A Canonical Particle Swarm Optimization(C-PSO) Approach to Identify High Utility Itemset

V. Jeevika Tharini[1], B.L.Shivakumar[2]

[1]Research Scholar, Sri Ramakrishna College of Arts & Science, Coimbatore,Email: jeevi153gv@gmail.com
[2]Principal, Sri Ramakrishna College of Arts & Science, Coimbatore

Received: 17.04.2024 Revised : 13.05.2024 Accepted: 08.05.2024

ABSTRACT
High Utility Itemset Mining (HUIM) is the advanced process of identifying highly profitable items by assessing the unit profit of each item within extensive transaction databases. Over recent years, HUIM has emerged as a crucial subject with broad applications. It facilitates the identification of profitable items based on factors such as profit and quantity, setting it apart from conventional algorithms. Numerous algorithms have been developed to mine High Utility Itemsets (HUIs), addressing the challenge of searching for these sets in databases containing many distinct items. HUIM incorporates optimization techniques to reduce complexities and discover optimal solutions. This research uses Canonical Particle Swarm Optimization (C-PSO) to identify high-profit items, effectively managing convergence properties.Canonical Particle Swarm Optimization (C-PSO) is motivated by the imperative to enhance the efficiency of HUIM. C-PSO aims to swiftly and accurately identify high-profit items within transaction databases by optimizing the search space. Its advantages lie in superior convergence properties, addressing challenges in navigating extensive item spaces, and its adaptability to diverse utility mining scenarios, making it a robust and superior choice compared to existing techniques. The experimental results, conducted on three benchmark datasets, illustrate that C-PSO outperforms existing state-of-the-art techniques in the context of HUIM.

Keywords: High Utility Item, fitness function, transaction weighted utility model, threshold value.

1. INTRODUCTION

Conventional mining algorithms like Association Rule Mining (ARM) and Frequent Item Mining (FIM) are mainly designed to compute the occurrence of individual items in the transactional databases,taking into account their confidence or minimum support values [1]. While these algorithms work well in identifying the most frequent itemsets, they are not very effective in finding dense itemsets, which yield high profits, although they may be rare. For instance, in a supermarket setting, the supermarket may sell sugar in large quantities daily, meaning frequent transactions, but the profit per unit is usually low. On the other hand, products such as cashew nuts may be sold with low frequency compared to biscuits, but their profit may be much higher [2]. This scenario shows how FIM and ARM have shortcomings in their little attention to the less frequently sold but highly profitable items. High Utility Itemset Mining (HUIM) techniques have been proposed to address these issues [3].

HUIM concentrates on mining itemsets with higher utility from the transaction databases. Utility mining considers the number of items and the profit per unit of each item to derive the utility of the itemset. In utility mining, the number of items appearing in a transaction is termed internal utility, and the profit of individual items is called external utility [4]. High Utility Itemsets (HUIs) are the itemsets which have utility greater than a specified minimum utility. This approach makes it possible to detect those items that, while occurring less frequently, produce a large proportion of the total profits. Utility mining can be applied to quantitative databases in which the transaction data contain the quantity of the items sold [5].

As a result of the development of HUIM, several algorithms and mechanisms have been developed to identify high-utility item sets. These algorithms are expected to overcome the shortcomings of this conventional mining process by targeting profitability rather than frequency. However, identifying profitable items has some drawbacks, like high time consumption and memory usage, the appearance of non-efficient candidate items, and the absence of required items at the stage of item creation. For such complexities, advanced optimization methods are needed [5].

Evolutionary computing, as a technique based on the principles of natural selection, is excellent for finding efficient optimal solutions [6]. Particle Swarm Optimization (PSO) is a bio-inspired optimization algorithm that belongs to the population-based optimization technique in which the solutions are updated based on pbest and gbest values. In PSO, particles are potential solutions that move through the solution space and update their positions according to their own experience as well as the experience of the other neighbouring particles [7]. This method blends the exploration and exploitation strategies to find the best solutions. However, PSO has been able to solve continuously optimizing problems but has struggled in discrete-valued search spaces.

Due to the limitations of the conventional algorithms in mining high-utility itemsets, researchers have developed several enhanced techniques. Such approaches frequently include components of evolutionary computing like PSO to improve the efficiency and effectiveness of the mining process [8]. For example, some studies integrate PSO with other optimization methods to deal with the discrete characteristics of utility mining issues. These techniques combine the best features of several methods to increase the efficiency and accuracy of the results [9].

The most prominent issue in the recent studies of HUIM is how to mine high utility itemsets since the utility values depend on users' preferences and are dynamic. The utility rate of a given product depends on the user, meaning these factors must be incorporated into the mining process. Many algorithms have been developed to solve the identification problems of the most profitable items. Still, they all have different issues: they take more time and memory, generate repeated and unnecessary items, and may lose essential items. To overcome these limitations, new optimization techniques are needed.

Optimization problems are significant in many application areas, such as data mining, to overcome the deficiencies of existing methods [10]. Therefore, by applying sophisticated optimization methods, researchers can enhance the performance and effectiveness of HUIM to find highly useful itemsets for profitability and importance. All these advancements not only increase the effectiveness of utility mining but also offer insights into the preferences and trends of the users, thus improving decision-making and strategic planning in different fields.

Conventional mining algorithms such as ARM and FIM cannot discover the most profitable items, hence the need to develop HUIM approaches. Utility mining considers the number of items and their profit, which helps generate PSO. An example of an evolutionary computing technique provides optimal solutions to enhance the mining process; however, they have issues in discrete search spaces. In this regard, developing sophisticated optimization methods and combined algorithms is critical, opening up new opportunities for high-utility itemset mining with higher efficiency.

The paper is organized as follows: Section 2 provides a review of the existing literature on utility mining. Section 3 outlines the architecture of the HUIM-based C-PSO and includes a running example. Section 4 presents the results and compares the performance of various algorithms. Finally, Section 5 offers conclusions and future recommendations.

2. RELATED WORKS

Fournier-Viger, P., et al [11] studied the restrictions of HUIM by explaining the issues of identifying Local HUI's (LHUI) and Peak HUI's (PHUI), which typically creates time periods of itemset and higher utility. PHUI-Miner and LHUI-Miner were developed to retrieve profitable patterns. LHUI used the data structure titled as LU-list, and further implements the general search procedure of HUI-miner. PHUI is appropriate in market basket analysis and efficiently discovered the time periods with HUI's in a prominent way. Non-redundant Peak HUI's (NPHUIs) is employed to identify the reduced set of patterns. All the above mentioned proposed algorithms were used in discovering the beneficial patterns where it identifies greater utility pattern rather than the high utility pattern.

Liu, M., et al [12] addressed the drawbacks of existing algorithms and designed an algorithm that incurs the features of list structure to store itemsets utility information and search space's heuristic information that is related to pruning. List structure has reduced the computation of utility values and several recursive generations of an itemset. The utility list is generated from the retrieved data that had prominently mined the HUI's. The main intent of HUI-Miner is to discover the HUI's without losing any of the important patterns in the database. HUI-miner showed the best result by acquiring lesser utilization of memory and reduced run time.

Ahmed, C. F., et al [13] proposed high utility pattern (HUP) mining, which computed the dissimilar profit values for every single item and the non-binary existence of items. Interactive as well as incremental data mining incorporates the previous structures of data and also lessens the redundant calculation whenever the threshold is changed or updated. HUP incorporated tree structures and achieved HUP mining. Each incidence of an item in the transaction is arranged in an item's lexicographic order that is called as Incremental HUP Lexicographic Tree (IHUPL-Tree) and it doesn't use any restructuring procedure to mine the incremental data. Based on the incidence of the items in a transaction, the items are arranged that are in decreasing order which is called an IHUP Transaction Frequency Tree (IHUPTF-Tree).

IHUPTWU-Tree is developed by incorporating the TWU model and the items were effectively mined with reduced time.
VikramGoyal et al [14] designed an algorithm called UP-Rare Growth and it implemented the UP-Tree data structure that fetched the rare HUI's from an operational database. UP-Rare Growth works on both the utility and incidence of itemsets together. The author has also suggested two more effective schemes to avoid the examination of inappropriate branches of the tree. Extensive experiments indicated that the developed algorithm outperforms the available algorithm in terms of generated candidate item count.
Peng, A. Y., et al [15] proposed a mHUIMiner algorithm for the proficient identification of HUI's. The main intent of HUIM is to forecast the items whose utilities are higher than or equivalent to a definite threshold value. The mHUIMiner integrated the structure of a tree to make the development process of itemset to neglect the unavailable itemset in the database. In mHUIMiner, complicated pruning schemes are not necessary that need costly computation overhead. In the sparse dataset, running time is comparatively reduced.
Kannimuthu, S., et al [16] had shown a contribution in reclaiming the HUIs which faces major issues namely threshold assignment and search space. When the items count and generation of items is huge then the search space eventually increased. The analyst had stated the threshold values without having any idea about the dataset. Based on the dataset's nature thresholds were generated automatically to avoid manual threshold assignment that is achieved using the GA. The auto-generation method showed better performance.To obtain HUI's effectively, HUIF-PSO (High Utility Itemset Framework based Particle Swarm Optimization) is improved utilizing the Frequent Pattern tree structure. Performance variables include the HUI's count, memory, and time utilization. The revised HUIF-PSO Tree method outperforms previous algorithms, according on the results of the experiments [17].
Song, W., et al [18] had shown recent advancements in bio-inspired computing and it had attracted attention that leads to the establishment of HUI mining algorithms. The HUI identification without any loss of HUI from the database is. It has maintained the optimal values of the population. Thus, the replication within populations was enhanced. It was developed using the GA, PSO and BAT algorithm. An extensive test report shows that the Bio-HUI framework outperforms the existing algorithm by comparing the performance factors efficiency, convergence speed, and result's quality.
Lin, J. C. W., et al [19] had conversed significant issues of HUIM. A heuristic HUPEumu-GRAM algorithm is proposed to reclaim HUIs based on GA. In PSO requirement of the parameter is less when compared to the GA-based approach. The discrete particles are encoded as the binary variables in the PSO. On the basis of the PSO algorithm, a new approach is formulated and it is called HUIM-BPSO$_{sig}$which efficiently acquire the HUIs. The combinational problem is reduced by the size of a particle assigned based on the TWU model in the evolution process. The sigmoid function is employed for updating the particles in the HUIM-BPSO$_{sig}$ algorithm.
Wu, et al [10] depicted the foremost issue in data mining which is HUI mining and it is a contrast factor of FIM. Factors like capacity and the profit were utilized for the HUI retrieval process. In HUIM, the process of handling the HUI identification space is tedious for varied size and the item of various type. The author had designed an algorithm using the optimization strategy called ACO based method which was mined effectively with the items holding greater profit values. PSO and GA are also used for acquiring utility items and these algorithm results in huge computational time. To attain the results with limited computational time ACO is introduced. HUIM-ACS (Ant colony system) used pruning in two ways which effectively map the solution into the routing graph. HUI's are reclaimed without any candidate edge from the initial stage of the process. HUIM-ACS outperforms other HUIM algorithms.
Dam, et al [21] developed utility mining with the approach called average utility measure that was the utility of the item which was divided by the count of the occurring item. The average utility of an item had no downward closure property and the maximal utility is assigned as an upper bound value. The process of estimation was drawn into two phases. The estimation process was initiated with the summation of the highest utility value and proceeded to the estimation of actual utility value with the upper bound value. Average utility mining has used higher threshold values compared to other algorithms.
Li, et al [22] designed one pass algorithms namely Mining HUI's based on BIT vector (MHUI-BIT) and Mining HUI's based on TIDlist (MHUI-TID). Data streams were a sequence of continuous data that arrived at a quick rate. Only few of the researchers have developed algorithms for data streams and hence, an efficient approach was needed for the retrieval of items from the streaming data. The lexicographical tree was used for representing the generation of candidate items. The test result showed the algorithm MHUI-BIT and MHUI-TID have obtained efficient results than other existing algorithms data streams.
The research gap identified in HUIM includes the following: the approach developed in HUIM does not consider frequency and utility in a balanced way; that is, it only finds high utility values (Fournier-Viger, P. et al. [11]); HUIM may have some computational issues (Liu, M. et al. [12]); HUIM has a problem in

dealing with incremental data (Ahmed Some of the other works focused on discovering closed itemsets or dynamic databases but were still not scalable for large datasets or real-time settings as subsequently optimized by VikramGoyal et al. [14] and Li et al. [22]. To address these gaps, the Canonical Particle Swarm Optimization (C-PSO) approach is proposed, which integrates the concepts of evolutionary computing with the latest heuristic methods to increase the speed, decrease the complexity, and increase the accuracy of HUI identification across multiple datasets, thereby providing a more equitable and inclusive solution for practical applications.

3. Proposed Methodology: HUIM-C-PSO Algorithm

In this approach, every particle is indicated by two kinds of vectors and it indicates position as well as velocity.

3.1. Modelling HUIM using C-PSO

Particle Swarm Optimization (PSO) is a bio-inspired method that can simply optimize real-world, non-linear, and complex optimization issue. Typically, every fish or bird is stated as “particle” and their flock stated as “population of a particle”. Every particle transmits around the widespread area of search space in accordance with objective function (OF). Movement of every particle is based on the neighbor and personal experiences. The particles are initialized randomly based on the position of particle that is relevant to constraint of issue. Entire particle position is stated as initial swarm or population. After generating random velocities of every particle and relevant to objective function, objective value is investigated. During the process of velocity update, constriction factor is introduced to control the convergence of a particle. After introduction of constriction factor, the velocity is updated using equation (1).

$$vl_i^{t+1} = CF(vl_i^t + c_1 RA_1^t(pb_i^t - x_i^t) + c_2 RA_2^t(gb_i^t - x_i^t))$$

$$CF = \frac{2r}{(|2-\varphi-\sqrt{(\varphi^2-4\varphi)}|)}$$

where the cognitive factor is indicated as c_1, the social factor is indicated as c_2, r,RA_1^t and RA_2^t indicates the random number ranges between 0 to 1, pb indicates personal best, gb indicates global best, the value of can be greater than 4, and the constriction factor is indicated as CF. The value of $c_1$and c_2 are assigned with 2.05, k is assigned with 1, and the CF is equivalent to 0.729.

The population size PS, 1-HTWUIs count is N_{ct}, and all the 1-HTWUIs are sorted with the assistance of lexicographic order across whole utility mining process. The velocity vector $V_i(1 \le i \le PS)$ with elements N_{ct} and every element V_i^j is considered as probability relevant to the velocity of the jth1-HTWUIs to be utilized in the updating position, the position vector $P_i(1 \le i \le PS)$ is considered as binary vector with elements N_{ct}, and every element P_i^j is either 0 or 1 that represent presence or absence of jth1-HTWUIs in P_i.

A velocity vector alters with regard to the prior positions for these two vectors, and a position vector changes in response to the velocity vector that indicates a new prospective itemset. If the position vector of a jth vector is composes a one item in the jth place that is according to whole order and indicates potential HUI. Otherwise, this item is not used and couldn't be in a potential HUI. The jth bit of a position vector P_i is initialized by zero or one by utilizing a roulette wheel selection with probabilities is given in equation (2)

$$p(p_i^j) = \frac{TWU\ (item_i)}{\sum_{k=1}^{N_{ct}} TWU\ (item_k)}$$

where N_{ct} is the count of 1-HTWUIs.

The fitness function is computed for each iteration of C-PSO to characterize the optimization problem. Let's say the itemset(IS) is indicated by a position vector, and the utility of IS is directly utilized as the fitness value in equation (3)

$$fitness(P_i) = u(IS)$$

It is also necessary to redefine the estimation of every position where the two position vectors are P_m and P_n with the element count N_{ct}. This is determined in equation (4)

$$dP = P_m - P_n = \{dP_i | 1 \le i \le N_{ct}\}$$

where

$$dP_i = \begin{cases} 1, & if\ p_m^i = 1\ and\ p_n^i = 0 \\ 0, & Ot⍰erwise \end{cases}$$

3.2. Illustrative Example

In a quantitative data store, $I = \{m_1, \mathrm{m}_2, \mathrm{m}_3, \dots\dots, \mathrm{m}_\mathrm{n}\}$ represents a set of n individual items, while $T = \{t_1, t_2, t_3 \dots\dots\dots\dots t_n\}$ denotes the transactions. Each transaction $\mathrm{T_r}$ is a subset of Iand has a unique identifier called TID. External utility values, shown in Table 2, represent the profit of each item, and internal utility values, shown in Table 1, indicate the quantity of purchased products. A minimal utility value is set based on user preferences. This study presents a running example with seven transactions and five items, D to H, as illustrated in Table 1.

Table 1. Quantitative data store

Tid	Transaction Item	Occurrence
A_1	{D,E,G}	1,4,1
A_2	{D,H}	5,2
A_3	{D,F,G}	3,4,6
A_4	{E,F,G,H}	1,2,1,5
A_5	{F.H}	5,1
A_6	{D,E,F,H}	1,2,5,1
A_7	{D,E,F,G,H}	1,2,5,3,1

Table 2. Profit values

Item	D	E	F	G	H
Profit	4	3	1	5	1

Definition 1: The utility of an item $\mathrm{m_j}$in the transaction $\mathrm{T_r}$is signified as $\mathrm{u(m_j, T_r)}$ and denoted as

$$\boldsymbol{u(m_j, T_r) = \mathbf{r}(m_j, T_r) \times \mathbf{pr}(m_j)}$$

Example: Item D's utility in A_1transaction is calculated as follows,
u(D, A_1)=r(D,A1) ×pr(D)
u(1,A_1)=r(1×4)=4

Definition 2: The utility of an item $\mathrm{m_j}$ in the data store DB is signified as $\mathrm{u(m_j)}$ and denoted as

$$\boldsymbol{u(m_j) = u(m_j, A_p) + \dots + u(m_n, A_n)}$$

Example: Item D's utility in entiretransaction is calculated as follows,
u(D) = u(D, A_1)+ u(D, A_2)+ u(D, A_3)+ u(D, A_6)+ u(D, A_7) = 4+20+12+4+4 =44

Definition 3: The utility of an itemset L in the transaction $\mathrm{T_r}$is signified as $\mathrm{u(L, T_r)}$, and denoted as

$$\mathbf{u(L, T_r)} = \sum_{\mathbf{m_j L \vee L \subseteq T_r}} \mathbf{u(m_j, T_r)}$$

Example: Itemset DE's utility in A_1transaction is calculated as follows,
u(DE, A_1) = u(D,A_1)+u(E,A_1) = 4+12 =16

Definition 4: The utility of an itemset L in the data storeDB is signified as u(L), and denoted as

$$\boldsymbol{u(L)} = \sum_{\mathbf{L} \subseteq \boldsymbol{T_r} \vee \boldsymbol{T_r} \subseteq \mathbf{DB}} \mathbf{u}(\mathbf{L}, \boldsymbol{T_r})$$

Example: Item DE's utility in entiretransaction is calculated as follows,
U(DE) = u(DE,A_1)+ u(DE,A_6)+ u(DE,A_7) = 16+10+10 = 36

Definition 5: The transaction utility of a transaction $\mathrm{T_r}$is signified as tu and denoted as

$$\boldsymbol{tu(T_r)} = \sum_{\mathbf{L} \subseteq \boldsymbol{T_r}} \mathbf{u}(\mathbf{L}, \boldsymbol{T_r})$$

Example: The utility of whole transaction A_1 is calculated as,
tu(A_1) = tu(D,A_1)+tu(E,A_1)+tu(G,A_1) = 4+12+5 = 21

Definition 6: The total transaction utility of a transaction $\mathrm{T_r}$is signified as tu and denoted as

$$\boldsymbol{Tu} = \sum_{\boldsymbol{T_r} \in \mathbf{DB}} \mathbf{Tu}(\boldsymbol{T_r})$$

Example: The utility of whole transaction from A_1 to A_7 is calculated as,
Tu = 44+27+21+55+10 = 157

To retrieve the HUIs and their relative utility from the temporal transaction database. The chief feature of HUIM-C-PSO from DB is to identify the set of HUI's.

In the designed HUIM-C-PSO algorithm, a traditional transaction weighted utility model is applied to figure out the HTWU. Based on the TWDC property of HTWUIs, the unsuitable items in the data store are proficiently removed. Thus, identification of HUI's estimation time is highly minimized. In the initialization, HTWUI's are identified. From the observed value, the minimum threshold value is assigned.

Table 3. Transaction Utility (TU) and Total Transaction Utility (TLTU)

Transaction	I-D	I-E	I-F	I-G	I-H	TU
A1	4	12	0	5	0	21
A2	20	0	0	0	2	22
A3	12	0	4	30	0	46
A4	0	3	2	5	5	15
A5	0	0	5	0	1	6
A6	4	6	5	0	1	16
A7	4	6	5	0	1	31
TLTU	44	27	21	55	10	157

Example: In the initialization phase TWU is calculated to evaluate the HTWUI's. Each item has its own utility rate, which is summed together in each transaction to yield tu. With the help of tu, an item's upper limit value is calculated. The predicted minimum utility value is 157×0.5 = 78.5. Table 4 shows the value of the identified HTWUI. In the solution space of L_i, DB, an initial population of particles is formed using this estimation of utility.

HTWUI(D) = D(A1)+D(A2)+D(A3)+D(A6)+D(A7)
= 21+22+46+16+31

HTWUI(D) = 136

The populations of the particle are spread in the data store of DB focused on the identified of HTWUI's in accordance with the minimal threshold population. The tltu is used to compute the least threshold level, and 0.6 is assigned as a user-defined number.

Minimum threshold value (β) = 157 × 0.6
= 94

β = 94

Table 4. HTWUI's

Item	TWU	HTWUI's
D	136	YES
E	83	NO
F	114	YES
G	97	YES
H	90	YES

Table 4 lists High Transaction Weighted Utility Items (HTWUIs), showing items with their Transaction Weighted Utility (TWU) values and indicating whether they meet the HTWUI criteria based on a predefined threshold. For example, item D with a TWU of 136 is marked as HTWUI.

Table 5. Particles Position

Particle position	**D**	**F**	**G**	**H**
P1	1	0	1	0
P2	1	0	0	1
P3	1	1	1	0
P4	0	1	1	1
P5	0	1	0	1
P6	1	1	0	0
P7	1	1	0	1

Table 5 displays particle positions in the Particle Swarm Optimization (PSO) algorithm, representing potential solutions with specific item combinations.

Table 6. List of HUIs

Particle position	Item	Fitness value
P1	DG	51
P2	DH	32
P3	DFG	46
P4	FG	41
P6	DF	34

Table 6 presents the final list of HUIs, showing the particle positions, the corresponding itemsets, and their fitness values. For example, the itemset DG has a fitness value of 51, indicating its high utility. Items having HUI are DG, DH, DFG, FG, and DF

4. RESULT AND DISCUSSION

This section illustrates the outcomes of the existing and proposed approach. The existing approaches namely HUIM-BPSO, HUIM-PSO, HUIM-GA are compared with

4.1. Execution Setup and Dataset Description

All the algorithms are instigated in Java and executed on a machine with a 3.20 GHz CPU. Comparison is performed for the proposed algorithm HUIM-C-PSO with the existing algorithms namely HUIM-BPSO, HUIM-BPSO-TREE, and HUIF-PSO for HUI mining.All the time, memory and high utility item measurements are carried with the Java API. Illustration of the algorithm is carried with three benchmark datasets having diverse characteristics. Candidates Count, Memory Usage and Execution Time for existing and proposed algorithm are observed against various threshold values. Table 7 shows the description of the dataset.

Table 7. Description of DATASET

Dataset Name	Instances	Attributes
Chess	28,056	6
Mushroom	8,124	22
Retail	88,162	10

4.2. Performance Evaluation

In this research work, HUI item count, memory usage and execution time are considered as performance factors. The execution time is measured in milliseconds (ms), memory usage in megabytes (MB) and candidate count in numbers (nos).

Comparison of Time Consumption

This section explains the time consumed in generating the candidate itemset and searching for profitable itemsets from the generated candidate itemset. The test compares three datasets' runtime with three chosen existing algorithms. The above results show that the proposed HUIM-C-PSO algorithm has yielded better outcomes than HUIM-BPSO, HUIM-BPSO-TREE [19] and HUIF-PSO algorithm [17]. The HUIM-C-PSO algorithm effectively reduces the combinational issue in the evolution procedure since it generates effective combinations of items from the dataset. The time taken to perform the existing and proposed approach is shown in Table 8 and Figure 1.

Table 8. Comparison of runtime

Dataset	Algorithm	T-35	T-30	T-25	T-20	T-15
Chess	HUI-BPSO	417823	408284	392331	388123	371231
	HUIM-BPSO-TREE	326404	345899	349708	350080	351938
	HUIF-PSO	160484	169115	198401	201208	218491
	HUIM-C-PSO	184611	193574	198472	201785	207891
Mushroom	HUI-BPSO	457896	459567	534897	598715	764582
	HUIM-BPSO-TREE	39389	82942	86657	89704	103105
	HUIF-PSO	43481	34202	35987	36987	37876

	HUIM-C-PSO	37001	37142	39547	41027	43120
Retail	HUI-BPSO	456142	497861	519677	567841	597271
	HUIM-BPSO-TREE	40928	41872	42983	43217	48264
	HUIF-PSO	35762	39827	40621	41982	42981
	HUIM-C-PSO	357162	372871	393876	403872	411098

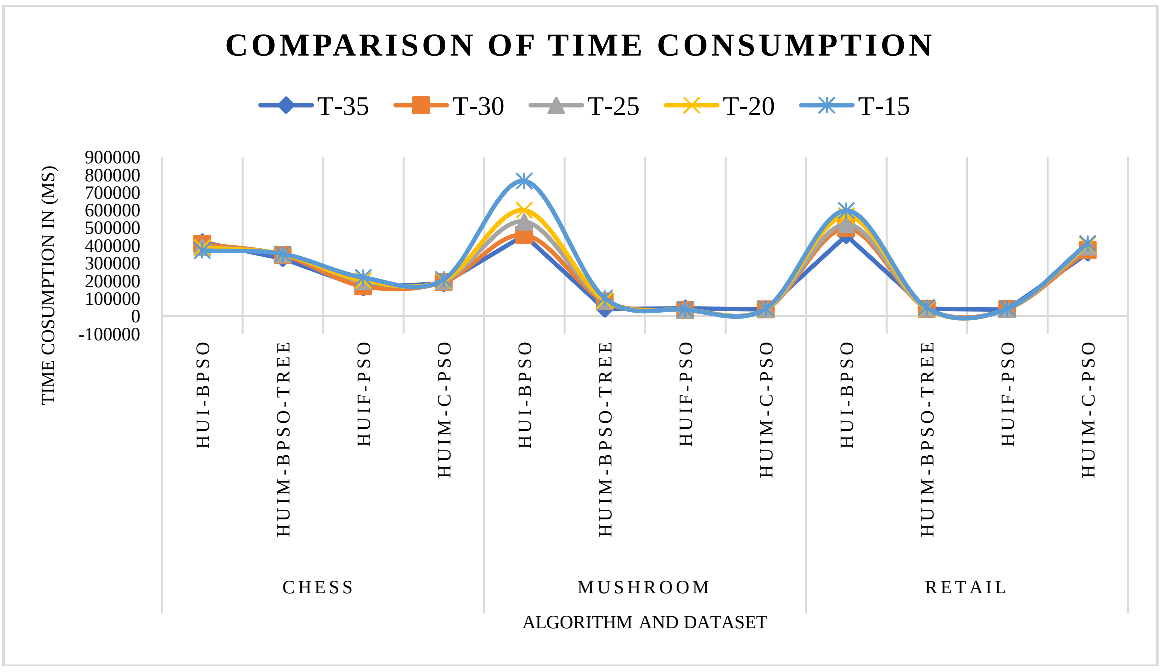

Figure 1. Comparison of runtime

Figure 1 shows that the proposed HUIM-C-PSO has minimal runtime than the HUIM-BPSO, HUIM-BPSO-TREE, and HUIF-PSO algorithms.

Comparison of Memory Usage

This section compares the memory space used to store the generated candidate itemset and the other processed information. The experimental evaluation of the HUIM-C-PSO, HUIM-BPSO, HUIM-BPSO-TREE, and HUIF-PSO algorithms with three benchmark datasets with several threshold values are indicated in Table 9 and Figure 2.

Table 9. Comparison of memory usage

Dataset	Algorithm	T-35	T-30	T-25	T-20	T-15
Chess	HUI-BPSO	76.08	72.69	69.56	65.34	61.34
	HUIM-BPSO-TREE	77.39	74.96	62.9	62.11	51.93
	HUIF-PSO	35	36.77	39.82	50.67	52.49
	HUIM-C-PSO	34	34.15	33.01	32.54	32.14
Mushroom	HUI-BPSO	38.65	42.89	45.88	48.69	52.36
	HUIM-BPSO-TREE	10.56	10.63	11.61	11.71	12.26
	HUIF-PSO	13.06	20.04	21.99	18.57	20.41
	HUIM-C-PSO	12.06	12.94	14.23	15.06	15.98
Retail	HUI-BPSO	41.82	45.76	47.65	49.76	51.76
	HUIM-BPSO-TREE	30.32	31.78	36.19	38.95	42
	HUIF-PSO	31.43	32.76	33.45	34.87	36.56
	HUIM-C-PSO	30.56	31.12	33.04	34.01	36.01

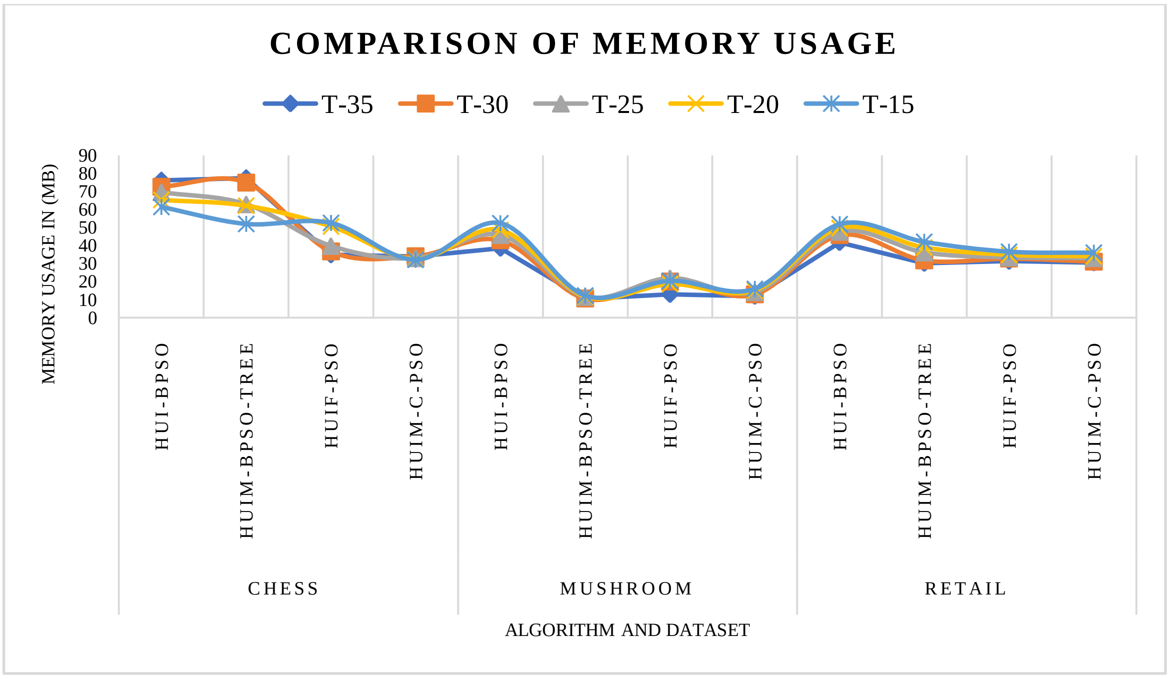

Figure 2. Comparison of memory usage

From the observation of Figure 2, it is found that the HUIM-C-PSO approach has given better results in terms of memory consumption that is minimal memory usage than the HUIM-BPSO, HUIM-BPSO-TREE, and HUIF-PSO algorithms.

Number of HUIs

The count of HUIs generated from the algorithms HUIM-C-PSO, HUIM-BPSO, HUIM-BPSO-TREE, and HUIF-PSO algorithms are discussed in this section. The TWU model is incorporated in the HUIM-C-PSO to determine the actual HUIs from the data store.

Table 10. Comparison of Count of HUIs

Dataset	**Algorithm**	**T-35**	**T-30**	**T-25**	**T-20**	**T-15**
Chess	HUI-BPSO	38821	31919	39798	40293	40928
	HUIM-BPSO-TREE	35998	35982	35270	33839	34862
	HUIF-PSO	33357	33419	33404	33753	33932
	HUIM-C-PSO	33876	32991	32198	32981	33914
Mushroom	HUI-BPSO	26548	26881	26938	27588	27823
	HUIM-BPSO-TREE	26088	39681	40146	48862	58941
	HUIF-PSO	25316	35986	36863	37388	49623
	HUIM-C-PSO	26316	35286	37863	38388	50623
Retail	HUI-BPSO	45098	45762	46761	47987	48768
	HUIM-BPSO-TREE	37675	38726	39871	41089	41872
	HUIF-PSO	38765	39796	40728	40981	41098
	HUIM-C-PSO	32876	35671	37987	39876	42892

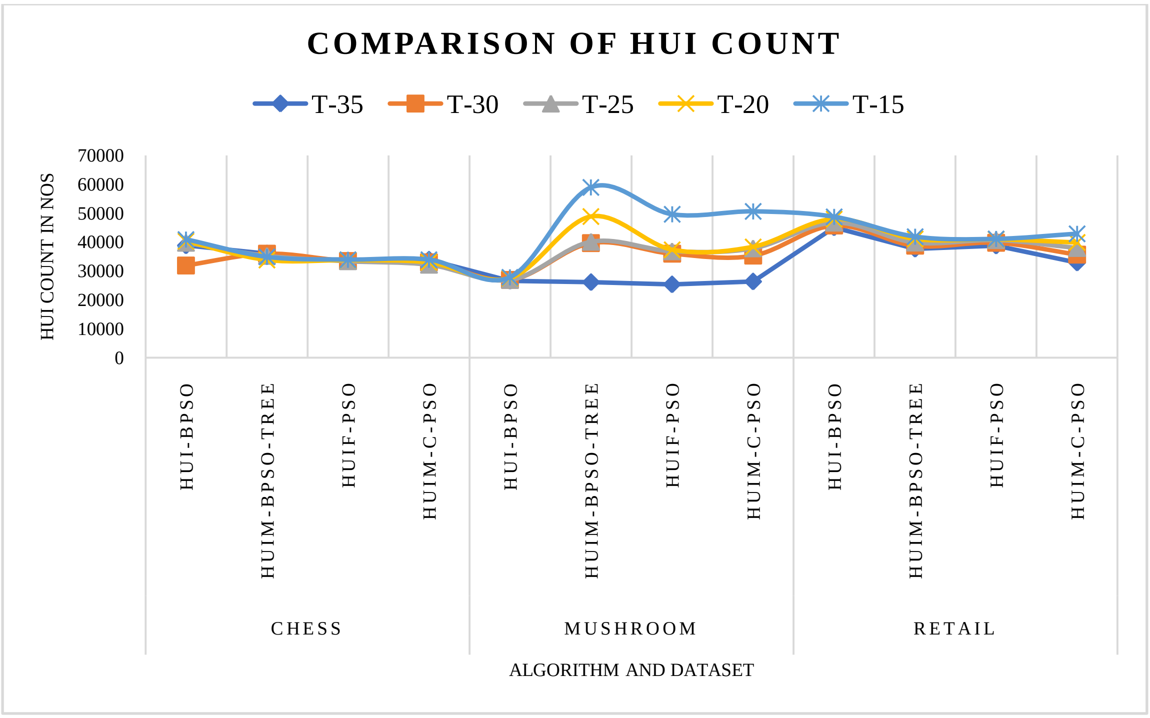

Figure 3. Comparison of Count of HUIs

Figure 3 shows that the HUIM-C-PSO algorithm produces almost similar count of the HUIs in the existing algorithms with minimal time consumption and memory usage.

5. CONCLUSION

HUIM has become a significant research area that can reveal very lucrative products. As many algorithms are available to extract the HUIs from the quantitative data repository efficiently, many researchers have used statistical analysis to identify critical information. The search process in HUI may require huge computational tasks. In recent years, bio-inspired algorithms have been widely applied in various fields, allowing the comparison of a new algorithm with it and demonstratingexcellent results.C-PSO is proposed for the HUIM algorithm which is a new approach for finding the HUIs. C-PSO is a population-based approach. The C-PSO-based mechanism is used for HUIM to retrieve the HUIs from the transactional database. A C-PSO-based algorithm is considered for mining the HUIs. The test is conducted on benchmark datasets to estimate the performance of the HUIM-C-PSO and HUIF-PSO-tree. The developed approach reduces the exploration space, time consumption and more number of HUI's are retrieved efficiently. Results depict that the proposed approach efficiently identifies the complete HUIs from the very condensing database and outperforms the existing algorithms. In the mere future, a novel algorithm for PPDM is also needed to achieve the intent of hiding the confidential HUI's so that the competitors not able to spot the items from the adjusted data store.

REFERENCE

[1] Agrawal, R., &Srikant, R. (1994, September). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large databases, VLDB (Vol. 1215, pp. 487-499).
[2] Chen, M. S., Han, J., & Yu, P. S. (1996). Data mining: an overview from a database perspective. IEEE Transactions on Knowledge and data Engineering, 8(6), 866-883.
[3] Tharini, V. J., & Shivakumar, B. L. High-Utility Itemset Mining: Fundamentals, Properties, Techniques and Research Scope. In Computational Intelligence and Data Sciences (pp. 195-210). CRC Press.
[4] Tharini, V. J. (2024). Cross-Entropy Assisted Optimization Technique for High Utility Itemset Mining from the Transactional Database. Communications on Applied Nonlinear Analysis, 31(3s), 90-104.
[5] Jeevika Tharini, V., & Vijayarani, S. (2020). Bio-inspired High-Utility Item Framework based Particle Swarm Optimization Tree Algorithms for Mining High Utility Itemset. In Advances in Computational Intelligence and Informatics: Proceedings of ICACII 2019 (pp. 265-276). Springer Singapore.
[6] Jeevika Tharini, V., Ravi Kumar, B., Sahaya Suganya Princes, P., Sreekanth, K., Kumar, B. R., &Sengan, S. (2024, January). Business Decision-Making Using Hybrid LSTM for Enhanced Operational

Efficiency. In International Conference on Multi-Strategy Learning Environment (pp. 155-166). Singapore: Springer Nature Singapore.

[7] J. Han, J. Pei, Y. Yin, ―Mining frequent patterns without candidate generation‖, in Proceedings of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12.

[8] Lan, G. C., Hong, T. P., & Tseng, V. S. (2014). An efficient projection-based indexing approach for mining high utility itemsets. Knowledge and information systems, 38(1), 85-107.

[9] Sivamathi, C., Vijayarani, S., & Jeevika Tharini, V. (2019). High on-shelf utility mining using an improved HOUI-mine algorithm. In International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018 (pp. 579-586). Springer International Publishing.

[10] Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science (pp. 39-43). Ieee.

[11] Fournier-Viger, P., Zhang, Y., Lin, J. C. W., Fujita, H., &Koh, Y. S. (2019). Mining local and peak high utility itemsets. Information Sciences, 481, 344-367.

[12] Liu, M., &Qu, J. (2012, October). Mining high utility itemsets without candidate generation. In Proceedings of the 21st ACM international conference on Information and knowledge management (pp. 55-64). ACM.

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

[14] VikramGoyalSiddharthDawar Ashish Sureka, ―High Utility Rare Itemset Mining over Transaction Databases‖, Databases in Networked Information Systems. DNIS 2015. Lecture Notes in Computer Science, vol 8999. Springer.

[15] Peng, A. Y., Koh, Y. S., & Riddle, P. (2017, May). mHUIMiner: A fast high utility itemset mining algorithm for sparse datasets. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 196-207). Springer, Cham.

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

[17] Jeevika Tharini, V., & Vijayarani, S. (2019, December). Bio-inspired High-Utility Item Framework based Particle Swarm Optimization Tree Algorithms for Mining High Utility Itemset. In International Conference on Advances in Computational Intelligence and Informatics (pp. 265-276). Springer, Singapore.

[18] Wu, J. M. T., Zhan, J., & Lin, J. C. W. (2017). An ACO-based approach to mine high-utility itemsets. Knowledge-Based Systems, 116, 102-113.

[19] Lin, J. C. W., Yang, L., Fournier-Viger, P., Wu, J. M. T., Hong, T. P., Wang, L. S. L., & Zhan, J. (2016). Mining high-utility itemsets based on particle swarm optimization. Engineering Applications of Artificial Intelligence, 55, 320-330.

[20] Wu, J. M. T., Zhan, J., & Lin, J. C. W. (2017). An ACO-based approach to mine high-utility itemsets. Knowledge-Based Systems, 116, 102-113.

[21] Dam, T. L., Li, K., Fournier-Viger, P., & Duong, Q. H. (2019). CLS-Miner: efficient and effective closed high-utility itemset mining. Frontiers of Computer Science, 13(2), 357-381.

[22] Li, H. F., Huang, H. Y., Chen, Y. C., Liu, Y. J., & Lee, S. Y. (2008, December). Fast and memory efficient mining of high utility itemsets in data streams. In 2008 eighth IEEE international conference on data mining (pp. 881-886). IEEE.