International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Improvised Swarm Based Discrete Data Mining Approach for High Utility Item Sets

Raja Rao Budaraju[1], Sastry Kodanda Rama Jammalamadaka[2*]

Submitted: 03/02/2024 **Revised**: 11/03/2024 **Accepted**: 17/03/2024

Abstract: Data mining techniques uncover valuable patterns hidden inside extensive databases to assist decision support systems in different practical applications. Association rule mining analyzes the transaction database to recognize patterns and provides insights into client behavior. Frequent itemset mining (FIM) detects a group of items that are commonly purchased together. A significant limitation of FIM is its disregard for the item's significance. The significance of an item is crucial in a practical application. Hence, it is imperative to identify the critical set of items that yields substantial profits, referred to as the HUIM (High-Utility Itemset Mining) problem. Various techniques may be employed to identify high utility itemsets from a transaction database. The HUIM approaches that employ the Utility list are relatively new and exhibit superior performance in terms of memory consumption and execution time. Primary constraint of these algorithms is the execution of expensive utility list join operations. This work presents a highly efficient optimization approach based on swarm intelligence for addressing the HUIM problem. Furthermore, the suggested method's execution time is assessed. Additionally, it is compared to relevant and advanced current approaches. Extensive tests were carried out on accessible benchmark datasets demonstrate that the suggested swarm-based methodology outperforms state-of-the-art approaches.

Keywords: *High Utility Itemset, discrete, Improved discrete cuckoo search*

1. Introduction

Data analysis systems are seeing significant growth due to their ability to uncover valuable insights that are concealed inside data. Frequent Itemset Mining (FIM) is a crucial activity in data analysis or data mining [1]. The process pulls often recurring events, patterns, or objects (either individual or grouped) from the data [2]. High Utility Itemset Mining (HUIM) [3] is a developing area of study that aims to address a clear drawback of FIM [1], which is the assumption that all patterns in a database are equally significant. The objective of HUIM is to identify patterns that have a significant utility in relation to their value to the user [4]. The creation of the utility function can stem from diverse criteria, yet its central aim remains quantifying the profit derived from retail product sales [5]. HUIM is often viewed as an extension of the FIM task, wherein each element within the input data receives a value denoting its significance or relevance to the particular problem at hand. An important feature of HUIM is its capacity to accommodate multiple instances of an item within a single transaction, whereas FIM solely displays the presence or absence of each item in each transaction.

The initial HUIM techniques [6] were released in the early 2000s with the goal of generating significant and practical findings. Subsequently, the problem has been widely explored in other application domains, including market basket analysis [5] and sentiment analysis [6], among others. At present, HUIM is a very active field of study that encompasses several effective algorithms [7, 8] for listing all itemsets that produce a benefit greater than a threshold established by the user. Several methods [8] utilize a way of generating and testing candidates in a level-wise manner, while others [9, 10] rely on pattern growth. Nevertheless, these specific HUIM algorithms may have significant delays when operating on extensive search areas. To be more explicit, the performance of accurate techniques tends to decrease as the amount of input data rises, particularly in terms of the number of transactions and the total number of distinct objects [6]. Consequently, performance becomes unsatisfactory for those who cannot tolerate such lengthy waiting times to acquire results. The objective for utilizing bio-inspired optimization methodologies, such as genetic algorithms [11] and particle swarm optimization [12], is to address the performance bottleneck in precise HUIM methods. In this research work, a swarm based optimization algorithm namely Improved Discrete Cuckoo Search algorithm (ICDS) is discretized to solve HUIM. The key contributions of the proposed work are as follows:

- The conventional Cuckoo Search algorithm has been adapted to address HUI problems that involve the representation of solution space in binary form.

[1]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur 522302, Andhra Pradesh, India.*
Email: rajaraob@yahoo.com
[2]*Department of Electronics and Computer Science, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur 522302, Andhra Pradesh, India.*
Email: drsastry@kluniversity.in
**(Corresponding Author)*

- The proposed model is evaluated with runtime, convergence towards optimal solution and identified HUI.
- The proposed model is being compared against contemporary existing methodologies in order to demonstrate its significance.

The following is the organizational scheme of the paper's following sections: Section 2 focuses on the previous research conducted on HUIM, specifically examining both exhaustive and evolutionary algorithms. Section 3 provides an explanation of the HUIM characteristics, and its representation in terms of mathematical notations. Section 4 outlines the suggested paradigm, namely discrete swarm optimization method. Section 5 outlines the empirical assessment of the suggested model. Section 6 provides the final conclusions of the work, and future perspectives.

2. Related Works

After the introduction of HUIM by Yao and Hamilton, numerous refinements have been introduced to enhance its mining efficiency. In their work, Liu et al. [13] devised the Two-Phase technique, which introduces the Transaction Weight Utility (TWU) as an innovative upper limit adhering to the downward closure property. This facilitates the elimination of unproductive candidates from the search space and ensures the comprehensive extraction of HUI.

However, the method described above requires extensive database examinations for all HUIs, and although TWU serves as a dependable upper limit, it lacks precision, resulting in prolonged processing times. Le et al. [14] introduced the DTWU-Mining approaches, which employ the diffset technique to diminish the number of candidates and database searches. This ensures that search time and memory usage are minimized through the utilization of this technology. The UPGrowth algorithm, developed by Tseng et al., [15], is derived from the FP-Growth method. In 2023, Qu et al., [16] introduced the Hamm method to improve the HUIM process in tree mining. They also proposed a new structure called TV that allows for the sequential mining of item sets with high utilization, eliminating the requirement for candidate creation.

Liu et al. [17] presented the HUIMiner approach, which applies a more stringent upper limit known as the residual utility and uses the Utility-List data format, as compared to TWU. The process of generating several candidates is now obsolete due to the implementation of the Utility-List structure. Because of this structure, the algorithm may do a single database search and merge the Utility-List of itemsets to form larger itemsets. Authors FournierViger et al. [18] introduced the FHM algorithm as an improvement to the HUI-Miner algorithm. The FHM algorithm achieves better performance by reducing the number of itemsets generated by two items. This reduction is accomplished via Estimated Utility Co-occurrence Pruning (EUCP). Author Krishnamoorthy [19] proposed the HUP-Miner technology as an enhancement to the FHM algorithm. This technique suggests more effective pruning strategies to further limit the search area. The ULB-Miner technique, introduced by Q.-H. H. Duong et al. [20], utilizes a utility-list buffer structure to decrease execution time and memory usage. In 2017, Krishnamoorthy introduced the HMiner algorithm, which incorporates multiple pruning techniques (TWU-prune, EUCS-prune, Uprune, LA-prune, and C-prune) along with a proposed approach for consolidating identical transactions. This strategy reduces runtime by consolidating "closed" transactions into a single repository. Additionally, the UBP-Miner method, developed by Wu et al. [21], aims to enhance the List-based approach by reducing the frequency of transaction scans. Benchmark assessments show that the UBP-Miner algorithm outperforms both the HUIMiner and ULB-Miner algorithms.

Moreover, alongside utility-list and tree-based techniques, the algorithms also manage transactions that exhibit partial similarity. EFIM stands out as one of the pioneering algorithms to adopt this approach, introduced by Zida et al. [22]. This algorithm employs a method to amalgamate similar transactions and rearrange objects within them, thereby facilitating their projection and merging to reduce memory usage. Additionally, the technique employs two precise and stringent upper bounds: the sub-tree and the local utility.

Yet, EFIM encounters an obstacle concerning the high costs incurred during database scanning. To address this issue, Nguyen et al. [23] introduced the iMEFIM method, which utilizes the P-Set framework to accurately record the positions of necessary transactions. This algorithm incorporates a mechanism to handle databases where each transaction has varying levels of significance.

In addition to the traditional algorithms for HUIM, there are various related problems that are commonly encountered in real-world scenarios. These include 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]. Conversely, deep learning models and soft computing have advanced significantly by resolving numerous issues across various domains [30–38].

3. Problem Definition

Let us consider a quantitative transaction dataset, denoted as $\mathbb{D}$, which is comprised of a collection of transactions. Specifically, $\mathbb{D}$ may be represented as a set of transactions, where each transaction is labeled with a unique identifier i. As an example, Table 1 illustrates a transactional dataset $\mathbb{D}$ that consists of seven transactions, each identified by a unique identifier. These transactions might reflect the purchasing patterns of clients or the specific behaviors of users. Let $I = \{I_1, I_2, \ldots, I_m\}$ exist in a limited number of distinct elements. Transaction Tr_i consists of m quantifiable items. Each item $i_j \in I$ has a value that is external, $U_E(i_j)$, which represents a product's weight or earnings from sales as indicated in Table 2. Additionally, each item $i_j \in I$ has an internal utility value, $U_I(i_j)$, which represents the amount of sales it makes or the number of times it is sold in each transaction $U_E(i_j)$. An itemset X can be conceptualized as a collection of k unique items and is commonly denoted as a k-itemset. A transaction denoted as Tr_i can be classified as a supporting transaction for X iff X is a subset of Tr_i.

Table 1. Transaction Database

Transaction ID	**Quantity**
Tr_1	(P, 1), (Q, 3), (T, 2)
Tr_2	(R, 4), (S, 5)
Tr_3	(T, 2), (U, 3)
Tr_4	(P, 5), (Q, 4), (R, 2), (S, 2), (T, 4)
Tr_5	(Q, 5), (R, 4), (T, 6), (U, 1)
Tr_6	(Q, 1), (R, 3), (T, 6)
Tr_7	(P, 2), (T, 3), (U, 4)

In order to gain a deeper understanding of the aforementioned introductory concepts, we will now examine the sample dataset $\mathbb{D}$ as presented in Table 1 There are a set of seven transactions denoted as $Tr_1, Tr_2, Tr_3, Tr_4, Tr_5, Tr_6$ and Tr_7. Moreover, the set $\mathbb{D}$ comprises six distinct elements denoted by the symbols P, Q, R, S, T, U. Table 2 presents the collection of positive unit earnings that are linked to the sale of these products. In essence, every transaction signifies the act of selling particular commodities. An illustration may be provided by transaction Tr_1 in database $\mathbb{D}$, which indicates that items P, Q *and* T were acquired in this particular transaction, with amounts of 1, 3, and 2, respectively.

Table 2. Items with Unit Profit

Item	**P**	**Q**	**R**	**S**	**T**	**U**
Profit/Unit	6	4	2	5	3	9

The utility of item $x \in Tr_i$ is computed as,

$$U(P, Tr_1) = U_I(A, Tr_1) \times U_E(A) = 1 \times 6 = 6$$

For an itemset $X \subseteq Tr_i$, $U(X)$ in Tr_i can be defined as,

$$U(\{P, Q\}, Tr_1) = U(P, Tr_1) \times U(Q, Tr_1) = 12 + 6 = 18$$

The total utility of an itemset X inside the dataset $\mathbb{D}$ is formally defined as

$$U(\{P, Q\}, \mathbb{D}) = (\{P, Q\}, Tr_1) + (\{P, Q\}, Tr_4) = 18 + 46 \\ = 64$$

The transaction utility of any transaction Tr_i in the set $\mathbb{D}$ may be computed as

$$U_T(Tr_3) = U(T, Tr_3) + U(U, Tr_3) = 6 + 27 = 33$$

HUI is used to describe an itemset that has a utility value that is either equal to or much higher than a minimum utility threshold (minUtil) that the user has chosen. The process of extracting HUI from transactional datasets has as its primary goal the identification of all HUIs that satisfy the minimal utility threshold that has been determined. With a minimum utility criterion of 70, the high-utility item sets that were derived from the dataset $\mathbb{D}$ that was provided are created and presented in the following manner:

Table 3. Itemset with Utility Value

Itemset	F	A, B	A, E	B, E	E, F	A, B, E	B, C, E	A, B, C, D, E
Utility	72	72	87	102	105	90	106	72

4. Proposed Model

The Cuckoo Search Algorithm [23] is derived from the brood parasite behavior of the cuckoo bird. Cuckoo birds deposit their eggs in the nests of other birds in order to rear their offspring. This behavior is derived from the cuckoo and has been formulated as an evolutionary method for optimization purposes. Computer Science employs two distinct methodologies for exploring optimal solutions in a vast search space: an approach to random walks that makes use of Levy flights (LFRW) and a method known as biased random walk (BSRW). It has been determined that the following is the formulation of the Levy Flight distribution, which the cuckoo use in order to do a random walk:

$$Levy(\beta) \sim \frac{\phi \times \mu}{|v|^{1/\beta}} \quad (2)$$

where random variables μ and v are both within the range of (0,1), whereas β represents the stability index. The formulation of ϕ can be expressed as

$$\phi = \left[\frac{\Gamma(1+\beta) \times sin\left(\frac{\pi \times \beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}}\right]^{1/\beta} \quad (3)$$

where Γ denotes the gamma function.

The levy flight distribution is utilized by the cuckoo search algorithm to provide random solutions, which aids in enhancing the exploration process. This has been devised as

$$LFRW_i = I_i + \alpha \oplus Levy(\beta) \quad (4)$$

The symbol $\alpha > 0$ denotes the magnitude of the step size. The step size α) can be adjusted to optimize the exploration of the search space.

In order to optimize the utilization of solutions, CS has been adjusted with a search mechanism that is based on the best solution achieved thus far. This is depicted as

$$LFRW_i = I_i + \alpha \oplus Levy(\beta) \times (I_i - I_{best}) \quad (5)$$

The LFRW technique is applied to I_i, and the best solution between I_i and $LFRW_i$ is preserved using the greedy method. After performing the $LFRW$ procedure on every solution in the population, the $BSRW$ method will be used to each person in order to explore the search space and get the optimal solution. Due to the tendency of the greedy technique to become stuck in local optima, the $BSRW$ algorithm is developed to explore the search space. The new solution is developed in $BSRW$ based on the following formulation.

$$BFRW_i = I_i + rand \times (I_q - I_p) \quad (6)$$

Let q and p be distinct random individuals, where | $p \neq q$. Unlike the $LFRW$ approach, which employs a greedy strategy to incorporate $LFRW$-generated solutions into the main population, $BFRW$ utilizes the probability fraction method ρ_a. ρ_a is a constant factor that is defined to be between 0 and 1. The answer in the main population can be substituted using the probability fraction approach ρ_a.

$$I_i = \begin{cases} BFRW_{i,j} & rand > \rho_a \\ I_{i,j} & otherwise \end{cases} \quad (7)$$

let "i" represent the current individual and "j" correspond to the dimension of each individual.

4.1. Improved Discrete Cuckoo Search Algorithm

Empirical results in traditional computer science algorithms demonstrate the presence of a defensive set of exploration that indicates a biased nature of the exploration. In order to extract road regions from high resolution satellite pictures, it is necessary to strike a balance between exploration and exploitation. This balance will help determine the best threshold values for each segmentation class. In computation, the process of exploration is managed in the BFRW section, leading to a larger emphasis on exploration rather than exploitation. This is crucial for effectively extracting roads from satellite photos. Therefore, to increase the likelihood of success in computer science, we integrate an enhanced form of search within the provided solution space, focusing on exploiting the potential of the LFRW method rather than only searching around the current best solution. Our proposed search approach involves searching for the answer based on both the overall best solution and the best solution found during each iteration. It can be expressed as

$$LFRW_i = \begin{cases} I_i + \alpha \oplus Levy(\beta) \times (I_i - I_{best}) & rand > rand \\ I_i + \alpha \oplus Levy(\beta) \times \left(I_i - I_{G,best}\right) & otherwise \end{cases} \quad (8)$$

In this context, $rand$ is a random variable that falls within the range of 0 to 1, and G is the generation number. The pseudo code for the Improved Discrete Cuckoo Search Algorithm has been included in Algorithm 2 as a result of this inclusion.

Algorithm: Discrete Cuckoo Search for HUIM

```
Initialize G to 1
Create Population at $G$, denoted as $Pop_G$, containing individuals: $I_{1,G}$ through $I_{N,G}$
Evaluate the fitness of $Pop_G$: $f(Pop_G)$
Obtain the best solution found so far: $I_{best}$
Obtain the best solution in iteration $G$: $I_{G,best}$

While the stopping condition is not met:
  For each individual $I$ in $Pop_G$:
    Generate $LFRW_i$ using Equation (8)
    Evaluate the fitness of $LFRW_i$: $f(LFRW_i)$
    Choose the best solution between $LFRW_i$ and $I_i$

  For each individual, $I$ in $Pop_G$:
    Generate $BFRW_i$ using Equation (7)
    Evaluate the fitness of $BFRW_i$: $f(BFRW_i)$
    Choose the best solution between $BFRW_i$ and $I_i$

  Update the best solution in iteration $G$: $I_{G,best}$ to be the minimum fitness in $Pop_G$
  If $I_{G,best}$ has better fitness than $I_{best}$:
    Update $I_{best}$ to be $I_{G,best}$

  Increment $G$ by 1
```

Output: $I_{G,best}$

5. Experimental Evaluation

5.1 Experimental Setup

The suggested algorithm is executed in MATLAB 2020 version on a computational device with an Intel Core i7 CPU running at 4.2 GHz, 16 GB of primary memory, and the Windows 11 operating system. The parameter configurations of the proposed algorithm are provided in Table 4.

Table 4. Parameter Settings

Parameters	Values
Size of the population	100
Total number of runs	10
Iterations / Run	500

5.2 Evaluation Model

The proposed algorithm is compared with seven existing models, notably HUIM-HC, HUIM-SA, and HUIM-AF. The suggested approach was evaluated using four real-life datasets: Chess, Mushroom, Accidents, and Connect datasets. The datasets are obtained from the SPMF data mining library [33]. The results are evaluated under the performance criteria's, including runtime, the number of identified HUIs, and convergence.

5.3 Runtime

Table 4 indicates the duration of execution for several HUI algorithms on the chess dataset, with respect to different minimal utility threshold values. When comparing the outcomes of the suggested model with the existing algorithms, HUIM-ICDS demonstrates a notable enhancement in terms of runtime.

Table 4. Execution Time on the Dataset-Chess

Min. utility threshold	**HUIM-HC**	**HUIM-SA**	**HUIM-AF**	**HUIM-IDCS**
27.5	85.58	37.75	50.25	33.59
27.0	84.14	35.94	44.07	29.08
26.5	86.77	35.15	49.03	27.05
26.0	84.52	36.49	46.41	31.45
25.5	83.10	38.71	52.58	29.76

On analyzing the average execution time on the chess dataset, HUIM-ICDS performs better than other approaches like HUIM-HC by 67.3%, HUIM-SA by 24.7%, and HUIM-AF by 42.8%. Figure 1 displays a graph that compares the runtime of the algorithms.

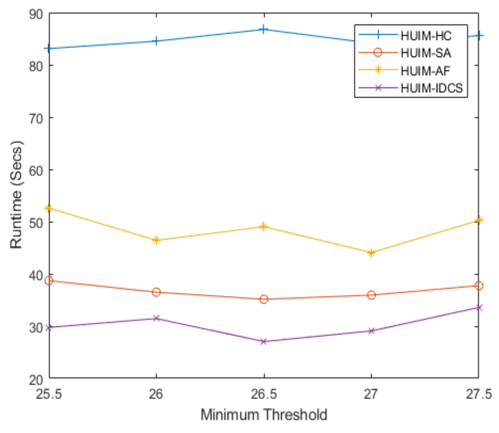

Fig. 1. Runtime on dataset - Chess

Table 5 depicts the duration of execution for several HUI algorithms on the Mushroom dataset, with respect to different minimal utility threshold values. When comparing the outcomes of the suggested model with the current algorithms, HUIM-ICDS demonstrates a noteworthy enhancement in terms of implementation.

Table 5. Execution time on the dataset-Mushroom

Minimum utility threshold	**HUIM-HC**	**HUIM-SA**	**HUIM-AF**	**HUIM-IDCS**
14.0	39.32	16.24	13.97	14.56
13.5	35.48	14.67	14.92	11.57
13.0	39.66	16.20	14.67	13.76
12.5	40.16	15.26	13.01	13.07
12.0	40.50	16.78	15.74	14.25

On analyzing the average execution time on the Mushroom dataset, HUIM-ICDS performs better than other approaches like HUIM-HC by 70.6%, HUIM-SA by 27.6%, and HUIM-AF by 20.7%. Figure 2 displays a graph that compares the runtime of the algorithms.

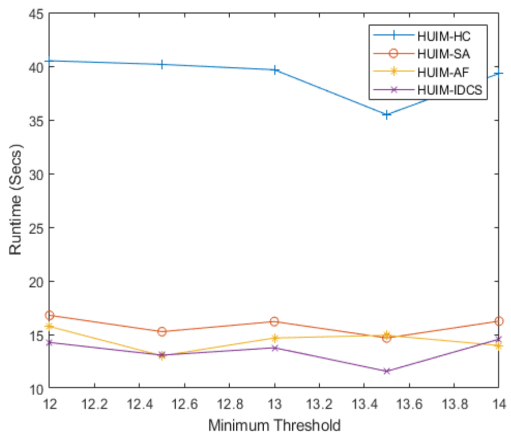

Fig. 2. Runtime on dataset - Mushroom

Table 6 displays the duration of execution on the accident dataset by various HUI algorithms with respect to different lowest utility threshold values. When comparing the outcomes of the suggested model with the current algorithms, HUIM-ICDS demonstrates a noteworthy enhancement in terms of implementation.

Table 6. Execution time on the dataset-Accident

Min. utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
13	732.76	304.65	327.54	243.43
12.8	800.02	333.06	281.79	266.18
12.6	770.39	241.67	226.24	203.89
12.4	766.54	270.06	331.66	229.77
12.2	771.43	202.32	287.84	156.13

On analyzing the average execution time on the accident dataset, HUIM-ICDS performs better than other approaches like HUIM-HC by 71.9%, HUIM-SA by 20.3%, and HUIM-AF by 26.04%. Figure 3 displays a graph that compares the runtime of the algorithms.

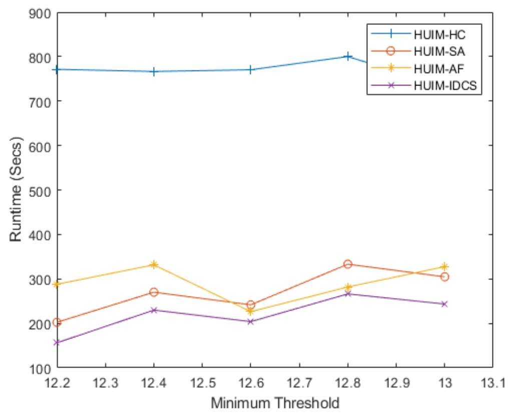

Fig. 3. Runtime on dataset - Accident

Table 7 depicts the duration of execution for several HUI algorithms on the connect dataset, with respect to different minimal utility threshold values. When comparing the outcomes of the suggested model with the existing algorithms, HUIM-ICDS demonstrates a notable enhancement in terms of implementation.

Table 7. Execution time on the dataset-Connect

Minimum utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
32.0	2781.16	803.76	1420.13	620.79
31.8	2441.67	828.14	1357.53	645.21
31.6	2477.80	1118.54	1402.52	838.48
31.4	2056.89	897.08	1338.70	717.70
31.2	2314.26	949.71	1333.12	779.00

On analyzing the average execution time on the accident dataset, HUIM-ICDS performs better than other approaches like HUIM-HC by 69.8%, HUIM-SA by 20.8%, and HUIM-AF by 46.9%. Figure 4 displays a graph that compares the runtime of the algorithms.

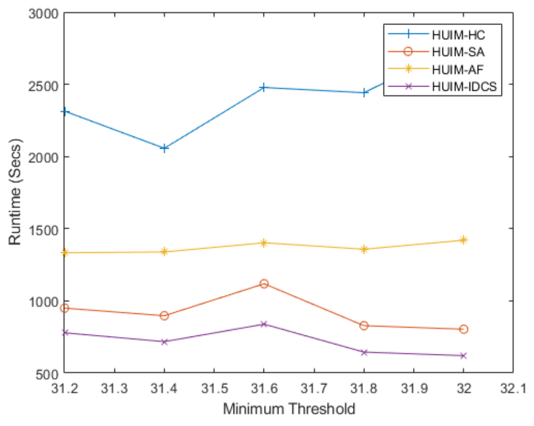

Fig. 4. Runtime on dataset – Connect

5.4 Discovered HUI's

Table 8 depicts the proportions of identified High Utility Items (HUIs) in the chess dataset using several HUI algorithms, with respect to varying minimum utility threshold values. When comparing the outcomes of the suggested model with the current techniques, HUIM-ICDS demonstrates a notable enhancement in terms of the Discovered HUI's.

Table 8. Number of HUI's on Dataset - Chess

Minimum utility threshold	HUIM-HC	HUIM-SA	HUIM-AF	HUIM-IDCS
27.5	100	96.66	95.42	100
27.0	100	99.20	96.98	100
26.5	100	99.95	99.07	100
26.0	100	100	100	100
25.5	100	100	100	100
Average	100	99.16	98.29	100

On examining the average percentage of detected HUI's on the chess dataset, HUIM-ICDS performs better than the current approaches, including HUIM-SA by 0.83%, HUIM-AF by 1.70% and competes equally with HUIM-HC.

Table 9 displays the proportion of identified High Utility Itemsets (HUIs) in the mushroom dataset using various HUI techniques, with respect to varying minimum utility threshold values. When comparing the outcomes of the suggested model with the existing methods, HUIM-ICDS demonstrates a notable enhancement in terms of the discovered HUI's.

Table 9. Number of HUI's on Dataset - Mushroom

Minimum utility threshold	**HUIM-HC**	**HUIM-SA**	**HUIM-AF**	**HUIM-IDCS**
14.0	100	99.03	75.45	100
13.5	100	97.77	88.01	100
13.0	100	99.22	98.75	100
12.5	100	100	100	100
12.0	100	100	100	100
Average	100	99.20	92.44	100

On examining the average percentage of detected HUI's on the mushroom dataset, HUIM-ICDS performs better than the current approaches, including HUIM-SA by 0.7%, HUIM-AF by 7.55% and competes equally with HUIM-HC.

Table 10 displays the proportion of identified HUIs in the accident dataset using several HUI algorithms, with respect to varying minimal utility threshold values. When comparing the outcomes of the suggested model with the existing methods, HUIM-ICDS demonstrates a notable enhancement in terms of the Discovered HUI's.

Table 10. Number of HUI's on Dataset - Accident

Minimum utility threshold	**HUIM-HC**	**HUIM-SA**	**HUIM-AF**	**HUIM-IDCS**
13	100	96.74	94.30	100
12.8	100	99.21	95.00	100
12.6	100	99.26	98.95	100
12.4	100	100.00	98.67	100
12.2	100	100	100	100
Average	100	99.04	97.39	100

On examining the average percentage of detected HUI's on the accident dataset, HUIM-ICDS performs better than the current approaches, including HUIM-SA by 0.9%, HUIM-AF by 2.61% and competes equally with HUIM-HC.

Table 11 displays the proportion of identified HUIs in the connect dataset using several HUI algorithms, with respect to varying minimal utility threshold values. When comparing the outcomes of the suggested model with the existing algorithms, HUIM-ICDS demonstrates a noteworthy enhancement in terms of the Discovered HUI's.

Table 11. Number of HUI's on Dataset - Connect

Minimum utility threshold	**HUIM-HC**	**HUIM-SA**	**HUIM-AF**	**HUIM-IDCS**
32	100	98.85	98.77	100
31.8	100	99.29	98.55	100
31.6	100	100	100	100
31.4	100	100	100	100
31.2	100	100	100	100
Average	100	99.63	99.46	100

On examining the average percentage of detected HUI's on the connect dataset, HUIM-ICDS performs better than the current approaches, including HUIM-SA by 0.3%, HUIM-AF by 0.5% and competes equally with HUIM-HC.

5.5 Convergence

Figure 5, 6, 7, and 8 displays the convergence graph illustrating the progress towards the optimal solution at each iteration. The graphs are plotted with an iteration interval of 50 for the chess, mushroom, accident, and connect datasets, respectively.

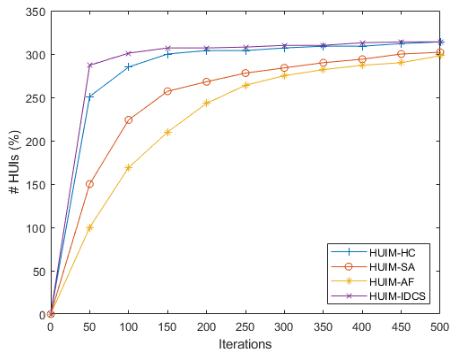

Fig. 5. Convergence while Discovering HUI on dataset - Chess

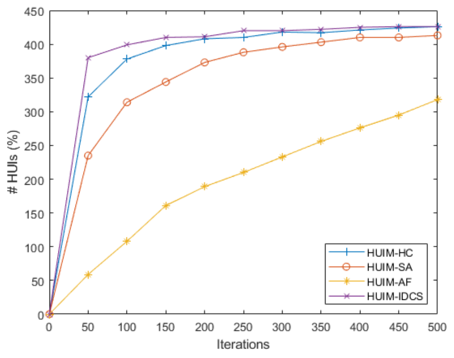

Fig. 6. Convergence while Discovering HUI on dataset - Mushroom

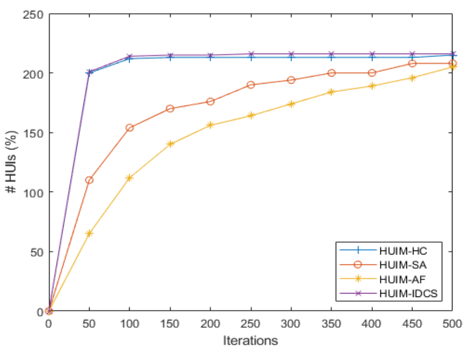

Fig. 7. Convergence while Discovering HUI on dataset - Accident

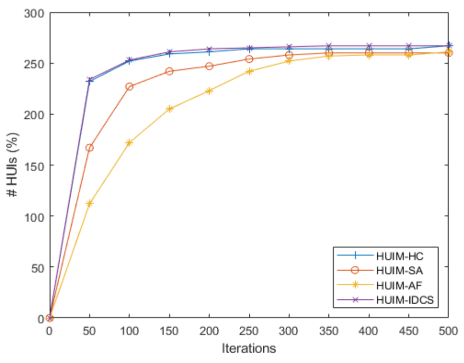

Fig. 8. Convergence while Discovering HUI on dataset - Connect

The convergence graph displays the cumulative count of HUIs identified at each interval of iteration time. Based on the data, it is evident that the suggested model surpasses all existing methods in terms of performance. Our suggested model outperforms HUIM-HC in terms of results, requiring less iterations. By analyzing the graphs, it can be noted that this model performs on par with other models at higher levels of achievement.

6. Conclusion

This study presents an enhanced Discrete Cuckoo Search technique designed in order to solve HUI issues that need huge datasets. Levy flights and oppositional learning are both included into the model that has been provided. The objective of this strategy is to streamline the process of selecting whether or not to include a certain item in the HUI by reducing the temporal complexity, which is characterized by an exponential component. The model we suggest significantly decreases the overall time complexity. The suggested model is evaluated against techniques such as HUIM-HC, HUIM-SA, and HUIM-AF utilizing performance criteria including runtime, number of identified high utility itemsets (HUIs), and convergence. Taking into consideration the results in comparison to other approaches that are already in existence, the experimental assessment reveals the significance of the proposed model.

References

[1] J.M. Luna, P. Fournier-Viger, S. Ventura, Frequent itemset mining: A 25 years review, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9 (6) (2019).

[2] C.C. Aggarwal, J. Han, Frequent Pattern Mining. Springer Publishing Company, 2014. ISBN: 978-3-319-07821-2.

[3] P. Fournier-Viger, J.C.-W. Lin, R. Nkambou, B. Vo, V.S. Tseng, High-Utility Pattern Mining: Theory, Algorithms and Applications, 1st edition., Springer Publishing Company, 2019.

[4] W. Gan, J.C. Lin, J. Zhang, P. Fournier-Viger, H. Chao, P.S. Yu, Fast utility mining on sequence data, IEEE Transaction on Cybernetics 51 (2) (2021) 487–500.

[5] P. Fournier-Viger, C. Wu, S. Zida, V.S. Tseng, FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning, in: Proceedings of the 21st International Symposium on Foundations of Intelligent Systems, ISMIS 2014, Roskilde, Denmark, June 25–27, 2014. Proceedings, volume 8502, Springer, 2014, pp. 83–92.

[6] P. Fournier-Viger, J.C.-W. Lin, R. Nkambou, B. Vo, V.S. Tseng, High-Utility Pattern Mining: Theory, Algorithms and Applications, 1st edition., Springer Publishing Company, 2019.

[7] W. Gan, J.C. Lin, P. Fournier-Viger, H. Chao, P.S. Yu, HUOPM: high-utility occupancy pattern mining, IEEE Transactions on Cybernetics 50 (3) (2020) 1195–1208.

[8] Y. Liu, W. Liao, A.N. Choudhary, 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, Hanoi, Vietnam, May 18–20, 2005, Proceedings, volume 3518, Springer, 2005, pp. 689–695.

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

[10] V.S. Tseng, B. Shie, C. Wu, P.S. Yu, Efficient algorithms for mining high utility itemsets from transactional databases, IEEE Transaction on Data and Knowledge Engineering 25 (8) (2013) 1772–1786.

[11] S. Kannimuthu, K. Premalatha, Discovery of high utility itemsets using genetic algorithm with ranked mutation, Applied Artificial Intelligence 28 (4) (2014) 337–359.

[12] J.C. Lin, L. Yang, P. Fournier-Viger, T. Hong, M. Voznák, A binary PSO approach to mine high-utility itemsets, Soft Computing 21 (17) (2017) 5103 5121.

[13] Y. Liu W.K. Liao A. Choudhary A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets 2005 Springer-Verlag 689 695 10.1007/11430919_79.

[14] Le, B., Nguyen, H., & Vo, B. (2011). An efficient strategy for mining high utility itemsets. International Journal of Intelligent Information and Database Systems, 5(2), 164–176

[15] Tseng, V. S., Wu, C. W., Shie, B. E., & Yu, P. S. (2010). UP-Growth: An efficient algorithm for high utility itemset mining. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 253–262. https://doi.org/10.1145/1835804.1835839.

[16] Qu, J.-F., Fournier-Viger, P., Liu, M., Hang, B., & Hu, C. (2023). Mining high utility itemsets using prefix trees and utility vectors. IEEE Transactions on Knowledge and Data Engineering, 1–14. https://doi.org/10.1109/TKDE.2023.3256126

[17] Liu, M., & Qu, J. (2012). Mining high utility itemsets without candidate generation. Proceedings of the 21st ACM International Conference on Information and Knowledge Management - CIKM '12, 55. https://doi.org/10.1145/2396761.2396773.

[18] Fournier-Viger, P., Wu, C.-W., Zida, S., & Tseng, V. S. (2014). FHM: Faster High-Utility itemset mining using estimated utility co-occurrence pruning. In In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 8502 LNAI (pp. 83–92). https://doi.org/10.1007/978-3-319-08326-1_9

[19] Krishnamoorthy, S. (2015). Pruning strategies for mining high utility itemsets. Expert Systems with Applications, 42(5), 2371–2381. https://doi.org/10.1016/j.eswa.2014.11.001

[20] Duong, Q.-H.-H., Fournier-Viger, P., Ramampiaro, H., Nørvåg, K., & Dam, T.-L.-L. (2018). Efficient high utility itemset mining using buffered utility-lists. Applied Intelligence, 48(7), 1859–1877. https://doi.org/10.1007/s10489-017-1057-2

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

[22] Zida, S., Fournier-Viger, P., Lin, J.-C.-W., Wu, C.-W., & Tseng, V. S. (2017). EFIM: A fast and memory efficient algorithm for high-utility itemset mining. Knowledge and Information Systems, 51(2), 595–625. https://doi.org/10.1007/s10115-016-0986-0

[23] Nguyen, L. T. T., Nguyen, P., Nguyen, T. D. D., Vo, B., Fournier-Viger, P., & Tseng, V. S. (2019). Mining high-utility itemsets in dynamic profit databases. Knowledge-Based Systems, 175, 130–144. https://doi.org/10.1016/j.knosys.2019.03.022

[24] Lan, W., Lin, J.-C.-W., Fournier-Viger, P., Chao, H.-C., Tseng, V. S., & Yu, P. S. (2021). A survey of Utility-Oriented pattern mining. IEEE Transactions on Knowledge and Data Engineering, 33(4), 1306–1327. https://doi.org/10.1109/TKDE.2019.2942594

[25] Vo, B., Nguyen, L. T. T., Bui, N., Nguyen, T. D. D., Huynh, V.-N., & Hong, T.-P. (2020). An efficient method for mining closed potential High-Utility itemsets. IEEE Access, 8, 31813–31822. https://doi.org/10.1109/ACCESS.2020.2974104

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

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

[28] Tung, N. T., Nguyen, L. T. T., Nguyen, T. D. D., Fourier-Viger, P., Nguyen, N.-T., & Vo, B. (2022a). Efficient mining of cross-level high-utility itemsets in taxonomy quantitative databases. Information Sciences, 587, 41–62. https://doi.org/10.1016/j.ins.2021.12.017

[29] Srivastava, G., Lin, J.-C.-W., Pirouz, M., Li, Y., & Yun, U. (2021). A Pre-Large WeightedFusion system of sensed High-Utility patterns. IEEE Sensors Journal, 21(14), 15626–15634. https://doi.org/10.1109/JSEN.2020.2991045

[30] Attuluri, S., Ramesh, M. Multi-objective discrete harmony search algorithm for privacy preservation in cloud data centers. Int. j. inf. tecnol. (2023). https://doi.org/10.1007/s41870-023-01462-w

[31] Hazzazi, Mohammad Mazyad, Sasidhar Attuluri,

Zaid Bassfar, and Kireet Joshi. 2023. "A Novel Cipher-Based Data Encryption with Galois Field Theory" *Sensors* 23, no. 6: 3287. https://doi.org/10.3390/s23063287

[32] R. R. Budaraju and O. S. Nagesh, "Multi-Level Image Thresholding Using Improvised Cuckoo Search Optimization Algorithm," *2023 3rd International Conference on Intelligent Technologies (CONIT)*, Hubli, India, 2023, pp. 1-7, doi: 10.1109/CONIT59222.2023.10205744.

[33] Hazzazi, Mohammad Mazyad, Raja Rao Budaraju, Zaid Bassfar, Ashwag Albakri, and Sanjay Mishra. 2023. "A Finite State Machine-Based Improved Cryptographic Technique" *Mathematics* 11, no. 10: 2225. https://doi.org/10.3390/math11102225

[34] Thirugnansambandam, K., Bhattacharyya, D., Frnda, J., Anguraj, D. K., & Nedoma, J. (2021). Augmented Node Placement Model in t-WSN Through Multi objective Approach. Comput. Mater. Contin. CMC Tech Sci. Press, 69, 3629-3644.

[35] Thirugnanasambandam, K., Raghav, R. S., Anguraj, D. K., Saravanan, D., & Janakiraman, S. (2021). Multi-objective Binary Reinforced Cuckoo Search Algorithm for Solving Connected Coverage target based WSN with Critical Targets. Wireless Personal Communications, 121(3), 2301-2325.

[36] Thirugnanasambandam, K.; Ramalingam, R.; Mohan, D.; Rashid, M.; Juneja, K.; Alshamrani, S.S. Patron–Prophet Artificial Bee Colony Approach for Solving Numerical Continuous Optimization Problems. *Axioms* 2022, *11*, 523. https://doi.org/10.3390/axioms11100523

[37] Thirugnanasambandam, K., Rajeswari, M., Bhattacharyya, D. et al. Directed Artificial Bee Colony algorithm with revamped search strategy to solve global numerical optimization problems. Autom Softw Eng 29, 13 (2022). https://doi.org/10.1007/s10515-021-00306-w

[38] Raghav, R. S., Thirugnanasambandam, K., Varadarajan, V., Vairavasundaram, S., & Ravi, L. (2022). 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, 10(3), 186-203.