Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 31 No. 3s (2024)

Cross-Entropy Assisted Optimization Technique for High Utility Itemset Mining from the Transactional Database

V. Jeevika Tharini[1], B. L. Shivakumar[2]
[1]Research Scholar, Sri Ramakrishna College of Arts & Science, Coimbatore
[2]Principal, Sri Ramakrishna College of Arts & Science, Coimbatore

Article History:
Received: *15-04-2024*
Revised: *21-05-2024*
Accepted: *02-06-2024*

Abstract

High Utility Itemset Mining (HUIM) is the process of discovering profitable itemsets in a transactional database with a high utility or profit range. This technique is mainly used to unearth the prominent patterns and insights from large volumes of data. The main goal of HUIM is to utilize optimization techniques that improve the efficiency of mining operations. This research article introduces a Genetic Algorithm (GA) integrated Bacterial Foraging Algorithm (BFA) based on the HUIM technique, which is efficient due to its exploitation and exploration ability. This incorporation intends to address the issue of early convergence in itemset mining, and Cross Entropy (CE) act as a genetic operator, which enhances the efficiency of the mining operation. The technique effectively perceives High Utility Itemsets (HUIs) within a significantly reduced duration of 16723 ms, accomplishing optimal outcomes. The strategy effectively reduces the likelihood of premature convergence and showcases exceptional efficiency, as seen by the successful retrieval of 3432 optimized itemsets. The suggested hybrid GA-BFA approach effectively addresses problems in itemset mining by achieving high-quality results in a small amount of time. The implementational outcomes of the proposed Hybridized Cross Entropy Assisted Genetic algorithm with Bacterial Foraging Algorithm (HCEG-BFA) are compared with existing state-of-art techniques where the HCEG-BFA outperforms existing state-of-art techniques.

Keywords: Cross entropy, crossover, BFA, GA, HUI mining, profit, memory usage, and runtime.

1. Introduction

Utility Mining (UM) is a sub-area of data mining where there is an interest in patterns, and knowledge about the value or utility of items in databases is identified [1]. UM is entirely different from traditional data mining, and the objective is to examine the utility of items based on their importance in a database [2]. UM searches for itemsets, sequences, or patterns that have some additional measure of utility. It allows decision-makers to process the information, make decisions based on information, and gather information that is relatable to their business [3].

High utility itemset mining (HUIM) is an emerging subfield in data mining that overcomes the limitations of traditional frequent itemset mining. While frequent itemsets are concerned with the presence of items, HUIM introduces utility, that is, the importance or value of each item. This sophisticated approach allows for the identification of groupings of items that together provide significant usefulness, providing a more profound understanding of the dynamics of the dataset [4, 5]. Real-time applications are associated with a wide range of industries, including retail and healthcare, where identifying patterns for more significant economic benefit or practical use is crucial. HUIM provides insights that enable decision-makers to optimize processes, customize tactics, and make informed decisions about the effectiveness of individual items across a dataset [6].

High Utility Itemsets (HUIs) are the combination of patterns mined that offer significant economic or practical implications. These itemsets prioritize items based not only on their frequency but also on their total contribution to usefulness [7]. Organizations benefit from being able to detect the profitable itemset combinations that would lead to a higher economic gain in market basket analysis cases [8]. By extracting HUIs, the relation between items can be thoroughly explained, which eventually leads to deciding with fine granularity. So, the novelty of the research in HUIM can be exploited to find exciting patterns, which, in turn, can lead to more efficient process operations, better resource allocation, and, in general, improved strategic decision-making [9, 10].

Bioinspired or Metaheuristic Optimization Algorithms are derived from nature's exceptional ability to solve intricate issues with efficiency [11]. Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) are algorithms that mimic biological or physical events [12]. These algorithms systematically search solution spaces, continuously adjusting and improving alternative solutions to identify the optimal outcomes. They are very flexible and can be used for many problems across areas, finding solutions to complex optimization problems with many factors and relationships that are non-linear [13, 14]. These algorithms are applied to engineering, logistics, finance, and machine learning to give adaptive and efficient solutions [15].

The integration of cross-entropy (CE), genetic algorithms (GA), and bacterial foraging algorithms (BFA) in HUIM gives an effective and robust method for recognizing patterns. Cross Entropy functions as a genetic operator that directs the process of searching by assessing disparities in distributions. This research improves the efficiency of High Utility Itemset Mining (HUIM) by leveraging the Genetic Algorithms (GA) and Bacterial Foraging Algorithm (BFA) in terms of exploration and exploitation. It aims to overcome the barriers due to early convergence, and at the same time, it searches for the solution space as broadly as possible. Cross Entropy (CE) is used as a genetic operator to enrich optimization. The primary goal is to improve the efficiency and efficacy of HUIM using the complementary capabilities of parallel bioinspired algorithms.

The primary intent is to bring a fundamentally more robust way of extracting utility-driven patterns to the data mining community. This looks at studying particularly those fields where the utility of the objects is at the heart of the economics or pragmatics of the field. The applied optimization techniques aim to accelerate the discovery of valuable item sets, leading to lower computational costs and faster knowledge discovery. The combination of these optimization strategies reduces computing difficulties, resulting in excellent outcomes and enabling the extraction of valuable patterns in various applications. This strategy effectively addresses the issue of premature convergence and enhances the process of identifying patterns that have economic and practical value. It optimizes decision support and facilitates knowledge discovery [16].

The remainder of the article is organized as follows: the overview of HUIM, motivation, and optimization approaches in HUIM identification is explained in Section 1, the recent research in HUIM identification using optimization approaches with its research gap is given in Section 2, materials and methods as well as the proposed Hybridized Cross Entropy Assisted Genetic algorithm with Bacterial Foraging Algorithm (HCEG-BFA) is detailed in Section 3, the experimental outcome and its comparative analysis is illustrated with discussion in Section 4, and the article is concluded with future research direction in Section 5,

2. Related Works

In this paper [17], the authors propose a method called Ant Colony-based High Utility Infrequent Itemset Mining (ACHUIIM) to efficiently mine high utility infrequent itemsets. The approach is based on the behaviour of ants depositing pheromones to find the shortest path to food. Similarly, the algorithm uses pheromone traces to guide the search for infrequent itemsets with high utility. By mimicking the behaviour of ants, ACHUIIM aims to uncover rare cases of highly profitable itemsets.

This paper [18] presents a method based on Binary Particle Swarm Optimization (BPSO) for high utility itemset mining without predefining a minimum utility threshold. Unlike traditional approaches, this method optimizes the search for high utility itemsets dynamically, adjusting the threshold as a post-processing step. The algorithm utilizes transactions with the highest utility as initial populations for particles, enhancing efficiency in discovering high utility itemsets.

In [19], the authors present hill climbing and simulated annealing-based algorithms, namely HUIM using Hill Climber and Simulated Annealing (HUIM-HCSA), for high utility itemset mining. These algorithms are adequate to work on an exponentially sized search space of the database and multiple items on each transaction. These algorithms provide faster speed compared to various heuristic and evolutionary techniques that convert the input database into a bitmap by using evolutionary techniques to enhance population diversity.

This researcher presents two algorithms in [20], Top-K High Utility itemset mining based on the Cross Entropy method (TKU-CE) and TKU-CE+, for heuristically mining the top-k HUIM. To update probabilities of itemsets with high-utility values, TKU-CE employs cross-entropy and combinatorial optimization methods. More effective TKU-CE+ includes a filter and an innovative mutation strategy to prevent it from getting fixed with ineffective items, and it pushes the sampling by sealing it with a mutation operator. The research clearly shows that both algorithms are efficient and scalable; moreover, they are able to mine real top-k high-utility itemsets without setting utility thresholds. The researchers also developed diverse optimisation algorithms [21 and 22].

To reduce the high computational complexity that is needed during HUIM, the authors propose a Pruned Matrix Aided Utility Tree (PMAUT) [23] in this research. PMAUT, with the matrix technique that is pruned primarily and the FP tree is constructed, can decrease the complexity of locating HUIs. The experimental outcome shows that the performance (time and memory usage) of the PMAUT algorithm is more effective than that of the existing techniques. These studies bring out alternate techniques and methodologies of HUIM and handle the challenges of computation complexity, threshold setting and scalability, thus enhancing the knowledge domain of data mining and decision support systems. Table 1 presents various methods, key factors, and research issues discussed above.

Table 1. Comprehensive Analysis of Literature Review

Reference	Algorithm	Inference	Methodology	Advantage	Research Gap
[17]	ACHUIIM	Mimics the behaviour of ants to find infrequent itemsets with high utility	Ant Colony Optimization	Efficiently mines high utility infrequent itemsets by guiding the search using pheromone traces.	Potential scalability issues with large datasets
[18]	BPSO	Dynamically optimizes the search for high utility itemsets without predefined minimum utility thresholds.	Particle Swarm Optimization	Utilizes transactions with the highest utility as initial populations for particles, enhancing efficiency	Further investigation is needed into the algorithm's robustness under various dataset characteristics.
[19]	HUIM-HC and HUIM-SA	Utilizes hill climbing and simulated annealing techniques, respectively, for handling exponential search space	Heuristic Algorithms	Faster performance compared to existing approaches	Evaluation of algorithm performance under different parameter settings
[20]	TKU-CE and TKU-CE+	Uses cross-entropy method for mining top-k high-utility itemsets without utility thresholds	Cross-Entropy Method	Efficiently discovers actual top-k high-utility itemsets without the need for utility thresholds.	Assessment of algorithm scalability with enormous datasets
[21]	HUIM-GA	Evaluates the impact of crossover operators in GAs for HUIM	Headless chicken test to compare GAs with normal and randomized crossover operators	Discovers most HUIs faster than exact algorithms	Performance varies with the presence of well-defined building blocks
[22]	HUIM-PSO	Improved HUIF-PSO with FP-tree structure outperforms existing algorithms	Utilizes frequent pattern tree structure to enhance HUIF-PSO	Better performance in HUI count, memory, and time usage	Requires evaluation in different real-world scenarios
[23]	PMAUT	Minimizes complexity by employing pruning techniques and constructing an FP tree	Pruning Technique with FP Tree Construction	Outperforms existing techniques in terms of time consumption and memory usage	Investigation into potential trade-offs between efficiency and accuracy

The research gap in mining HUIM is the need for more robust and scalable algorithms that can handle the computational complexity during the process of discovering HUIs in extensive transactional databases. The traditional approaches often encounter the challenge of scalability, more time for execution, and sometimes interference, such as parameter optimization, which limits their applicability for real-time scenarios. The optimized BFA, along with other meta-heuristic techniques like GA and CE, use mathematical single information for the solution. Still, the proposed HCEG-BFA-based techniques are a combination of robust Cross Entropy (CE) Method, Genetic Algorithm (GA), and Bacterial Foraging Algorithm (BFA). The hybrid approach improves the exploration but refines solutions iteratively to overcome local optima, resulting in a practical, scalable, and resistant algorithm to find high utility itemsets in large databases.

3. Hybridized Cross Entropy Assisted Genetic algorithm with Bacterial Foraging Algorithm (HCEG-BFA)

During the process of optimization, the values of the Genetic Algorithm (GA) and Bacterial Foraging Algorithm (BFA) are initialized. In contrast, the bacterial positions are generated and act as a solution space across the search space. The positions are created and updated in accordance with the principles of BFA. Pi indicates every position of the bacteria, and it acts as a distinct candidate solution. P= indicates the position of bacteria.$\{P_1, P_2,, \dots \dots, P_N\}$ and the population size of the bacteria is given as N, which is the count of the position of bacteria. In Genetic Algorithm (GA), populations are initialized as G_j, where the whole population is indicated using $G = \{G_1, G_2, \dots \dots \dots, G_N\}$. The process of optimization is initiated with the initialization process. Iterative evolution and enhancement are accomplished using a Hybridized Cross Entropy Assisted Genetic algorithm with Bacterial Foraging Algorithm (HCEG-BFA). The proposed methodology, HCEG-BFA, is given in Figure 1.

3.1. Initialization

The bacterial positions are indicated by $S_i = (s_{i1}, s_{i2}, \dots \dots, s_{iM})$ where M gives the total count of the item. The occurrence of an item in the position of bacteria is p_i, which is shown as sij=1, and the absence is indicated by s_{ij}=0. The binary string S_1, S_2,…..S_{12} is given in Table 3.

Table 3. Binary Representation of Bacteria at Different Positions

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
P_1	1	1	0	1	0	1	1	0	0
P_2	0	0	0	1	0	0	0	1	0
P_3	1	1	0	0	0	1	1	1	0
P_4	0	1	1	1	1	1	0	0	0
P_5	0	0	0	1	0	1	0	0	0
P_6	0	1	1	0	1	1	0	0	0
P_7	1	0	0	0	0	0	1	0	0
P_8	0	0	1	0	1	0	1	0	0
P_9	1	1	1	1	1	0	1	0	0
P_{10}	0	1	0	1	1	0	1	0	0
P_{11}	0	0	0	0	0	1	1	0	0
P_{12}	0	1	1	0	1	0	0	0	1

3.2. Fitness Evaluation

The bacterial position p_i of fitness function $F(p_i)$ is estimated as the total utilities of the items indicated by p_i. Let M be the total count of items and the binary digits s_{ij} of the item in the position of bacterial position p_i. The utility of every item is indicated by utility (j). The fitness value is given in Equation 1.

$$F(p_i) = \sum_{j=1}^{M} s_{ij} \times Utility(j) \text{ ---------(1)}$$

The above formula estimates the fitness value by adding the product of every binary digit s_{ij} and the relevant jth items utility value. It is used to evaluate the bacterial positions that contribute to the whole utility value acquired from the combination of items in the population space. The higher fitness value indicates an efficient solution with potentially effective HUIs. The fitness value of bacterial positions is given in Table 4.

Table 4. Fitness Value of Bacterial Position

Bacterial Position	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	$Total_{fitness}$
Fitness Value	18	7	13	22	13	14	5	9	18	19	5	7	150

Example: Bacterial Position (p_1) = (1,2,0,2,0,1,2,0,0)

$F(p_1)$=(1*Utility(I_A))+(2*Utility(I_B))+(0*Utility(I_C))+(2*Utility(I_D))+(0*Utility(I_E))+(1*Utility(I_F))+(2*Utility(I_G))+(0*Utility(I_H))+(0 *Utility(I_I))

$F(p_1)=(1*1)+(2*2)+(0*1)+(2*5)+(0*4)+(1*3)+(2*1)+(0*2)+(0*1)$

$F(p_1)=18$

The summation of the fitness value of all bacterial positions to acquire the total fitness using Equation 2.

$$Total_{fitness} = \sum_{j=1}^{N} F(P_j) \text{---------(2)}$$

3.3. Selection Mechanism

For every bacterial position (p_1), estimate its selection probability $P(p_i)$ utilizing Equation 3.

$$P(p_i) = \frac{F(p_i)}{\sum_{j=1}^{N} F(p_j)} \text{---------(3)}$$

Example: $P(p_1) = \frac{F(p_1)}{\sum_{j=1}^{N} F(p_j)} = \frac{18}{150} \approx 0.12$

The selection probability for every bacterial position ($p_1, p_2, \ldots . p_{12}$) is given as 0.12, 0.0467, 0.0867, 0.1467, 0.0867, 0.0933, 0.0333, 0.06, 0.12, 0.1267, 0.0333, and 0.0467.

3.4. Genetic Operators

The solution space is effectively exploited using the prominent operations mutation and crossover that has the exploration and exploitation ability in solution space. These operators are responsible for generating the new itemset by modifying and combining the existing ones. Genetic recombination is accomplished using a genetic operator called crossover, where it exchanges the genetic data among the two parent solutions to generate offspring solutions. In the hybridized BFA with GA, crossover operates over the bacterial position pairs indicated as binary strings.

Typically, it selects the crossover points from the binary strings and the substrings are swapped beyond the crossover point amongst two parent solutions. The exchange of genetic data generates two offspring solutions with the combination of personalities from the parents. The processes of crossover are given in Equation 4 and 5.

$$Offspring_1 = Substring_1^{Parent_1} + Substring_2^{Parent_2}\text{---------(4)}$$

$$Offspring_2 = Substring_1^{Parent_2} + Substring_2^{Parent_1}\text{---------(5)}$$

where $Substring_1^{Parent_1}$ indicates parent₁'s substring of the binary strings before the point of crossover, $Substring_2^{Parent_2}$ indicates parent₁'s substring of the binary strings before the point of crossover and vice versa for the subsequent string.

Example: The bacterial positions of the parents are parent1 is {1101001100} and parent2 is {0011110101}. The crossover is performed on every pair is given as $Substring_1^{Parent_1}$ is {1101}, $Substring_2^{Parent_2}$ is {1011}, $Substring_1^{Parent_2}$ is {0011}, and $Substring_2^{Parent_1}$ is {1001}. The acquired offsprings are $Offspring_1$ is {11011011}, and $Offspring_2$ is {00111001}. The same process is applied to all the parent solutions, and the solution is acquired for all the bacterial positions.

The random alteration in the genes of the genetic operator is accomplished using the mutation operation. It perturbs the positions of bacteria indicated as binary strings, which permits the identification of new regions in the solution space. The process of mutation is given in Equation 6.

$$Mutated_{solution} = Original\ Solution \oplus Mutation\ Mask\text{---------(6)}$$

where the bitwise XOR operation is given amongst the mutation mask, and the original solution is indicated as ⊕. The mutation mask encompasses bits, which are randomly created bits to be mutated in the solution space.

Example: The probability of mutation is given as 0.1 (10%), and every string has a 10% chance of mutation. For masking, a random number is created that is less than the probability. For the original binary string 110101100, the randomly generated mutation mask is 001010100, whereas the mutated solution is 111111000. The process of candidate generation from the mutated binary strings is {I_A}, {I_B}, {I_D}, {I_E}, {I_F}, {I_G}, {I_A, I_B}, {I_A, I_D}, {I_A, I_E}, {I_A, I_F}, {I_A, I_G}, {I_B, I_D}, {I_B, I_E}, {I_B, I_F}, {I_B, I_G}, {I_D, I_E}, {I_D, I_F}, {I_D, I_G}, {I_E, I_F}, {I_E, I_G}, {I_F, I_G}, {I_A, I_B, I_D}, {I_A, I_B, I_E}, {I_A, I_B, I_F}, {I_A, I_B, I_G}, {I_A, I_D, I_E}, {I_A, I_D, I_F}, {I_A, I_D, I_G}, {I_A, I_E, I_F}, {I_A, I_E, I_G}, {I_A, I_F, I_G}, {I_B, I_D, I_E}, {I_B, I_D, I_F}, {I_B, I_D, I_G}, {I_B, I_E, I_F}, {I_B, I_E, I_G}, {I_B, I_F, I_G}, {I_D, I_E, I_F}, {I_D, I_E, I_G}, {I_D, I_F, I_G}, {I_E, I_F, I_G}, {I_A, I_B, I_D, I_E }, {I_A, I_B, I_D, I_F}, {I_A, I_B, I_D, I_G}, {I_A, I_B, I_E, I_F}, {I_A, I_B, I_E, I_G}, {I_A, I_B, I_F, I_G}, {I_A, I_D, I_E, I_F}, {I_A, I_D, I_E, I_G}, {I_A, I_D, I_F, I_G}, {I_A, I_E, I_F, I_G}, {I_B, I_D, I_E, I_F}, {I_B, I_D, I_E, I_G}, {I_B, I_D, I_F, I_G}, {I_B, I_E, I_F, I_G}, { I_D, I_E, I_F, I_G}, {I_A, I_B, I_D, I_E, I_F}, {I_A, I_B, I_D, I_E, I_G}, {I_A, I_B, I_D, I_F, I_G}, {I_A, I_B, I_E, I_F, I_G}, {I_A, I_D, I_E, I_F, I_G }, {I_B, I_D, I_E, I_F, I_G}, {I_A, I_B, I_D, I_E, I_F, I_G}.

3.5. BFA Dynamics

The probability distribution is initialized using the cross-entropy method, which is based on the current population when identifying the HUIs. The primary intent of this update is to alter the probability, and the exploration is achieved with the support of a higher fitness value. The process is carried iteratively to calculate the probability distribution. This adaptive framework assists in efficient guiding towards the near-optimal or optimal solution. The probability distribution initialization is given in Equation 7.

$$P^{(0)}=\frac{1}{N}\sum_{i=1}^{N}P_i^{(0)}\text{---------}(7)$$

where the initial value of probability distribution is given as $P^{(0)}$, size of the population is given as N, and the ith solution binary string in the solution space is given as $P_i^{(0)}$.

The probability is evaluated using the BFA fitness function and updated using Equation 8.

$$\mathrm{P}^{(t+1)}=\frac{\sum_{i=1}^{N}\mathrm{P}_i^{(t)}\cdot \mathrm{I}\left(\mathrm{F}\left(\mathrm{P}_i^{(t)}\right)\right)>\text{threshold}}{\sum_{i=1}^{N}\mathrm{I}\left(\mathrm{F}\left(\mathrm{P}_i^{(t)}\right)\right)>\text{threshold}}\text{---------}(8)$$

where the updated probability for the iterations t+1 is given as $\mathrm{P}^{(t+1)}$, the binary string of the solution at the tth iteration is given as $\mathrm{P}_i^{(t)}$, the fitness value is $\mathrm{F}\left(\mathrm{P}_i^{(t)}\right)$, and the indicator function is $\mathrm{I}(\cdot)$. This function gives 1 if it is true and 0 for false.

The new solutions are sampled for the subsequent generation, and the process is carried iteratively. The iteration is continued till it reaches the convergence. The cross-entropy values are updated dynamically and give higher fitness values.

Example: The population size is 12, and the population initialization is $\mathrm{P}^{(0)}=\{\mathrm{P}_1^{(0)},\mathrm{P}_2^{(0)},\dots\dots,\mathrm{P}_{12}^{(0)}\}$. The fitness of the population is given as $\{\mathrm{F}\left(\mathrm{P}_1^{(0)}\right)=18,\mathrm{F}\left(\mathrm{P}_2^{(0)}\right)=7,\mathrm{F}\left(\mathrm{P}_3^{(0)}\right)=13,\mathrm{F}\left(\mathrm{P}_4^{(0)}\right)=22,\mathrm{F}\left(\mathrm{P}_5^{(0)}\right)=13,\mathrm{F}\left(\mathrm{P}_6^{(0)}\right)=14,\mathrm{F}\left(\mathrm{P}_7^{(0)}\right)=5,\mathrm{F}\left(\mathrm{P}_8^{(0)}\right)=9,\mathrm{F}\left(\mathrm{P}_9^{(0)}\right)=18,\mathrm{F}\left(\mathrm{P}_{10}^{(0)}\right)=19,\mathrm{F}\left(\mathrm{P}_{11}^{(0)}\right)=5,\mathrm{F}\left(\mathrm{P}_{12}^{(0)}\right)=7$. The threshold value assigned is 10, and the acquired solution space is $\{\mathrm{F}\left(\mathrm{P}_1^{(0)}\right)=18,,\mathrm{F}\left(\mathrm{P}_3^{(0)}\right)=13,\mathrm{F}\left(\mathrm{P}_4^{(0)}\right)=22,\mathrm{F}\left(\mathrm{P}_5^{(0)}\right)=13,\mathrm{F}\left(\mathrm{P}_6^{(0)}\right)=14,\mathrm{F}\left(\mathrm{P}_9^{(0)}\right)=18,\mathrm{F}\left(\mathrm{P}_{10}^{(0)}\right)=19$. The updated value is given as $\mathrm{P}^{(t+1)}\approx(0,0.142857,0,0.142857,0,0,0.142857,0.142857,0.142857,0)$. Based on the probability threshold, the non-profitable itemsets are eliminated and the reduced set is $\{I_A, I_D\}$, $\{I_A, I_G\}$, $\{I_B, I_D\}$, $\{I_D, I_G\}$, $\{I_E, I_F\}$, $\{I_E, I_G\}$, $\{I_F, I_G\}$, $\{I_A, I_B, I_D\}$, $\{I_A, I_B, I_E\}$, $\{I_A, I_B, I_F\}$, $\{I_A, I_B, I_G\}$, $\{I_A, I_D, I_E\}$, $\{I_A, I_D, I_F\}$, $\{I_A, I_D, I_G\}$, $\{I_A, I_E, I_F\}$, $\{I_A, I_E, I_G\}$, $\{I_A, I_F, I_G\}$, $\{I_B, I_D, I_E\}$, $\{I_B, I_D, I_F\}$, $\{I_B, I_D, I_G\}$, $\{I_B, I_E, I_G\}$, $\{I_B, I_F, I_G\}$, $\{I_D, I_E, I_F\}$, $\{I_D, I_E, I_G\}$, $\{I_D, I_F, I_G\}$, $\{I_E, I_F, I_G\}$, $\{I_A, I_B, I_D, I_E\}$, $\{I_A, I_B, I_D, I_F\}$, $\{I_A, I_B, I_D, I_G\}$, $\{I_A, I_B, I_E, I_F\}$, $\{I_A, I_B, I_E, I_G\}$, $\{I_A, I_B, I_F, I_G\}$, $\{I_A, I_D, I_E, I_F\}$, $\{I_A, I_D, I_E, I_G\}$, $\{I_A, I_D, I_F, I_G\}$, $\{I_A, I_E, I_F, I_G\}$, $\{I_B, I_D, I_E, I_F\}$, $\{I_B, I_D, I_E, I_G\}$, $\{I_B, I_D, I_F, I_G\}$, $\{I_B, I_E, I_F, I_G\}$, $\{I_D, I_E, I_F, I_G\}$, $\{I_A, I_B, I_D, I_E, I_F\}$, $\{I_A, I_B, I_D, I_E, I_G\}$, $\{I_A, I_B, I_D, I_F, I_G\}$, $\{I_A, I_B, I_E, I_F, I_G\}$, $\{I_A, I_D, I_E, I_F, I_G\}$, $\{I_B, I_D, I_E, I_F, I_G\}$, $\{I_A, I_B, I_D, I_E, I_F, I_G\}$.

3.6. Iteration Refinement

The iteration is continued via all the above steps for a distinct count of generation or criterion of convergence met. Each iteration fine-tunes the population's ability to identify and indicate HUIs, which can enhance the overall performance of the proposed algorithm. The process of proposed HCEG_BFA is given in Algorithm 1.

Algorithm 1: HCEG-BFA for High Utility Itemset Discovery

```
1. InitializePopulation()
2. EvaluateFitness()
3. Set iteration = 0
4. while not ConvergenceCriteriaMet or iteration < MaxIterations do
5.     Chemotaxis()
6.     Reproduction()
7.     EliminationDispersal()
8.     EvaluateFitness()
9.     UpdateProbabilityDistribution()
10.    Sampling()
11.    increment iteration
12. end while
13. return BestSolutionFound
```

4. Result and Discussion

This section deals with the experimental discussion on diverse datasets, and it illustrates with detailed comparative analysis. The description of the dataset and experimental setup are also discussed. The proposed HCEG-BFA is executed on four real-time and benchmark datasets, namely mushroom, chess, retail, and foodmart. The proposed approach HCEG-BFA is compared with existing HUIM techniques, namely HUIM-BPSO [18], HUIM-HCSA [19], HUIM-GA [21], and HUIM-PSO [22].

4.1. Dataset Description and Experimental Setup

The proposed and existing techniques are implemented on a computer system with CPU 3.600 GHz 8 Core and Windows 10 64 GB RAM with 64 bits. The implementation is carried out in a Java net beans environment. The research work utilized publicly available and standard datasets. The dataset Foodmart is sparse, and Ecommerce is dense, with real-time utility values, whereas the remaining holds the synthetic utility value [25]. The characteristics of the dataset are given in Table 5.

Table 5. Dataset Description

Dataset	Type	Average Transaction Length	Transaction	Item
Chess	Dense	37	3,196	75
E-commerce	Dense	11.71	14,975	3,468
Foodmart	Sparse	4.42	4,141	1,559
Mushroom	Dense	23	8,416	119

The dataset utilized in this research is acquired from the public data mining library. The transaction about the retail store is given in the Foodmart dataset, the steps involved in the chess game are given in the Chess dataset, the characteristics of mushroom species are given in the mushroom dataset, and the transactions that happened in the online store are given in E-commerce dataset. For the utility mining experiments, the termination criterion for the algorithms is assigned as 10,000 iterations, and the size of the initial population is given as 30. The threshold value α is given with diverse ranges.

4.2. Performance Metrics

The performance of the proposed HCEG-BFA and existing techniques, namely HUIM-BPSO, HUIM-HCSA, HUIM-GA, and HUIM-PSO, are evaluated using the performance metrics, namely memory usage, execution time, HUI count, and convergence. The amount of memory utilized by the HUIM technique during the process of HUI identification is stated as memory usage. Memory usage can be measured in megabytes and bytes. The time used to complete the implementation of HUIM algorithms to retrieve the HUIs is termed runtime. The runtime is measured in milliseconds or seconds. Convergence monitors the performance nature of HUIM algorithms over different iterations and monitors for change or stabilized performance where it reaches some even solution over a predefined count of iterations. The count of HUIs retrieved by the HUIM technique is recorded as HUI count. The memory usage and runtime are given in Equations 9 and 10.

$$\text{Memory}_{\text{usage}} = \text{Memory}_{\text{afterexecution}} - \text{Memory}_{\text{beforexecution}} \text{---------}(9)$$

$$\text{Runtime} = \text{Time}_{\text{Completion}} - \text{Time}_{\text{start}} \text{---------}(10)$$

4.3. Experimental Discussion

Table 6. Comparison of Runtime in Seconds

Dataset	Minimum Threshold Value	HUIM-PSO	HUIM-BPSO	HUIM-GA	HUIM- HCSA	HCEG-BFA
Mushroom	20k	623.56	567.56	453.67	367.78	155. 46
	60k	600.67	483.37	413.45	321.67	105.78
	100k	581.67	442.78	389.81	300.67	77.56
	140k	552.34	418.75	324.67	287.56	59.67
	180k	500.78	387.56	298.68	265.78	36.68
	220k	473.67	361.25	246.87	215.88	30.09
	260k	423.78	289.56	219.56	180.67	27.69
	300k	389.67	238.76	182.78	154.45	21.56
Chess	20k	652.87	612.45	586.56	534.56	123.67
	60k	634.12	594.89	542.78	489.46	108.56
	100k	587.32	563.94	512.56	432.78	99.56
	140k	521.78	512.56	498.45	400.77	69.46
	180k	457.12	467.89	453.67	352.45	53.79
	220k	387.98	368.67	356.23	319.78	44.56
	260k	341.98	311.87	289.56	253.56	39.73
	300k	318.9	298.92	252.89	214.46	34.89
	340k	256.98	251.78	229.78	177.56	31.45
	380k	213.78	198.67	186.76	143.56	29.56

	400k	190.56	182.56	181.45	136.67	25.56
Foodmart	2.5k	38.56	14.99	21.56	33.56	5.02
	3.5k	33.45	15.09	21.62	35.2	5.12
	4.5k	29.45	15.24	22.67	36.9	5.2
	5.5k	28.09	15.39	23.1	37.56	5.4
	6.5k	26.67	15.45	23.89	37.98	5.61
	7.5k	25.98	15.57	24.32	39.12	5.7
	8.5k	24.67	15.62	24.81	39.76	5.75
	9.5k	23.45	15.69	25.12	39.93	5.79
	10.5k	21.56	15.81	26.34	41.23	5.85
	11.5k	19.56	15.91	27.45	43.45	5.99
	12.5k	17.45	16	28.46	44.67	6.08
E-commerce	20k	62.556	51.016	66.149	59.3297	26.664
	60k	62.575	57.682	62.289	52.2134	27.473
	100k	73.184	44.156	58.714	59.1536	25.369
	140k	63.327	50.892	71.082	56.3317	29.251
	180k	66.841	54.086	60.505	68.0221	27.844
	220k	62.517	61.454	64.586	59.0026	27.162
	260k	78.799	71.512	67.086	59.9599	26.080
	300k	60.199	65.371	70.741	53.1328	25.946
	340k	75.103	44.088	61.258	53.3603	29.769
	380k	81.897	57.864	58.193	49.3999	26.702
	400k	77.449	50.640	51.588	53.9557	25.324

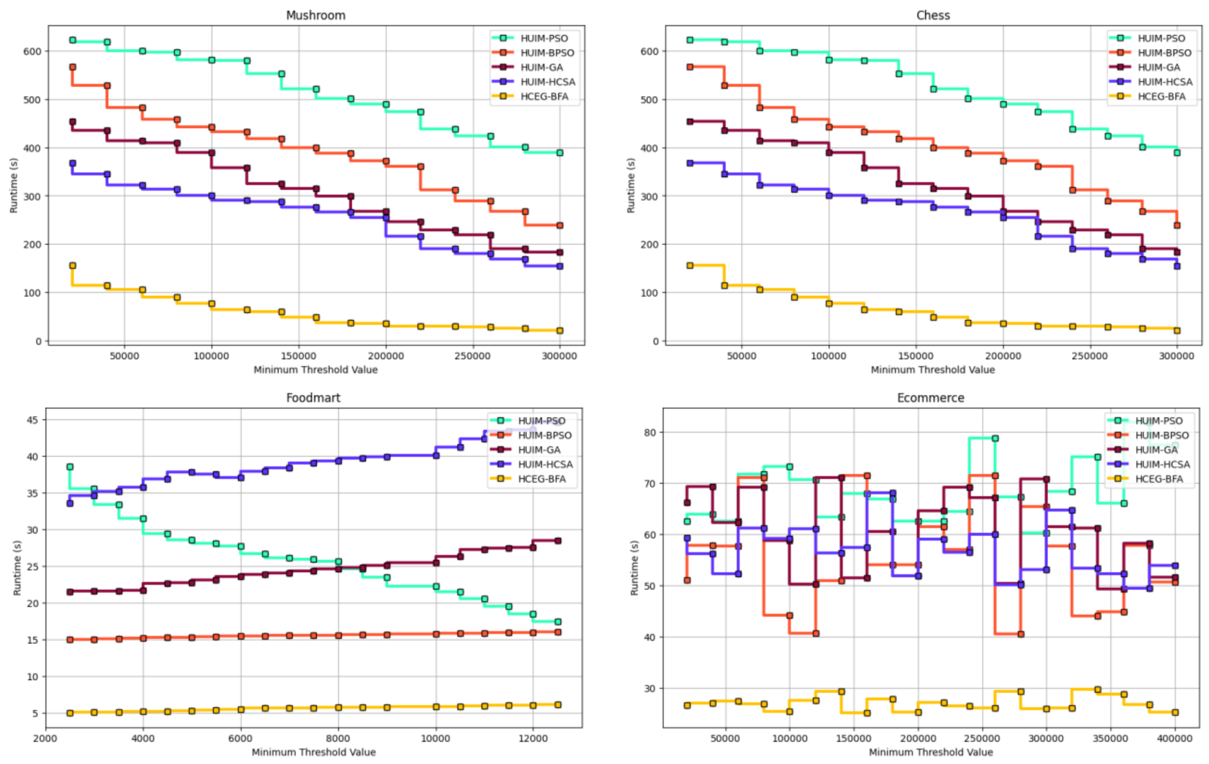

Figure 1. Comparison of Performance of Runtime for Different Datasets with Minimum Threshold Values

After analyzing the average runtimes across various minimum threshold values for each algorithm across all datasets, the results indicate notable differences in their performance. On average, HCEG-

BFA exhibited the fastest execution time, with an average runtime of 25.48 seconds across all datasets. HUIM-BPSO followed closely behind with an average runtime of 45.24 seconds, while HUIM-HCSA and HUIM-PSO showed slightly longer average runtimes of 47.92 and 48.97 seconds, respectively. HUIM-GA demonstrated the most extended average runtime among the algorithms, with an average of 51.69 seconds. The comparative analysis further revealed that HCEG-BFA consistently outperformed other algorithms across all three datasets, displaying the shortest average runtime. Specifically, in the Chess dataset, HCEG-BFA showcased the lowest average runtime, closely trailed by HUIM-HCSA and HUIM-BPSO. These findings suggest that HCEG-BFA stands out as the most efficient algorithm for association rule mining tasks across diverse datasets, offering significantly faster execution times compared to other algorithms.

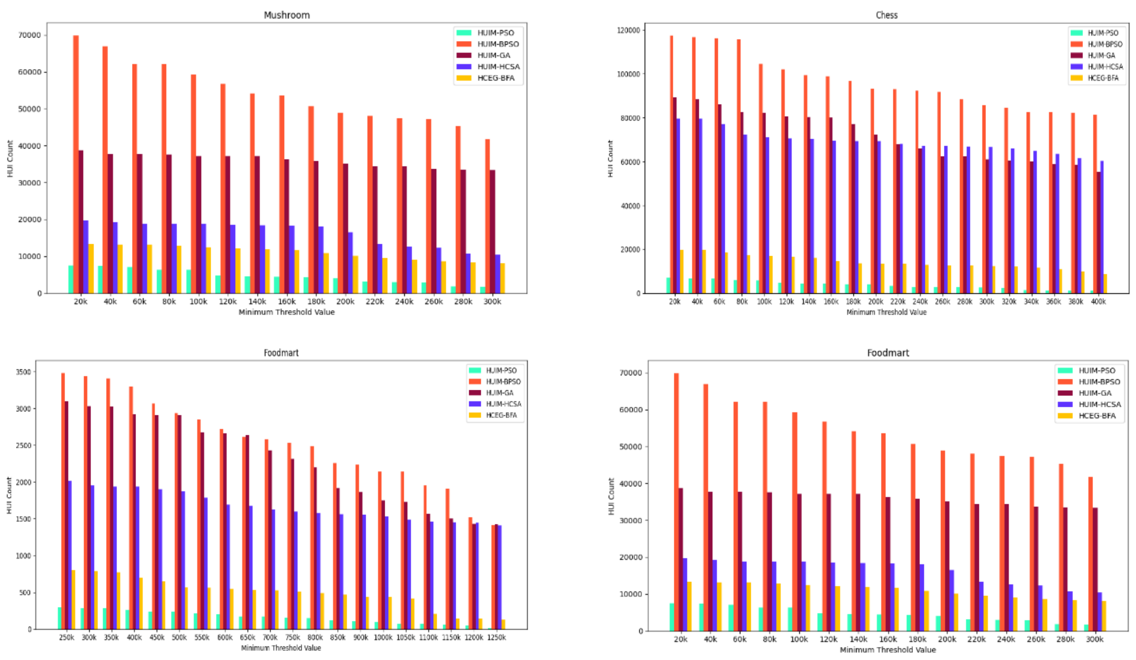

Figure 2. Comparison of Discovered HUIs

For instance, at a 20k minimum threshold, HCEG-BFA produces 13,272 frequent itemsets, compared to 7,558 by HUIM-PSO, 69,759 by HUIM-BPSO, 38,682 by HUIM-GA, and 19,653 by HUIM-HCSA. This trend persists across varying thresholds, indicating the superior performance of HCEG-BFA in terms of both efficiency and HUI count. Similarly, in the Chess dataset, HCEG-BFA outperforms other algorithms in runtime efficiency while generating competitive HUI counts. For example, at a 20k minimum threshold, HCEG-BFA produces 19,844 frequent itemsets, compared to 6,941 by HUIM-PSO, 117,381 by HUIM-BPSO, 89,276 by HUIM-GA, and 79,674 by HUIM-HCSA. HCEG-BFA consistently exhibits the lowest memory consumption across Mushroom, Foodmart, and Ecommerce datasets, with values between 82.94 MB to 98.91 MB, 25.12 MB to 76.11 MB, and 43.84 MB to 269.02 MB respectively, while maintaining competitive HUI counts and runtime efficiency, underscoring its robust performance.

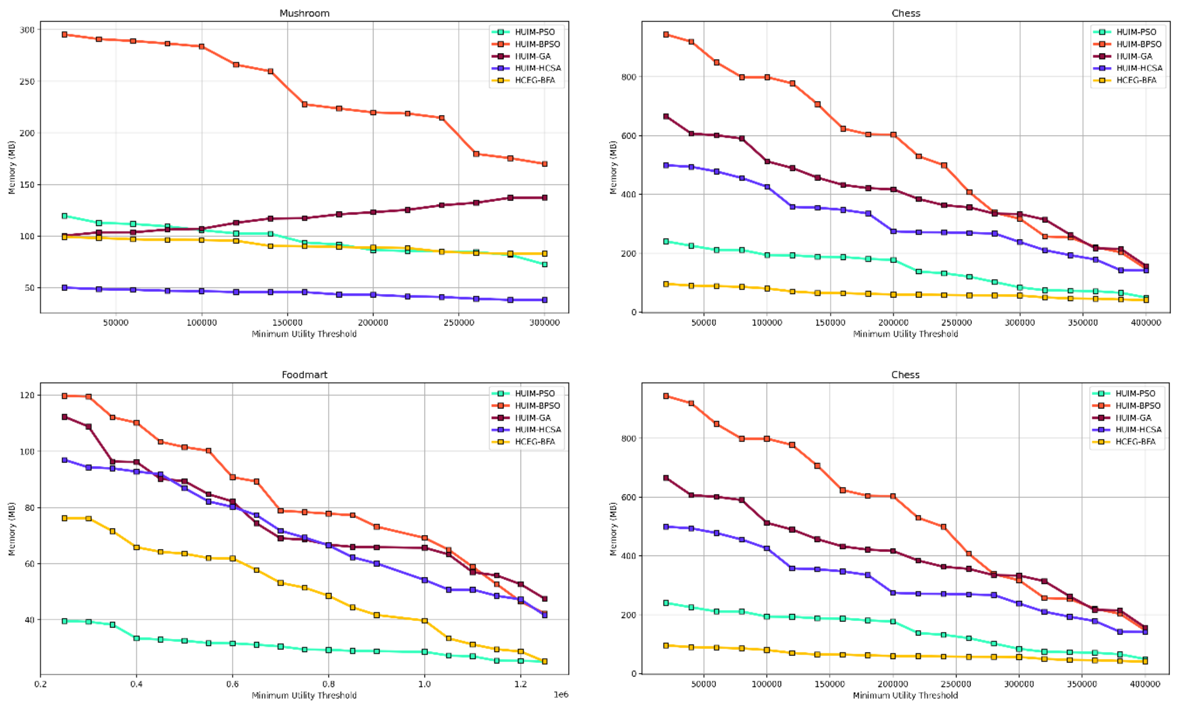

Figure 3. Comparison of Memory Usage

Analyzing memory consumption across diverse datasets and algorithms reveals notable efficiency and performance. For all datasets, the Hybridized Cross Entropy Assisted Genetic algorithm with Bacterial Foraging Algorithm (HCEG-BFA) consistently shows lower memory utilization compared to Particle Swarm Optimization (PSO), Binary Particle Swarm Optimization (BPSO), Genetic Algorithm (GA), and Hybrid Crossover-based Semi-Genetic Algorithm (HCSA). HCEG-BFA maintains memory consumption below 100 MB across the Mushroom, Foodmart, and E-commerce datasets. In the Mushroom dataset, HCEG-BFA's memory values range from 82.94 MB to 98.91 MB; in Foodmart, 25.12 MB to 76.11 MB; and in E-commerce, 43.84 MB to 269.02 MB, highlighting its effectiveness in memory optimization.

5. Conclusion

Cross Entropy-assisted metaheuristic optimization, specifically HCEG-BFA, has proven to be highly effective for high utility itemset mining (HUIM) across various transactional databases. Our analysis of datasets like Mushroom, Chess, Foodmart, and E-commerce indicates that HCEG-BFA consistently exhibits superior performance in terms of memory efficiency and runtime compared to other algorithms such as PSO, BPSO, GA, and HCSA. Notably, HCEG-BFA maintains memory consumption under 100 MB in all examined datasets, showcasing its ability to manage resources effectively. This robustness makes it an ideal candidate for real-world applications where computational efficiency is critical. The consistent performance of HCEG-BFA across diverse datasets underscores its versatility and potential as a powerful tool in the domain of HUIM. Future work could focus on integrating association rule generation with HUIM, leveraging HCEG-BFA to not only identify high utility itemsets but also derive meaningful rules, thereby enhancing its applicability in decision-making processes.

Reference

[1] Chen, J., Yang, S., Ding, W., Li, P., Liu, A., Zhang, H., & Li, T. (2024). Incremental high average-utility itemset mining: survey and challenges. *Scientific Reports*, *14*(1), 9924.

[2] Cheng, Z., Fang, W., Shen, W., Lin, J. C. W., & Yuan, B. (2023). An efficient utility-list based high-utility itemset mining algorithm. *Applied Intelligence*, *53*(6), 6992-7006.

[3] Haipeng, J., Guoqing, W., Mengdan, S., Feng, L., Yunfei, S., & Wei, F. (2024). PHUI-GA: GPU-based efficiency evolutionary algorithm for mining high utility itemsets. *Journal of Systems Engineering and Electronics*.

[4] Tharini, V. J., & Shivakumar, B. L. (2022). High-utility itemset mining: fundamentals, properties, techniques and research scope. In *Computational intelligence and data sciences* (pp. 195-210). CRC Press.

[5] GAO, Z., HAN, M., LIU, S., LI, A., & MU, D. (2023). Survey of high utility itemset mining methods based on intelligent optimization algorithm. *Journal of Computer Applications*, *43*(6), 1676.

[6] Chen, J., Yang, S., Ding, W., Li, P., Liu, A., Zhang, H., & Li, T. (2024). Incremental high average-utility itemset mining: survey and challenges. *Scientific Reports*, *14*(1), 9924.

[7] 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.

[8] Tung, N. T., Nguyen, T. D. D., Nguyen, L. T., Tho, Q. T., & Mai, A. (2023). A Nature-Inspired Method to Mine Top-k Multi-Level High-Utility Itemsets. *Cybernetics and Systems*, 1-22.

[9] Vijayarani, S., Sivamathi, M. C., & Tharini, M. V. J. (2018). Comparative Analysis of On-Shelf Utility Mining Algorithm.

[10] Boukhalat, A., Heraguemi, K., Benouis, M., Akhrouf, S., & Bouderah, B. (2022, September). A Survey on Using Evolutionary Approaches-based high-utility itemsets Mining. In *Artificial Intelligence Doctoral Symposium* (pp. 43-57). Singapore: Springer Nature Singapore.

[11] GAO, Z., HAN, M., LIU, S., LI, A., & MU, D. (2023). Survey of high utility itemset mining methods based on intelligent optimization algorithm. *Journal of Computer Applications*, *43*(6), 1676.

[12] Kumar, R., & Singh, K. (2023). High utility itemsets mining from transactional databases: a survey. *Applied Intelligence*, *53*(22), 27655-27703.

[13] Yang, H., Lu, Y., & Zhang, B. (2020, October). A Survey of High-utility Itemsets Mining. In *Journal of Physics: Conference Series* (Vol. 1624, No. 3, p. 032034). IOP Publishing.

[14] NELLUTLA, A., & Srinivasan, N. (2022). A Survey on Analysis of Data Mining Algorithms for High Utility Itemsets. *El-Cezeri*, *9*(3), 1085-1100.

[15] Zhang, Q., Fang, W., Sun, J., & Wang, Q. (2019). Improved genetic algorithm for high-utility itemset mining. *IEEE Access*, *7*, 176799-176813.

[16] Vijayarani, S., Tharini, V. J., & Sivamathi, C. (2019). Business intelligence for evaluating the intangible benefits of on-shelf high utility itemset from the temporal transaction database. *International Journal of Engineering and Advanced Technology (IJEAT)*, *8*(6).

[17] Arunkumar, M. S., Suresh, P., & Gunavathi, C. (2020). High utility infrequent itemset mining using a customized ant colony algorithm. *International Journal of Parallel Programming*, *48*(5), 833-849.

[18] Gunawan, R., Winarko, E., & Pulungan, R. (2020). A BPSO-based method for high-utility itemset mining without minimum utility threshold. *Knowledge-Based Systems*, *190*, 105164.

[19] Nawaz, M. S., Fournier-Viger, P., Yun, U., Wu, Y., & Song, W. (2021). Mining high utility itemsets with hill climbing and simulated annealing. *ACM Transactions on Management Information System (TMIS)*, *13*(1), 1-22.

[20] Song, W., Zheng, C., Huang, C., & Liu, L. (2022). Heuristically mining the top-k high-utility itemsets with cross-entropy optimization. *Applied Intelligence*, *52*(15), 17026-17041.

[21] Nawaz, M. S., Fournier-Viger, P., Song, W., Lin, J. C. W., & Noack, B. (2021). Investigating crossover operators in genetic algorithms for high-utility itemset mining. In *Intelligent Information and Database Systems: 13th Asian Conference, ACIIDS 2021, Phuket, Thailand, April 7–10, 2021, Proceedings 13* (pp. 16-28). Springer International Publishing.

[22] 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.

[23] Tharini, V. J., & Shivakumar, B. L. (2023). An efficient pruned matrix aided utility tree for high utility itemset mining from transactional database. *International Journal of Intelligent Systems and Applications in Engineering*, *11*(4s), 46-55.

[24] https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php