Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

Mining High Utility Item Sets Through a Swarm-Based Optimization Method

Raja Rao Budaraju[1], Sastry Kodanda Rama Jammalamadaka[1*], Sasi Bhanu Jammalamadaka[2], Balakrishna Kamesh Duvvuri[3]

[1]Department of Computer Science and Engineering, KL Deemed to be University, Vaddeswarm, 522501, Guntur District, India
[2]Department of Computer Science and Engineering, CMR College of Engineering and Technology, Hyderabd, 501401, India
[3]Department of Computer Science and Engineering, MLR Institute of Technology, Hyderabad, 500043, India
E-mail: drsastry@kluniversity.in

Received: 10 May 2025; **Revised:** 14 May 2025; **Accepted:** 14 May 2025

Abstract: Data mining identifies patterns in vast databases to support practical decision-making. Using association rule mining, the patterns in the transaction database are found and reveal client behavior. Frequent Itemset Mining (FIM) Finds Often-Buyd Items. FIM's disdain for item significance is a drawback. Practical applications depend on the relevance of the item. Thus, the High-Utility Itemset Mining (HUIM) challenge requires identifying the most profitable items in the transaction database. High-Utility Items (HUI) in the transaction database can be identified using several methods. HUIM methods based on utility lists are novel and outperform traditional methods in terms of memory consumption and execution time. The main limitation of this algorithm is the costly joins of the utility list. This research presents a highly effective swarm intelligence-based method for optimizing HUIM issues. The execution time of the suggested method is assessed. In addition, it is compared to advanced current methods. Testing on publicly available benchmark data sets shows that the swarm-based strategy outperforms current methods. This technique has wide applications in retail analytics, healthcare, fraud detection, finance, supply chain management, and recommendation systems, enabling businesses and researchers to optimize decision making and extract meaningful insights from transactional data.

***Keywords*:** high utility itemset, discrete, improved discrete cuckoo search

MSC: 68T37

1. Introduction

Due to their ability to reveal hidden insights, data analysis systems are experiencing significant growth. FIM is essential to data analysis and mining [1]. Frequently occurring events, patterns, or objects (individually or collectively) are extracted [2]. HUIM [3] is a growing field that addresses FIM's premise that all database patterns are equally relevant [1]. HUIM seeks patterns with high utility relative to user value [4]. The utility function was originally designed to measure retail sales profit [5]. It can be generated from numerous criteria. HUIM is an extension of FIM that assigns values to input data items based on their importance or weight in the problem. HUIM allows multiple repetitions of an item in a transaction, while FIM indicates whether each instance occurs. HUIM techniques in the early 2000s [6] yielded

DOI: https://doi.org/10.37256/cm.6320257213

major and helpful discoveries. Market basket analysis [5] and sentiment analysis [6] examined the issue. This hot issue has several effective algorithms [7, 8] for listing all item sets that benefit more than a threshold of the user set. Some methods [8] generate and test candidates' level-wise, while others [9, 10] use pattern growth. However, HUIM algorithms can slow down massive searches. More particular, accurate techniques perform worse with more input data, especially for transactions and objects [6]. Performance disappoints those who cannot wait for results. Genetic algorithms [11] and particle swarm optimization [12] aim to address the precise limitation of the HUIM performance. This work discretises ICDS, a swarm-based optimisation method, to solve HUIM. Key contributions of the proposed study include modifying the Cuckoo Search algorithm to address HUI issues in binary solution spaces, evaluating the suggested model, examining its runtime and convergence towards optimal solutions, and assessing HUI to demonstrate its significance. The suggested model is also compared to recently developed methods.

The proposed model builds on work like Utility-List Based (ULB)-Miner and Efficient Fast Itemset Miner (EFIM) by addressing critical limitations in execution time and memory efficiency in HUIM. ULB-Miner introduced a buffer structure for the utility list that optimized memory usage and reduced redundant database scans, thus improving execution time. However, it still struggled with scalability and could not process large and complex data sets. Similarly, EFIM focused on memory-efficient algorithms by consolidating similar transactions and applying stringent utility upper bounds. EFIM incurred high computational costs in handling massive datasets despite improvements due to extensive subtree and utility pruning techniques. The proposed model improves these foundational approaches by integrating swarm intelligence and the discrete cuckoo search algorithm, which optimizes memory and runtime. Leveraging Levy Flight and Biased Flight/Random Walk (BFRW) strategies achieves faster convergence towards high-quality solutions while maintaining global exploration. Additionally, population-based optimization reduces the need for exhaustive candidate generation and database scans, unlike traditional methods such as EFIM and ULB-Miner. This significantly improves execution time while conserving memory, making the model more suitable for handling large-scale HUIM problems. Experimental results on various datasets further validate that the proposed model consistently outperforms ULB-Miner and EFIM in both run-time efficiency and memory utilization. The novelty of the proposed model includes:

1. Integrating Levy flights for exploration in discrete spaces.
2. Extending the binary solution space for improved optimization.
3. Evaluated the proposed model in 9 different datasets to prove its significance.
4. The proposed model improved with an average efficiency of 12.1% of accuracy in terms of identifying appropriate item sets that holds high utilization in the given dataset.

The technical challenge on this HUIM by Iterative Depth-first Combinatorial Search (IDCS) was handling the combinatorial explosion of item sets while ensuring efficient convergence toward optimal solutions. This was addressed by integrating Levy Flight Random Walk (LFRW) for global exploration and BFRW for local exploitation, which dynamically balance search diversity and refinement, preventing premature convergence while improving computational efficiency. The structure of the following sections is as follows. Section 2 discusses precise and bioinspired HUIM algorithms from previous research. Section 3 describes the HUIM problem and presents a mathematical model. Section 4 presents the discrete swarm optimization paradigm. Section 5 empirically assesses the proposed model. Section 6 presents the final conclusions and prospects.

2. Related work

The Ao and Hamilton HUIM improved mining efficiency. The Transaction Weight Utility (TWU) from Liu et al. is a novel upper limit that meets the downward closure property and fully extracts Highly-Utility Itemsets (HUI) [13]. The Two-Phase technique requires many database scans to find all HUIs, and the TWU is a reliable but imprecise upper limit, increasing processing time.

[14] reduced Dynamic Time Warping Utility (DTWU)-based Mining candidates and database scans with a diffset. This minimizes search and memory. [15] generated UPGrowth from Frequent Pattern (FP)-Growth. In 2023, [16]

improved the HUIM tree mining with Hamm. They introduced TV prefix Tree and utility one-phase vector mining of high-utility itemsets without candidate formation.

[17] developed HUIMiner, which uses data from the utility list and a tighter upper residual utility restriction than TWU. The Utility-List eliminates the need for multiple applicants. This structure allows the application to scan the database once and consolidate the utility list itemsets to create larger ones.

[18] upgraded HUI-Miner using FHM. The FHM method reduces the two-item itemsets to increase performance. EUCP and EUCS data formats mitigate this. Duong et al. suggested HUP-Miner to update the FHM algorithm in 2017 [19]. Better pruning choices narrow the search with this strategy.

[20] reduce execution time and memory use with a buffer layout of utility lists. Duong et al. released the HMiner algorithm in 2017, which includes TWU-, EUCS-, Uprune, LA-, and C-prunes, as well as a technique for grouping identical transactions. Having a single repository for "closed" transactions reduces the program's runtime. UBP-Miner minimizes transaction scans to improve list-based [21]. UBP-Miner outscored HUIMiner and ULB-Miner in benchmarks.

Algorithms handle partially identical transactions in addition to utility-list and tree-based techniques. EFIM was an early algorithm that used this approach. Nguyen et al. proposed the algorithm [22]. The method consolidates similar transactions and reorganizes elements within them to project and combine them, saving memory. The upper boundaries of the subtree and the local utility are also stringent. EFIM struggles with high database scanning expenses. The P-Set structure of the incremental Mining of Efficient Frequent Itemsets with Memory consideration (iMEFIM) method [23] accurately records the transaction positions. The algorithm manages databases with different transaction utilities.

Practical challenges arise with HUIM techniques, Utility-Oriented Pattern Mining [24], closed HUIS [25], maximal HUIS [26], uncertainty HUIS [27], Cross-Level HUIS [28], and weighted high-utility pattern mining [29]. Soft computing and deep learning have been developed by solving domain-wide problems [30–37].

In 2024, [38] proposed work on Single-Valued Neuromorphic Fuzzy Sombor Numbers, exploring a novel approach based on fuzzy logic to model trade flows between countries via sea routes, which incorporates uncertainty and complex decision-making factors. By integrating neuromorphic computing with fuzzy Sombor indices, the method optimizes high-utility trade transactions by analyzing profitability, shipping costs, and fluctuations in market demand. This approach improves traditional techniques (HUIM) by enabling more adaptive and precise decision making in global trade, logistics, and supply chain management.

Despite numerous advancements in HUIM algorithms, many suffer from limited scalability and excessive computational overhead in discrete solution spaces. This study addresses these gaps by introducing an enhanced Cuckoo Search algorithm tailored for HUIM.

3. Problem definition

HUIM is an extension to the classical FIM problem, and it is a more advanced data mining problem because it focuses on identifying sets of items that occur frequently and have a high utility, where utility is often defined as profit, cost, weight, etc. The HUIM problem, given a transaction database where each transaction includes the item with its quantity and the utility (e.g., the profit per unit), is to identify the item sets that contribute the most to the overall utility of the database. In this section, a mathematical interpretation of HUIM is provided.

A quantitative transaction dataset, D, has transactions R. A set of transactions with unique identifiers is D. The Transactional Data Set D has seven transactions with unique identifiers in Table 1. Transactions may reveal customer buying histories or user behavior. Let $K = K_1, K_2, ..., K_m$ be a finite set of unique items. Quantitative items M are in R_i transactions. According to Table 2, each item $M_N \in K$ has an external utility value, $U_E(i_j)$, representing its sales revenue or weight. In each transaction, U_E, each item $M_N \in K$ has an internal utility value, $U_I(M_N)$, representing its sale amount or frequency. A k-itemset has k different objects. If X is a subset $R_i's$, R_j supports K.

Table 1. Transaction database

Tran ID	Quantity (Qty)
$R1$	$(PP, 1), (QQ, 3), (TT, 2)$
$R2$	$(RR, 4), (SS, 5)$
$R3$	$(TT, 2), (UU, 3)$
$R4$	$(PP, 5), (QQ, 4), (RR, 2), (SS, 2), (TT, 4)$
$R5$	$(QQ, 5), (QR, 4), (TT, 6), (UU, 1)$
$R6$	$(QQ, 1), (RR, 3), (TT, 6)$
$R7$	$(PP, 2), (TT, 3), (UU, 4)$

To further comprehend the introductory ideas, we shall study Table 1's sample dataset D. There are seven transactions: $R1$, $R2$, $R3$, $R4$, $R5$, $R6$, and $R7$. The symbols PP, QQ, RR, SS, TT, and UU represent six elements in D. Table 2 shows positive unit earnings from these products. Every transaction involves selling goods. In transaction $R1$ in database D, items PP, QQ, and TT were purchased for 1, 3, and 2, respectively.

Table 2. Profits associated with items

Item	PP	QQ	RR	SS	TT	UU
Profit/Unit	006	004	002	005	003	009

The item x, which is a utility item $x \in R_1$, can be computed using equation (1).

$$UU(PP, R1) = [UU]_I(AA, R1)[UU]_E(AA) = 01 * 06 = 06 \tag{1}$$

For any itemset $K \subseteq R_1$, $UU(K) \in R_1$, the utility can be computed using equation (2).

$$U(PP, QQ, R1) = UU(PP, R1)UU(QQ, R1) = 012 + 06 = 018 \tag{2}$$

The entire utility of item K of the D data set can be computed using equation (3)

$$UU(\{PP, QQ\}, DD) = (\{PP, QQ\}, R1) + (\{PP, QQ\}, R4) = 0.18 + 0.46 = 0.64 \tag{3}$$

The utility of a transaction R_i existing in the dataset D can be computed using equation (4).

$$U_T(R3) = UU(T, R3) + UU(UU, R3) = 06 + 027 = 033 \tag{4}$$

HUIs exceed a user-selected minimum utility threshold. HUIS are extracted from transactional datasets to locate all HUIS that match minimal utility criteria. Table 3 shows the HUI of the dataset with a minimum utility threshold of 70 from dataset *D*.

Table 3. Fixing utility values to the item set

Itemset	*FF*	*AA*, *BB*	*AA*, *EE*	*BB*, *EE*	*EE*, *FF*	*AA*, *BB*, *EE*	*BB*, *CC*, *EE*	*AA*, *BB*, *CC*, *DD*, *EE*
Utility	072	072	087	0102	0105	090	0106	072

4. Proposed method

4.1 *Conventional cuckoo search algorithm*

The parasitism of cougars in fawns inspired the cougar search algorithm [23]. In other birds' nests, cuckoos raise their young. As an evolutionary optimizing method, cuckoo behavior evolved. Levy Flights Random Walk (LFRW) and Biased Random Walks (BSRW) help computer scientists identify optimal solutions in huge search spaces. Cuckoo's Levy Flight distribution for random walks:

$$X \sim \text{Levy}(\beta) = \frac{\phi \times \mu}{|v|^{1/\beta}} \tag{5}$$

where β refers to the stability index v and μ in the range (0, 1). The parameter gamma function is an integral part of the stability index, contributing to stabilising the search process in the context of the proposed optimization method. The Levy flight distribution in expression (5) allows a mix of short-range and long-range movements, mimicking the natural foraging behavior of cuckoo birds. This power-law step size distribution helps avoid local optima by allowing occasional long jumps, making the search process more efficient in high-dimensional spaces. Physically, Levy flights are observed in animal movement, human travel, and financial market fluctuations, which makes them well suited for optimizing high-utility itemset mining by striking a balance between exploration and exploitation. A mathematical and contextual description (Eq. (6)) has been added to illustrate its functionality in balancing exploration and exploitation. The parameter ϕ is expressed as

$$\phi = \left[\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}} \right]^{\frac{1}{\beta}} \tag{6}$$

Where Γ specifies a gamma function.

The cuckoo search algorithm generates random solutions using the Levy flying distribution, improving exploration. This is designed as

$$\text{LFRW}_i = I_i + \alpha \oplus \text{Levy}(\beta) \tag{7}$$

The notation $\alpha > 0$ indicates the step size magnitude. Adjust the step size α to optimise search space exploration. To maximize solution use, CS searches for the best solution. The derivation of equation (8) begins with initializing a

population of candidate solutions, each representing a potential answer. The algorithm employs the Levy Flight Random Walk (LFRW) to explore the solution space by generating steps with large variability, calculated using a random value scaled by parameters such as the Levy distribution exponent. The updated solution is expressed as a solution in Eq. (8), where α controls the step size. A greedy strategy refines these solutions by retaining only those improving fitness levels. To overcome local optima and ensure robust exploration, Biased Random Walk (BFRW) is applied using probabilistic weights to diversify the search. These steps are executed iteratively, balancing exploration and exploitation, leading to effective algorithm convergence. This appears as

$$\mathrm{LFRW}_i = I_i + \alpha \oplus \mathrm{Levy}(\beta) \times (I_i - I_{\mathrm{best}}) \tag{8}$$

The greedy strategy preserves the proper solution within the range of I_i and LFRW_i after applying the LFRW technique. After performing LFRW on every solution, the BSRW method is used for each person to explore the search space and find the best solution. Since greedy techniques tend to get caught in local optima, the BSRW algorithm explores the search space. BFRW improves on the solutions generated by LFRW, with probabilistic parameters (ρ_a) playing a key role in improving population diversity. This illustrates how the refinement process balances exploration and exploitation, leading to more effective convergence toward optimal solutions. BSRW can be computed using the following equation.

$$\mathrm{BFRW}_i = I_i + \mathrm{rand} \times (I_q - I_p) \tag{9}$$

Consider q and p as different random people with $p \neq$ and q. BFRW uses the probability fraction method ρ_a, unlike LFRW, which uses a greedy strategy to incorporate LFRW-generated solutions into the main population. ρ_a is a constant factor between 0 and 1. To substitute the main population response, use the probability fraction approach ρ_a.

$$I_{i,\,j} = \begin{cases} \mathrm{BFRW}_{i,\,j}, & \text{if rand} > \rho_a \\ I_{i,\,j}, & \text{otherwise} \end{cases} \tag{10}$$

where i represents the current individual and j corresponds to the dimension of every solution.

4.2 *Improved discrete cuckoo algorithm*

Traditional computer science algorithms show a protective set of exploration, indicating prejudice. Exploration and exploitation must be balanced to derive road regions from high-resolution satellite images. A balance will identify the appropriate threshold values for each segmentation class. Exploration is prioritised over overexploitation in computation because the BFRW section manages exploration. Road extraction from satellite pictures requires this. Instead of searching for the best answer, we extend the LFRW method's solution space search to boost computer science achievement. Our search approach uses the best solutions from each iteration to find the answer. Represent it as

$$\mathrm{LFRW}_i = \begin{cases} I_i + \alpha \oplus \mathrm{Levy}(\beta) \times (I_i - I_{\mathrm{best}}), & \mathrm{rand} > \mathrm{rand} \\ I_i + \alpha \oplus \mathrm{Levy}(\beta) \times (I_i - I_{G,\,\mathrm{best}}), & \text{otherwise} \end{cases} \tag{11}$$

where rand is the random variable, whose value is set between 0 and 1, and the generation number is G. The pseudo-code of the Extended and Improved Algorithm, which is based on the discrete cuckoo Algorithm, is shown below:

Algorithm 1 Extended and Improved Algorithm (Based on the Discrete Cuckoo Algorithm)
Data: Population size N, maximum number of generations G, stopping condition
Result: Optimized solution

Let Pop_G be a population containing individuals such as $I_{1,G}$ to $I_{N,G}$;
Calculate the fitness of Pop_G: Denoted as $f(\text{Pop\_}G)$;
Let I_{best} be the current solution ;
Let $I_{G,\text{best}}$ be the best solution in the g^{th} iteration ;
while *stopping condition*
  for *each I in Pop_G*
    Compute $LFRW_i$ {Equation 8} ;
    Compute fitness of $LFRW_i$: $f(LFRW_i)$;
    Select the best of $LFRW_i$ and I_i ;
  end
  for *each I in Pop_G*
    Compute $BFRW_i$ {Equation 9} ;
    Compute fitness of $BFRW_i$: $f(BFRW_i)$;
    Select the best of $BFRW_i$ and I_i ;
  end
  Update $I_{G,\text{best}}$ as the individual with the minimum fitness in Pop_G ;
  if $f(I_{G,best}) < f(I_{best})$ **then**
    Update I_{best} with $I_{G,\text{best}}$;
  end
  Increment G by 1 ;
end
Output: $I_{G,\text{best}}$.

The above algorithm generates a population of possible solutions (itemsets) and evaluates their utility using a fitness function. We design the algorithm based on two mechanisms. LFRW enables candidate solutions to be explored broadly, making large, random jumps to avoid local maxima traps, whereas BFRW makes less drastic adjustments to explore promising candidate solutions. After assessing these new solutions, the best candidates are selected to replace the weaker ones, gradually improving the set of items.

With each iteration of the algorithm, the best solution of the current generation is updated and kept if it is better than the last known best solution. Using this dynamic update mechanism, the algorithm converges to an increasingly valuable set of items over time. Thus, the proposed algorithm excels at identifying profitable and superior patterns, and combining these results is particularly beneficial in transactional data sets. This is achieved by combining the best of both worlds: the algorithm explores multiple branches based on the nature-inspired randomness inherent to Genetic Algorithms while selecting only optimal paths through adaptive-based selection.

5. Results and discussions

5.1 *Experimental setup*

The improved algorithm runs on MATLAB 2020 on a computer with a 4.2 GHz Intel Core i7 CPU, 16 GB of primary memory, and Windows 11. Table 4 shows the parameters of the proposed algorithm.

Table 4. Parametric assignments

Parameters	Values
Population size	0100
Runs size	010
Number of iterations/Run	0500

5.2 *Evaluation model*

The proposed method is compared to three models: HUIM-Hybrid Concept (HC), Simulated Annealing (SA), and Adaptive Filtering (AF). Chess, Mushroom, Accidents, and Connect were used to test our method. The SPMF data mining library includes datasets [33]. Performance of runtime, HUI count, and convergence measurement.

5.3 *Results*

Table 5 shows the execution time of the chess data set of many HUI algorithms for different minimum utility threshold values. Current approaches take longer than HUIM-ICDS. Table 4 shows that HUIM-ICDS outperforms HUIM-HC by 67.3%, SA by 24.7%, and AF by 42.8%. Figure 1 demonstrates performance as the threshold changes.

Table 5. Performance analysis through execution times

Min. utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
27.5	085.580	037.750	050.250	033.590
27	084.140	035.940	044.070	029.080
26.5	086.770	035.150	049.030	027.050
26	084.520	036.490	046.410	031.450
25.5	083.100	038.710	052.580	029.760

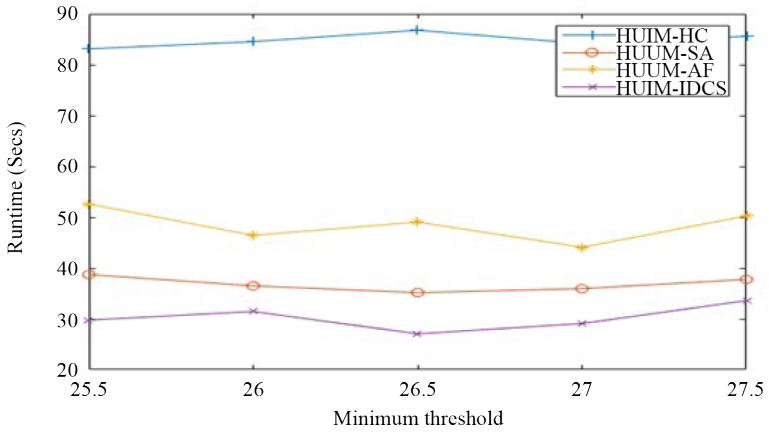

Figure 1. Runtime on dataset-Chess

Table 6 shows the execution time of many HUI algorithms in the Mushroom data set to vary the minimum utility threshold values. Compared to current algorithms, HUIM-ICDS performs better. In the mushroom data set, HUIM-ICDS outperforms HUIM-HC by 70.6%, HUIM-SA by 27.6%, and HUIM-AF by 20.7% in average execution time. Figure 2 shows a graph of the runtime of the algorithm.

Table 6. Performance analysis through execution times

Min. utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
27.5	085.580	037.750	050.250	033.590
27	084.140	035.940	044.070	029.080
26.5	086.770	035.150	049.030	027.050
26	084.520	036.490	046.410	031.450
25.5	083.100	038.710	052.580	029.760

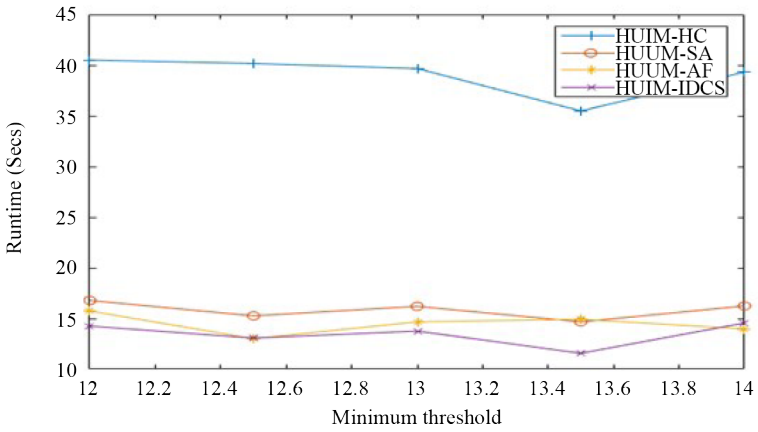

Figure 2. Performance analysis using dataset-Mushroom

Table 7 shows the execution times of the HUI algorithm in the accident dataset for different minimal utility threshold values. Compared to current algorithms, HUIM-ICDS performs better. In the average execution time of the accident dataset, HUIM-ICDS outperforms HUIM-HC by 71.9%, HUIM-SA by 20.3% and HUIM-AF by 26.04%. Figure 3 shows a runtime graph of the algorithm.

Table 7. Performance analysis using dataset-Accident

Min. utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
013.00	0732.76	0304.65	0327.54	0243.43
012.80	0800.02	0333.06	0281.79	0266.18
012.60	0770.39	0241.67	0226.24	0203.89
012.40	0766.54	0270.06	0331.66	0229.77
012.20	0771.43	0202.32	0287.84	0156.13

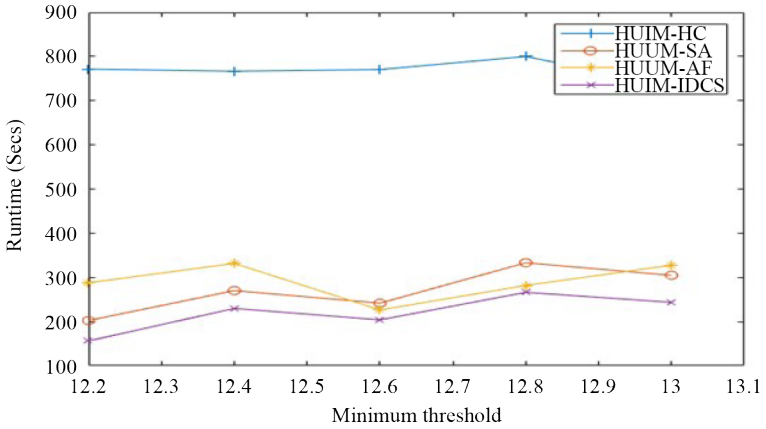

Figure 3. Performance analysis using dataset-Mushroom

Table 8 shows the performance of HUI Algorithms, considering different data sets and thresholds on utility values. As shown in the table, HUIM-ICDS outperformed other algorithms. Regarding the average execution time of the accident dataset, HUIM-ICDS outperforms HC by 69.8%, SA by 20.8%, and AF by 46.9%. The graph of the algorithm runtime is shown in Figure 4.

Table 8. Performance analysis considering dataset-Connect

Minimum utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
032.00	02781.160	0803.760	01420.130	0620.790
031.80	02441.670	0828.140	01357.530	0645.210
031.60	02477.800	01118.540	01402.520	0838.480
031.40	02056.890	0897.080	01338.700	0717.700
031.20	02314.260	0949.710	01333.120	0779.000

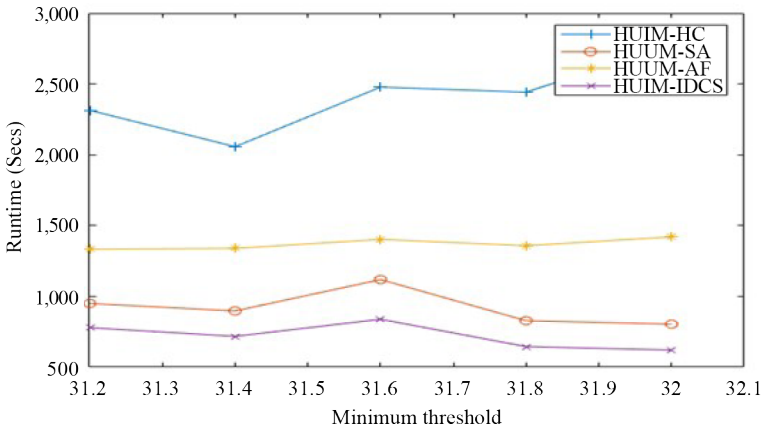

Figure 4. Performance analysis using dataset-Mushroom

5.4 *Discovered HUI's*

Table 9 presents the proportions of HUIS in the chess dataset identified by various HUI methods at different minimum utility threshold values. HUIM-ICDS improves the discovered HUIS compared to current methods. Regarding average HUI detection on the chess dataset, HUIM-ICDS outperforms HUIM-SA by 0.83%, HUIM-AF by 1.70%, and HUIM-HC by the same margin. Table 10 shows the fraction of the mushroom dataset HUIS identified using various HUI algorithms with different minimum utility threshold values. Compared to existing approaches, HUIM-ICDS yields a greater number of HUIS. HUIM-ICDS achieves 0.7% more HUIS than HUIM-SA in the mushroom data set and 7.55% more than HUIM-AF and HC. Table 11 presents the fraction of the accident dataset HUIS identified using various HUI methods at different levels of minimal utility threshold. HUIM-ICDS improves the discovered HUIS compared to previous approaches. Regarding the detection of average HUI in the accident data set, HUIM-ICDS outperforms HUIM-SA by 0.9%, HUIM-AF by 2.61% and HUIM-HC by the same margin. Table 12 compares the proportion of HUIS identified in the connected dataset using several HUI methods with different minimal utility thresholds. HUIM-ICDS improves the discovered HUIS compared to previous techniques. Regarding the detection of average HUI on the connected dataset, HUIM-ICDS outperforms HUIM-SA by 0.3%, HUIM-AF by 0.5% and HUIM-HC by the same margin.

Table 9. Computation of HUIs using Chess dataset

Minimum utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
027.50	0100.00	096.66	095.42	0100.00
027.00	0100.00	099.20	096.98	0100.00
026.50	0100.00	099.95	099.07	0100.00
026.00	0100.00	0100.00	0100.00	0100.00
025.50	0100.00	0100.00	0100.00	0100.00
Average	0100.00	099.16	098.29	0100.00

Table 10. Computation of HUIs using dataset-Mushroom

Minimum utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
014.00	0100.00	099.03	075.45	0100.00
013.50	0100.00	097.77	088.01	0100.00
013.00	0100.00	099.22	098.75	0100.00
012.50	0100.00	010.00	100.00	0100.00
012.00	0100.00	0100.00	100.00	0100.00
Average	0100.00	099.20	92.44	0100.00

Table 11. Computation of HUI's using dataset-Accident

Minimum utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
013.00	0100.00	096.74	094.30	0100.00
012.8	0100.00	099.21	095.00	0100.00
012.6	0100.00	099.26	098.95	0100.00
012.4	0100.00	0100.00	098.67	0100.00
012.2	0100.00	0100.00	0100.00	0100.00
Average	0100.00	099.04	097.39	0100.00

Table 12. Computation of HUIs using the dataset-Connect

Minimum utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
032.00	0100.00	098.85	098.77	0100.00
031.80	0100.00	099.29	098.55	0100.00
031.60	0100.00	0100.00	0100.00	0100.00
031.40	0100.00	0100.00	0100.00	0100.00
031.20	0100.00	0100.00	0100.00	0100.00
Average	0100.00	099.63	099.46	0100.00

5.5 *Convergence*

Figures 5, 6, 7, and 8 show the convergence graph for each iteration towards the ideal solution. The graphs have 50 iterations for the chess, mushroom, accident, and link datasets. The convergence graph displays the cumulative HUI count for each iteration time interval. The data show that the proposed model outperforms all others. Our model outperforms HUIM-HC and requires fewer iterations. The graphs show that this model performs similarly to higher-achieving models.

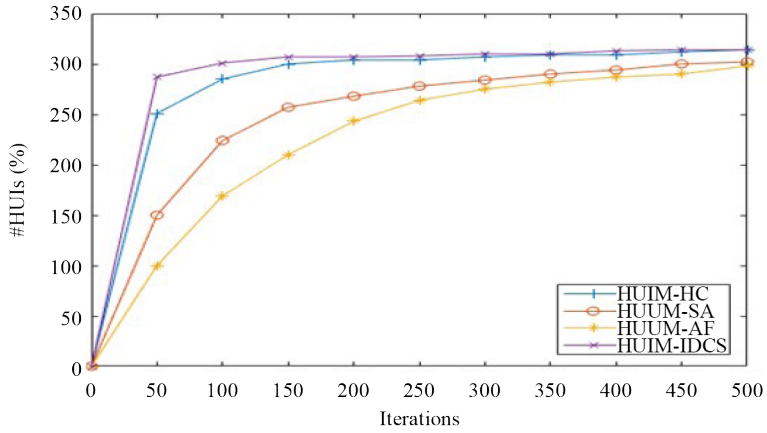

Figure 5. Convergence while discovering HUI on the dataset-Chess

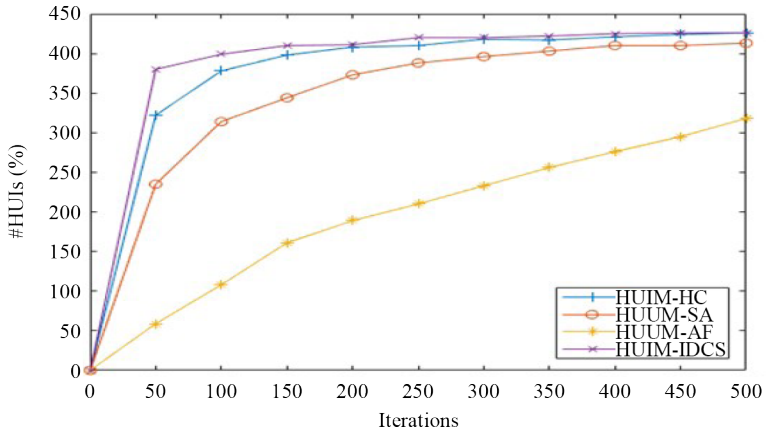

Figure 6. Convergence while discovering HUI on the Mushroom dataset

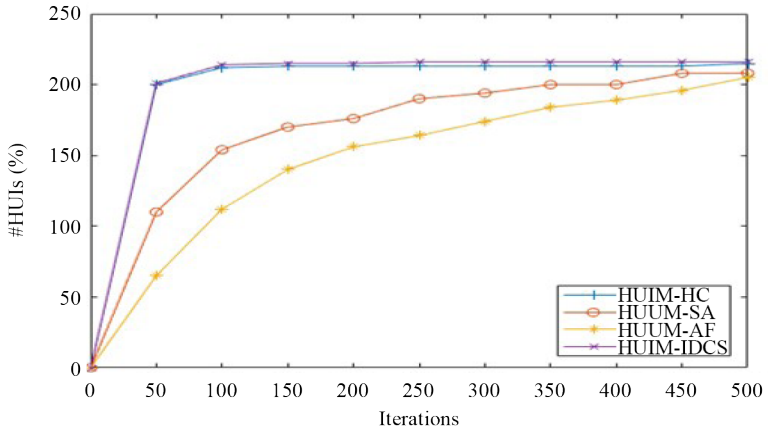

Figure 7. Convergence while discovering HUI on the Accident Data set

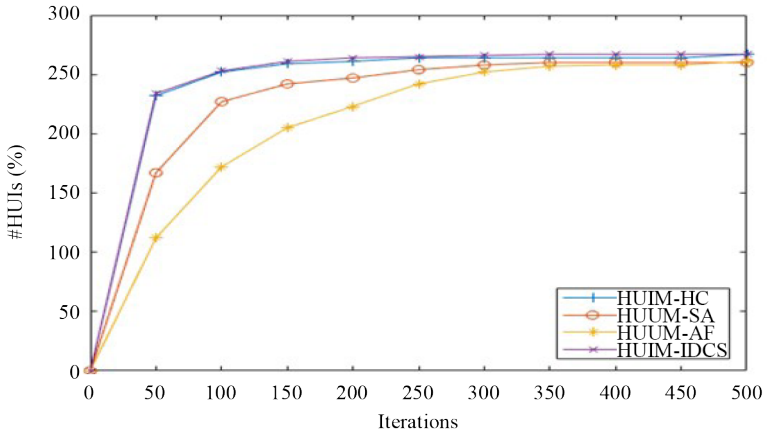

Figure 8. Convergence while discovering HUI on the Connect Data set

5.6 *Time and space complexity*

The time complexity of the existing method HUIM-HC holds a cap O open paren cap N close paren cap N, which is the number of iterations. HUIM-SA holds $O(N)$, where N is the number of iterations. HUIM-AF holds $O(M * D*)$re M is the number of ants; cap D is the dimensionality of the problem and N is the number of iterations. The proposed HUIM-IDCS holds re N is the number of iterations, P is the number of solutions, and D is the number of dimensions. In text, HUIM-HC holds $O(D)$, HUIM-SA holds $O(D)m$, HUIM-AF holds $O(M * D * D^2)$ and HUIMIDCS holds $O(N \times D)$, where D refers to the dimension of the solution space, M refers to the number of ants and N refers to the number of solutions in the population size.

6. Conclusions

The authors propose an enhanced version of the Discrete Cuckoo Search (DCS) to solve the HUI problem in a large dataset. This contribution is primarily due to the proposed model's use of Levy flights and oppositional learning, significantly reducing the complexity of determining whether to include the specified item in the HUI. This is evidenced by significant advancements in the results of time complexity, which are crucial to efficiently processing vast amounts of data. The number of HUIS and convergence performance were compared with those of other methods, such as HUIM-HC, HUIM-SA, and HUIM-AF, for the proposed model. The model is more accurate in detecting HUI than these methods and can be particularly useful for industries that perform accurate pattern recognition, such as retail analytics and supply chain optimization. Although these contributions are valuable, the proposed model has limitations, including that its performance is highly sensitive to parameter tuning, such as the discovery probability and the number of nests. The basic premise of the algorithm can remain unchanged, but care must be taken when selecting the parameters. Moreover, although the model performs better on large-scale transactions than conventional approaches, it does not consider scenarios involving very large data, which is common in big data, where the quality of item sets formed would lower the quality of correct identification. In addition, the sweep approach is designed for static data. It is ineffective when the database is frequently updated or the data is a constant stream of real-time data, which is a potential avenue for further improvement. After all, other options for HUI detection were more robust for handling noisy transactions, thus dealing with them more effectively than those used in state-of-the-art methods.

Author contributions

All authors contributed equally to this work.

Conflict of interest

There are no conflicts of interest for this study.

References

[1] Luna JM, Fournier-Viger P, Ventura S. Frequent itemset mining: A 25 years review. *WIREs Data Mining and Knowledge Discovery*. 2019; 9(6): e1329. Available from: https://doi.org/10.1002/widm.1329.

[2] Aggarwal CC, Han J. *Frequent Pattern Mining*. Springer Publishing Company; 2014.

[3] Fournier-Viger P, Lin JCW, Nkambou R, Vo B, Tseng VS. *High-Utility Pattern Mining: Theory, Algorithms and Applications*. Springer Publishing Company; 2019.

[4] Gan W, Lin JC, Zhang J, Fournier-Viger P, Chao H, Yu PS. Fast utility mining on sequence data. *IEEE Transactions on Cybernetics*. 2021; 51(2): 487-500. Available from: https://doi.org/10.1109/TCYB.2020.2970176.

[5] Fournier-Viger P, Wu C, Zida S, Tseng VS. FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In: *Proceedings of the 21st International Symposium on Foundations of Intelligent Systems*. Springer; 2014. p.83-92.

[6] Fournier-Viger P, Lin JCW, Nkambou R, Vo B, Tseng VS. *High-Utility Pattern Mining: Theory, Algorithms and Applications*. Springer Publishing Company; 2019.

[7] Gan W, Lin JC, Fournier-Viger P, Chao H, Yu PS. High-utility occupancy pattern mining. *IEEE Transactions on Cybernetics*. 2020; 50(3): 1195-1208. Available from: https://doi.org/10.1109/TCYB.2019.2896267.

[8] Liu Y, Liao W, Choudhary A. 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*. Springer; 2005. p.689-695.

[9] Ahmed CF, Tanbeer SK, Jeong B, Lee Y. Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*. 2009; 21(12): 1708-1721. Available from: https://doi.org/10.1109/TKDE.2009.46.

[10] Tseng VS, Shie B, Wu C, Yu PS. Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Transactions on Knowledge and Data Engineering*. 2013; 25(8): 1772-1786. Available from: https://doi.org/10.1109/TKDE.2012.59.

[11] Kannimuthu S, Premalatha K. Discovery of high utility itemsets using genetic algorithm with ranked mutation. *Applied Artificial Intelligence*. 2014; 28(4): 337-359. Available from: https://doi.org/10.1080/08839514.2014.891839.

[12] Lin JC, Yang L, Fournier-Viger P, Hong T, Voznák M. A binary PSO approach to mine high-utility itemsets. *Soft Computing*. 2017; 21(17): 5103-5121. Available from: https://doi.org/10.1007/s00500-016-2106-1.

[13] Liu Y, Liao WK, Choudhary A. A two-phase algorithm for fast discovery of high utility itemsets. In: *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer; 2005. p.689-695.

[14] Tseng VS, Wu CW, Shie BE, Yu PS. UP-growth: An efficient algorithm for high utility itemset mining. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2010. p.253-262. Available from: https://doi.org/10.1145/1835804.1835839.

[15] Qu JF, Fournier-Viger P, Liu M, Hang B, Hu C. Mining high utility itemsets using prefix trees and utility vectors. *IEEE Transactions on Knowledge and Data Engineering*. 2023; 35(10): 10224-10236. Available from: https://doi.org/10.1109/TKDE.2023.3256126.

[16] Liu M, Qu J. Mining high utility itemsets without candidate generation. In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. New York: Association for Computing Machinery; 2012. p.55-64 Available from: https://doi.org/10.1145/2396761.2396773.

[17] Fournier-Viger P, Wu CW, Zida S, Tseng VS. FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. *Lecture Notes in Computer Science*. 2014; 8502: 83-92. Available from: https://doi.org/10.1007/978-3-319-08326-1_9.

[18] Krishnamoorthy S. Pruning strategies for mining high utility itemsets. *Expert Systems with Applications*. 2015; 42(5): 2371-2381. Available from: https://doi.org/10.1016/j.eswa.2014.11.001.

[19] Duong QHH, Fournier-Viger P, Ramampiaro H, Nørvåg K, Dam TLL. Efficient high utility itemset mining using buffered utility-lists. *Applied Intelligence*. 2018; 48(7): 1859-1877. Available from: https://doi.org/10.1007/s10489-017-1057-2.

[20] Wu P, Niu X, Fournier-Viger P, Huang C, Wang B. UBP-Miner: An efficient bit based high utility itemset mining algorithm. *Knowledge-Based Systems*. 2022; 248: 108865. Available from: https://doi.org/10.1016/j.knosys.2022.108865.

[21] Zida S, Fournier-Viger P, Lin JCW, Wu CW, Tseng VS. EFIM: A fast and memory efficient algorithm for high-utility itemset mining. *Knowledge and Information Systems*. 2017; 51(2): 595-625. Available from: https://doi.org/10.1007/s10115-016-0986-0.

[22] Nguyen LTT, Nguyen P, Nguyen TDD, Vo B, Fournier-Viger P, Tseng VS. Mining high-utility itemsets in dynamic profit databases. *Knowledge-Based Systems*. 2019; 175: 130-144. Available from: https://doi.org/10.1016/j.knosys.2019.03.022.

[23] Lan W, Lin JCW, Fournier-Viger P, Chao HC, Tseng VS, Yu PS. A survey of Utility-Oriented pattern mining. *IEEE Transactions on Knowledge and Data Engineering*. 2021; 33(4): 1306-1327. Available from: https://doi.org/10.1109/TKDE.2019.2942594.

[24] Vo B, Nguyen LTT, Bui N, Nguyen TDD, Huynh VN, Hong TP. An efficient method for mining closed potential High-Utility itemsets. *IEEE Access*. 2020; 8: 31813-31822. Available from: https://doi.org/10.1109/ACCESS.2020.2974104.

[25] Duong H, Hoang T, Tran T, Truong T, Le B, Fournier-Viger P. Efficient algorithms for mining closed and maximal high utility itemsets. *Knowledge-Based Systems*. 2022; 257: 109921. Available from: https://doi.org/10.1016/j.knosys.2022.109921.

[26] Ahmed U, Lin JCW, Srivastava G, Yasin R, Djenouri Y. An evolutionary model to mine high expected utility patterns from uncertain databases. *IEEE Transactions on Emerging Topics in Computational Intelligence*. 2021; 5(1): 19-28. Available from: https://doi.org/10.1109/TETCI.2020.3000224.

[27] Tung NT, Nguyen LTT, Nguyen TDD, Fournier-Viger P, Nguyen NT, Vo B. Efficient mining of cross-level high-utility itemsets in taxonomy quantitative databases. *Information Sciences*. 2022; 587: 41-62. Available from: https://doi.org/10.1016/j.ins.2021.12.017.

[28] Srivastava G, Lin JCW, Pirouz M, Li Y, Yun U. A pre-large weighted fusion system of sensed high-utility patterns. *IEEE Sensors Journal*. 2021; 21(14): 15626-15634. Available from: https://doi.org/10.1109/JSEN.2020.2991045.

[29] Attuluri S, Ramesh M. Multi-objective discrete harmony search algorithm for privacy preservation in cloud data centers. *International Journal of Information Technology*. 2023; 15: 3983-3997. Available from: https://doi.org/10.1007/s41870-023-01462-w.

[30] Hazzazi MM, Attuluri S, Bassfar Z, Joshi K. A novel cipher-based data encryption with galois field theory. *Sensors*. 2023; 23(6): 3287. Available from: https://doi.org/10.3390/s23063287.

[31] Budaraju RR, Nagesh OS. Multi-level image thresholding using improvised cuckoo search optimization algorithm. In: *3rd International Conference on Intelligent Technologies*. Hubli, India: IEEE; 2023. p.1-7. Available from: https://doi.org/10.1109/CONIT59222.2023.10205744.

[32] Hazzazi MM, Budaraju RR, Bassfar Z, Albakri A, Mishra S. A finite state machine-based improved cryptographic technique. *Mathematics*. 2023; 11(10): 2225. Available from: https://doi.org/10.3390/math11102225.

[33] Thirugnansambandam K, Bhattacharyya D, Frnda J, Anguraj DK, Nedoma J. Augmented node placement model in t-WSN through multi objective approach. *Computers, Materials & Continua*. 2021; 69: 3629-3644. Available from: https://doi.org/10.32604/cmc.2021.018939.

[34] Thirugnansambandam K, Raghav RS, Anguraj DK, Saravanan D, Janakiraman S. Multi-objective binary reinforced cuckoo search algorithm for solving connected coverage target based WSN with critical targets. *Wireless Personal Communications*. 2021; 121(3): 2301-2325. Available from: https://doi.org/10.1007/s11277-021-08824-2.

[35] Thirugnansambandam K, Ramalingam R, Mohan D, Rashid M, Juneja K, Alshamrani SS. Patron-prophet artificial bee colony approach for solving numerical continuous optimization problems. *Axioms*. 2022; 11: 523. Available from: https://doi.org/10.3390/axioms11100523.

[36] Thirugnansambandam K, Rajeswari M, Bhattacharyya D, et al. Directed Artificial Bee Colony algorithm with revamped search strategy to solve global numerical optimization problems. *Automated Software Engineering*. 2022; 29(13). Available from: https://doi.org/10.1007/s10515-021-00306-w.

[37] Raghav RS, Thirugnansambandam K, Varadarajan V, Vairavasundaram S, Ravi L. Artificial bee colony reinforced extended kalman filter localization algorithm in internet of things with big data blending technique for finding the accurate position of reference nodes. *Big Data*. 2022; 10(3): 186-203. Available from: https://doi.org/10.1089/big.2020.0203.

[38] Anwar S, Azeem M, Jamil MK, Almohsen B, Shang Y. Single-valued neutrosophic fuzzy Sombor numbers and their applications in trade flows between different countries via sea route. *Journal of Supercomputing*. 2024; 80: 19976-20019. Available from: https://doi.org/10.1007/s11227-024-06169-8.