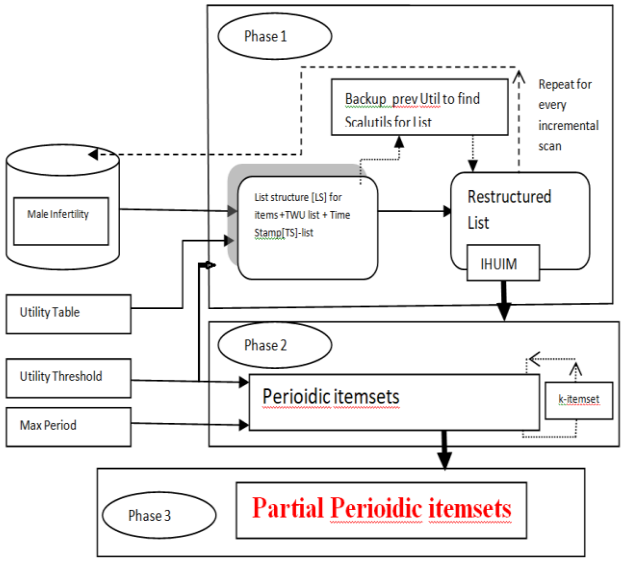


Fig 3: The proposed model showing the three phases 1)Incremental HUIM (IHUIM) , 2) periodic, IPHUIM and 3) partial periodic IPPHUIM methods.

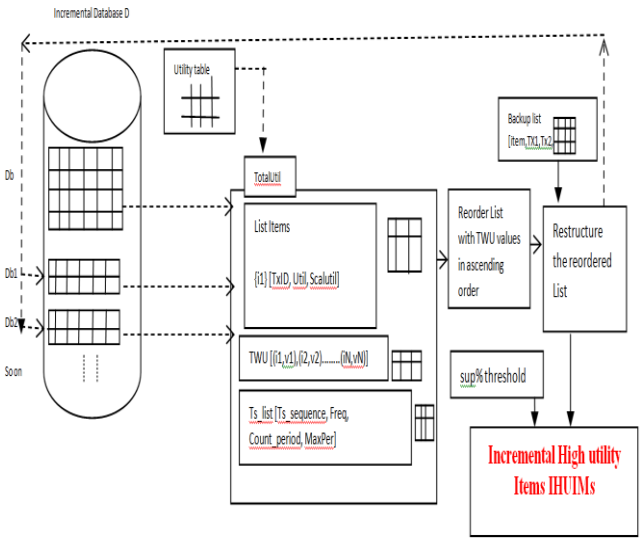


Fig 4: Phase 1: Incremental high utility itemsets , IHUIMs

The Second phase is about generating the periodic HUIMs with the help of Timestamp list we maintained from phase 1 (Fig 4), utility threshold and Max periodicity. If the itemsets util < utility threshold, minutil or period > max periodicity, the items are pruned and not promising items. We repeat the process for K-itemsets and generate all possible periodic itemsets. The strategy we use to store the itemsets is called the K-itemset_List Structure as shown in Tab 9.

V PROPOSED ALGORITHM

In phase 1 Algorithm , The inputs are incremental database 'db' such as (Tab 4) and the utility table such as (Tab 2). The database 'db' is scanned incrementally for every transaction and in every transaction, the items are identified (line 1) and created a List with their transaction item as seen in Fig 5.

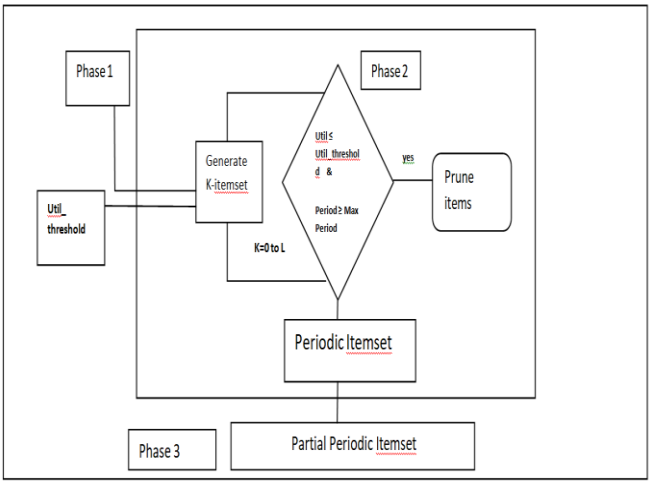


Fig 5: phase 1 to phase 3 algorithm flow to find partial periodic itemset

Calculated util values (internal * external utility value) and scalutil (initially zero) for that transaction Txid(Line 2 &3) . If in further scans, the item is already available, the list gets appened as the scans takes place(line 5) . For every scan we compute total utity TotUtil , the sum of all utilities in the transaction(line 6), Total weighted utilities(Line 7) and time stamps, their frequency, periods, max periodicity of that transaction(line 8). Being dynamic, from the original database to every increment we reorder the TWU values in ascending order (from Line 7) and Restructure the List in that

order with scalutuils (Line 9) using the backuptable. The backup table (Tab 8) is the previous values of Util for that
The Util value is compared to the Util_threshold and if its less than threshold, prune the items(remove from the list)(line 11). The remaining promising items are the resulted IHUIMs(Line 13). The following figure explains the data flow in all the three algorithms.In phase 2, we take the IHUIMs as input along with Util_threshold, Max_Periodicity which are user defined. We donot take either too short or too long periods or threshold values. The discussion in the Experimental study in section 6 will explain the values in detail. We have to find K-itemset periodic items by pruning the unpromising items which doesn't satisfy both Util < Util_threshold, and period > Max_period(line 3). The remaining itemsets are the resulted periodic IHUIMs , also called PHUIMs(Line 6)

Phase 1: Algorithm 1: Incremental High utility itemset mining algorithm(IHUIM)

Input: Incremental database [db] , utility_table , Util_threshold
Output: high utility itemsets, HUIs

1. Scan the database "db" for every transaction "Tx" do // T1, T2, T3, so on in database dynamically
2. If there is no Liststructure for items in transaction "Tx" then // no prev created list available for items i1,i2,....in T1 , T2,.....
3. Construct the Liststructure List_item(Txid, Util , scaleutil)
4. Else Append the ListStructure with List_item(Txid, Util,
5. scaleutil)
6. Total_Util = sum(util) of all items in Tx // see definition
7. Construct TWU_List(items)
8. Construct Ts_List(TimeStamp, freq, per_count, max_period)
9. Construct Restructure-list(Txid, Util, scaleutil) // Restructure the List
10. New_util =Util + scaleutil
11. If New_util <= Util_threshold // definition of HUIM
12. Prune items
13. Return HUIs // the items remaining after the pruning

List_item (Txid, Util, Scaleutil)

1. For every item i in Tx do
2. Txid = Current Tx number of the database
3. Util = IUtil * EUtil of item // see definition
4. Scaleutil = 0

Restructure_list(Txid, Util, scaleUtil)

1. Reorder the List_item(Txid, Util, scaleutil) in the increasing order of TWU_List
2. For each item from the last item [highest TWU_value item] repeat
3. Scaleutil = backup(scaleutil)

Backup(scaleutil)

1. For the first item , i.e, nth item , scale util = 0
2. For n-i item to 1 repeat
3. Scaleutil = Util(n-i+1)

TWU_List(items)

1. Item_name = item
2. TWU= sum (Tot_util(Tx)) of the item // see definition
3. Sort the item based on TWU value

TS_list(TimeStamp, freq, per_count, max_period)

1. For each transcation Tx[i]
2. For each item in transaction Tx[i]
3. TimeStamp= append(Tx[i] ->)// append all transactions visited to the time stamp chain
4. Frequency= count(Tx[i]) // no. of transactions for that item
5. Init= Tx[0]
6. Diff =Diff + [Tx[i]-Tx[i-1]]
7. Per_count= init +Diff // sum of all period values
8. max_period = max(Tx[i]) //Maximum value of all the period values

Phase 2: algorithm : Incremental Periodic high utility itemset mining: IPHUIM

Input: Util_threshold, Max_Per, TS_list , HUIs
Output: Perioidic High utility items , PHUIMs

1. For itemsets K=0 to L Length do
2. Util_thresh = Util_threshold% *TotUtil(Db) // total utility of database , sum of all TotUtil from phase1
3. If Util <= Util_thresh & max_period > Max_Per
4. Prune items // remove from the list
5. Return PHUIMs // the periodic items remaining after pruning

We further move to phase 3 and filter the subsets of all PHUIMs(line 2& 3) and remove the unpromising items which are not PHUIMs but subsets, return the PPHUIMs which are the final efficient, highly profitable itemsets.

Phase 3: algorithm: Incremental Partial Periodic high utility itemset mining: IPPHUIM

Input: PHUIMs
Output: Partial PHUIMs , PPHUIMs

1. For every PHUIMs do
2. Compute the subset of all PHUIMS
3. Choose only the subsets which are PHUIMs already
4. Prune other items
5. Return PPHUIMs // partial periodic HUIMS

VI WORKED EXAMPLE & EXPERIMENTAL STUDY

A. Worked Example

In Phase 1: IHUIM, Take Tab 4 & Tab 2 as inputs for the algorithm and scan the database as below. Here we only scan each row only once throughout the algorithm. There are three lists we follow in scanning and placing the values in lists.

1) Utility List – structure for items { Tab 5.1 to Tab 5. 7}
2) TWU List { Tab 6.1 to Tab 6.5 }
3) Time stamp, Ts_list {Tab 7.1 to Tab 7.4 }

First row scan from Tab 4: Transaction 1 : T1 {abcd}

The initial list structure is formed (in Tab 5.1) by taking the Transaction Id , Initial utility of item in the transaction and Remaining utility to be zero.

{a} for Tx1 is 1, Util is the utility measure ,2[IntUtil]*2[ExtUtil]= 4 and Sutil=0. Similarly utility can be measured for b,c and d. and the TotWtUtil of each item are simultaneously computed while updating the TWUList:

Tab 6.1, a TWU list TotUtil of Tx1: 21

a	b	c	D
21	21	21	21

The timestamp list TS_list helps in monitoring the time stamp, its frequency and perioidicty and notes the max period of that time stamp item as shown in Tab 7.1

Tab 7.1, a time stamp list, TS_list

TimeStamp item	Time stamp [TS]	Frequency(f) =count(TS)	Period count(+)	Max Period =(max value in the period count)
a	1->	1	1+	1
b	1->	1	1+	1
c	1->	1	1+	1
d	1->	1	1+	1

Now scanning the 2nd row {abdge}, here we notice that a,b and d were already constructed at the first scan, so we only extend the already existing list a,b,d and construct new lists for only g and e The same is shown as Tab 5.2, and the TWU list will be updated as follows:

Tab 6.2 , updated TWUList after Tx2

A	b	c	D	e	g
52	52	21	52	31	31

Tab 7.2, updated Timestamp list, TS_list on every scan

TimeStamp item	Time stamp [TS] ->	Frequency(f) =count(TS)	Period count (+) values	Max Period =(max value in the period count)
a	1->2->	2	1+(2-1)+	1
b	1->2->	2	1+(2-1)+	1
c	1->	1	1+	1
d	1->2->	2	1+(2-1)+	1
e	2->	1	2+	2
g	2->	1	2+	2

Similarly, forming the List structure with the values in all the items, and re ordered the list with the ascending order of TWU.Tx1 to Tx6 is the original database, let's consider only Tx1 to Tx6 in constructing the total List structure after the 6th scan 'cefgh' of Tx6.

Tab 6.3, final TWUList for the original database Db scan

a	b	c	D	e	F	g	h	j
129	107	73	107	119	52	61	66	22

Tab 7.3 Timestamp_list after the original database scan

TimeStamp item	Time stamp [TS] ->	Frequency(f) =count(TS)	Period count (+) values	Max Period =(max value in the period count)
a	1->2->3->4->5	5	1+(2-1)+(3-2)+(4-3)+(5-4)+	1
b	1->2->3->5	4	1+(2-1)+(3-2)+(5-3)	2
c	1->4->6	3	1+(4-1)+(6-4)	3
d	1->2->3->5	4	1+(2-1)+(3-2)+(5-3)	2
e	2->4->5->6	4	2+2+1+1	2
f	4->6	2	4+2	4
g	2->6	2	2+4	4
h	5->6	2	5+1	5
j	4	1	4	4

With the above scans, the original database is completely scanned and we need to reorder the list structure based on TWU values in ascending order (from Tab 5.3 and Tab 6.3),we get the reordered list as Tab 5.4

The ascending order as per TWU values is j<f<g<h<c<b<d<e<a

Restructure the reorder list with Sutil values as Tab 5.5.After the reorder as per TWU values, we need to restructure the list which is ordered with the help of scaled values saved in the backup.The Sutil of the most valued TWU item is assumed to be "0" and the backup list (Tab 8) is updated with the Util values of the transactions for that item. Here "a" is the top most TWU, so Sutil for itemset {a} is 0 for all transactions as shown in Tab 5.5. the remaning Sutils are updated with the previous holding values of items in their transactions. For e.g 7: Sutil of {d} ={4,11,8,16} which is obtained from the previous itemset {e} values of those transactions used in item "d".

Tab 8 , the backup table to follow up previous transaction value to use in Sutil.

Item/transaction	1	2	3	4	5	6
a	4	8	8	4	10	0
e	4	11	8	7	16	15
d	14	26	9	7	31	15
b	20	29	15	7	34	15
c	21	29	15	8	34	17
h	21	29	15	8	36	21
g	21	31	15	8	36	22
f	21	31	15	16	36	30
j	21	31	15	22	36	30

{e} has values {4,11,8,7,16,15} for transactions {1,2,3,4,5,6}. {d} has transactions {1,2,3,5} and values from {e} are {4,11,8,16} respectively. The Sutil values of {d} = {4,1,,8,16}

We repeat the same until the ascending order is complete for all items in the list, and the final Sutil values can be seen in Tab 5.5 and backup values can be seen in Tab 8 for restructuring.Phase 1 for incremental database db1: adding Tx7 [bdefi], Tx8 [acfj]. The Tab 5.6 will be updated accordingly and the order of List changes as TWUList values changes (Tab 6.4)

Tab 6.4, final TWUList for the incremental database Db1 scan

a	b	C	d	e	f	g	h	i	j

160	163	104	163	175	139	61	66	56	53

Now List structure will be changed as per the Db1 scan TWUList ascending order j<i<g<h<c<f<a<b<d<e and the List is reordered based on this order, restructured with Sutil and the final output after db1 scan (increment) is as in Tab 5.7.Similarly, the db2 incremental database scan has transactions Tx9 [abcfj] ,Tx10 [dgh] can be inserted, reordered and restructured with the TWUlist Tab 6.5 values in ascending order i<g<h<j<c<e<f<d<a<b and the restructured utility list structure after db2 scan is shown in Tab 5.8

Tab 6.5, final TWUList for the incremental database Db2 scan

a	b	C	d	e	f	g	h	i	J
209	212	153	189	175	188	87	92	56	102

Phase 2:

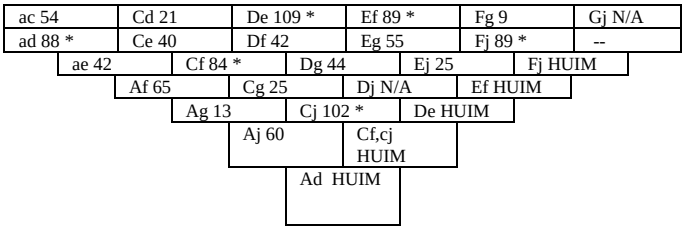
Tab 5.8 is the input for the next phase , here we try to find out the K-length itemset for their high utility. We also give a threshold value for utility (util_thresh%) and max_period as input.

Say, util_thresh%= 25%,

Then, TotUtil(Db) = 21+ 31+ 19+ 22+ 36+ 30+ 56+31+49+26=321. So util_threshold = util_thresh% * TotUtil(Db) = 321 * 0.25 = 80.18.

Any item that has utility value ≤ 80.18 will not be high utility itemset, since we considered the upper bounds. We check the sum of Util & Sutil ≤ util_threshold. If so, the items are pruned from the list and the left out items are productive high utility items. We repeat the process from 0-itemset to K-itemset until there can be no more decomposition.1-itemset = {a, b, c, d, …..} and 2-itemset = {ab,ac,ad,bc,bd,……} and so on. After applying the 25% threshold%, only a,c,d,e,f,g,h,j are considered and b,i are pruned. Again for profitable perioidic items, we apply max_period to prune the items further. In the example, say Max_period=4, any item with period ≥Max_period are pruned. If Max_period=4, then only a,b,c,d,e,f,g,j are better and h,i are pruned. Now from both, the common itemset with k=1 are a, c, d, e, f, g, j.For k=2, ac, ad, ae, af, ag, aj, cd, ce, cf, cg, cj, de, df, dg, dj, ef, eg ,ej, fg, fj, gj the threshold values are as in the step Tab 11.1 and max_period ≥4 further prunes 'ad'

Tab 9.1 , 2-itemset List which is a step table with k-itemset values and the values denoted by * are periodic itemsets



For K=3, cfj, cef,def, efj . The below explains the way to compute cfj and iutils are summed up and Sutils are taken with minimum Sutil values of both.

Tab 10: List structure of 2-itemset and its resulted Tab 9.2

{cf}

4	9	4
6	9	0
8	17	2
9	16	27

{cj}

4	7	15
8	17	14
9	10	39

{cfj}

4	16	4
8	34	2
9	26	27

Tab 9.2 , 3-itemset list

Cfj	Cef
109 *	58

Therefore, 1-itemset, a,c,d,e,f,g,j.

2-itemset, AD, Cf, Cj,de,ef,fj and 3-itemset is CFJ.

{ a,c,d,e,f,g,j, AD, Cf, Cj,de,ef,fj, CFJ}

Now cfj is the most high utility itemset which is periodic, but the length cant be further extended in this example, so the perioidic itemsets of k-itemsets are return.

Phase 3: if we observe, cfj, Tab 10, there are no other full periods for it and we might lose the periodic observations and their profits associated. Therefore, the partial periods of cfj in the HUIM list available are c, cf, cj which are subsets or partial perioidic items to cfj. Sometimes they yield even better results without loosing their quality compared to a fully observed period. Hence the partial periodic itemsets are c,cf,cj,cfj.

The example when applied to MI dataset (Fig 2), we could find the itemsets such as *oxidation_stress(os),smoke(s), works_in_industry (w), smoke & works_in_industry (s_w),smoke & works_industry & AO-VitE(s_w_vitE),AO-VitE&AO-Coenzyme(vitE_CoE),AO-VitE&AO-COEnzyme& sperm_concentration(VitE_CoE_spC),AO-VitE&AO-Coenzyme & AO-Selenium(VitE_CoE_Sel) , AO-VitE&AO-Coenzyme & AO-Selenium & Lowsperm_motility* (VitE_CoE_Sel_spM) were the most profitable itemsets which are partial periodic in nature.From above itemsets , we can implement rules that are highly profitable to further find,the relation among the combination of antioxidants and the treatment. For e.g 10: if Antioxidant Vit E & antioxidant Coenzyme in combination was given to the patient, its more likely the treatment is effective for sperm_concentration. Also proven by showell et al (2014) [32] and the threshold value is taken at 55%, max_period_length=5 for high expected accurate results.

B. Experimental study

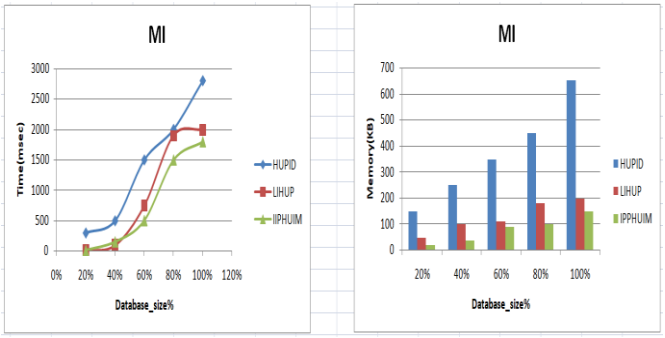
The expirements performed for the proposed method using a windows10 Enterprise 64 bit OS with processor Intel(R) Core(TM) i7-6700HQ CPU @2.60GHz & installed memory (RAM) with 8GB and real datasets are used where real patients details and experimented, the dataset used is about the male infertility factors (ramaraju et al (2014)[30] and worked with our model and algorithms are developed in java language. The graphs are drawn in MS Excel.

Tab 11 : Statistical data about the IVF_dataset Infertility dataset , MI dataset Male infertility , IVF_MI dataset both_IVF and MI attributes dataset.

Dataset	IVF_dataset	MI dataset	IVF_MI dataset
Size	1.2MB	650KB	820KB
Attributes (Items)	14	20	34
Records (Transactions)	10000	2500	3800
Avg Length of transaction	5	6	15

Max Length of transaction	10	14	28

The min_util threshold should not be either too low or too high. The performance comparision with regard to running time is measured with varied minutil for each dataset. The range from 25% to 55% shows the larger number of HUIMs and also more running time in Fig 7. The period_length should be minimal, if the period_length is increasing , the running time is also increasing. The less the minutil and the more the periodic length, the execution time is more. For comparisons, we considered our proposed algorithm IPPHUIM , LIHUP algorithm ,Unil Yun et al (2017) [15] and HUPID algorithm by Unil Yun et al(2015) [14] which are both incremental database which scans the database only once but the latter works on static databases.As the database size initially takes less time for execution, with every increment the runtime is gradually increasing as in Fig 6. The Fig 17 a) the running time and memory consumption of Male infertility dataset (MI dataset) which we used for experimenting the algorithm extensively. 17 b) is the IVF dataset that shows the running time and memory consumption and 17 c) shows the IVF_MI dataset with its running time and memory consumption. The graph with x-axis of database size and y-axis with the time in milli seconds are scaled for runtime and its evident that in all the datasets, the IPPHUIM algorithm is more effective consumes minimal execution time compared to LIHUP & HUPID algorithms. The right side figures of Fig 6 shows the experimental results of memory consumption of the three datasets and the memory utilised by IPPHUIM is very less compared to the comparable algorithms. The x-axis scaled the database size and y-axis with the memory in KB or MB as per the statistics available in Tab 11.The experimental values of utilthreshold% are user defined and through various threshold inputs, its visible where the minutil% is effective in the datasets. At 55% the yield is more and also the execution time is comparative less than other thresholds and the memory usage is also minimal. Like Threshold, period length is also a challenge and is user defined. The Fig 8 shows the runtime of period length, as the period length incresses the time and memory isFig 6: a)Database size running time and memory consumption for MI dataset , b)Database size running time and memory consumption for IVF dataset and c)Database size running time and memory consumption for IVF_MI dataset



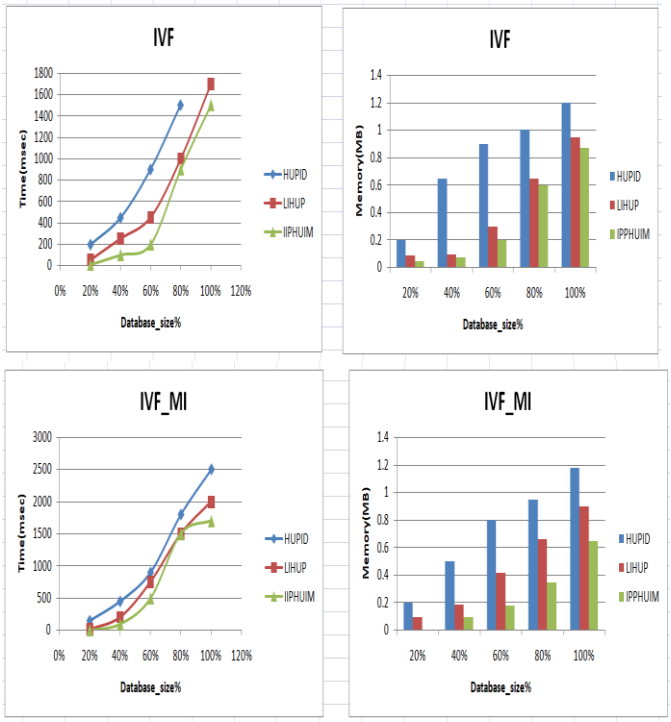
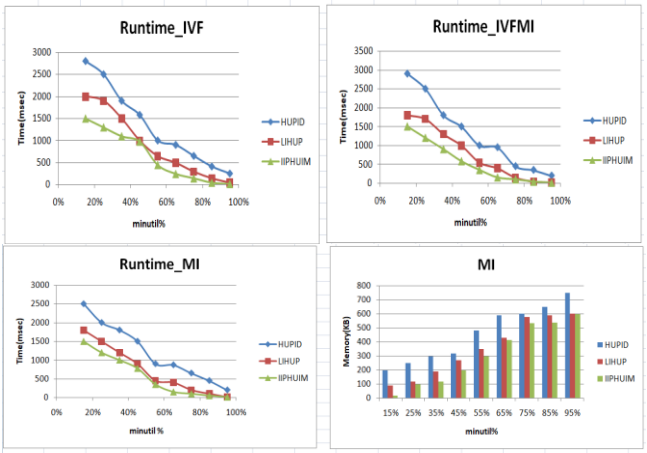
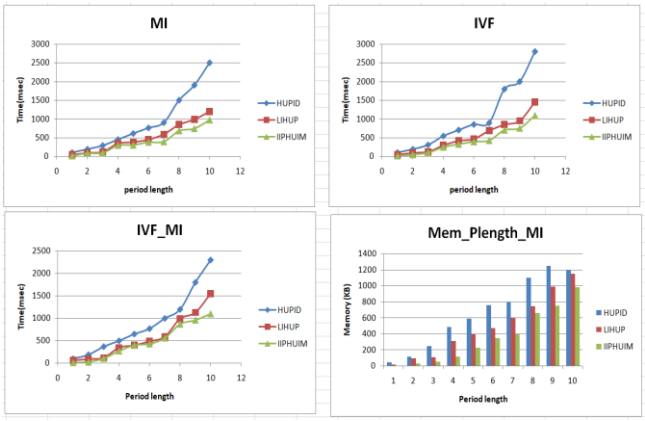


Fig 7: The runtime and memory consumption for various minutil thresholds. From clockwise runtime of IVF dataset, Runtime of IVF_MI, memory consumption of MI and runtime of MI datasets.



also consumed more, so the period length must not be too high and an average period length is always helpful. At length 6 the experimental study in the MI_dataset, length 4 in the IVF_MI dataset, periodic lengths between 4-6, the yield is high with minimal runtime and memory. Its visible at length 4, memory consumption is optimal and execution time is very less.

Fig 8: Periodic length and its runtime and memory consumption in dynamic databases, clockwise runtime of MI, runtime of IVF, Memory consumption of MI dataset and runtime of IVF_MI



Therefore the proposed algorithm is identified as the efficient high utility miner for dynamic datasets satisfying the profitable partial periodic itemsets with execution and memory consumption challenges.

VII CONCLUSION

Our article is a novel method of finding partial periodic high utility itemsets in a transaction database efficiently,scalably. The database is not just scanned once but the novel List structures for timestamp and K-itemsets reduced the memory space avoiding the enumerated trees and the joining operations and the pruning is made very simple. Since the datasets used in the experiments are real datasets, an extensive study was worked and compared with LIHUP algorithm which also challenges incremental databases with one scan. The thresholds are used in both phase 1 and phase 2 for pruning, but attimes too many thresholds prunes the valuable data at the beginning itself , so it can be taken optional in phase 1 while at use if the dataset size and attributes are dense. We have used Max_periodicity for pruning the periodic itemsets,but we can be soft with the algorithm using average periodicity.
In future work, we would like to work on novel periodicity measures, and apply gain ratios to the itemsets which can yield the optimal solutions to the itemset mining.

ACKNOWLEDGMENT

This work has been supported by “KRISHNA IVF CLINIC RESEARCH LAB, VISHAKAPATNAM” for providing the real datasets.

REFERENCES

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: International Conference on Very Large Data Bases, pp. 487–499 (1994)
2. Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: SIAM International Conference on Data Mining, pp. 211–225 (2004)
3. Yung Liu, Woe heng Liao, Alok Choudhary, “ A two-phase algorithm for fast discovery of high utility itemsets “, Advances in Knowledge Discovery and Data Mining, Berlin, Germany: Springer,Vol. 35, no.18,(2005), pp. 689–695.
4. V. S. Tseng, B.-E. Shie, C.-W. Wu and P. S. Yu, Up growth: An efficient algorithm for high-utility itemset mining, In Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 253–262, 2010.
5. Mengchi Liu, Junfeng Qu, “Mining High utility itemsets without candidate generation”, CIKM 2012 Procs of the 21at ACM conference on information and knowledge management, (2012), pp. 55-64.
6. Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu, “Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases”, IEEE Transactions on Knowledge and Data Engineering, Vol.25, No. 8, (2013), pp 1772-1786.
7. Vincent S. Tseng, Cheng-Wei Wu, Viger, Philip S. Yu, “ Efficient Algorithms for Mining Top-K High Utility Itemsets”, IEEE Transactions on Knowledge and Data Engineering, vol28, no.1, (2016),pp. 54-67.
8. Jerry ChunWeiLin, JiexiongZhang,Phillippe Foumier Viger, TzungPeiHong,JiZhang, “A two-phase approach to mine short-period high-utility itemsets in transactional databases”, Advanced Engineering informatics, vol 33, (2017),pp. 29-33.
9. Vincent S. Tseng, Cheng-Wei Wu, Viger, Philip S. Yu, “ Efficient Algorithms for Mining Top-K High Utility Itemsets”, IEEE Transactions on Knowledge and Data Engineering, vol28, no.1, (2016),pp. 54-67.
10. Unil Yun,Donggyu Kim, “Mining of high average-utility itemsets using novel list structure and pruning strategy”,Future Generation Computer Systems,vol 68, (2017), pp. 346-360.
11. T. P.Hong, C. W. Lin, Y. L,Wu, ”Incrementally fast updated frequent pattern trees”, Expert Systems with Applications, vol 34, no.2, (2008), pp. 2424–2435.
12. Lin C. W., Hong T. P., Lan G. C., Wong J. W., Lin W.-Y, “ Incrementally mining high utility patterns based on pre-large concept”, Applied Intelligence, vol 40, no.2, (2014),pp. 343–357.
13. Lin, C. W., Lan, G. C., & Hong, T. P. (2012). An incremental mining algorithm for high utility itemsets. Expert Systems with Applications, 39(8), 7173–7180.
14. UnilYun, Heungm Ryang,” Incremental high utility pattern mining with static and dynamic databases”, Applied intelligence, Vol 42, no 2, Mar(2015),pp. 323-352.
15. Yun, Unil, Heungmo Ryang, Gangin Lee, and Hamido Fujita. "An efficient algorithm for mining high utility patterns from incremental databases with one database scan." *Knowledge-Based Systems* 124 (2017): 188-206.
16. Gangin Lee, Unil Yun, “Single-pass based efficient erasable pattern mining using list data structure on dynamic incremental databases”, Future generation computer systems, vol 80, 2018, pp. 12-28.
17. Carson kai sang-leung, Quamrul.I.khan, Zhan Li, Tariqul Hoque, “Can Tree: a canonical order tree for incremental frequent-pattern mining”, Knowledge and Information systems, vol 11, no 3, (2007), pp. 287–311.
18. Philippe Fournier-Viger, Jerry Chun-Wei Lin,Quang-Huy Duong, Thu-Lan Dam, “PHM: Mining Periodic High-Utility Itemsets ”,ICDM2016,advances in data mining,applications and theoretical aspects, (2016), pp. 64-79.
19. Fournier-Viger, Philippe, Jerry Chun-Wei Lin, Ted Gueniche, and Prashant Barhate. "Efficient incremental high utility itemset mining." In *Proceedings of the ASE BigData & SocialInformatics 2015*, p. 53. ACM, 2015.
20. A.K.chanda,C.F.Ahmed, md Samiuallah, CK.Leung,” A new framework for mining weighted periodic patterns in time series databases”,Expert Systems with Applications,Vol 79,(2017),pp. 207-224.
21. Ashis Kumar chanda, Chowhury Farhan ahmed,md.Samiullah, Swapnil Saha, Manziba Akanda Nishi, “An efficient approach to mine flexible periodic patterns in time series databases”,Engineering Applications of Artificial intelligence, Vol 44, (2015),pp.46-63.
22. R. Uday Kiran,P. Krishna Reddy, Masaru Kitsuregawa,“Efficient discovery of periodic-frequent patterns in very large databases”,The journal of Systems and Software, Vol 112, (2016), pp.110-121.
23. J. Han, G. Dong and Y. Yin, “Efficient mining of partial periodic patterns in time series databases.”,The 15th International Conference on Data Engineering, 1999, pp. 106-115.
24. Hong, Tzung-Pei, Jen-Hao Hsu, Guo-Cheng Lan, Kung-Jiuan Yang, Shyue-Liang Wang, and Jerry Chun-Wei Lin. "High utility partial periodic pattern mining." In *Proceedings of the 4th Multidisciplinary International Social Networks Conference on ZZZ*, p. 35. ACM, 2017.
25. Aref, W.G., Elfeky, M.G. and Elmagarmid, A.K., 2004. Incremental, online, and merge mining of partial periodic patterns in time-series databases. *IEEE Transactions on Knowledge and Data Engineering*, *16*(3), pp.332-342.
26. Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Le, Y. K. (2009). Efficient tree structures for high utility pattern mining in

incremental databases. IEEE Transactions on Knowledge and Data Engineering, 21(12), 1708–1721.
27. Tanbeer, Syed Khairuzzaman, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee. "Discovering periodic-frequent patterns in transactional databases." In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 242-253. Springer, Berlin, Heidelberg, 2009.
28. Suvarna, U., and Y. Srinivas. "Efficient High-Utility Itemset Mining Over Variety of Databases: A Survey." *Soft Computing in Data Analytics*. Springer, Singapore, 2019. 803-816.
29. Kumar, Naina, and Amit Kant Singh. "Reactive oxygen species in seminal plasma as a cause of male infertility." *Journal of gynecology obstetrics and human reproduction*(2018).
30. Vijaya Bharathi, B., G. Jaya Prakash, K. M. Krishna, C. H. Ravi Krishna, T. Sivanarayana, K. Madan, G. A. Rama Raju, and A. Annapurna. "Protective effect of alpha glucosyl hesperidin (G-hesperidin) on chronic vanadium induced testicular toxicity and sperm nuclear DNA damage in male Sprague Dawley rats." *Andrologia* 47, no. 5 (2015): 568-578.
31. Gan, Wensheng, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, Tzung-Pei Hong, and Hamido Fujita. "A survey of incremental high-utility itemset mining." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, no. 2 (2018): e1242.
32. Showell, Marian G., Rebecca Mackenzie-Proctor, Julie Brown, Anusch Yazdani, Marcin T. Stankiewicz, and Roger J. Hart. "Antioxidants for male subfertility." *Cochrane database of systematic reviews* 12 (2014).
33. Suvarna, U., and Y. Srinivas. " A Classified Medical Infertility Dataset Using High Utility Item Set Mining
" *Inderscience* submitted not yet published.

AUTHORS PROFILE

Suvarna U. Research scholar, Gitam University,Vishakapatnam, AP, India
Email: viksonio@gmail.com

Srinivas Y Professor, Gitam University, Vishakapatnam, AP, India
Email: sriteja.y@gmail.com