Applied Artificial Intelligence, 28:337–359, 2014
Copyright © 2014 Taylor & Francis Group, LLC
ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839514.2014.891839

Taylor & Francis
Taylor & Francis Group

DISCOVERY OF HIGH UTILITY ITEMSETS USING GENETIC ALGORITHM WITH RANKED MUTATION

S. Kannimuthu[1] and K. Premalatha[2]
[1]*Department of CSE, Coimbatore Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India*
[2]*Department of CSE, Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, India*

□ *Utility mining is the study of itemset mining from the consideration of utilities. It is the utility-based itemset mining approach to find itemsets conforming to user preferences. Modern research in mining high-utility itemsets (HUI) from the databases faces two major challenges: exponential search space and database-dependent minimum utility threshold. The search space is extremely vast when the number of distinct items and the size of the database are very large. Data analysts must specify suitable minimum utility thresholds for their mining tasks, although they might have no knowledge pertaining to their databases. Moreover, a utility-mining algorithm supports only an itemset with positive item values. To evade these problems, two approaches are presented for mining HUI containing negative item values from transaction databases: with/without specifying the minimum utility threshold through a genetic algorithm with ranked mutation. To the best of our knowledge, this is the first work on mining HUI with negative item values from transaction databases using a genetic algorithm. Experimental results show that approaches described in this article achieve better performance in terms of scalability and efficiency.*

INTRODUCTION

One of the primary research areas of artificial intelligence (AI) is data mining. The proliferation of data in a diversity of areas presents a new set of challenges and opportunities in the way information is searched and retrieved. Because of the great volume of data that must be analyzed, data mining techniques must be used. Thus, data mining researches have been increasing in the last years. Data mining, or knowledge discovery in databases (KDD), refers to the extraction of valid, novel, potentially useful, and ultimately understandable patterns/knowledge in data (Fayyad,

Address correspondence to S. Kannimuthu, Department of CSE, Coimbatore Institute of Engineering and Technology, Coimbatore 641008, Tamil Nadu, India. E-mail: kannimuthu.me@gmail.com

Piatetsky-Shapiro, and Smyth 1996). Knowledge can be learned from the experience or obtained from the data. For example, car mechanics often use reasoning to find the cause for failure and apply an appropriate action based on their knowledge of mechanical science. Business analysts in a bank can assess credit card risks and decide to supply credit cards to customers by analyzing transaction data. Hence, knowledge discovery is used for solving complex problems. Data mining approaches may generate different kinds of knowledge such as association rules, classification rules, clusters, and others.

The problem of extracting association rules has received significant research interest, and numerous algorithms for mining association rules have been developed (Agrawal, Imielinski, and Swami 1993; Agrawal and Srikant 1994, 1995; Agrawal and Shafer 1996). 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 (Agrawal and Shafer 1996). Association rule mining (ARM) algorithms consider only the presence or absence of items in a complete transaction; they do not reflect semantic factors such as cost, profit, and so forth. High utility pattern mining algorithms (Ahmed et al. 2009; Ahmed, Tanbeer, and Jeong 2010a; Chu, Tseng, and Liang 2009; Erwin, Gopalan, and Achuthan 2007a; Li 2011; Liu, Liao, and Choudhary 2005a; Tseng, Chu, and Liang 2006; Tseng et al. 2010; Wu et al. 2012; Yao and Hamilton 2006; Yin, Zheng, and Cao 2012) resolve the problems in ARM by considering nonbinary values in transactions and different profit values of every item.

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

The most important factor in determining the product's success in marketing and retail is the price of a product. In fact, quality and appearance play a vital role in a customer's decision, but buyers are being manipulated by the pricing, and they are often not even aware of it. Many supermarkets may wish to promote certain products to attract customers and increase sales. *Loss leader strategy* is one type of sales promotion method in which a business offers an item or a product at a low price (or for free) that is not profitable, for the sake of offering another product or an item at a greater profit or to attract new customers. In this scenario, supermarkets may give an item as a gift (i.e., for free) whenever the customer buys the specific item.

An item that is given for free is considered to be a negative value. This kind of practice is mainly done by the supermarkets to promote the product, as well as to earn high profits with these free items. For example, if a customer buys three items of I_4, he/she will get one item I_3 for free from the supermarket (refer to Table 1). In this case, the supermarket gets six dollars of profit from each unit of item I_4 sold, and drops two dollars for each unit of item I_3 given to the customer (refer to Table 2). Although the supermarket loses two dollars for giving item $I_3$for free, they can earn eighteen dollars from selling three units of item I_4. Finally, they have a net gain of sixteen dollars from this kind of product promotion.

Existing utility mining algorithms in the literature have failed to extract HUI from the databases that have negative item values. To address this issue, Chu, Tseng, and Liang (2009) developed a scheme named high utility itemsets with negative item values (HUINIV-Mine) for mining HUI with negative item values from large databases. The HUINIV-Mine scheme makes use of a two-phase (TP; Liu, Liao, and Choudhary 2005b) algorithm to mine HUI from the databases. The TP algorithm is based on a candidate generate-and-test approach, which takes a significant amount of time to mine HUI.

The major challenges faced by the data analyst are:

1. Search space for HUI mining is exponential. The major factors decide the search space is the size of the transaction and the number of distinct items in a transaction database.
2. Data analysts need to specify minimum utility threshold to mine the HUI. There are many algorithms and technologies for discovering HUI that have been proposed by researchers. These techniques largely focus on improving scalability and efficiency. Utility-mining algorithms suggested in the literature are based mostly 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.

TABLE 1 Offer Table

Selling item ID /quantity	Offering item ID/quantity
(I_1,2)	(I_2,1)
(I_4,3)	(I_3,1)

TABLE 2 Profit Table

Item	I_1	I_2	I_3	I_4	I_5	I_6
Profit	5	−3	−2	6	10	8

To avoid these problems, genetic algorithm (GA)-based techniques are designed to mine HUI from the transaction database effectively. Charles Darwin's *The Origin of Species* publication in 1859 brought about GAs detailing how complex, problem-solving organisms could be created and improved through an evolutionary process of random trials, sexual reproduction, and selection (Beasley, Bull, and Martin 1993). GAs are used to construct a version of biological evolution on computers. GAs have been successfully adopted in a wide range of optimization problems such as control, design, scheduling, robotics, signal processing, game playing, and combinatorial optimization (Berson and Smith 2004). Data mining is also one of the important application areas of GAs.

The main contributions of this article are summarized as follows:

1. A novel evolutionary approach called *high utility pattern extraction using genetic algorithm with ranked mutation using minimum utility threshold* ($HUPE_{UMU}$-GARM)**,** which makes use of GA, is proposed for mining HUI. In this approach, the data analyst inputs the 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 *high utility pattern extraction using genetic algorithm with ranked mutation without using minimum utility threshold* ($HUPE_{WUMU}$-GARM) using GA is proposed for mining HUI. This approach generates optimal HUIs without specifying a minimum utility threshold.

The rest of this article is organized as follows. **"**Basic Concepts and Definitions" describes the basic concepts and definitions of utility mining and GA. The next section presents the "Related Works" and is followed by "Extracting HUI with GA." The proposed approaches are discussed in "$HUPE_{UMU}$-GARM" and "$HUPE_{WUMU}$-GARM." Experimental results are made in "Experimental Evaluation." The final section is our "Conclusion."

BASIC CONCEPTS AND DEFINITIONS

This section explains the basic concepts and definitions of utility mining and GA.

Utility Mining

HUI mining is a research area of utility-based descriptive data mining, aimed at finding itemsets that make the best contribution to the total utility. This section starts with the definition of a set of terms, which leads to the

formal definition of a utility mining problem that is given in Yao, Hamilton, and Butz (2004).

Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items; $D = \{T_1, T_2, \ldots, T_n\}$ be a transaction database in which each transaction $T_j \in D$ is a subset of I; $o(I_p, T_q)$ is the local transaction utility value, which represents the quantity of item I_p in transaction T_q. For example, $o(I_1, T_9) = 2$, in Table 3. In the Utility table (Table 2), $s(I_p)$, external utility, is the value associated with item I_p. This value reflects the importance of an item, which is independent of transactions. For example, in Table 2, the external utility of item I_1,$s(I_1)$ is 5. Utility$u(I_p, T_q)$, the quantitative measure of utility I_p in transactionT_q, is defined as $o(I_p, T_q) \times s(I_p)$. For example,$u(I_1, T_9) = 2 \times 5$ in Table 3. The utility of an itemset X in transactionT_q, $u(X, T_q)$, is defined as $\sum_{I_p \in X} u(I_p, T_q)$, where $X = \{I_1, I_2, \ldots, I_k\}$ *symbols*is a k-itemset, $X \subseteq T_q$ and $1 \leq k \leq m$. The utility of an itemset X, $u(X)$, is defined as

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

The main task is to find all the HUI using utility mining. An itemset X is a *high utility itemset* if$u(X) \geq minUtil$, where $X \subseteq I$. For example, in Table 3, $u(I_1, T_9) = 2 \times 5 = 10$, $u(\{I_1, I_4\}, T_9) = u(I_1, T_9) + u(I_4, T_9) = 2 \times 5 + 3 \times 6 = 28$, and $u(\{I_1, I_4\}) = u(\{I_1, I_4\}, T_2) + u(\{I_1, I_4\}, T_9) = 46 + 28 = 76$. If $minUtil = 150$, then $\{I_1, I_4\}$ is not an HUI. The utility mining approach does not support the downward-closure property (Agrawal and Srikant 1994). Hence, combinations of all items are generated and the same should be processed to ensure that no HUI will be lost.

Liu, Liao, and Choudhary (2005b) presented the TP algorithm for mining HUI. The TP algorithm has two phases. In the first phase, the transaction utility (TU) for all the transactions is calculated. Next, the set of all single itemsets is identified and transaction-weighted utilization (TWU) for

TABLE 3 Quantity Table

TID/ITEM	I_1	I_2	I_3	I_4	I_5	I_6
T_1	2	1	0	0	2	2
T_2	2	1	2	6	0	1
T_3	0	0	2	6	0	1
T_4	2	1	0	0	0	0
T_5	0	0	2	6	0	1
T_6	2	1	0	0	2	0
T_7	2	1	0	0	0	1
T_8	0	0	1	3	0	2
T_9	2	1	1	3	0	1
T_{10}	0	0	1	3	1	1

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 (Liu, Liao, and Choudhary 2005b) First phase may overestimate some low utility itemsets, but it never underestimates any itemsets. In the second phase, one extra database scan is performed in order to filter the overestimated itemsets.

Consider Table 3: it has 10 transactions, $tu(T_1)$ is the transaction utility of T_1 and will be $tu(T_1) = u(I_1, T_1) + u(I_2, T_1) + u(I_5, T_1) + u(I_6, T_1) = 2 \times 5 + 1 \times (-3) + 2 \times 10 + 2 \times 8 = 43$; ($tu(T_1)$ is 46 if items containing negative values are not considered). The transaction utilities for all the transactions are listed in Table 4. Transaction-weighted utilization of an itemset I_1, denoted as TWU(I_1), is the sum of the transaction utilities of all transactions containing I_1. By observing Table 4 and Table 5, TWU is calculated: $\text{TWU}(I_1) = TU(T_1) + TU(T_2) + TU(T_4) + TU(T_5) + TU(T_6) + TU(T_7) + TU(T_9) = 46 + 54 + 10 + 30 + 18 + 36 = 194$.

TABLE 4 Transaction Utility with/without Negative Item Values of the Transaction Database

TID	TU with negative item values	TU without negative item values
T_1	43	46
T_2	47	54
T_3	40	44
T_4	7	10
T_5	40	44
T_6	27	30
T_7	15	18
T_8	32	34
T_9	31	36
T_{10}	34	36

TABLE 5 Transaction Database

TID/ITEM	I_1	I_2	I_3	I_4	I_5	I_6
T_1	1	1	0	0	1	1
T_2	1	1	1	1	0	1
T_3	0	0	1	1	0	1
T_4	1	1	0	0	0	0
T_5	0	0	1	1	0	1
T_6	1	1	0	0	1	0
T_7	1	1	0	0	0	1
T_8	0	0	1	1	0	1
T_9	1	1	1	1	0	1
T_{10}	0	0	1	1	1	1

Genetic Algorithm

GA 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 to find the best solution based on fitness functions under a set of multiple constraints. The GA 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 GA are mentioned in Table 6.

RELATED WORKS

In recent manuscripts dedicated to the subject of utility mining, we were able to identify different factors that influence the performance of the algorithms, for example, utility measures used, data structures adopted, and pruning strategies applied. We summarized these papers in Table 7.

Yao, Hamilton, and Butz' (2004) theoretical analysis of the utility mining problem presented the foundation for future utility mining algorithms. They proposed Apriori-like algorithms, called UMining and UMining_H (Yao and Hamilton 2006) to extract HUI level by level. Mining HUI often leads to the generation of a large number of patterns. Unlike frequent itemset mining (FIM), the utility mining model does not satisfy the anti-monotone property. Several pruning strategies (Li, Yeh, and Chang 2008;

TABLE 6 Terminologies Used in Genetic Algorithm

Terminology	Description
Locus	A position in the genome is called locus.
Allele	It is the value of the gene (or genes).
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 indicates 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.
Generation	A generation is an iteration of the genetic algorithm. Usually, the initial random generation is known as generation zero.
Selection	The process of picking effective chromosomes from the population for a later breeding is called 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 at an optimal solution is called mutation.
Search space	It is the space of all feasible solutions.

TABLE 7 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 of a transaction	*minUtil* threshold	Execution time in seconds
TP algorithm (Liu, Liao, and Choudhary 2005b)	Liu et al.	List	2005	TWDC Property	Real-world dataset	1112949	46086	6	1	25.76
Parallel implementation of TP algorithm (Liu, Liao, and Choudhary 2005a)	Liu et al.	List	2005	TWDC Property	T10I6D1000K	100000	1000	10	2	20
					T20I6D1000K	100000	1000	20	2	50
UMining (Yao and Hamilton 2006)	Yao et al.	List, Hash Table	2006	Upper Bound Property	Synthetic dataset	11160188	100	5	1	4200
UMining_H (Yao and Hamilton 2006)	Yao et al.	List, Hash Table	2006		Synthetic dataset	11160188	100	5	1	3720
THUI-Mine (Tseng, Chu, and Liang 2006)	Liu et al.	List	2006	TWDC Property	T10I6D100K	100000	1000	10	1	80
CTU-Mine (Erwin, Gopalan, and Achuthan 2007a)	Liu et al.	CTU-Tree	2007	TWDC Property	T10I5D50K	50000	1000	10	1	700
CTU-PRO (Erwin, Gopalan, and Achuthan 2007b)	Liu et al.	CTU-Tree	2007	TWDC Property	T10N5D100K	100000	1000	10	1	20
HUQA (Yen and Lee 2007)	Yen and Lee	List	2007	Support Bound Property	T6I4D1000K	1000000	1000	6	1	2000
					T6I4D100K	100000	5000	6	0.5	230
HUINIV (Chu, Tseng, and Liang 2009)	Liu et al.	List	2009	TWDC Property	T10I6D1000K	100000	1000	10	1	900
					BMS-POS	515597	1657	6.53	1	3200
IHUPTWU (Ahmed et al. 2009)	Liu et al.	$IHUP_{TWU}$-Tree	2009	TWDC Property	Real-time retail dataset	88162	16470	10.3	1	100
HUPMS (Ahmed, Tanbeer, and Jeong 2010a)	Liu et al.	HUS-Tree	2010	TWDC Property	T10I4D100K	100000	870	10	2	350
					BMS-POS	515597	1657	6.53	4	550
IHUP-FPG (Tseng et al. 2010)	Liu et al.	IHUP-Tree	2010	TWDC Property	T10I6D100K	100000	1000	10	0.2	555
UP-FPG (Tseng et al. 2010)	Liu et al.	UP-Tree	2010	TWDC Property, DGU, DGN, DLU, and DLN strategies	T10I6D100K	100000	1000	10	0.2	407

(*Continued*)

TABLE 7 (*Continued*)

Algorithm	Measures used	Data structures used	Year	Pruning strategies used	Dataset	No. of transactions	Number of items	Average length of a transaction	*minUtil* threshold	Execution time in seconds
UP-UPG (Tseng et al. 2010)	Liu et al.	UP-Tree	2010	TWDC Property, DGU, DGN, DLU, and DLN strategies	T10I6D100K	100000	1000	10	0.2	283
IHUP-FPG (Tseng et al. 2010)	Liu et al.	IHUP-Tree	2010	TWDC Property	BMS-Web-View-1	59602	497	2.51	2.9	401000
UP-FPG (Tseng et al. 2010)	Liu et al.	UP-Tree	2010	TWDC Property, DGU, DGN, DLU, and DLN	BMS-Web-View-1	59602	497	2.51	2.9	2
UP-UPG (Tseng et al. 2010)	Liu et al.	UP-Tree	2010	TWDC Property, DGU, DGN, DLU, and DLN strategies	BMS-Web-View-1	59602	497	2.51	2.9	2
IHUP-FPG (Tseng et al. 2010)	Liu et al.	IHUP-Tree	2010	TWDC Property	Chess	3196	76	37	55	10055
UP-FPG (Tseng et al. 2010)	Liu et al.	UP-Tree	2010	TWDC Property, DGU, DGN, DLU, and DLN strategies	Chess	3196	76	37	55	151
MHUI-TID (Li 2011)	Liu et al.	LexTree-maxHTU	2011	TWDC Property	T10I4D100K	100000	870	10	2	700
					BMS-POS	515597	1657	6.53	4	1150
TKU (Wu et al. 2012)	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
					Chain store dataset	1, 112, 949	46, 086	7.2	0.01	1400
USpan (Yin, Zheng, and Cao 2012)	Liu et al.	LQS-Tree	2012	TWDC Property,	C10T2.5S4I2.5DB10kN1k	10000	1000	10	0.001	100
				Width pruning, and Depth pruning	C8T2.5S6I2.5DB10kN10k	10000	10000	8	0.002	120
HUI-Miner (Liu and Qu 2012)	Liu et al.	Utility List	2012	Pruning using iutils and rutils	T10I4D100K	100000	1000	10	0.03	100

Liu, Liao, and Choudhary 2005b; Tseng et al. 2010; Wu et al. 2012; Yao and Hamilton 2006; Yen and Lee 2007; Yin, Zheng, and Cao 2012) have been proposed to improve the performance of the utility mining algorithms.

A vital research issue extended from the utility mining is the discovery of high utility patterns in data streams (Tseng, Chu, and Liang 2006; Ahmed, Tanbeer, and Jeong 2010a; Li 2011) because of the wide applications on various domains. Discovering high utility sequential patterns can be considered as a special type of mining, which was first introduced by Yin, Zheng, and Cao (2012). They presented an algorithm named USpan to efficiently mine high utility sequential patterns using lexicographic quantitative sequence trees.

Traditional utility mining algorithms have failed to find HUI with negative values from large databases. This issue has been investigated by Chu, Tseng, and Liang (2009) who propose the HUINIV-Mine algorithm. HUINIV-Mine uses the TP algorithm to mine HUI. Mining web access sequences (WASs) can extract very useful knowledge from web logs with wide-ranging applications. Ahmed, Tanbeer, and Jeong (2010b) presented a pioneering work with regard to mining high utility WASs: two tree structures, called utility-based WAS tree (UWAS-tree) and incremental UWAS-tree (IUWAS-tree) were proposed for mining WASs in static and incremental databases.

In recent years, researchers focused on utility-based web traversal pattern mining approaches (Zhou et al. 2007; Ahmed, Tanbeer, and Jeong 2011) and defined high utility web traversal patterns. One of the issues with the existing approaches is that all patterns satisfying the user-specified *minUtil* threshold would become high utility patterns, regardless of their pattern length. Thus, if the length of a pattern increases automatically, the utility of the pattern will also increase. Moreover, longer patterns with less page utility in a transaction may result in higher values, and they will be treated equally with shorter patterns with more page utility. Thilagu and Nadarajan (2012) present an efficient mining of web traversal patterns by considering high average utility patterns rather than patterns with only actual utility. Average utility of a pattern can be defined as the actual utility of the pattern divided by its length.

Unfortunately, recent studies show that utility-mining algorithms may suffer from some weaknesses such as (1) large search space and (2) problems with database-dependent minimum utility threshold. Many HUI mining algorithms (Ahmed et al. 2009; Ahmed, Tanbeer, and Jeong 2010a; Erwin, Gopalan, and Achuthan 2007b; Li 2011; Tseng, Chu, and Liang 2006; Tseng et al. 2010; Wu et al. 2012; Yin et al. 2012) have been proposed to reduce the search space. These algorithms use a tree-based approach. It is widely acknowledged that a tree-based approach achieves a better performance than Apriori-like approaches because it finds HUIs without generating candidate itemsets. Liu, Liao, and Choudhary (2005b) propose the TP algorithm, which efficiently extorts HUI from the databases. The

TP algorithm utilizes TWDC property 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_{UMU}-GARM to efficiently mine HUI using GA.

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

EXTRACTING HIGH UTILITY ITEMSETS WITH GENETIC ALGORITHM

Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items, D= $\{T_1, T_2, \ldots, T_n\}$ be a transaction database where each transaction $T_j \in D$ is a subset of I. An itemset X is an HUI if it satisfies the *minUtil* threshold, or, $u(X) \geq minUtil$; *minUtil* is a threshold that is defined by the user. This section portrays the building blocks of the GA used in this work.

Encoding

In this subsection, encoding is presented first. Three genetic operators and a population initialization and a fitness function based on the Yao, Hamilton, and Butz (2004) measure. Finally, these modules are put together into the algorithms HUPE_{UMU}-GARM and $\text{HUPE}_{\text{WUMU}}$-GARM. Figure 1 shows the configuration of the genes (items) in the chromosome. Binary encoding is used in this approach. Binary value "1" represents presence of an item and "0" represents absence of an item in an itemset. Chromosome length is fixed and it is equal to the number of distinct items (n), which is obtained from the transaction database.

Population Initialization

Given an itemset length k, all the genes (items) in a chromosome are encoded as 0. The initial population is produced using a random number

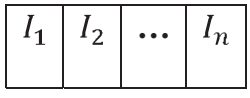

FIGURE 1 Chromosome.

generator. If the generated random number is r, then the chromosome is encoded as 1 at rth position. This represents i_r item presence in a chromosome (itemset). On 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 condition $k \leq n$. The algorithm for population initialization is given in Figure 3.

Example. Consider the chromosome length to be 7 and the itemset length to be 4. Random numbers generated by a random number generator are {1, 3, 5, 6}.

The representation of a chromosome is shown in Figure 2.

Fitness Function

The main goal of this work is to generate the HUI from the transaction database. Hence, the fitness function is essential for determining the chromosome (itemset) that satisfies the *minUtil* threshold. In HUPE_{UMU}-GARM and $\text{HUPE}_{\text{WUMU}}$-GARM algorithms, we use Yao Hamilton, and Butz's (2004) utility measure *u(X)* as the fitness function

$$f(X) = u(X) = \Sigma_{T_q \in D \wedge X \subseteq T_q} u\left(X, T_q\right). \tag{2}$$

Genetic Operators

This subsection expresses three genetic operators: selection, crossover, and mutation.

Selection

Selection is the one footstep of the GA in which an individual chromosome is chosen from a population for later breeding (crossover). If all the genes in the chromosome have negative utility value, then it will not be considered for later breeding. The selection operator acts as a filter of chromosomes with considerations of their fitness value. There are many selection approaches available in the literature. In this work, roulette wheel selection (Holland 1975) is used.

1	0	1	0	1	1	0

FIGURE 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 ≤ 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
```

FIGURE 3 Algorithm for initializing the population.

Crossover

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

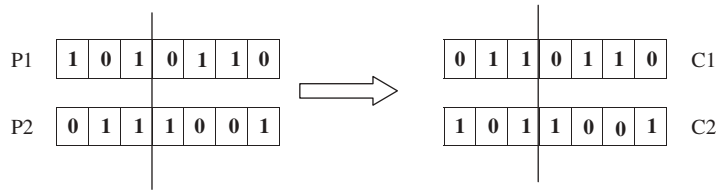

FIGURE 4 Single-point crossover.

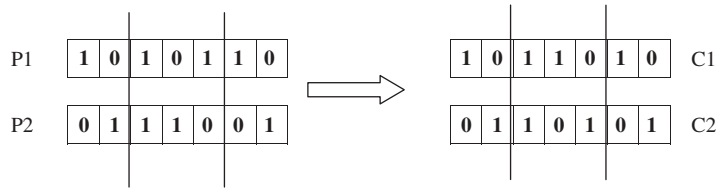

FIGURE 5 Two-point crossover.

In its simplest form, the operator works by exchanging substrings after a randomly selected crossover point. The illustration of the crossover operation is given in Figures 4 and 5.

Mutation

The mutation operator is used to maintain diversity from one generation of a population to the next generation**.** Mutation changes one or more gene values in a chromosome from its previous generation. The simple GA follows user-defined mutation probability (P_m) for the mutation operation. Normally, the mutation probability should be set to low. If it is too high, the search will become a primitive random search. The adaptive mutation rate gives better performance than a fixed mutation rate (Premalatha and Natarajan 2009).

$$P_m = \left(P_m^{\max} - \frac{P_m^{\max} - P_m^{\min}}{N_i} \times T \right) \times \frac{\text{Rank}}{R} \tag{3}$$

$P_m^{\max}$: Maximum mutation rate
$P_m^{\min}$: Minimum mutation rate
N_i: Number of iteration
T: Time epoch or iteration number
R: Total number of ranks

In the proposed work, the mutation probability is reduced when the generation number increases, and the mutation rate of the offspring is adapted

with respect to its fitness value. Initially, the large mutation rate is employed to explore more on the search space. The offspring are ranked based on their fitness values. The mutation rate of the offspring is assigned with respect to its rank. The higher-ranked offspring mutated at a lower rate when compare to other offspring that possess lower ranks. The highest-fitness offspring may reach the optimal solution in close proximity. So the minimum rate avoids the distraction in the search space, and P_m can be computed by using Equation (3).

Evaluation

The evaluation step intends to select the chromosomes for the next generation. In this work, the elitist selection (Holland 1975) method is used. This method copies the chromosome(s) with higher fitness value to the new population.

Termination Criteria

The termination criterion is the criterion by which the GA decides whether to continue searching or to 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

HUPE$_{UMU}$-GARM

Mining HUI with minimum utility threshold using GA is proposed, which effectively extracts useful patterns from the databases. The GA is chosen because it is a promising solution for global search, and it is capable of discovering HUI with corresponding parameters quantity and profit. We use two measures such as TWU (Liu, Liao, and Choudhary 2005b) for removing unpromising items from the transaction database at an earlier stage and the Yao, Hamilton, and Butz (2004) measure for fitness value computation.

The steps of the HUPE$_{UMU}$-GARM algorithm are the following (see Figure 6):

1. Scan the transaction database to compute TU of each transaction without considering negative item values. At the same time, the TWU of each single item is also accumulated.

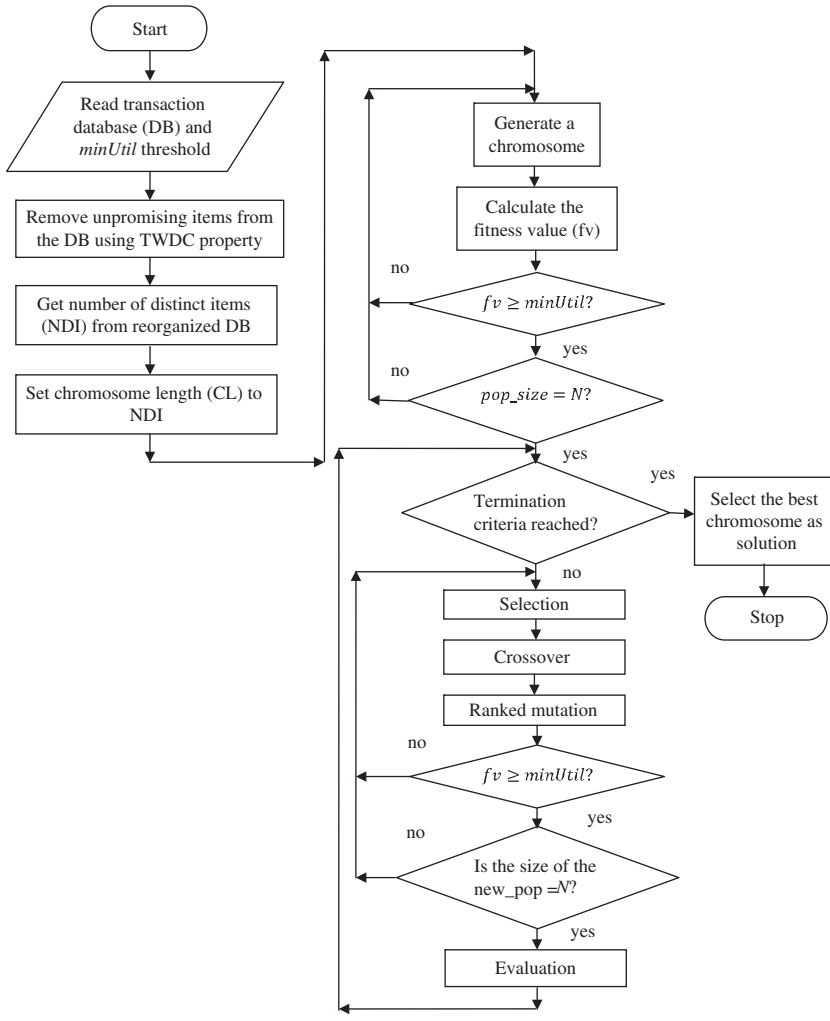

FIGURE 6 Flow chart of $HUPE_{UMU}$-GARM approach.

2. If the TWU of an item is less than the minimum utility threshold, its supersets are unpromising to be HUI (TWDC property). So, remove the corresponding unpromising items from the transaction database. The transaction database after the above reorganization is called the 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 (in a generation process, if all the genes in the chromosome have negative utility value, then it will not be considered).
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 the population.
10. Perform mutation on the individuals of the 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 the size of the new population reaches N. If yes, go to Step 12, else go to Step 8.
12. Evaluate the individuals by using elitist selection of N chromosomes from the new and old populations for the next generation.

HUPE$_{WUMU}$-GARM

The proposed HUPE$_{WUMU}$-GARM approach is used to mine optimal HUIs without specifying a *minUtil* threshold. In this approach, the Yao, Hamilton, and Butz (2004) measure is used for fitness value computation.

The steps of HUPE$_{UMU}$-GARM algorithm are the following (see Figure 7):

1. Examine the transaction database to find the number of distinct items (NDI). Set chromosome length (CL) to NDI.
2. Generate a chromosome of length CL (in a generation process, if all the genes in the chromosome have negative utility value, then it will not be considered).
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, which will compose the next generation with the best parents.
7. Perform crossover on the selected individuals of the population.
8. Perform mutation on the individuals of the population with mutation probability $P_{m.}$ Calculate the fitness value (fv) for the individuals.
9. Check if the size of the new population reaches N. If yes, go Step 10, else go to Step 6.
10. Evaluate the individuals by using the elitist selection of N chromosomes from new and old populations for the next generation.

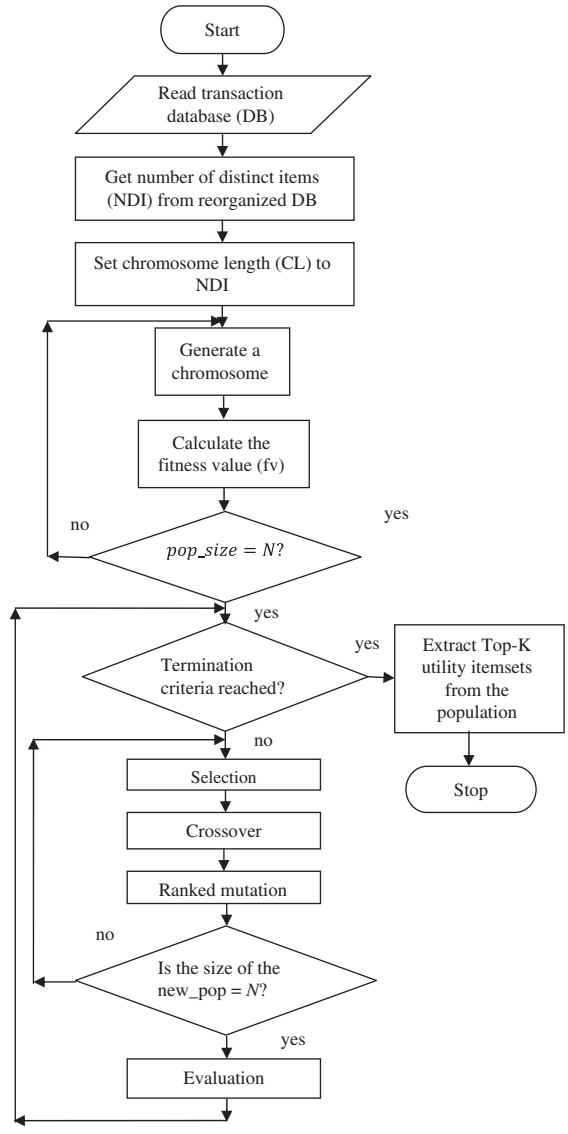

FIGURE 7 Flow chart of $HUPE_{WUMU}$-GARM approach.

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 an IBM data generator (IBM Quest Market-Basket Synthetic Data Generator). Dataset "T10.I4.D10K" means an average transaction size of 10, an average size of

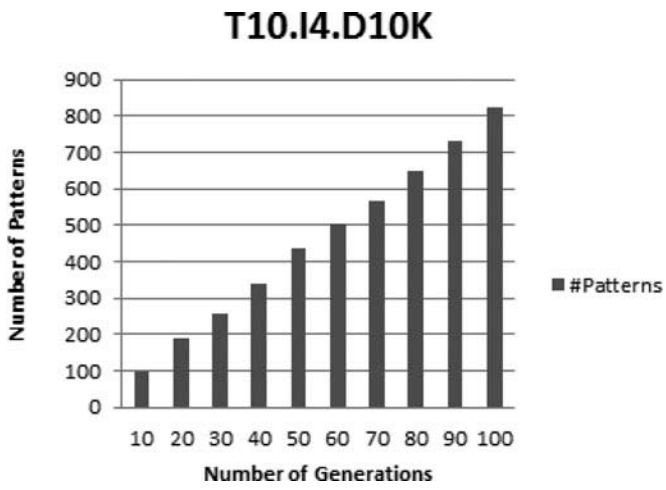

FIGURE 8 Plot of number of generations versus number of patterns generated when $minUtil = 2\%$.

the maximal potentially frequent itemsets of 4, and 10,000 generated transactions. This dataset contains 100 distinct items. The 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.

HUPE$_{UMU}$-GARM

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

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

HUPE$_{WUMU}$-GARM

The next experiment was carried out with the HUPE$_{WUMU}$-GARM approach in different numbers of generations by keeping 100 distinct

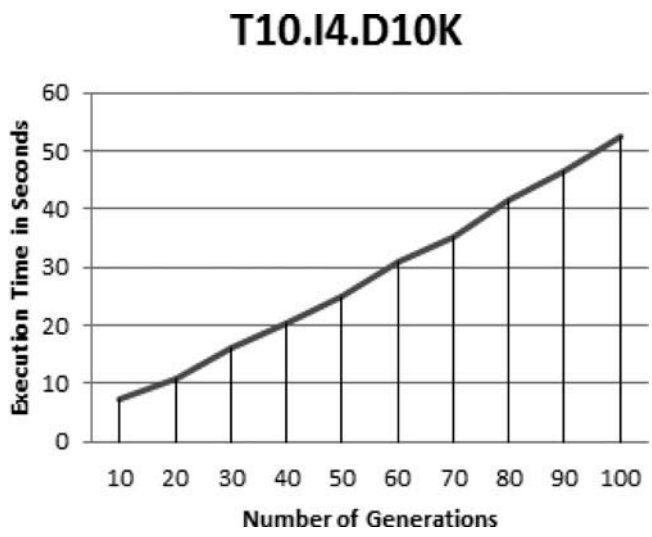

FIGURE 9 Plot of number of generations versus execution time generated when $minUtil = 2\%$.

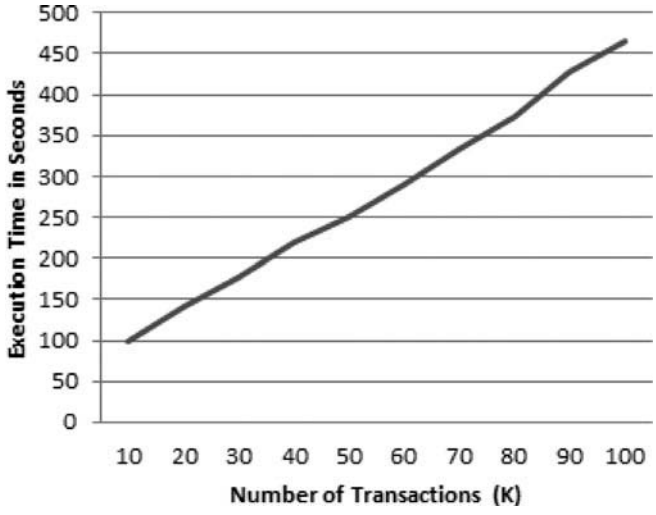

FIGURE 10 Scalability with number of transactions.

items as fixed. The test results are illustrated through the Figure 11 and Figure 12, respectively. As with the $HUPE_{UMU}$-GARM, execution time of the $HUPE_{WUMU}$-GARM approach is proved to be considerably less. Also with the $HUPE_{WUMU}$-GARM approach was an experiment in increasing the size of the transaction from 10K to 100K. This approach shows linear scalability and scales much better, which is illustrated in Figure 13.

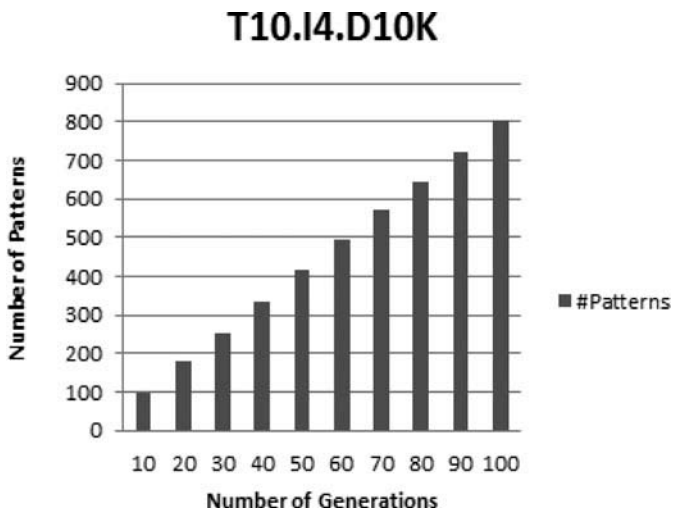

FIGURE 11 Plot of number of generations versus number of patterns generated.

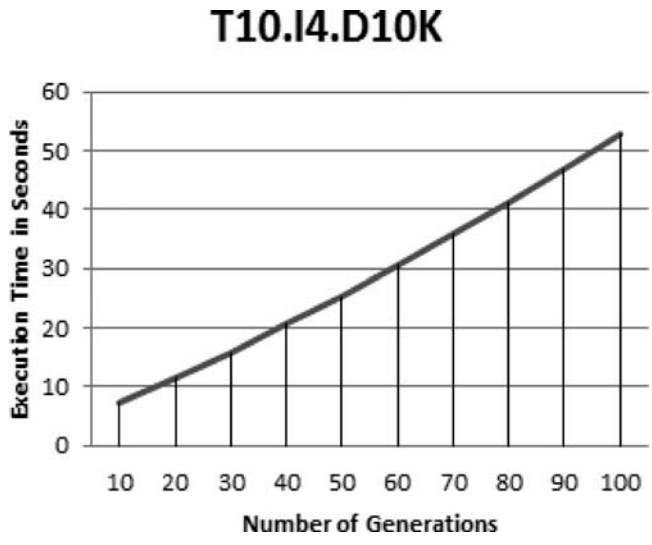

FIGURE 12 Plot of number of generations versus execution time generated.

CONCLUSION

HUI mining is a very important research issue in data mining and knowledge discovery. The approaches mentioned in this article exploit a GA; they can handle large numbers of distinct items and transactions. These approaches are the best choice when the execution time and memory requirement are significant issues. GAs have proved to be robust, general-purpose search techniques. They have a central place in data mining

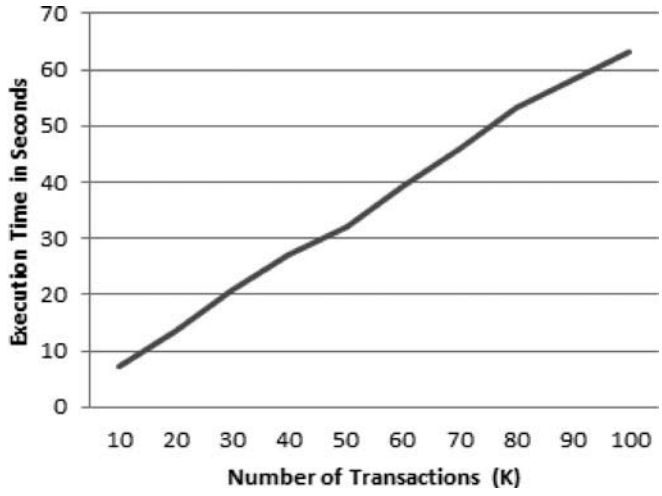

FIGURE 13 Scalability with number of transactions.

applications because of their ability to explore large search spaces efficiently. Two major challenges in utility mining, that is, exponential search space and database-dependent minimum utility threshold are studied, and an attempt is made to resolve the problems by proposing a GA-based approach to mine HUIs from the databases. These approaches work well on itemsets containing negative item values. Experimental results show that our proposed approaches scale well and retrieve the HUIs from the databases containing negative item values efficiently.

REFERENCES

Ahmed, C. F., S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee. 2009. Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering* 21(12):1708–1721.

Ahmed, C. F., S. K. Tanbeer, and B.-S. Jeong. 2010a. Efficient mining of high utility patterns over data streams with a sliding window method. In *Software engineering, artificial intelligence, networking and parallel/distributed computing*, ed. R. Lee, Studies in Computational Intelligence 295:99–113. Berlin, Heidelberg: Springer.

Ahmed, C. F., S. K. Tanbeer, and B.-S. Jeong. 2010b. Mining high utility web access sequences in dynamic web log data. In *11th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing*, 76–81. IEEE Conference Publications.

Ahmed, C. F., S. K. Tanbeer, and B. S. Jeong. 2011. A framework for mining high utility web access sequences. *IETE Technical Review* 28(1):3–16.

Agrawal, R., T. Imielinski, and A. N. Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD conference, 207–216.* ACM Press.

Agrawal, R., and R. Srikant. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th very large data bases (VLDB) conference*, 487–499. San Francisco, CA, USA: Morgan Kaufman.

Agrawal, R., and J. C. Shafer. 1996. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering* 8(6):962–969.

Agrawal, R., and R. Srikant. 1995. Mining sequential patterns. In *Proceedings of the 11th international conference on data engineering*, 3–14. Washington, DC, USA: IEEE Computer Society.

Beasley, D., D. R. Bull, and R. R. Martin. 1993. An overview of genetic algorithms. *University Computing* 15(2):58–69.

Berson, A., and S. J. Smith. 2004. *Data warehousing, data mining and OLAP*. India: Tata McGraw-Hill.

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

Erwin, A., R. P. Gopalan, and N. R. Achuthan. 2007a. CTU-mine: An efficient high utility itemset mining algorithm using the pattern growth approach. In *Proceedings of the 7th international conference on computer and information technology*, 71–76. Aizu-Wakamatsu City, Fukushima, Japan, October 16–19.

Erwin, A., R. P. Gopalan, and N. R. Achuthan. 2007b. A bottom-up projection based algorithm for mining high utility itemsets. In *Proceedings of the 2nd workshop on integrating ai and data mining (AIDM 2007)*, 84:3–11. Darlinghurst, Australia: Australian Computer Society.

Fayyad, U. M., G. Piatetsky-Shapiro, and P. Smyth. 1996. From data mining to knowledge discovery: An overview. *Advances in knowledge discovery and data mining*, 1–34, Menlo Park, CA, USA: AAAI/MIT Press.

Holland, J. 1975. *Adaptation in natural and artificial systems*. Ann Arbor, MI, USA: University of Michigan Press.

IBM Quest Market-Basket Synthetic Data Generator, http://www.cs.loyola.edu/~cgiannel/assoc_gen.html

Li, H.-F. 2011. MHUI-max: An efficient algorithm for discovering high-utility itemsets from data streams. *Journal of Information Science* 37(5):532–545.

Li, Y.-C., J.-S. Yeh, and C.-C. Chang. 2008. Isolated items discarding strategy for discovering high utility itemsets. *Data and Knowledge Engineering* 64:198–217.

Liu, Y. W.-K. Liao, and A. Choudhary. 2005a. A fast high utility itemsets mining algorithm. In *Workshop on utility-based data mining (UBDM 2005)*, 90–99. Chicago, IL, USA: ACM.

Liu, Y. W.-K. Liao, and A. Choudhary. 2005b. A two-phase algorithm for fast discovery of high utility itemsets. In *Proceedings of the 9th Pacific-Asia conference on advances in knowledge discovery and data mining (PAKDD 2005)*, ed. T. B. Ho, D. Cheung, and H. Liu. Lecture Notes in Artificial Intelligence 3518: 689–695. Berlin, Heidelberg: Springer-Verlag.

Premalatha, K., and A. M. Natarajan. 2009. Genetic algorithm for document clustering with simultaneous and ranked mutation. *Journal of Modern Applied Science* 3(2):75–82.

Thilagu, M., and R. Nadarajan. 2012. Efficiently mining of effective web traversal patterns with average utility. *Procedia Technology Journal: 2nd International Conference on Communication, Computing, and Security* 6:444–451.

Tseng, V. S., C.-J. Chu, and T. Liang. 2006. Efficient mining of temporal high utility itemsets from data streams. In *Proceedings of the 2nd international workshop on utility-based data mining*, 18–27. Philadelphia, PA, USA.

Tseng, V. S., C.-W. Wu, B.-E. Shie, and P. S. Yu. 2010. UP-growth: An efficient algorithm for high utility itemset mining. In *Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining*, 253–262. New York: ACM Press.

Wu, C. W., B.-E. Shie, P. S. Yu, and V. S. Tseng. 2012. Mining top-k high utility itemsets. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining (ACM, KDD'12)*, 78–86. Beijing, China.

Yao, H., and H. J. Hamilton. 2006. Mining itemset utilities from transaction databases. *Data and Knowledge Engineering* 59:603–626.

Yao, H., H. J. Hamilton, and C. J. Butz. 2004. A foundational approach to mining itemset utilities from databases. In *Proceedings of the 3rd SIAM international conference on data mining*, 482–486. Orlando, Florida, USA.

Yen, S.-J., and Y.-S. Lee. 2007. Mining high utility quantitative association rules. *Data warehousing and knowledge discovery*, Lecture Notes on Computer Science 4654: 283–292. Berlin, Heidelberg: Springer.

Yin, J., Z. Zheng, and L. Cao. 2012. USpan: An efficient algorithm for mining high utility sequential patterns. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining (ACM, KDD'12)*, 660–668. Beijing, China

Zhou, L., Y. Liu, J. Wang, and Y. Shi. 2007. Utility-based web path traversal pattern mining. Paper presented at the 7th IEEE international conference on data mining workshops, Omaha, Nebr., October 28–31.