Mining High Utility Itemset with Hybrid Ant Colony Optimization Algorithm

Keerthi Mohan[1], Anitha J[2]
Research Scholar[1], Professor[2],
Department of Computer Science and Engineering, RV Institute of Technology and Management, Bangalore, India

***Abstract*—A significant area of study within data mining is high-utility itemset mining (HUIM). The exponential problem of broad search space usually comes up while using traditional HUIM algorithms when the database size or the number of unique objects is huge. Evolutionary computation (EC) -based algorithms have been presented as an alternate and efficient method to address HUIM problems since they can quickly produce a set of approximately optimum solutions. In transactional databases, finding entire high-utility itemset (HUIs) still need a lot of time using EC-based methods. In order to deal with this issue, we propose a hybrid Ant colony optimization-based HUIM algorithm. Genetic operators' crossover is applied to the generated solution by the ant in the Ant Colony optimization algorithm. In this study, a single-point crossover is employed wherein, the crossover point is selected randomly and a mutation operator is applied by changing one or many random bits in a string. This technique requires less time to mine the same number of HUIs than state-of-the-art EC-based HUIM algorithms.**

Keywords*—Utility mining; high utility itemset; ant colony optimization; genetic algorithm; evolutionary computation

I. Introduction

The purpose of data mining is to uncover valuable insights or cognition from data which can aid in decision-making, prediction, and revealing hidden patterns or relationships within the data. Frequent itemset mining (FIM) is a crucial data mining approach aimed at identifying groups of items or elements that commonly occur together in large datasets. It is often applied in association rule learning, where the objective is to uncover relationships or patterns among items in transactional data, such as in market basket analysis. Although FIM mines itemsets based on the count of occurrence, the item's utility (profit) is not taken into consideration. To address this shortcoming, the concept of High Utility Itemset (HUI) was introduced. While traditional frequent itemset mining identifies itemsets based on their frequency of occurrence, HUIM takes into account the actual value or utility of items, which can include factors like profit, revenue, or user preference. An itemset is deemed high utility if its utility exceeds the user specified threshold. Here, a minimum utility value will be set, either manually or through a probabilistic function. Traditional HUIM algorithms encounter and have to deal with the exponential search space problem with increasing number of transactions and database items. Evolutionary algorithms such as genetic algorithms (GA), Ant Colony Optimization (ACO) algorithm, Particle swarm Optimization (PSO), and Artificial bee colony (ABC) algorithm are some efficient algorithms that solve the exponential search space problem. In order to find near-optimal solutions based on fitness functions, evolutionary algorithms effectively search enormous problem spaces. For mining HUIs beyond a minimum utility threshold, evolutionary-based HUIM algorithms are quicker than conventional precise approaches, although they still require a lot of time. Enhancing the diversity of the generated population is one way of reducing the time. Hybrid approach balances exploration and exploitation, handles sizably voluminous and unevenly distributed datasets preponderant, and acclimates well to dynamic environments. It outperforms standalone PSO, GA, or ACO in terms of haste, scalability, and solution quality. The Hybrid Ant Colony Optimization (GA-ACO) algorithm ameliorates HUI mining by cumulating the iterative refinement of ACO with the expeditious ecumenical search of GA, which expedites convergence and truncates the early-stage impuissance's of ACO.

II. Related Work

Agrawal and Srikant [1] explored association rules in large sales transaction databases. Agrawal, Imielinski, and Swami [2] used novel estimation and pruning techniques for efficient and fast mining of association rules. Tseng et al. [3] developed algorithms for mining high utility itemsets from a transactional database. They presented two algorithms, Utility-Pattern-Growth (UP-Growth) and also UP-Growth+, to carry out the mining of high utility itemset with a collection of effective approaches for pruning potential itemsets. They stored information associated with high utility itemsets into a tree-based data structure called "utility pattern tree" in such a way that candidate itemsets could be developed with only two scans of the database and obtained significant improvement in extracting high utility itemsets. Chan, Yang, and Shen [4] proposed a technique that targets mining the top-K high utility closed patterns directly aligned with a specific business objective. Their experimental results demonstrate that the algorithm does not require a user-defined minimum utility, making it practical and effective in real-world applications.

Yao, Hamilton, and Butz [5] examined the utility relationships among itemsets and identified two key properties: the utility bound property and the support bound property. They also developed a mathematical model for utility mining based on these principles. Ahmed et al. [6] introduced tree structures to efficiently conduct incremental and interactive high-utility pattern (HUP) mining. Their first tree structure, the Incremental HUP Lexicographic Tree (IHUPL-Tree), is organized by an item's lexicographic order, capturing incremental data without requiring any restructuring. The second tree structure, the IHUP Transaction FrequencyTree (IHUPTF-Tree), optimizes size by arranging items in descending order of transaction frequency. To

reduce mining time, the third structure, the IHUP Transaction-Weighted Utilization Tree (IHUPTWU-Tree), is based on the descending TWU (Transaction-Weighted Utilization) values of items. The authors demonstrated that these tree structures are highly efficient and scalable for incremental and interactive HUP mining.

In study [7], Liu and Qu introduced a data structure called the "utility-list" and introduced an algorithm named HUIMiner for mining high utility itemsets. The HUI-Miner algorithm introduces utility-lists to store utility and pruning information, enabling it to mine high utility itemsets without engendering candidate itemsets. This eliminates the computational overhead of traditional algorithms that process sizably voluminous numbers of low-utility candidates. By directly fixating on high utility itemsets, HUI-Miner evades the costly processes of itemset generation and utility computation. Performance comparisons with other state-of-the-art algorithms across multiple databases show that HUI-Miner offers better improvment in both runtime and memory utilization, making it a more efficient approach for mining high utility itemsets.

Fournier-Viger et al. [8] presented a new algorithm called FHM- Fast High-Utility Miner which improved the performance of HUI-Miner algorithm. FHM's key innovation lies in its strategic analysis of item cooccurrences, which substantially decreases the number of necessary join operations during the mining process. Testing on real-world datasets demonstrated FHM's efficiency, reducing join operations by as much as 95% and operating up to six times faster than its predecessor, HUI-Miner.

Zida et al. [9] introduced EFIM, a new algorithm for mining high-utility itemsets. EFIM's effectiveness arises from two main components: newly developed upper-bounds called the sub-tree utility and the local utility, and a streamlined array-based method named Fast Utility Counting. Their approach allows for linear time and space complexity when calculating these upper-bounds. To further enhance efficiency, EFIM incorporates techniques for database projection and transaction merging, also achieving linear time and space complexity. These methods significantly reduce the computational cost associated with database scans. Comprehensive experiments across diverse datasets demonstrated EFIM's superior performance.

Peng, Koh, and Riddle [10] introduced mHUIMiner, which leverages a tree-based framework to steer the process of itemset growth. This approach efficiently eliminates the need to examine non-existent database itemsets. mHUIMiner emerges as different from other methods by avoiding complex and computationally intensive pruning mechanisms.

One of the earliest works on mining high utility itemsets using evolutionary algorithms from transaction databases is using Genetic Algorithm (GA) proposed by Kannumuthu and Premalatha [11]. They proposed two approaches to mine high utility itemsets from transaction databases with or without specifying minimum utility threshold by using genetic algorithm. Results from experiments showed that authors' GA approaches accomplished better performance in terms of scalability and efficiency.

Lin et al. [12] proposed using discrete Particle Swarm Optimization (PSO) to represent particles as binary variables. They introduced HUIM-BPSO, an efficient PSO-based algorithm designed to discover High-Utility Itemsets (HUIs) effectively. Their algorithm employs the transaction-weighted utility (TWU) model to identify high-transaction-weighted utilization 1-itemsets (1-HTWUIs). They established that their method mitigated the combinatorial complexity typically encountered during the evolutionary process.

Lan, Hong, and Tseng [13] introduced a novel pattern category called high transaction-weighted utility itemsets. Three key factors are taken into consideration in this approach: individual item profits, item quantities within transactions, and the overall contribution of each transaction to the database. To uncover these high transaction-weighted utility itemsets, the authors developed a two-stage mining algorithm. They demonstrated their methods through experiments conducted on synthetic datasets, which revealed promising performance results.

A 2-phase algorithm was introduced by Liu, Liao, and Choudhary [15] which aimed at streamlining the high-utility itemset mining process. Their approach effectively minimized the candidate set while ensuring the comprehensive discovery of all high-utility itemsets. Their algorithm exhibited efficiency in both time and memory consumption across various database types - synthetic and real-world. Their method successfully handles large-scale databases that typically pose significant challenges for existing methods.

A new algorithm called HUIM-ACS for high-utility itemset (HUI) mining was introduced by Wu, Zhan, and Lin [16]. Their algorithm is an enhanced version of the standard ant colony optimization (ACO) technique called ant colony system (ACS). HUIM-ACS provides several benefits compared to existing methods like genetic algorithms and particle swarm optimization. It constructs solutions in a way that avoids impractical outcomes, maps the entire solution space to ensure comprehensive coverage, and uses pruning processes to improve efficiency. The algorithm also prevents redundant evaluations of solutions, saving computational resources. Experimentation on real-life datasets showed that HUIM-ACS outperforms other heuristic HUI mining algorithms in both the number of HUIs discovered and convergence speed. While most evolutionary algorithms can't guarantee finding the global optimum, HUIM-ACS's comprehensive exploration of the solution space could potentially address this limitation in high utility itemset mining.

Song and Huang [17] introduced Bio-HUI, a framework that adapts bio-inspired algorithms for high-utility itemset mining. Unlike traditional approaches, Bio-HUI selects discovered HUIs as targets for the next generation, enhancing population diversity. The proposed framework was implemented using genetic algorithms, particle swarm optimization and bat algorithms. Extensive testing on public datasets demonstrates that these Bio-HUI-based methods outperform current state-of-the-art algorithms in efficiency, result quality, and convergence speed. The authors established that their method shows significant promise in advancing the field of high-utility itemset mining by leveraging the strengths of bio-inspired computational techniques.

Nawaz et al. [18] proposed two new algorithms for high-utility itemset mining (HUIM): HUIM-HC based on Hill Climbing and HUIM-SA based on Simulated Annealing. These algorithms aim to focus on the limitations of existing evolutionary and heuristic methods, which are typically affected by long runtimes and may miss many high-utility itemsets (HUIs). Their techniques used efficient utility computation and search space pruning through bitmap transformation of the input database. Their algorithms also improved population diversity by using discovered HUIs as targets for subsequent generations, rather than simply maintaining current optimal values. Their experimentation using real-life datasets demonstrated that HUIM-HC and HUIM-SA outperform state-of-the-art heuristic and evolutionary HUIM algorithms in terms of speed. Their techniques represent a significant step forward in addressing the challenges of high-utility itemset mining, offering improved efficiency and effectiveness over existing methods.

Song and Nan [19] introduced a new high-utility itemset mining algorithm called HUIM-ACO, based on ant colony optimization. HUIM-ACO uses a constructive approach to produce candidate itemsets, represented as search paths. Pheromone values are stored in a matrix to guide the search process, and an efficient enumeration technique is used to discover more itemsets. Their experimentation showed that HUIM-ACO outperforms existing algorithms in terms of speed and the count of high-utility itemsets discovered.

Li et al. [20] also proposed a high-utility itemset mining algorithm called HUIM-ACO, based on ant colony optimization. HUIM-ACO uses a constructive approach to generate candidate itemsets, represented as search paths. Pheromone values are stored in a matrix to guide the search process, and an efficient enumeration technique is used to discover more itemsets. Experimental results demonstrated that HUIM-ACO outperforms existing algorithms in terms of both speed and the count of high-utility itemsets discovered.

Han et al. [22] introduced a new high-utility itemset mining algorithm that incorporates two key strategies: positional evolution based on the female elephant factor to reduce the search space and improve efficiency, and two-phase population diversity maintenance to prevent premature convergence. Their experimentation showed that this algorithm outperforms existing heuristic methods in terms of both speed and effectiveness.

In order to obtain a near-optimal solution based on fitness functions under a number of constraints, evolutionary algorithms can search through enormous problem spaces. Even though the current evolutionary-based HUIM algorithms can mine all HUIs that meet the minimum utility threshold faster than classic exact methods, they can still be quite time-consuming. This is mostly because search and evolutionary algorithms, which perform well in problems with fewer optimal solutions, keep the best values from one population to direct the next. However, because it overemphasizes previously discovered optimal values within a small number of iterations, this strategy runs the risk of missing some itemsets in High Utility Itemset Mining (HUIM), when results are many and unevenly distributed. Enhancing the diversity of the generated population, as suggested by Song and Huang [17], is one way to solve this issue. Rather than selecting HUIs with high utility values from the current population, this method applied roulette wheel selection to all of the identified HUIs in order to probabilistically choose the initial target of the next population.

To enrich the efficiency, this work suggests a hybrid approach that coalesces Ant Colony Optimization (ACO) and Genetic Algorithms (GA). Despite achieving expeditious global convergence, GAs have trouble with feedback and may carry out dispensable iterations, which can truncate precision. The lack of pheromone trails causes ACO to perform poorly at first, but it excels at updating information for optimal convergence. To get beyond these restrictions, the suggested genetic ant colony algorithm makes utilization of the advantages of both approaches. We modeled the HUIM problem from the perspective of the hybrid genetic algorithm, using the work done in study [17] as a starting point.

III. Problem Statement

The problem of HUIM is: Given a transaction database (*TD*), its profit table (*ptable*) and a user specified minimum utility threshold, the problem of HUIM is to identify all itemsets that have utility values equal to or greater than *min _util*. Let us say that Table I represents the Transaction Database of items procured and Table II represents the external profit value of each item.

TABLE I. Transaction Database

TD	Transactions	TU
*T*0	(a, 3) (c, 12) (e, 3)	54
*T*1	(b, 4) (d, 2) (e, 1) (f, 5)	47
*T*2	(a, 3) (c, 2) (e, 1)	16
*T*3	(a, 2) (d, 2) (f, 1)	15
*T*4	(a, 1) (c, 5) (d, 7)	52
*T*5	(b, 1) (d, 4) (f, 2)	29

TABLE II. Profit Table

Item	a	b	c	d	e	f
profit	2	7	3	5	4	1

From the transaction database in Table I and profit table from Table II, the utility value of an item *e* in the transaction T_0 is represented as $u(e,T_0) = 3 \times 4 = 12$. The utility of itemset {a, c} in transaction T_0 is $u(\{a,c\},T_0) = u(a,T_0) + u(c,T_0) = 3 \times 2 + 12 \times 3 = 42$. Along similar lines, the utility value of an itemset {a, c} in the transaction database TD will be $u(\{a,c\},T_0) + u(\{a,c\},T_2) + u(\{a,c\},T_4) = 42+12+17 = 71$. The transaction utility (TU) of an entire transaction T_0 is represented as $TU(T_0) = u(\{a,c,e\},T_0) = 54$. If the threshold utility *min util* = 80 then the itemset {a, c} is not considered an HUI as the itemset utility value $u(\{a,c\}) <$ *min util*. However, as the itemset {a, c} is part of three transactions T_0, T_2 and T_4 the TWU (Transaction Weighted-Utilization) is computed as $TWU(\{a,c\}) = TU(T_0) + TU(T_2) + TU(T_4)$. The value of this expression is 122. Since

TWU of the itemset $\{a,c\} > min\ util$ this itemset is considered an HTWUI (high transaction-weighted utilization itemset).

A. Terminologies

In this section, we define the terminologies related to the problem statement. Let $I = \{i_0,i_1,...,i_m\}$ represent a finite set of m itemsets in the transaction database $TD = \{T_0,T_1,...,T_d\}$. Each transaction T_k in TD is a subset of I which has a unique identifier $k(1 \le k \le n)$ and is called TID. Also, the set $X \subseteq I$ is called an *itemset* and an itemset which consists of k items is called a k-itemset. An itemset X is contained inside a transaction T_k if $X \subseteq T_k$. Every item i_j in T_k has a positive number $q(i_j,T_k)$ which is called it's *internal utility* and indicates the quantity (or occurrence) of i_j in T_k. The external utility$p(i_j)$, represents the unit profit value of the item i_j. The profit table $ptable = \{p_1,p_2,...,p_m\}$ depicts the profit value p_j of each item i_j in I.

The overall utility value of an item i_j in a transaction T_k is defined by the following equation as:

$$u(i_j,T_k) = p(i_j) \times q(i_j,T_k) \quad (1)$$

In any given transaction T_k, the utility if the itemset X is represented as $u(X,T_k)$ and characterizes the amount of money from the sale of X in that transaction [14]. Also, the overall utility value of an itemset X in TD denoted by $u(X)$ represents the total amount of money that the itemset yields for all transactions where X is purchased in the database. These two ideas are formally defined as:

$$u(X,T_k) = \sum_{i_j \in X \wedge X \subseteq T_k} u(i_j,T_k) \quad (2)$$

$$u(X) = \sum_{X \subseteq T_k \wedge T_k \in TD} u(X,T_k) \quad (3)$$

The user set minimum utility threshold (δ) is the percentage of total sum of all TU values in the input database. The minimum-utility value is defined as:

$$min\_util = \delta \times \sum_{T_k \in TD} TU(T_k) \quad (4)$$

An itemset X is considered an HUI if $u(X) \ge min\_util$. Search space reduction is often carried out in HUIM by defining another term called transaction weighted-utilization (TWU) which is an upper bound on the utility value of an itemset and its supersets.The TWU of an itemset X is the total sum of transaction utility values of all the transactions that contain X and is defined as:

$$TWU(X) = \sum_{X \subseteq T_k \wedge T_k \in TD} TU(T_k) \quad (5)$$

An itemset X is considered a high transaction weightedutilization itemset (HTWUI) if $TWU(X) \ge min\ util$; otherwise X is considered a low transaction weighted-utilization itemset (LTWUI).

IV. Proposed Methodology

This study addresses the challenge of high-utility itemset mining by proposing a hybrid algorithm approach which combines Genetic Algorithms (GA) and Ant Colony Optimization (ACO). Both GA and ACO are iterative optimization techniques, which forms the basis for their integration. GAs excel at rapid global convergence but struggle with feedback information. Once a solution reaches a certain range, GAs tend to perform redundant iterations, potentially reducing the accuracy of the final result. In contrast, ACO continuously gathers and updates information so that it converges to the optimal solution, leveraging its global search capabilities and parallel processing. However, ACO's initial performance is hindered by the lack of early pheromone trails. To address this limitation, we propose a genetic ant colony algorithm which capitalizes on the complementary strengths of both methods. The ACO algorithm is applied to itemsets, utilizing its ability to gather and update information continuously. Subsequently, GA is employed on the discovered High Utility Itemsets (HUI) to streamline the algorithm. In each iteration, mutation generates an HUI where crossover can be applied. GAs, inspired by natural selection and genetics, model the evolution of potential solutions through selection, crossover, and mutation. They are versatile and applicable to various optimization and search problems. ACO, on the other hand, mimics the foraging behavior of ants, simulating how they find paths by depositing and following pheromone trails. ACO excels in exploring complex solution spaces, adapting to changing conditions, and finding near-optimal solutions mainly for problems which have large search spaces. The hybrid approach aims to mitigate the premature convergence problem often encountered in genetic algorithms. By combining GA and ACO, the algorithm strikes a balance between exploitation and exploration. The ACO/GA hybrid typically reduces the number of offspring produced by constructing solutions gradually. This results in fewer new solutions needing to be stored in memory compared to a standard GA, leading to a larger pool of offspring for selection and significantly reduced memory usage.

A. Encoding and Pruning

An efficient representation method for mining HUIs is transformation of initial transaction database into a bitmap [16]. Here, the transaction is encoded using binary notation; an entry of '0' denotes the absence of an item, whereas an entry of '1' denotes its presence. The bitmap cover of an itemset X is computed as $Bit(X)$ = bitwise-AND $_{i \in X}(Bit(i))$. This indicates that X is a bit vector that is produced by applying a bitwise-AND operation to the bitmap covers of each and every item in X. For two itemsets X and Y, $Bit(X \cup Y)$ can be evaluated as $Bit(X) \cap Bit(Y)$, the bitwise-AND of $Bit(X)$ and $Bit(Y)$.

TABLE III. Bitmap Representation

T_k	a	b	c	d	e	f
$T0$	0	1	1	0	1	0
$T1$	1	0	1	0	1	1
$T2$	1	0	1	0	1	0
$T3$	0	1	0	1	0	1
$T4$	1	0	1	1	0	0
$T5$	1	0	0	1	0	1

Table III illustrates the bitmap of Table I's database, for reference. The column vectors $B(a)$ = 011011 and $B(c)$ = 111010, respectively, represent the bitmap covers of items a and c. The bitmap cover of itemset $\{a,c\}$ is the column vector obtained by performing the bitwise-AND of $B(a)$ and $B(c)$, that is $B(\{a,c\})$ = 011010. The study in [17] proposed a promising encoding vector for speeding up the process of mining HUI and is employed in this proposed work too.

Let's say that V represents an encoding vector that contains $0s$ and/or $1s$ and corresponds to a solution. If $Bit(X)$ only contains $0s$ then V is called an unpromising encoding vector ($UPEV$), otherwise V is called a promising encoding vector (PEV). Since an empty encoding vector indicates that the itemset does not contain any *HTWUI*, it is simple to understand that each itemset (solution) X that is represented by a *UPEV* cannot be an HUI.

This type of solution can significantly cut down on runtime because it does not require the fitness value to be computed. This technique is called PEV-Check (PEVC) pruning approach [17]. Every newly generated solution goes through this strategy to make sure that the solution actually exists in the database. The pseudocode of this strategy is given as follows:

Algorithm 1: Encoding and Pruning

Step 1	Determine which of the elements in the encoding vector (EV) are represented by $1s$ and stores it in VN after looking for $1s$ in the vector.
Step 2	Initialize a variable XV with bitmap cover of the first item in EV.
Step 3	Start a loop, for each item i_k, perform bitwise AND operation on XV with bitmap cover of i_k.
Step 4	If the resulting bit vector is a UPV, then the item is not kept in XV and the bits of i_k in EV is changed from 0 to 1.
Step 5	Repeat Steps 3 to 4 till VN is empty

B. Population Initialization

The initial population for hybrid ACO/GA is generated randomly. Algorithm 2 initially searches the database for all 1-HTWUIs, and as 1LTWUIs cannot be a part of any HUI, they are subsequently removed. After that, a bitmap is created from the database. After that, a for loop creates the first individuals one at a time, assigning a random number of 1s to each person in the i^{th} bit vector, where n is an integer between 1 and |1 - HTWUIs |. A bit vector with 1s in it is formed. The following formula indicates the likelihood that the bit corresponding to i_j will be set to 1 is as follows:

$$P_j = \frac{TWU(i_j)}{\sum_{k=1}^{|1-HTWUIs|} TWU(i_k)} \quad (6)$$

Algorithm 2: Population Initialization

```
Step 1   Perform a single scan on D to remove 1-LTWUIs
         and identify all 1-HTWUIs
Step 2   Transform the database D to a bitmap
         representation of D;
Step 3  Start a for loop from i = 1 to P
        3.1Generate a random number n_i,an integer
           between 1 and |1-HTWUIs |;
        3.2Generate V_i with n_i bits set to 1using the Eq 6
        3.3   if n_i > 1 then
                V_i = PEV C(V_i);
              endif
           endfor
end
```

C. Hybrid ACO Algorithm

The primary objective of ACO algorithm is to take full advantage of the features of ACO and GA for mining itemsets with high utility. Both ACO and GA gives best result for mining HUI. In the proposed technique, population is initialized by following Algorithm 2. The entire number of transactions in the database is represented by the number of ants' SN. Pheromones are initialized with the utility values, iff TWU(X,Tc) ≥ min util. Genetic operators crossover is applied on the generated solution by the ant. In this study, a single-point crossover is utilized. The crossover point is selected randomly. Now mutation operator is applied by changing one or more random bits in a string when Pm ≥ randomly generated value.

Algorithm 3: Hybrid ACO algorithm

```
Step 1   Population initialization
Step 2   Pheromone initialization with utility value
Step 3   iter = 1
Step 4   while (iter ≤ max iter)
Step 5   for each transaction in D
           Start each ant to visit each vertex
         endfor
Step 6   Apply genetic operators crossover
         on the selected population
Step 7    Perform mutation on the population of
          individuals with mutation probability pm.
          7.1   Verify the fitness value of individuals,
          7.2    If fv ≥ min util go to Step 6
          7.3    Else go to Step 8
Step 8    Pheromone update
Step 9    End of while
Step 10   Output all HUIs
```

V. Result and Discussion

This part includes a discussion of the findings and experimental assessment of the suggested algorithms. Experiments were performed on a computer equipped with an 8-core 3.6 GHz CPU and 8 GB RAM running Windows 10. The program was developed in Java.

A. Dataset

Four standard benchmark datasets- Chess, Mushroom, Connect, and Accident - are utilized to assess the performance and effectiveness of the suggested algorithm. The algorithm performance is also evaluated against a real dataset downloaded from the UCI repository. The benchmark datasets were sourced from SPMF data mining library [23].

A widely used dataset from the UCI Machine Learning Repository, the Mushroom dataset is often employed in association rule mining and High-Utility Itemset (HUI) mining applications. There are 8,124 different species of mushrooms in it, and each one is characterized by 22 different categories, including gill spacing, color, cap shape, and odour. Every instance has a label designating it as deadly or edible. The dataset can be modified in the context of HUI mining by giving utility values (such as profit or significance) to particular features in order to discover desirable itemsets based on utility thresholds as opposed to just frequency. When it comes to HUI mining, the Mushroom Dataset presents a problem because it requires categorical data to be transformed into a format that makes utility-based itemset mining feasible. In order to match with the utility mining paradigm, this frequently entails defining the utility of goods (attributes) and transactions (mushroom instances) in novel ways.

The chess dataset is widely used in High-Utility Itemset (HUI) mining as a benchmark for assessment of the performance of various algorithms. This dataset represents chess games transactionally, with each transaction denoting a series of moves or itemsets. Finding itemsets (e.g., common move sequences) with high utility, like winning strategies, is the goal. In this case, the utility values could stand for the significance or regularity of moves within the dataset. As the chess dataset is structured, repetitious, and dynamic [21], it's a great resource for researching utility-based itemset mining. Every transaction in HUI mining can be compared to a position in a chess game. Each item can stand for a particular move or the location of a chess piece.

Based on actual accident data, the Accident _10% dataset is a 10% sampling of the entire dataset. The dataset comprises transactions that typically contain attributes related to an accident, such as conditions, location, and involved entities. The utility values assigned to each attribute indicate its significance or influence. When mining for interesting patterns, this dataset aids researchers in assessing the accuracy, scalability and execution time of HUI algorithms.

High-utility itemset (HUI) mining research frequently uses the Connect dataset as a benchmark. It is derived from a Connect-4 game and is sourced from the UCI Machine Learning Repository. With 43 categorical attributes, the dataset consists of 67,557 instances, each of which represents a unique board configuration in the game. Each configuration can be looked at as a transaction in the context of HUI mining, where the objective is to mine itemsets (board configurations) that offer high utility or significance, frequently based on different utility values ascribed to different configurations. The Connect dataset is a valuable resource for examining recurring and noteworthy patterns in gaming, since its utility may be linked to winning plays or strategies in the Connect-4 game.

The proposed algorithm is also tested against a real dataset - the online retail dataset downloaded from UCI Repository. The dataset has 541909 instances with six features. A UKbased online retailer's sales transactions are included in this dataset from a period of December 2010 to December 2011. The dataset includes 8 attributes with approximately 4000 distinct items. Cancelled orders, returns (negative quantities), and possibly outliers are included in the dataset. In order to handle returned items or eliminate invalid transactions, preprocessing is frequently necessary. The dataset exhibits a high degree of transaction data skewness because it includes both huge bulk transactions and numerous low-quantity sales. The identification of high-utility itemsets is impacted by this skewness. Since utility values were not included in the dataset, they had to be manually determined using the following technique so that they could be used in HUI mining where *Utility = quantity × unit price*. Table IV shows characteristics of datasets and Table V shows sample online retail dataset.

TABLE IV. Characteristics of Datasets

Dataset	Trans. Len	No. of items	No. of Trans	Type
Mushroom	23	119	8,416	Dense
Chess	37	75	3,196	Dense
Connect	43	129	67,557	Dense
Accident 10%	34	468	34,018	Dense

TABLE V. Sample Online Retail Dataset

Invoice No	Stock Code	Description	Quantity	Invoice Date	Unit Price	CustomerID	Country
536365	71053	Jam making set	5	12/1/2010	2.25	13047	United Kingdom
536366	85123A	Jam making set	1	12/1/2010	2.25	12583	France
536367	71057	White Hanging Heart	6	12/1/2010	3.39	17850	United Kingdom
536367	22993	White Hanging Heart	6	12/1/2010	4.25	12678	France

B. Runtime efficiency

Experiments are conducted to assess the efficiency of the proposed algorithm with regards to runtime. The efficiency is tested by varying the minimum utility value.

Table VI summarizes the minimum utility value set for the dataset.

TABLE VI. MINIMUM UTILITY VALUES

Dataset	Minimum Utility Threshold				
Chess	28.5	29	29.5	30	30.5
Mushroom	14	14.5	15	15.5	16
Connect	31.8	32	32.2	32.4	32.6
Accident 10%	12.6	12.8	13	13.2	13.4

Fig. 1, 2, 3, and 4 show the execution time of our algorithm presented in this work. The runtime efficiency of our proposed algorithm is compared with HUIM-GA. Hybrid ACO/GA produces better runtime efficiency because it focuses on promising areas early on, reducing unnecessary evaluations of low utility items. The incremental building of solutions impacts the efficiency of our proposed algorithm. Also, the adaptive mechanisms such as pheromone evaporation leads to an efficient search, thus reducing runtime.

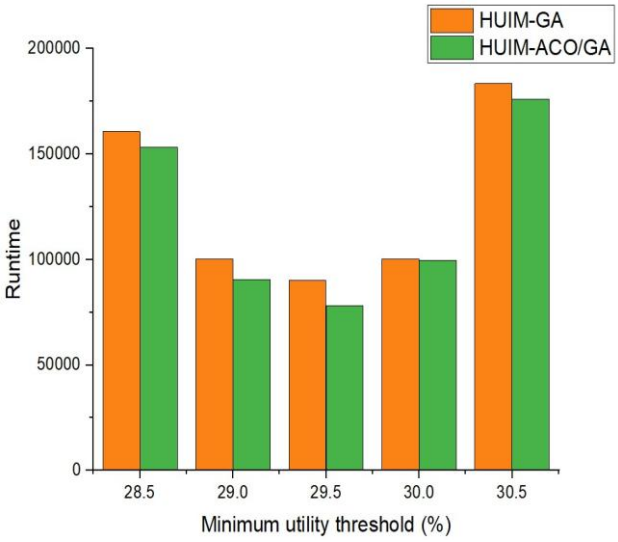

Fig. 1. Chess dataset (based on runtime).

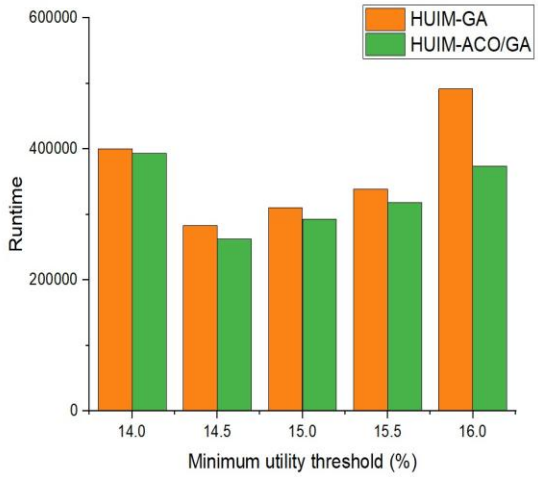

Fig. 2. Mushroom dataset (based on runtime).

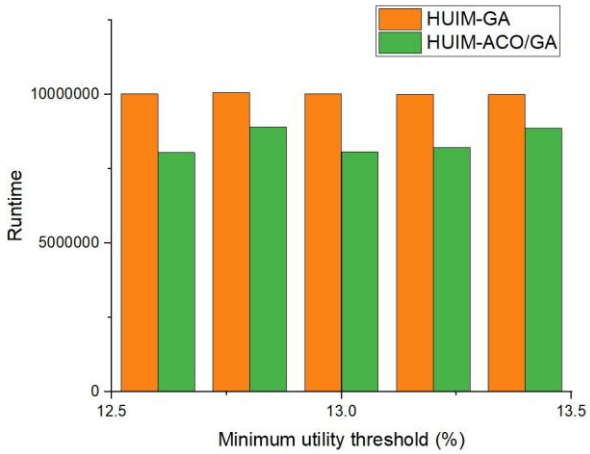

Fig. 3. Accident_10% dataset (based on runtime).

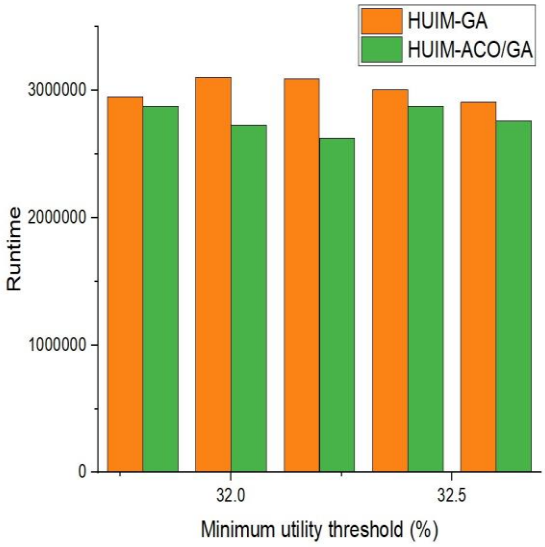

Fig. 4. Connect dataset (based on runtime).

A real-time retail dataset is also used to evaluate the recommended algorithm's runtime efficiency. Here at minimum utility threshold of 10%, maximum number of HUI's were identified. Fig. 5 demonstrates the result.

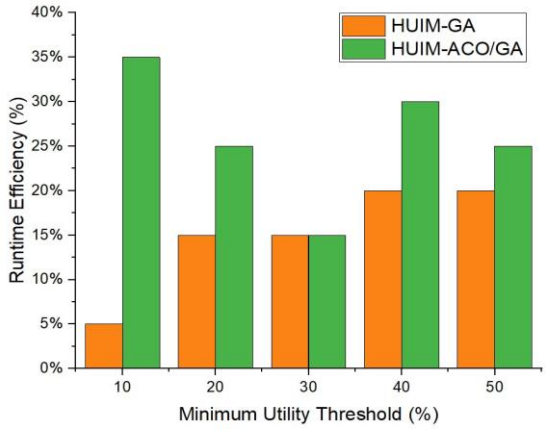

Fig. 5. Retail dataset (based on runtime).

C. Discovered number of High utility items(HUI)

This section analyzes the performance of the hybrid ACO/GA approach by evaluating the number of HUIs identified at various threshold levels. The proposed algorithm discovers the maximum number of HUIs. Fig. 6 to 10 demonstrate the graphical representation of the experimental results. It may be noticed that hybrid ACO/GA discovers more number of HUIs in most cases. However, for the real time retail dataset, the number of HUIs identified by hybrid ACO/GA algorithm were comparable to that of HUIM-GA algorithm as may be seen in Fig. 10.

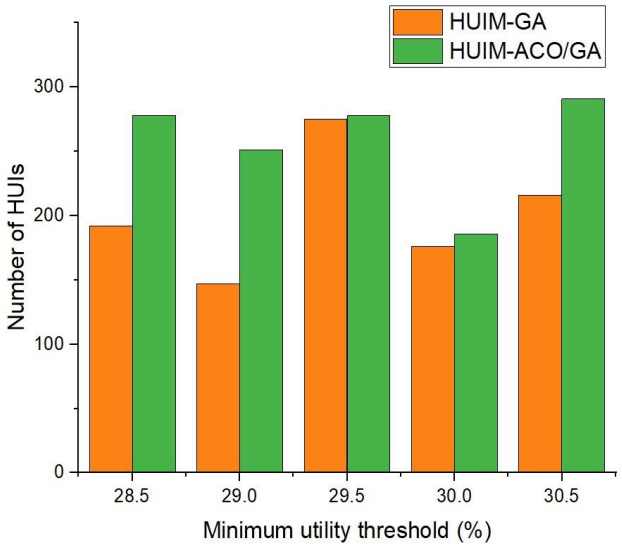

Fig. 6. Chess dataset.

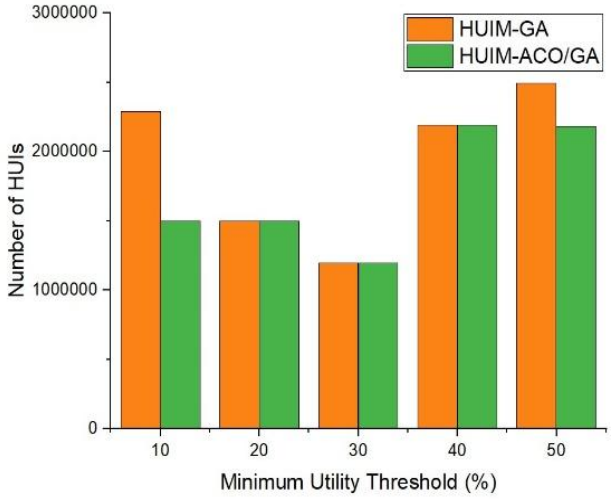

Fig. 7. Mushroom dataset (based on number of HUIs).

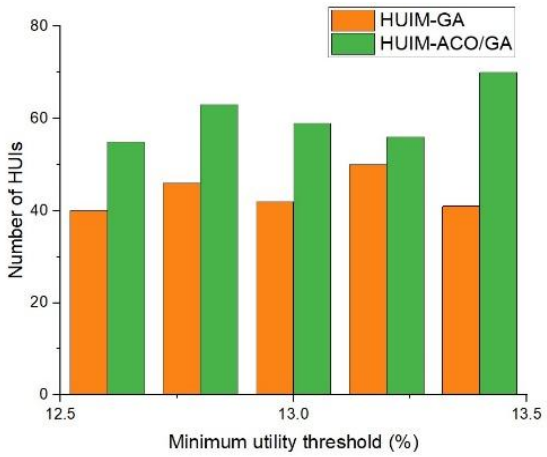

Fig. 8. Accident_10%dataset (based on number of HUIs).

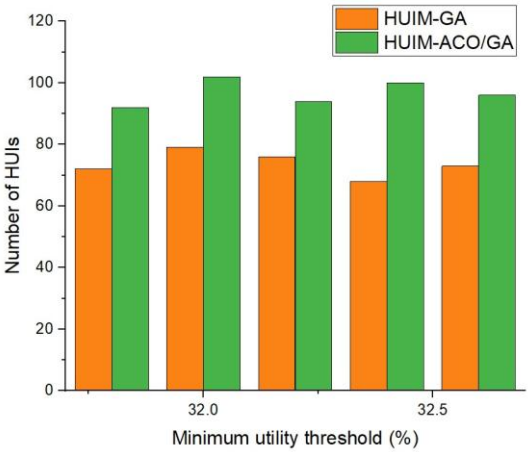

Fig. 9. Connect dataset (based on number of HUIs)..

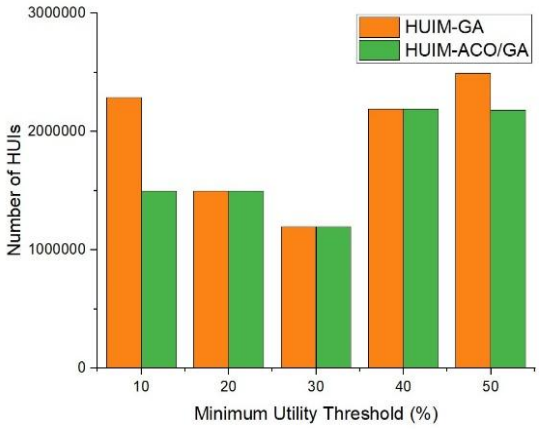

Fig. 10. Retail dataset (based on number of HUIs)..

D. Convergence

Using the four datasets - Chess, Mushroom, Accident, and Connect, 4000 fitness evaluations were conducted to assess the convergence properties of the proposed approach. Minimum utility threshold was kept at the value which gives the maximum number of HUIs discovered in each dataset. Premature convergence of genetic algorithm is avoided here. Convergence curves obtained on the Chess, Mushroom and Retail datasets on both hybrid ACO/GA are represented in Fig. 11, 12 and 13 respectively.

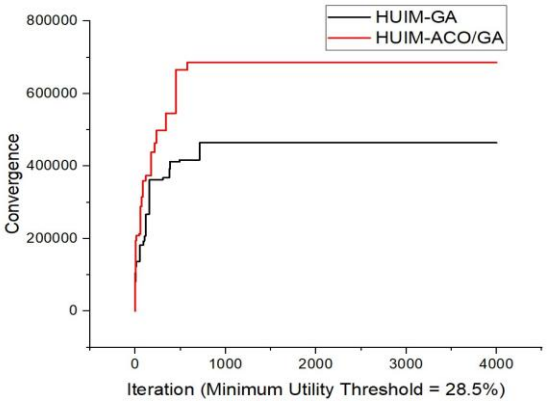

Fig. 11. Chess dataset (convergence).

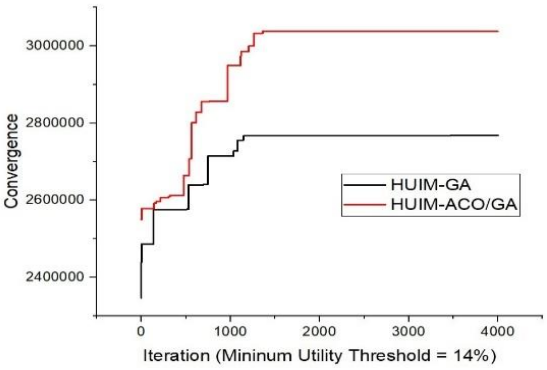

Fig. 12. Mushroom dataset (convergence).

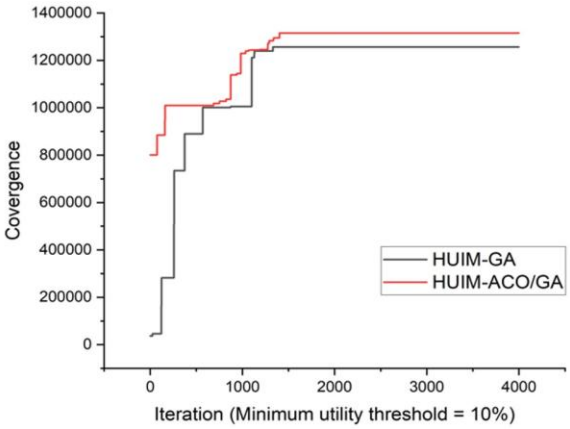

Fig. 13. Retail dataset (convergence).

VI. Conclusion and Future Enhancement

With a wide range of applications, HUIM is a significant data mining task. When mining HUIs, EC-based HUIM algorithms outperform conventional HUIM algorithms such as HUPEumu-GRAM, HUIM-BPSOsig, HUIM-BPSO, and BioHUIF-GA. Even though these algorithms offer an effective method for extracting HUIs from the transition datasets, finding the full or significant proportion of HUIs still takes a lot of effort. While hybrid ACO-GA algorithm yields good results on generic datasets, the suggested technique uses more resources to create HUIs on real-time datasets. Our proposed algorithm yields the same significant number of HUIs as other algorithms and after a certain period of time both algorithms converge and the quantity of HUI generated is sparse. The suggested algorithm will be used to mine high utility itemsets from the e-commerce dataset in order to implement dynamic pricing strategies. This will allow for real-time price modifications based on patterns of customer demand and product profitability. Dynamic pricing is made possible by mining HUIs in e-commerce datasets to find high-utility items and product groups. Through value-driven tactics, this aids companies in maintaining client loyalty, increasing revenue, and optimizing pricing in real-time. Our next improvement can be to use fewer resources to address the HUIM issue and improvise our proposed algorithm.

References

[1] Agarwal R., and Srikant R., “Fast algorithms for mining association rules”, In the Proceedings of 20th International Conf. Very Large Data Bases, pp.487-499, 1994.

[2] Agrawal R., Imielinski T, Swami A, “Mining association rules between´ sets of items in large databases”, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data - SIGMOD ‘93. pp.207, 1993.

[3] Tseng V.S., Shie Bai-En, Wu Cheng-Wei and Yu Pillip S., “Efficient Algorithms for Mining High Utility Itemset from Transactional Databases”, IEEE Transactions on Knowledge and Data Engineering, Vol. 25, No. 8, pp. 1172-1786, 2013.

[4] Chan R., Yang Q., and Shen Y., “Mining High Utility Itemsets”, Proceedings of the IEEE 13th International Conference on Data Mining, Melbourne, Florida, pp.19- 26, 2003.

[5] Yao H., Hamilton H.J., and Butz C.J., “A Foundational Approach to Mining Itemset Utilities from Databases”, Proceedings of the 2004 SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, pp.482-486, 2004.

[6] Ahmed C.F., Tanbeer S.K., Jeong Byeong-Soo, and Lee Young-Koo, “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.

[7] Liu M. and Qu J., “Mining High Utility Itemsets without Candidate Generation”, Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 55-64, 2012.

[8] Philippe Fournier Viger, Cheng-Wei Wu, Souleymane Zida, Vincent S. Tseng, “FHM: Faster High-Utility Itemset Mining using Estimated

Utility Co-occurrence Pruning”, Proc. 21st International Symposium on Methodologies for Intelligent Systems (ISMIS 2014), Springer, LNAI, pp 83-92, 2014.

[9] Artificial Intelligence Advances in Artificial Intelligence and Soft Computing pp 530-546, 2015.

Peng A., Koh Y. S, and Riddle P. J., “mHUIMiner: A Fast High Utility Itemset Mining Algorithm for Sparse Datasets” Proceedings 21st Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp.196-207, 2017.

[10] Utility Itemset Mining Algorithm for Sparse Datasets" Proceedings 21st Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp.196-207, 2017.

[11] Souleymane Zida, Philippe Fournier-Viger, Jerry Chun-Wei Lin, ChengWei Wu, Vincent S. Tseng, "EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining", Mexican International Conference on

[12] Kannumuthu S., and Premalatha K., "Discovery of High Utility Itemsets Using Genetic Algorithm" International Journal of Engineering and Technology (IJET), Vol. 5 No. 6, pp. 4866-4880, 2013.

[13] Lin C.W, Yang L., Fournier-Viger P., Hong T.P., and Voznak M., "A Binary PSO Approach to Mine High-Utility Itemsets", Soft Computing, Vol. 21, pp. 5103-5121, 2016.

[14] Lan G.C., Hong T.P.; and Tseng V.S., "Mining High-Transaction Weighted Utility Itemsets", 2010 Second International Conference on Computer Engineering and Applications, Bali, Indonesia, pp. 314-318, 2010.

[15] Lichman M., UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, 2013.

[16] Liu Y., Liao Wk., and Choudhary, A., "A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets", Lecture Notes in Computer Science, Vol. 3518, pp. 689-695, 2005.

[17] Wu J.M-T, Zhan J., and Lin, J. C-W, "An ACO-based Approach to Mine High-Utility Itemsets", Knowledge-Based Systems, Vol. 116, pp. 102113, 2017.

[18] Song W., and Huang C., "Mining High Utility Itemsets Using BioInspired Algorithms: A Diverse Optimal Value Framework", IEEE Access, Vol. 6, pp. 19568-19582, 2018.

[19] Nawaz M. S., Fournier-Viger P., Yun U., Wu Y., and Song, W., "Mining High Utility Itemsets with Hill Climbing and Simulated Annealing". ACM Transactions on Management Information Systems, Vol. 13 No. 1, pp. 1-22, 2021.

[20] Song W., and Nan J., "Mining High Utility Itemsets using Ant Colony Optimization" Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery, pp. 98-107, 2020.

[21] Li Y., Zhao Y., Shang Y., and Liu J., "An improved firefly algorithm with dynamic self-adaptive adjustment", PLoS One, Vol. 16, 2021.

[22] Han M., He F., Zhang R., Li C., and Meng F., "Mining High Utility Itemsets with Elephant Herding Optimization", https://doi.org/10.21203/rs.3.rs-3881656/v1, 2024.

[23] Fournier-Viger P., Gomariz A., Gueniche T., Soltani A., Wu C.W., and Tseng V.S., "SPMF: A Java Open-Source Pattern Mining Library", Journal of Machine Learning Research, Vol. 15, pp. 3569-3573, 2014.