Discovery of High Utility Itemsets Using Genetic Algorithm

S. Kannimuthu[#1], Dr. K.Premalatha[*2]

[#]Assistant Professor, Department of CSE, Coimbatore Institute of Engineering and Technology,
Coimbatore-641109, Tamil Nadu, India
[1]kannimuthu.me@gmail.com
[*]Professor, Department of CSE, Bannari Amman Institute of Technology,
Sathyamangalam-638401, Tamil Nadu, India
[2]kpl_barath@yahoo.co.in

***Abstract*--Contemporary research in mining high utility itemsets from the databases faces two major challenges: exponential search space and database-dependent minimum utility threshold. The search space is very huge when number of distinct items and size of the database is very large. Data analysts must specify suitable minimum utility thresholds for their mining tasks though they may have no knowledge pertaining to their databases. To evade these problems, two approaches are presented to mine high utility itemsets from transaction databases with or without specifying minimum utility threshold by using genetic algorithm. To the best of our knowledge, this is the first work on mining high utility itemsets from transaction databases using Genetic Algorithm (GA). Experimental results show that below mentioned GA approaches achieve better performance in terms of scalability and efficiency.**

***Keywords:* Association Rule Mining, Data Mining, Genetic Algorithm, Scalability, Utility Mining.**

I. INTRODUCTION

Data mining is the assemblage of techniques used to discover deep and imperceptible relationships from the database. Meaningful patterns in data will boost profitability, improve company productivity, and give the organization an edge in today's highly competitive environment. Knowledge discovery is being used by banks, investment houses, retailers and suppliers, marketing departments, engineering staff, customer service departments and large number of diverse organizations to make effective decisions. Data mining approaches may generate different kinds of knowledge such as association rules, classification rules, clusters etc,

The problem of extracting association rules has received significant research interest and numerous algorithms for mining association rules have been developed [1]-[4]. Mining association rules from the transaction databases is a two step process: 1) Finding all itemsets that are present in at least s% of transactions (frequent itemsets) and 2) Generating rules from each large itemset [4]. Association Rule Mining (ARM) algorithms only consider the presence or absence of items in a complete transaction. It does not reflect semantic factors like cost, profit etc., High utility pattern mining algorithms [5]-[19] resolves the problems in ARM by considering non-binary values in transactions and different profit values of every item.

Utility value for an item is defined by the user is not available in the transaction databases. It reflects user preference and can be represented by an external utility function or utility table. Utility table defines utilities of all items in a given database. Moreover, we also need internal utilities like quantity of items in transactions. Utility function is represented to compute utility of an itemset takes both internal and external utility of all items in an itemset. Consider, u(.) is the utility function. An itemset I is a high utility itemset if it satisfies the *minUtil* threshold, i.e., $u(I) \geq minUtil$. *minUtil* is a threshold that is defined by the user. Utility value of an itemset can be measured in terms of cost, profit, or other measures of user preference.

The major challenges faced by the data analyst are:

1) Search space for high utility itemset mining is exponential. The major factors decide the search space is size of the transaction and number of distinct items in a transaction database.

2) Data analyst need to specify minimum utility threshold to mine high utility itemsets. There are many algorithms and technologies for discovering high utility itemsets have been proposed by the researchers. These techniques largely focus on improving scalability and efficiency. Utility mining algorithms suggested in the literature are mostly based on the assumption that users can specify the minimum utility threshold appropriate to their databases. But, setting the minimum utility threshold is by no means an easy task.

To avoid these problems, genetic algorithm based techniques are designed to mine high utility itemsets from the transaction database effectively. Charles Darwin's "The Origin of Species" publication in 1859 brought about genetic algorithm detailing how complex, problem-solving organisms could be created and improved through an evolutionary process of random trials, sexual reproduction, and selection [20]. GAs are used to construct a version of biological evolution on computers.GA have been successfully adopted in a wide range of optimization problems such as control, design, scheduling, robotics, signal processing, game playing and combinatorial optimization [21]. Data mining is also one of the important application areas of genetic algorithms.

The main contributions of this paper are summarized as follows.

1. A novel evolutionary approach called **HUPE$_{\mathbf{UMU}}$-GA** is proposed to mine HUI which makes use of GA. In this approach, data analyst inputs minimum utility threshold (*minUtil*) value along with transaction database. This approach is preferred, when search space and memory usage are an issue.

2. An effective approach called **HUPE$_{\mathbf{WUMU}}$-GA** is proposed to mine HUI using genetic algorithm. This approach generates optimal HUIs without specifying minimum utility threshold.

The rest of this paper is organized as follows. Section II describes the basic concepts and definitions of utility mining and genetic algorithm. Section III presents the related works. Extracting high utility itemsets with genetic algorithm is given in Section IV. The proposed approaches are discussed in Section V and VI. Experimental results are discussed in Section VII. Conclusions are finally given in Section VIII.

II. BASIC CONCEPTS AND DEFINITIONS

This section exemplifies the basic concepts and definitions of utility mining and genetic algorithm.

A. Utility Mining

Utility mining can be viewed as the extension of ARM used to extract all high utility itemsets from the transaction database. Utility value of an itemset can be computed in terms of cost, profit or other interpretation of user preferences. An item set *x*is said to be a high utility itemset if and only if u(x) ≥ *minUtil*, where *minUtil* is a user defined minimum utility threshold. The formal definition of utility mining problem discussed in [22] is given below.

TABLE I
Transaction Table

TID/ITEM	**A**	**B**	**C**	**D**	**E**
T_1	1	0	0	2	1
T_2	0	1	2	6	0
T_3	3	0	0	5	0
T_4	1	0	0	0	1
T_5	0	1	2	6	0
T_6	0	1	1	0	2
T_7	2	0	0	0	0
T_8	3	0	0	1	0
T_9	0	1	1	4	0
T_{10}	1	0	0	0	1

TABLE II
Profit table

Item	**A**	**B**	**C**	**D**	**E**
Profit	5	3	2	6	10

$I = \{i_1, i_2, \ldots, i_m\}$ is a set of items, D= $\{T_1, T_2, \ldots, T_n\}$ be a transaction database where each transaction $T_i \in D$ is a subset of I. $o(i_p, T_q)$ local transaction utility value, represents the quantity of item i_p in transaction T_q. For example, $o(A, T_8) = 3$, in Table I. $s(i_p)$, external utility, is the value associated with item i_p in the Utility table. This value reflects the importance of an item, which is independent of transactions. For example, in Table II, the external utility of item A,$s(A)$ is 5.

$u(i_p, T_q)$, utility, the quantitative measure of utility i_p in transactionT_q, is defined as $o(i_p, T_q) \times s(i_p)$. For example,$u(A, T_8) = 3 \times 5$ in Table I. $u(X, T_q)$, utility of an itemset X in transactionT_q , is defined as $\sum_{i_p \in X} u(i_p, T_q)$, where $X = \{i_1, i_2, \ldots, i_k\}$is a k-itemset, $X \subseteq T_q$ and $1 \le k \le m$. $u(X)$, utility of an itemset X, is defined as

$$\sum_{T_q \in D \wedge X \subseteq T_q} u(X, T_q) \tag{1}$$

We find all the high utility itemsets using utility mining. An itemset X is a *high utility itemset* if $u(X) \geq minUtil$, where $X \subseteq I$and minUtil is the minimum utility threshold.

For example, in Table I, $u(A, T_8) = 3 \times 5 = 15$, $u(\{A, D\}, T_8) = u(A, T_8) + u(D, T_8) = 3 \times 5 + 1 \times 6 = 21$, and $u(\{A, D\}) = u(\{A, D\}, T_1) + u(\{A, D\}, T_3) + u(\{A, D\}, T_8) = 17 + 45 + 21 = 83$. If minUtil=80, then $\{A, D\}$ is a high utility itemset. However, if an item is a low utility item, its superset may be a high utility itemset. In this database, A is a low utility item (u(A)=55<80), but its superset {A,D} is a high utility itemset (u({A,D})=83>80). Utility mining approach does not support downward-closure property [7]. Hence, we generate combinations of all items and same should be processed to ensure that no high utility itemset will be lost.

Liu et al. [13] presented the Two-Phase (TP) algorithm for mining high utility itemsets. TP algorithm has two phases. In the first-phase, transaction utility (tu) for all the transactions is calculated. Next, set of all single itemset is identified and transaction-weighted utilization (twu) for the same is calculated by scanning the database. Combinations of high transaction weighted utilization itemsets are added into the candidate set at each level during the level-wise search. This phase maintains a Transaction-Weighted Downward Closure (TWDC) property [14]. First-phase may overestimate some low utility itemsets, but it never underestimates any itemsets. In second-phase, one extra database scan is performed to filter the overestimated itemsets.

TABLE III
Transaction utility for the transactions in a database

TID	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**T_7**	**T_8**	**T_9**	**T_{10}**
TU	27	43	45	15	43	25	10	21	29	15

Consider the Table I, we have 10 transactions, $tu(T_1)$ be the transaction utility of T_1will be$tu(T_1) = u(A, T_1) + u(D, T_1) + u(E, T_1) = 1 \times 5 + 2 \times 6 + 1 \times 10 = 27$. Transaction utility for all the transactions is listed in Table III. Transaction-weighted utilization of an itemset A, denoted as twu (A), is the sum of the transaction utilities of all transactions containing A. In Table I, $twu(A) = tu(T_1) + tu(T_3) + tu(T_4) + tu(T_7) + tu(T_8) + tu(T_{10}) = 27 + 45 + 15 + 10 + 21 + 15 = 133$.

TABLE IV
Terminologies used in Genetic Algorithm

Terminology	Description
Locus	A position in the genome is called locus.
Allele	The value at a particular locus in the genome is called the allele.
Genome	A particular feature in the chromosome is corresponding to a genome.
Chromosome	A collection of genomes representing a prospective solution to the problem is called a chromosome (individual).
Fitness function	It is a measure associated with the collective objective functions that indicate the fitness of a particular chromosome.
Survival of the fittest	The fittest individuals are preserved and reproduce, which is referred to as survival of the fittest.
Selection	The process of picking effective chromosomes from the population for a later breeding is called as selection.
Crossover	The process of creating a new chromosome by mating two or more valuable chromosomes is known as crossover.
Mutation	The process of randomly changing the value of an allele to arrive an optimal solution is called as mutation.

B. Genetic algorithm

Genetic algorithm is a directed optimization and search technique that can solve highly complex and often highly nonlinear problems. It is used to investigate very large problem spaces and find the best solution based on fitness functions under a set of multiple constraints. The genetic algorithm starts with a large population of feasible solutions and through the application of crossover and mutation evolves a solution that is better than any previous solution over the lifetime of the genetic analysis. The basic terminologies used in genetic algorithm are mentioned in Table IV.

III. RELATED WORKS

In recent manuscripts dedicated to the subject of utility mining we were able to identify that different factors influence the performance of the algorithms, i.e., utility measures used, data structures adopted and pruning strategies applied. We summarized these papers in Table V. Yao. et al.'s [22] theoretical analysis of utility mining problem presented the foundation for future utility mining algorithms. They proposed Apriori-like algorithms, called UMining and UMining_H [18], to extract high utility itemsets level by level. Mining high utility itemset often leads to the generation of a large number of patterns. Unlike Frequent Itemset Mining (FIM), utility mining model does not satisfy anti-monotone property. Several pruning strategies [13], [16]-[19], [23], [24] have been proposed to improve the performance of the utility mining algorithms.

Vital research issue extended from the utility mining is the discovery of high utility patterns in data streams [6], [11], [15] due to the wide applications on various domains. Discovering high utility sequential patterns can be considered a special type of mining which was first introduced in [19]. They presented an algorithm named USpan to efficiently mine high utility sequential patterns using lexicographic quantitative sequence tree. Traditional utility mining algorithms are failed to find high utility itemsets with negative values from large databases. This issue has been investigated through Chun-Jung et al. [8] by proposing High Utility Itemsets with Negative Item Values-Mine (HUINIV-Mine) algorithm. HUINIV-Mine uses TP algorithm to mine high utility itemsets. Mining Web Access Sequences (WASs) can extract very useful knowledge from web logs with wide-ranging applications. Chowdhury et al. [7] presented a pioneering work to mine high utility WASs. Two tree structures, called utility-based WAS tree (UWAS-tree) and incremental UWAS-tree (IUWAS-tree) proposed for mining WASs in static and incremental databases.

Unfortunately, recent studies show that utility mining algorithms may suffer from some weaknesses such that (1) large search space and (2) problems with database dependent minimum utility threshold. Many HUI mining algorithms [5], [6], [9]-[11], [15]-[17], [19] have been proposed to reduce the search space. These algorithms use tree-based approach. It's widely acknowledged that tree-based approach achieves a better performance than Apriori-like approaches since it finds HUIs without generating candidate itemsets. Y.Liu et al. [13] proposes the TP algorithm which efficiently extort high utility itemsets from the databases. TP algorithm utilizes Transaction-Weighted Downward Closure (TWDC) to reduce the search space. It was observed that many unpromising candidates were still generated when we use this strategy. This issue can be tackled by using HUPE$_{\text{UMU}}$-GA to efficiently mine HUI using genetic algorithm.

Setting a suitable minimum utility threshold is a difficult problem for data analysts. If *minUtil* threshold is set too low, huge number of high utility itemsets will be generated, which may cause the mining algorithms to become inefficient or even run out of memory. In contrast, if *minUtil* threshold set too high, no HUI will be extracted. We address this problem by proposing a new approach called HUPE$_{\text{WUMU}}$-GA to mine Top-K high utility itemsets without specifying *minUtil* threshold.

IV. EXTRACTING HIGH UTILITY ITEMSETS WITH GENETIC ALGORITHM

Let $I = \{i_1, i_2, \dots, i_m\}$ is a set of items, D= $\{T_1, T_2, \dots, T_n\}$ be a transaction database where each transaction $T_i \in D$ is a subset of I. An itemset X is a high utility itemset if it satisfies the *minUtil* threshold, i.e., $u(X) \geq minUtil$. *minUtil* is a threshold which is defined by the user. This section portrays the building blocks of genetic algorithm used in the proposed works.

TABLE V
Performance of different utility mining algorithms

Algorithm	Measures used	Data structures used	Year	Pruning Strategies used	Dataset	No. of transactions	Number of items	Average length a transaction	minUtil Threshold	Execution time in seconds
TP algorithm [13]	Y. Liu et al.	List	2005	TWDC Property	Real world data set	1112949	46086	6	1	25.76
Parallel implementation of TPalgorithm[14]	Y. Liu et al.	List	2005	TWDC Property	T10I6D1000K	100000	1000	10	2	20
					T20I6D1000K	100000	1000	20	2	50
UMining [18]	Yao et al.	List, Hash Table	2006	Upper bound property	Synthetic dataset	11160188	100	5	1	4200
UMining_H [18]					Synthetic dataset	11160188	100	5	1	3720
THUI-Mine [15]	Y. Liu et al.	List	2006	TWDC Property	T10I6D100K	100000	1000	10	1	80
CTU-Mine [9]	Y. Liu et al.	CTU-Tree	2007	TWDC Property	T10I5D50K	50000	1000	10	1	700
CTU-PRO [10]	Y. Liu et al.	CTU-Tree	2007	TWDC Property	T10N5D100K	100000	1000	10	1	20
HUQA [24]	Yen and Lee	List	2007	Support Bound property	T6I4D1000K	1000000	1000	6	1	2000
					T6I4D100K	100000	5000	6	0.5	230
HUINIV[8]	Y. Liu et al.	List	2009	TWDC Property	T10I6D1000K	100000	1000	10	1	900
					BMS-POS	515597	1657	6.53	1	3200
IHUPTWU [5]	Y. Liu et al.	IHUP$_{TWU}$-	2009	TWDC Property	Real-time retail	88162	16470	10.3	1	100
HUPMS [6]	Y. Liu et al.	HUS-Tree	2010	TWDC Property	T10I4D100K	100000	870	10	2	350
					BMS-POS	515597	1657	6.53	4	550
IHUP-FPG [16]	Y. Liu et al.	IHUP-Tree	2010	TWDC Property	T10I6D100K	100000	1000	10	0.2	555
UP-FPG [16]	Y. Liu et al.	UP-Tree	2010	TWDC Property, DGU, DGN, DLU and DLN strategies	T10I6D100K	100000	1000	10	0.2	407

Algorithm	**Measures used**	**Data structures used**	**Year**	**Pruning Strategies used**	**Dataset**	**No. of transactions**	**Number of items**	**Average length a transaction**	**minUtil Threshold**	**Execution time in seconds**
UP-UPG [16]	Y. Liu et al.	UP-Tree	2010	TWDC Property, DGU, DGN, DLU and DLN strategies	T10I6D100K	100000	1000	10	0.2	283
IHUP-FPG [16]		IHUP-Tree		TWDC Property	BMS-Web-View-1	59602	497	2.51	2.9	401000
UP-FPG [16]		UP-Tree		TWDC Property, DGU, DGN, DLU and DLN strategies	BMS-Web-View-1	59602	497	2.51	2.9	2
UP-UPG [16]		UP-Tree			BMS-Web-View-1	59602	497	2.51	2.9	2
IHUP-FPG [16]		IHUP-Tree		TWDC Property	Chess	3196	76	37	55	10055
UP-FPG [16]		UP-Tree		TWDC Property, DGU, DGN, DLU and DLN strategies	Chess	3196	76	37	55	151
MHUI-TID [11]	Y. Liu et al.	LexTree-maxHTU	2011	TWDC Property	T10I4D100K	100000	870	10	2	700
					BMS-POS	515597	1657	6.53	4	1150
TKU[17]	Y. Liu et al.	UP-Tree	2012	TWDC Property, MC, PE, NU, MD and SE pruning strategies	Microsoft Foodmart Dataset	4,141	1,559	4.4	0.01	40
					Mushroom	8,124	119	23.0	0.01	1000
					Chainstore dataset	1,112,949	46,086	7.2	0.01	1400
USpan[19]	Y. Liu et al.	LQS-Tree	2012	TWDC Property, Width pruning and Depth Pruning	C10T2.5S4I2.5DB10kN1k	10000	1000	10	0.001	100
					C8T2.5S6I2.5DB10kN10k	10000	10000	8	0.002	120

A. Encoding

In this subsection, encoding is presented first. Three genetic operators, a population initialization and a fitness function based on the Yao et al. [22] measure. Finally, these modules are put together into the algorithm HUPE$_{UMU}$-GA and HUPE$_{WUMU}$-GA. Figure 1 shows the configuration of the genes (items) in the chromosome. Binary encoding is used in this approach. Binary value '1' represent presents of an item and '0' represent absence of an item in an itemset. Chromosome length is fixed and it is equal to number of distinct items (*n*) which is obtained from the transaction database.

i_1	i_2	$\dots$	i_n

Fig. 1. Chromosome

1	**0**	**1**	**0**	**1**	**1**	**0**

Fig. 2. Chromosome representation for the itemset $\{i_1, i_3, i_5, i_6\}$

```
population initialize (n, k)
begin
    i←0;
    for j← 0 to n do
    begin
        pop[j]←0;
    end
    while $i \leq k$do
    begin
        rand_no←rand(k);
        if pop[rand_no]≠1 then
        begin
            pop[rand_no]←1;
            i←i+1;
        end
    end
    return pop;
end
```

Fig. 3. Algorithm for initializing the population

B. Population Initialization

Given an itemset length 'k', all the genes (item) in a chromosome are encoded as '0'. The initial population is produced using random number generator. If the generated random number is 'r', then the chromosome is encoded as '1' at r^th position. This represent i_r item presents in a chromosome (itemset). Upon randomly generating an item in a chromosome, it is checked against other items already generated in the same chromosome and if the item is present a new number is randomly generated until it is unique. This is repeated until generating 'k' unique random numbers. This process should hold the conditionk $\leq$ n. The algorithm for population initialization is given in Figure 3.

Example:

Consider the chromosome length is seven and the itemset length is four. Random numbers generated by random number generator is {1, 3, 5, 6}.

The representation of a chromosome is shown in Figure 2.

C. Fitness function

The main goal this work is to generate the high utility itemsets from the transaction database. Hence, the fitness function is essential for determining the chromosome (itemset) which satisfy *minUtil* threshold. In HUPE$_{UMU}$-GA and HUPE$_{WUMU}$-GA algorithm, we use Yao et al.'s [22] utility measure *u(X)* as the fitness function

$$f(X) = u(X) = \sum_{T_q \in D \wedge X \subseteq T_q} u(X, T_q) \quad (2)$$

D. Genetic Operators

This subsection express three genetic operators, select, crossover and mutation.

1) Selection

Selection is the one footstep of the genetic algorithm in which individual chromosome is chosen from a population for later breeding (crossover). Selection operator acts as a filter of chromosome with considerations of their fitness value. There are many selection approaches available in the literature. In this work, roulette wheel selection [25] is used.

2) Crossover

Crossover is a significant feature of genetic algorithms. Crossover obtains two individuals called parents and constructs two new individuals called the offspring by swapping parts of the parents.

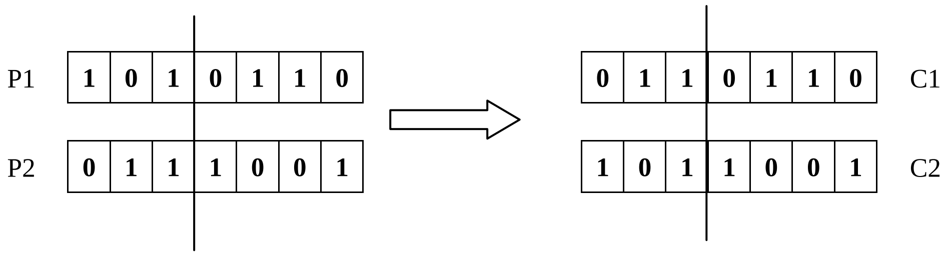

Fig. 4. Single-point crossover

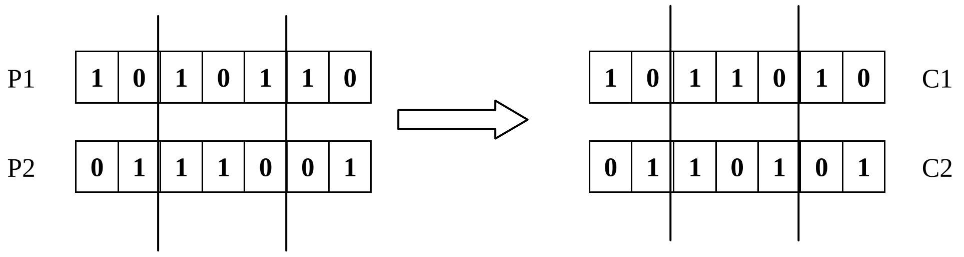

Fig. 5. Two-point crossover

In its simplest form, the operator works by exchanging sub-strings after a randomly selected crossover point. The illustration of crossover operation is given Figure 4 and 5.

3) Mutation

Mutation serves to prevent premature loss of population diversity by randomly sampling new points in the search space. Mutation rates are kept small however, otherwise the process degenerates into a random search. In this approach, mutation is applied by changing one or more random bits in a string when $P_m \geq$randomly generated probability value.

E. Evaluation

Evaluation step intends to select the chromosomes for next generation. In this work, elitist selection [25] method is used. This method copies the chromosome(s) with higher fitness value to new population.

F. Termination Criteria

It is the criterion by which the GA decides whether to continue searching or stop the search. The possible terminating conditions are listed below

1) Fixed number of generations reached

2) The solution's fitness with highest ranking at a fixed number of generations.

3) Manually inspecting the solution

4) Combinations of the above

V. HUPE$_{UMU}$-GA

Mining high utility itemset with minimum utility threshold using genetic algorithm is proposed that effectively extracts useful patterns from the databases. The steps in HUPE$_{UMU}$-GA are depicted in Figure 6. A genetic algorithm is chosen because it is a promising solution for global search and it is capable of discovering high utility itemsets with corresponding parameters quantity and profit. We use two measures such as TWU [13] for removing unpromising items from the transaction database at earlier stage and Yao et al. [22] measure for fitness value computation.

The steps of HUPE$_{UMU}$-GA algorithm are the following:

1. Scan the transaction database to compute transaction utility of each transaction. At the same time, TWU of each single item is also accumulated.

2. If the TWU of an item is less than minimum utility threshold, its supersets are unpromising to be high utility itemsets (TWDC property). So remove the corresponding unpromising items from the transaction database. The transaction database after the above reorganization is called reorganized transaction database.

3. Get the number of distinct items (NDI) from the reorganized transaction database. Set chromosome length (CL) to NDI.

4. Generate a Chromosome of length CL.

5. Evaluate the individual by calculating the fitness value (fv) and check $fv \geq minUtil$. If yes go to Step 6, otherwise go to Step 4.

6. Check whether the population size is equal to 'N'. If yes go to Step 7, else go to Step 4.

7. If the termination criterion is fulfilled, then present the best individual(s) in the population as the output of the evolutionary process and terminate. Otherwise, continue.

8. Select m individuals using roulette wheel selection that will compose the next generation with the best parents.

9. Perform crossover on the selected individuals of population.

10. Perform mutation on the individuals of population with mutation probability $P_{m.}$ Calculate the fitness value (fv) for the individuals and check $fv \geq minUtil$. If yes go to step 8, else go to Step 11.

11. Check if size of the new population reaches 'N'. If yes go Step 12, else go to Step 8.

12. Evaluate the individuals by using elitist selection of 'N' chromosomes from new and old population for next generation.

VI. HUPE$_{WUMU}$-GA

The proposed HUPE$_{WUMU}$-GA approach is used to mine optimal HUIs without specifying *minUtil* threshold. In this approach, Yao et al. [22] measure is used for fitness value computation. Flow of activities in HUPE$_{WUMU}$-GA is represented in Figure 7.

The steps of HUPE$_{WUMU}$-GA algorithm are the following:

1. Read the transaction database to find the number of distinct items (NDI). Set chromosome length (CL) to NDI.

2. Generate a Chromosome of length CL.

3. Evaluate the individual by calculating the fitness value (fv).

4. Check whether the population size is equal to 'N'. If yes go to Step 5, else go to Step 2.

5. If the termination criterion is fulfilled, then present the Top-K utility itemsets from the population as the output of the evolutionary process and terminate. Otherwise continue.

6. Select m individuals using roulette wheel selection that will compose the next generation with the best parents.

7. Perform crossover on the selected individuals of population.

8. Perform mutation on the individuals of population with mutation probability $P_{m.}$ Calculate the fitness value (fv) for the individuals.

9. Check if size of the new population reaches 'N'. If yes go Step 10, else go to Step 6.

10. Evaluate the individuals by using elitist selection of 'N' chromosomes from new and old population for next generation.

VII. EXPERIMENTAL EVALUATION

The experiments were performed on a machine with 2.33 GHz Intel® Core™ 2 Quad CPU and 2GB RAM, running on Windows 7. The experiments in Figures 8-13 were carried out on synthetic datasets from IBM data generator [26]. Dataset "T10.I4.D10K" means an average transaction size of 10, an average size of the maximal potentially frequent itemsets of 4 and 10,000 generated transactions. This dataset contains 100 distinct items. Quantity for each item in a transaction is also kept in the dataset (value in range of 1 to 10). Utility values for the items were assigned randomly in the profit table.

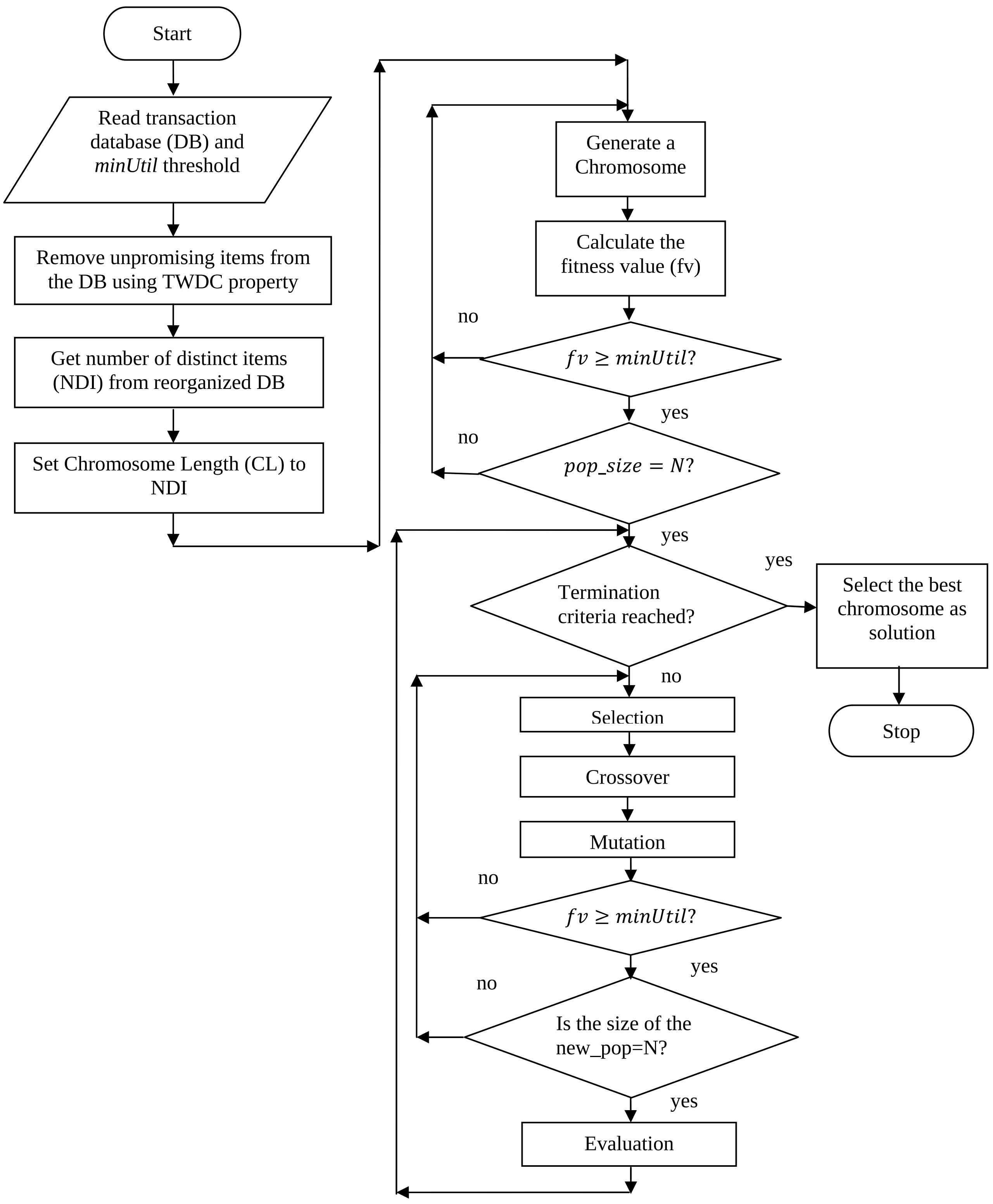
Start
Read transaction database (DB) and minUtil threshold
Remove unpromising items from the DB using TWDC property
Get number of distinct items (NDI) from reorganized DB
Set Chromosome Length (CL) to NDI
Generate a Chromosome
Calculate the fitness value (fv)
$fv \geq minUtil?$
no
yes
$pop\_size = N?$
no
yes
Termination criteria reached?
yes
no
Select the best chromosome as solution
Stop
Selection
Crossover
Mutation
$fv \geq minUtil?$
no
yes
Is the size of the new_pop=N?
no
yes
Evaluation

Fig. 6. Flow chart of $HUPE_{UMU}$-GA approach

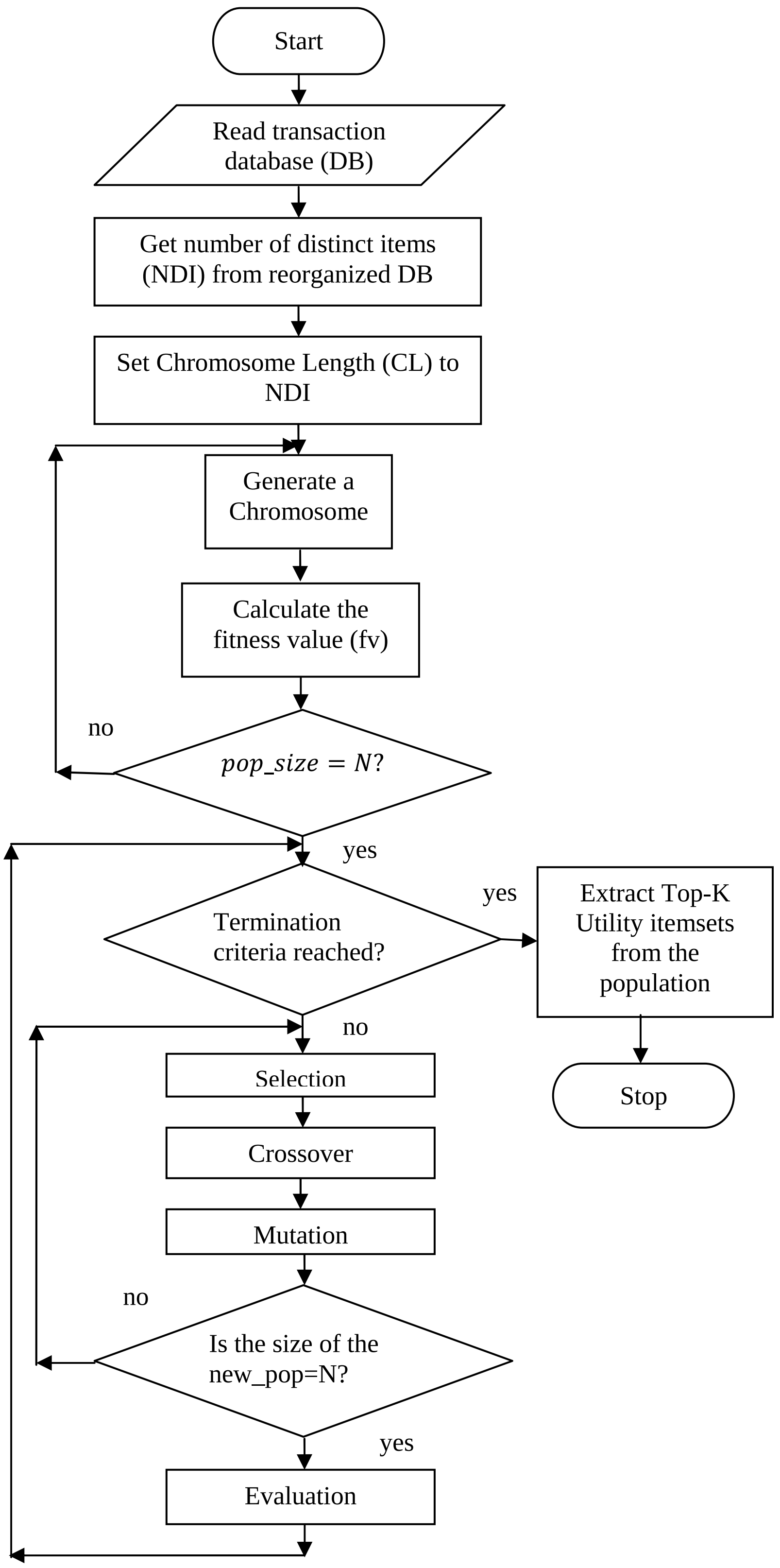
Start
Read transaction database (DB)
Get number of distinct items (NDI) from reorganized DB
Set Chromosome Length (CL) to NDI
Generate a Chromosome
Calculate the fitness value (fv)
no
pop_size = N?
yes
Termination criteria reached?
yes
Extract Top-K Utility itemsets from the population
Stop
no
Selection
Crossover
Mutation
no
Is the size of the new_pop=N?
yes
Evaluation

Fig. 7. Flow chart of $\mathrm{HUPE}_{\mathrm{WUMU}}$-GA approach

A. $HUPE_{UMU}$-GA

The first experiment was conducted with **$\mathrm{HUPE}_{\mathrm{UMU}}$-GA** approach in different number of generations by keeping *minUtil*=2% threshold and 100 distinct items as fixed. The test results for the dataset T10.I4.D10K is illustrated through the Figure 8 and Figure 9 respectively. It can be observed that the execution time of **$\mathrm{HUPE}_{\mathrm{UMU}}$-GA** approach for mining high utility itemsets from the databases proved to be significantly less.

To test the scalability of **HUPE$_{UMU}$-GA** with the number of transactions, experiments on IBM synthetic dataset are used. The *minUtil* threshold is set to 2%. The results are presented in Figure 10. **HUPE$_{UMU}$-GA** show linear scalability with the number of transactions from 10K to 100K. **HUPE$_{UMU}$-GA** scales much better and it supports large dataset.

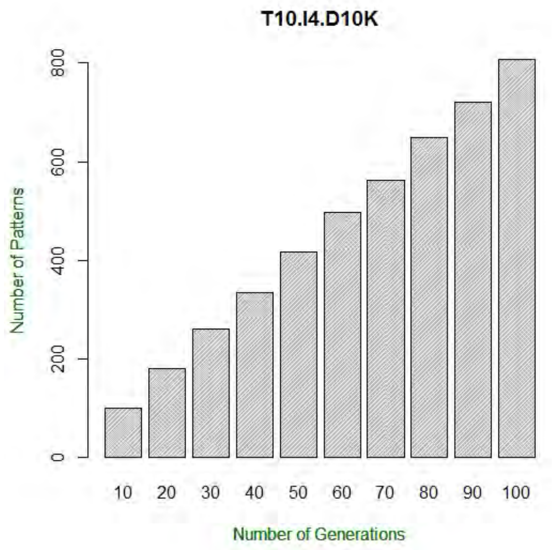

Fig. 8. Plot of number of generations versus number of patterns generated when *minUtil*=2%

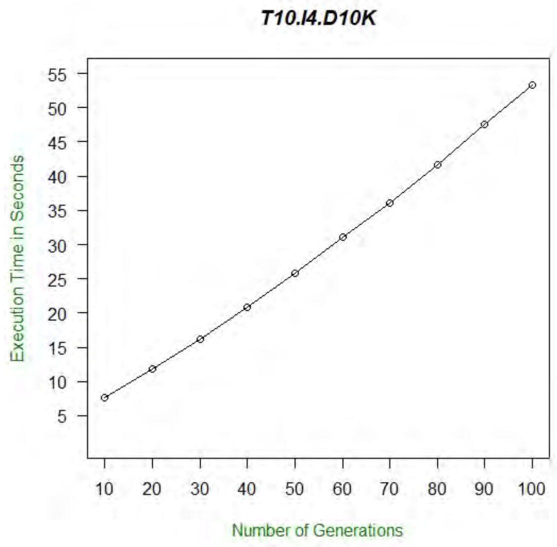

Fig. 9. Plot of number of generations versus execution time generated when minUtil=2%

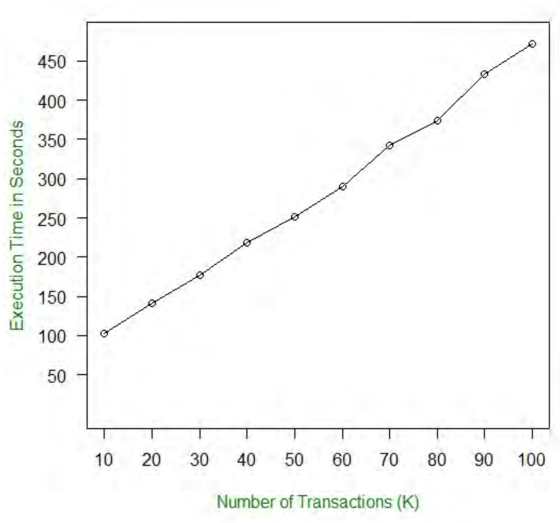

Fig. 10. Scalability with number of transactions

B. HUPE$_{WUMU}$-GA

The next experiment was carried out with **HUPE$_{WUMU}$-GA** approach in different number of generations by keeping 100 distinct items as fixed. The test results are illustrated through the Figure 11 and Figure 12 respectively. Like **HUPE$_{UMU}$-GA,** execution time of **HUPE$_{WUMU}$-GA** approach is proved to be considerably less. HUPE$_{WUMU}$-GA approach is also experimented by increasing the size of transaction from 10K to 100K. This approach shows linear scalability and scales much better which illustrated in Figure 13.

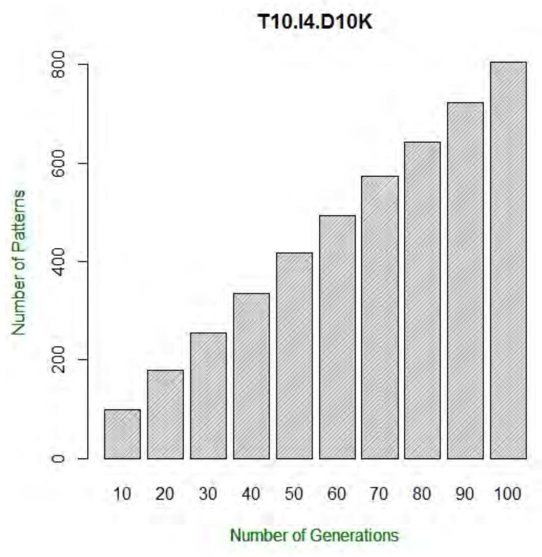

Fig. 11.Plot of number of generations versus number of patterns generated

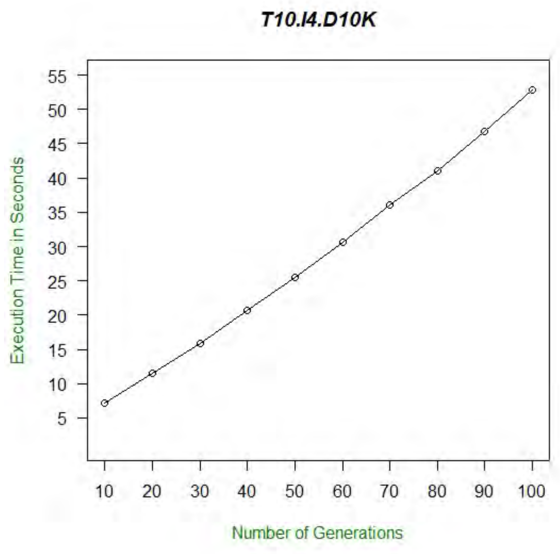

Fig. 12.Plot of number of generations versus execution time

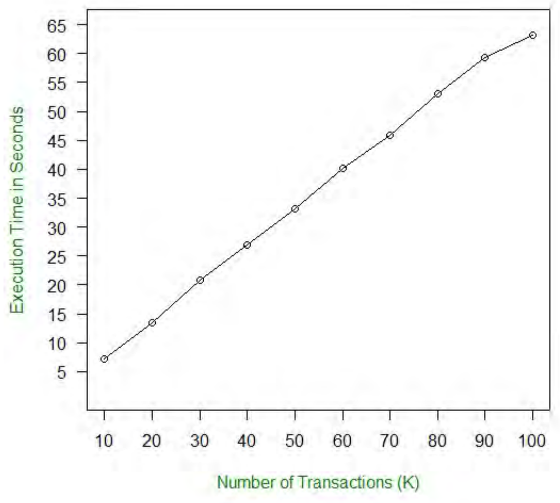

Fig. 13. Scalability with number of transactions

VIII. CONCLUSION

The approaches mentioned in this paper exploits a genetic algorithm, they can handle large number of distinct items and transactions. These approaches are the best choice when the execution time and memory requirement are the big issues. Genetic algorithms have proved to be a robust general purpose search techniques. They have a central place in data mining applications due to their ability to explore large search spaces efficiently. Two major challenges in utility mining i.e., exponential search space and database-dependent minimum utility threshold are studied and made an attempt to resolve the problems by proposing GA based approach to mine HUIs from the databases. Experimental results show that our proposed approaches scales well and retrieves the HUIs from the databases efficiently.

REFERENCES

[1] R. Agrawal, T. Imielinski and A. N. Swami, “Mining association rules between sets of items in large databases”. In Proceedings of ACM SIGMOD Conference, pp. 207-216, 1993.

[2] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules”. In Proceedings of the 20th VLDB Conference, Santiago, Chile, pp. 487-499, 1994.

[3] R. Agrawal and R. Srikant, “Mining Sequential Patterns”. In Proceedings of the 11thInternational Conference on Data Engineering, pp. 3-14, 1995.

[4] R. Agrawal and John C. Shafer, “Parallel Mining of Association Rules”, IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 8, No. 6, pp. 962-969, 1996.

[5] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong and Young-Koo Lee, "Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases", IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No. 12, pp. 1708-1721, 2009.
[6] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer and Byeong-Soo Jeong, "Efficient Mining of High Utility Patterns over Data Streams with a Sliding Window Method", In Proceedings of 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2010), pp. 99-113, 2010.
[7] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer and Byeong-Soo Jeong, "Mining High Utility Web Access Sequences in Dynamic Web Log Data", 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 76-81, 2010.
[8] Chun-Jung Chu, Vincent S. Tseng and Tyne Liang. "An efficient algorithm for mining high utility itemsets with negative item values in large databases", Applied Mathematics and Computation, Elsevier Journal, Vol. 215, No. 2, pp. 767–778, 2009.
[9] Alva Erwin, Raj P. Gopalan and N.R. Achuthan, "CTU-Mine: An Efficient High Utility Itemset Mining Algorithm Using the Pattern Growth Approach", In Proceedings of 7th International Conference on Computer and Information Technology, pp. 71-76, 2007.
[10] Alva Erwin, Raj P. Gopalan and N.R. Achuthan, "A Bottom-Up Projection Based Algorithm for Mining High Utility Itemsets", 2nd Workshop on Integrating AI and Data Mining (AIDM 2007), pp. 3-11, 2007.
[11] Hua-Fu Li, "MHUI-max: An efficient algorithm for discovering high-utility itemsets from data streams", Journal of Information Science, Vol. 37, No. 5, pp. 532-545, 2011.
[12] Ying Liu, Wei-keng Liao and Alok Choudhary, "A Fast High Utility Itemsets Mining Algorithm", UBDM'2005, pp. 90-99, 2005.
[13] Ying Liu, Wei-keng Liao and AlokChoudhary, "A two-phase algorithm for fast discovery of high utility itemsets". In 9th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD 2005), Lecture Notes in Computer Science, 3518, Springer-Verlag, Berlin, pp. 689–695, 2005.
[14] Shankar.S, Dr.Purusothaman and T, Jayanthi.S. "Novel Algorithm for Mining High Utility Itemsets", In Proceedings of the 2008 International Conference on Computing, Communication and Networking, ICCCN 2008, pp. 1-6.
[15] Vincent S. Tseng, Chun-Jung Chu and Tyne Liang, "Efficient Mining of Temporal High Utility Itemsets from Data streams", In Proceedings of the Second International Workshop on Utility-Based Data Mining, pp. 18-27, 2006.
[16] Vincent S. Tseng, Cheng-Wei Wu, Bai-En Shie and Philip S. Yu, "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, 2010.
[17] Cheng Wei Wu, Bai-En Shie, Philip S. Yu and Vincent S. Tseng, "Mining Top-K High Utility Itemsets", ACM, KDD'12, Beijing, China, pp. 78-86, 2012.
[18] H.Yao and Howard J. Hamilton, "Mining itemset utilities from transaction databases", Data & Knowledge Engineering, Elsevier Journal, Vol. 59, pp. 603-626, 2006.
[19] Junfu Yin, Zhigang Zheng and Longbing Cao, "USpan: An Efficient Algorithm for Mining High Utility Sequential Patterns", ACM Conference on KDD'12, Beijing, China, pp. 660-668, 2012.
[20] David Beasley, David R.Bull and Ralph R.Martin, "An Overview of Genetic Algorithms", University Computing, Vol. 15, No. 2, pp. 58-69, 1993.
[21] Alex Berson and Stephen J. Smith, "Data Warehousing, Data Mining and OLAP", Tata McGraw-Hil, India, 2004.
[22] H.Yao, Howard J. Hamilton and Cory J. Butz, "A Foundational Approach to Mining Itemset Utilities from Databases", Proceedings of the 3rd SIAM International Conference on Data Mining, Orlando, Florida, pp. 482-486, 2004.
[23] Yu-Chiang Li, Jieh-Shan Yeh and Chin-Chen Chang, "Isolated items discarding strategy for discovering high utility itemsets", Data and Knowledge Engineering, Elsevier Journal, Vol. 64, pp. 198-217, 2008.
[24] Show-Jane Yen and Yue-Shi Lee, "Mining High Utility Quantitative Association Rules", DaWaK 2007, Lecture Notes on Computer Science, Springer-Verlag, pp. 283-292, 2007.
[25] Holland J, "Adaptation in Natural and Artificial Systems", ed. Ann Arbor, University of Michigan Press, 1975.
[26] IBM Quest Market-Basket Synthetic Data Generator, http://www.cs.loyola.edu/~cgiannel/assoc_gen.html