JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

SHUFFLED COMPLEX EVOLUTION OF ARTIFICIAL BEE COLONY ALGORITHM BASED HIGH UTILITY ITEMSETS MINING

Dr. S Thirumaran[1], Dr. T Anitha[2]
[1st] Assistant Professor, Department of Computer Applications,
Alagappa Government Arts College, Karaikudi, India.
thirumaran.s@gmail.com
[2nd] Assistant Professor, Department of Mathematics , Alagappa Chettiar Government College of Engineering and Technology, Karaikudi.

Abstract

High utility itemset mining is a recent research area in data mining. It attracts many researchers as it extracts profitable products from a database. It considers both the quantity and profit of products. Several algorithms have been presented to mine high-utility itemsets. There are some bio-inspired algorithms like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO) algorithms are also used to retrieve high utility itemsets. In this paper a novel Shuffled Complex Evolution of Artificial Bee Colony (SCE-ABC) algorithm is used to retrieve high utility itemsets. In the SCE-ABC, a population of points is sampled randomly in the feasible space. Then the population is partitioned into several complexes, which are made to evolve based on ABC. At periodic stages in the evolution, the entire population is shuffled and points are reassigned to complexes. The performance of the SCE-ABC algorithm is compared to the existing SCE-ABC,ABC, PSO and GA algorithms and results are compared. A benchmark dataset, mushroom and foodmart datasets are used for performance analysis. Execution times of the algorithms and the memory space used by the algorithms are considered performance factors.

Keywords: Artificial Bee Colony, High Utility itemsets, Shuffled Complex Evolution, Utility mining.

1. Introduction

Association rule mining was a popular data mining techniques used in business analysis. It finds frequently occurring items in a transaction database, based on the occurrences of the items in a database. It does not consider the profit or quantity of the items. For many decision making process, the profit and quantity of the items are also highly important. Utility mining is another research area in data mining, which considers both the profit and quantity of the items. It extracts high profitable items from a quantitative database. An item, whose utility value is greater than mentioned threshold, is known as high profitable item. Chu et al first proposed the concept of utility mining problem instead of frequent itemsets mining[1]. Liu et al discovered the high utility itemsets, by keeping quantity of item as the internal utility and the unit profit of items as the external utility[2]. According to Liu et al, the product of internal and external utility is termed as utility function of that item. There are some bio-inspired algorithms like Genetic Algorithm (GA), particle swarm optimization (PSO), and Ant Colony Optimization (ACO) algorithms are also used to retrieve high utility itemsets. A shuffled complex evolution of Particle Swarm Optimization (PSO) algorithm called SCE-PSO was proposed by Yan and his co-authors in 2007 [3]. The SCE- PSO algorithm was used to mine high utility item sets from a transaction database. In SCE-PSO a population of points is sampled randomly in the feasible space. Then the population is partitioned into several complexes, which is made to evolve based on PSO. From the experiments, it was found that the SCE-PSO based approach was better than the GA and PSO algorithms.

The Artificial Bee Colony (ABC) based optimization algorithm is also one of bio-inspired algorithm, exhorted from the foraging behaviour of honey bees in searching foods. The ABC algorithm is a swarm based meta-heuristic algorithm that was introduced by Karaboga in 2005 for optimizing numerical problems[4]. It was inspired by the intelligent foraging behavior of honey bees. The algorithm is specifically based on the model proposed by Tereshko and Loengarov in 2005 for the foraging behaviour of honey bee colonies[5]. The model consists of three essential components: employed and unemployed foraging bees, and food sources. The first two components, employed and unemployed foraging bees, search for rich food sources, which is the third component, close to their hive. The model also defines two leading modes of behaviour which are necessary for self-organizing and collective intelligence: recruitment of foragers to rich food sources resulting in positive feedback and abandonment of poor sources by foragers causing negative feedback. The advantages of ABC algorithms are it employs only three control parameters (population size, maximum cycle number and limit) that are to be predetermined by the user, quite simple, flexible and robust[6].

Based on the advantages of ABC algorithms, in this paper, a novel shuffled complex evolution of ABC algorithm called SCE-ABC is proposed to retrieve the high-utility itemsets. The difference between ABC and SCE-ABC is that, in SCE-ABC the population of points is sampled randomly in the feasible space. In ABC only a part of the honey bees is considered for its position update based on the visual information and the quality of the new food source. But in SCE-ABC, the neighborhoods are constantly changing, according to the fitness development for the individual honey bees. Thus SCE-ABC combines the advantages of the ABC and the concept of complex shuffling.

2. Artificial Bee Colony Algorithm

In ABC, the colony of artificial bees contains three groups of bees: employed bees associated with specific food sources, onlooker bees watching the dance of employed bees within the hive to choose a food source, and scout bees searching for food sources randomly[6]. Both onlookers and scouts are also called unemployed bees. Initially, all food source positions are discovered by scout bees. Thereafter, the nectar of food sources are exploited by employed bees and onlooker bees, and this continual exploitation will ultimately cause them to become exhausted. Then, the employed bee which was exploiting the exhausted food source becomes a scout bee in search of further food sources once again. In other words, the employed bee whose food source has been exhausted becomes a scout bee. In ABC, the position of a food source represents a possible solution to the problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of employed bees is equal to the number of food sources (solutions) since each employed bee is associated with one and only one food source.

The general scheme of the ABC algorithm is as follows:

Initialization Phase
REPEAT
  Employed Bees Phase
  Onlooker Bees Phase
  Scout Bees Phase
Store the best solution achieved so far
UNTIL(maximum number of iterations reached)

2.1 Initialization Phase

All the vectors of the population of food sources, $\overrightarrow{x_m}$ are initialized ($m = 1 \dots SN$, where SN is population size) by scout bees and control parameters are set. Since each food source, $\overrightarrow{x_m}$, is a solution vector to the optimization problem, each $\overrightarrow{x_m}$ vector holds n variables, $(x_{mi}, i = 1 \dots n)$, which are to be optimized so as to minimize the objective function. The following definition is used for initialization purposes.

$$x_{mi} = l_i + \text{rand}\,(0,1) * (u_i - l_i) \qquad (1)$$

where l_i and u_i are the lower and upper bound of the parameter x_{mi} respectively.

2.2 Employed Bees Phase

Employed bees search for new food sources $\overrightarrow{v_m}$ having more nectar within the neighbourhood of the food source $\overrightarrow{x_m}$ in their memory. They find a neighbour food source and then evaluate its profitability (fitness). For example, they can determine a neighbour food source $\overrightarrow{v_m}$ using the following formula.

$$v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki}) \qquad (2)$$

where $\overrightarrow{x_k}$ is a randomly selected food source, i is a randomly chosen parameter index and ϕ_{mi} is a random number within the range $[-a, a]$. After producing the new food source $\overrightarrow{v_m}$, its fitness is calculated and a greedy selection is applied between $\overrightarrow{v_m}$ and $\overrightarrow{x_m}$.

The fitness value of the solution, $fit_m(\overrightarrow{x_m})$, might be calculated for minimization problems using the following formula,

$$fit_m(\overrightarrow{x_m}) = \begin{cases} \frac{1}{1+f_m(\overrightarrow{x_m})} & \text{if } f_m(\overrightarrow{x_m}) \geq 0 \\ 1 + abs\left(f_m(\overrightarrow{x_m})\right) & \text{if } f_m(\overrightarrow{x_m}) < 0 \end{cases} \qquad (3)$$

where $f_m(\overrightarrow{x_m})$ is the objective function value of solution $\overrightarrow{x_m}$.

2.3 Onlooker Bees Phase

Unemployed bees consist of two groups of bees: onlooker bees and scouts. Employed bees share their food source information with onlooker bees waiting in the hive and then onlooker bees probabilistically choose their food sources depending on this information. In ABC, an onlooker bee chooses a food source depending on the probability values calculated using the fitness values provided by employed bees. For this purpose, a fitness based selection technique can be used, such as the roulette wheel selection method (Goldberg, 1989).

The probability value p_m with which $\overrightarrow{x_m}$ is chosen by an onlooker bee can be calculated by using the following expression.

$$p_m = \frac{fit_m(\overrightarrow{x_m})}{\sum_{m=1}^{SN} fit_m(\overrightarrow{x_m})} \quad (4)$$

After a food source $\overrightarrow{x_m}$ for an onlooker bee is probabilistically chosen, a neighbourhood source $\overrightarrow{v_m}$ is determined by using equation (2), and its fitness value is computed. As in the employed bees phase, a greedy selection is applied between $\overrightarrow{v_m}$ and $\overrightarrow{x_m}$. Hence, more onlookers are recruited to richer sources and positive feedback behaviour appears.

2.4 Scout Bees Phase

The unemployed bees who choose their food sources randomly are called scouts. Employed bees whose solutions cannot be improved through a predetermined number of trials, specified by the user of the ABC algorithm and called "limit" or "abandonment criteria" herein, become scouts and their solutions are abandoned. Then, the converted scouts start to search for new solutions, randomly. For instance, if solution $\overrightarrow{x_m}$ has been abandoned, the new solution discovered by the scout who was the employed bee of $\overrightarrow{x_m}$ can be defined by (1). Hence those sources which are initially poor or have been made poor by exploitation are abandoned and negative feedback behaviour arises to balance the positive feedback.

3. Shuffled Complex Evolution of Artificial Bee Colony Algorithm

A shuffled complex evolution of PSO algorithm called SCE-PSO was first proposed by Yan and his co-authors [3]. In SCE-PSO a population of points is sampled randomly in the feasible space. Then the population is partitioned into several complexes, which is made to evolve based on PSO. At periodic stages in the evolution, the entire population is shuffled and points are reassigned to complexes to ensure information sharing. Based on the advantage of information sharing among the individuals, a novel shuffled complex evolution of ABC algorithm called SCE-ABC algorithm is proposed in this paper to retrieve high utility itemsets.

In the original ABC, the bee is guided by the best solution found i.e., global exploration. In recent years, many researchers found that each bee's best current performance of its neighbors to replace the best previous one of the whole bees i.e., local exploration. SCE-PSO is a local exploration model to define the neighborhood relations.

4. Preliminaries in High Utility Itemset Mining

Let $I = \{i1, i2, \ldots, im\}$ be a finite set of m distinct items. D is a quantitative database that contains a set of transactions $\{T1, T2, \ldots, Tn\}$, Each transaction $T_i \in D$ contain items I. Each item I have internal utility that represents the number of items purchased and an external utility that was represented by a separate table. A minimum utility threshold , δ was set according to users' preference.

Definition 1. The utility of an item in a transaction Tq, $U(iTq)$ is defined as:

$U(iTq) =$ internal utility (i) * external utility (i). Definition 2. The utility of an itemset X in transaction $TqU(X, Tq)$ is defined as:

$U(X, Tq) = \Sigma$ (ui, Tq) for all if X.

Definition 3. The utility of an itemset X in a database $DU(X)$ is defined as:

$U(X) = \Sigma U(X, Tq)$. For all $X \in D$.

Definition 4. The transaction utility of a transaction Tq $tu(Tq)$ is defined as:

$Tu(tq) = \sum U(X, Tq)$

Definition 5. The transaction-weighted utility of an itemset X in a database D, twu(X), is defined as: $twu\ (X) = \Sigma tu(Tq)$

Definition 6. An itemset X in a database D is a high transaction- weighted utilization itemset(HTWUI) if and if its twu is greater than minimum utility threshold.

Definition 7. An itemset X in a database D is a high-utility itemset (HUI) if its utility value is greater than the minimum utility threshold value.

5. Related Works

Liu et al. first developed a two- phase (TWU) model and designed the transaction-weighted downward closure (TWDC) property to early prune the unpromising high utility itemsets (HUIs)[7]. But the algorithm cannot discover the complete HUIs.

Kannimuthu and his co-author adopted the genetic algorithm approach and developed the high utility pattern extracting using genetic algorithm with ranked mutation using minimum utility threshold (HUPEumu-GRAM) to mine HUIs [8]. Another algorithm called HUPEwumu-GRAM was also designed to mine HUIs without specified minimum utility threshold.

An efficient PSO-based algorithm namely HUIM-BPSO sig is proposed to efficiently find HUIs[9]. It first sets the number of discovered high-transaction weighted utilization1-itemsets (1-HTWUIs)as the size of a particle based on transaction-weighted

utility(TWU) model, which can greatly reduce the combinational problem in evolution process. The sigmoid function is adopted in the updating process of the particles of the designed HUIM-BPSO sig algorithm.

An ACO based HUIM-ACS algorithm was proposed to efficiently find high utility itemsets[10]. The HUIM-ACS algorithm maps the completed solution space into the routing graph and includes two pruning processes. Therefore, it guarantees that it obtains all of the HUIs when there is no candidate edge from the starting point. In addition, HUIM-ACS does not estimate the same feasible solution again in its process in order to avoid wasting computational resource.

An algorithm named THUI (temporal high utility itemsets) was proposed by Chu et al and it is the first algorithm for finding temporal high utility itemsets in temporal databases[11]. The algorithm integrated the advantages of the Two-Phase algorithm and the SWF algorithm and augment with the incremental mining techniques for mining temporal high utility itemsets efficiently

Lan et al proposed a projection-based average-utility itemset mining (PAI) algorithm to reveal HAUIs using a level-wise approach[12]. Based on the proposed upper-bound model, the number of unpromising candidates can be greatly reduced compared to previous work based on the TWU model.

6. Proposed SCE-ABC algorithm-based Framework for High Utility Itemset Mining

The following strategy list the steps to calculate high utility itemsets based on SCE-ABC algorithm. Before applying the algorithms, HTWUIs was calculated as per definition 6. Only the items whose HTWUI greater than minimum utility threshold are considered as promising items and other items are ignored at this level itself.

Step 1: Initializing.

Select $p \geq 1, m \geq 1$, where, p is the number of partitions, m is the number of points in each complex. In utility mining m is taken as number of HTWUIs and p can be any number of partitions depending upon the size of database.

Step 2: Arranging.

Arrange the items in HTWUIs in ascending order and initialize the items in population.

Step 3: Fitness function.

Compute the function value f at each point X. Here utility function of the item given in Definition 1 was taken as fitness function.

Step 4: Array Function.

Store the functions in an array $\mathrm{A} = \{\mathrm{Xi}, \mathrm{fi}\}$ for $\mathrm{i} = 1,2 \ldots \mathrm{m}\}$

Step 5: Partitioning.

Partition the array A into p complexes $A1, A2, .., Ap$, each containing points m.

Step 6: Evolving.

Evolve each complex Ak using traditional ABC algorithm separately. In traditional ABC algorithm, the itemsets are generated from HTUWIs. Utility values of items are calculated as given in introduction part. Then the utility value of items are compared with minimum utility threshold, if it is greater than minimum threshold, then the item is considered as high utility items. Else the velocity of items is updated using equation 1. This is repeated until no items are present.

Step 7: Complexes shuffling.

Do the above steps for all items in A.

Step 8: Termination condition.

If termination condition is met, then, stop. Otherwise, return to Step 4.

7. Experimental Results

Experiments are conducted to analyses the performance of proposed SCE-ABC algorithm. The proposed SCE-ABC was compared with existing algorithms SCE-PSO, ABC, PSO, and GA algorithms. The algorithms are implemented in MATLAB and executed in a PC with an Intel Core i7-4500CPU and 6 GB of RAM, running the 64-bitMicrosoft Windows 10 operating system. Two benchmark datasets Mushroom and Foodmart are used in this experiment[13]. The algorithms are compared in terms of execution time, memory space used and number of high utility itemsets. Table 1 shows the dataset characteristics used in the experiment[13]. Table 2 shows the performance analysis of SCE-ABC, SCE-PSO, ABC, PSO, and GA algorithms in Foodmart and Mushroom dataset.

Table 1. Dataset Characteristics

S.No	Dataset	Number of transactions	Maximum number of items in transactions
1	Mushroom	8124	23
2	Foodmart	4144	8

Table 2. Comparison of execution time, Memory space and Number of high utility itemsets of algorithms

Dataset	Algorithms	Execution time (s)	Number of high utility items retrieved	Memory space (MB)
Mushroom	GA	30.78	60	20.82
	PSO	22.47	68	17.64
	ABC	18.35	70	15.59
	SCE-PSO	14.98	72	14.31
	SCE-ABC	12.04	81	12.14
Foodmart	GA	7.33	60	241.09
	PSO	6.97	88	105.56
	ABC	5.26	91	96.26
	SCE-PSO	4.91	95	89.62
	SCE-ABC	4.08	99	85.37

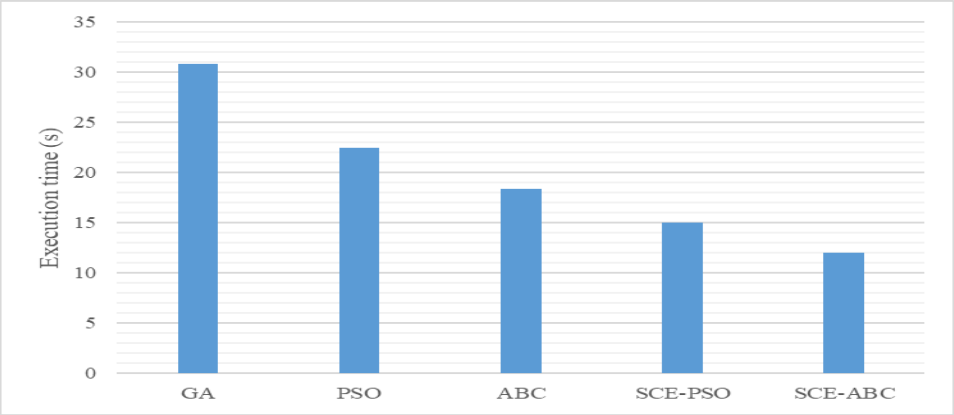

Figure 1 Comparison of execution time – Mushroom dataset

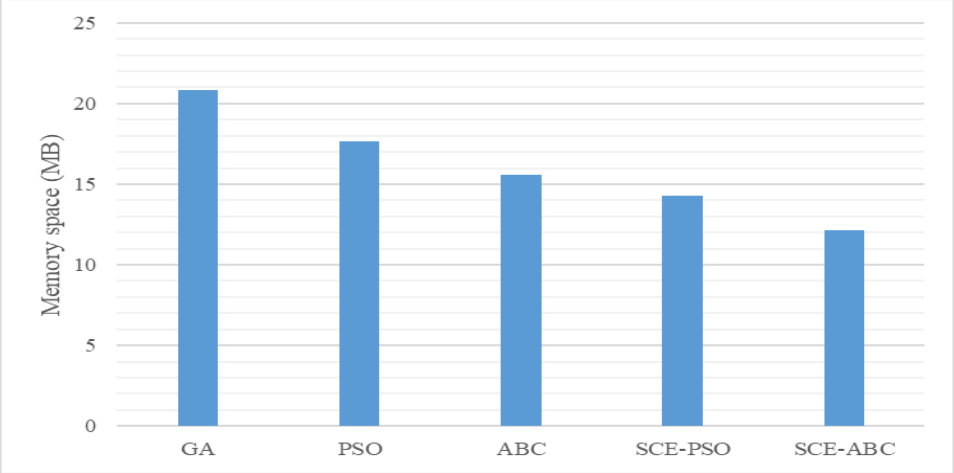

Figure 2 Comparison of memory consumption – Mushroom dataset

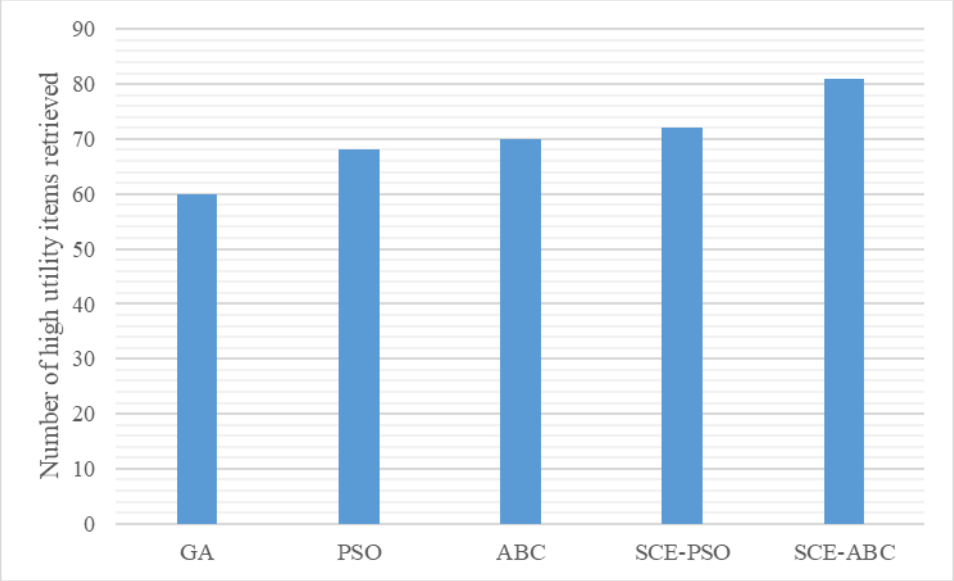

Figure 3 Comparison of number of high utility items retrieved – Mushroom dataset

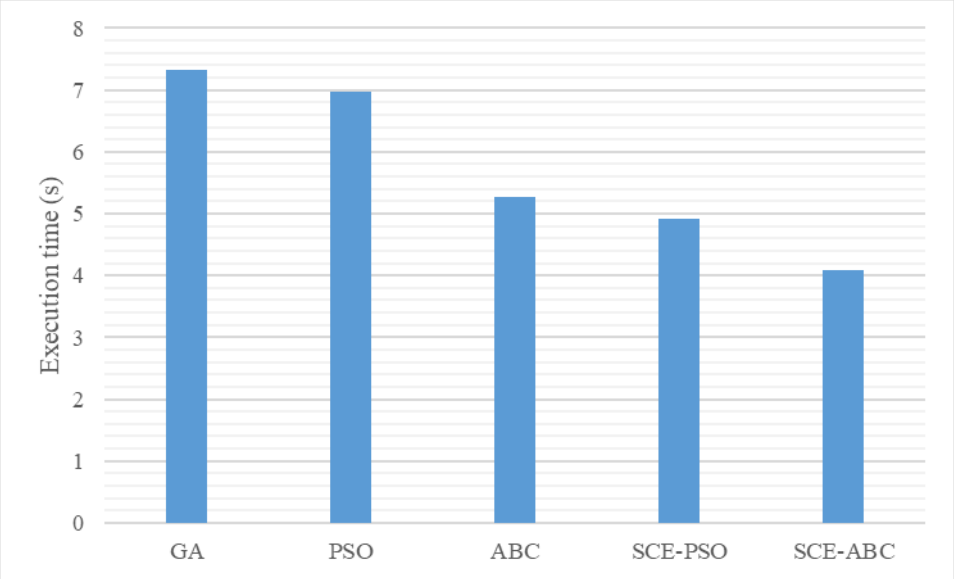

Figure 4 Comparison of execution time – Foodmart dataset

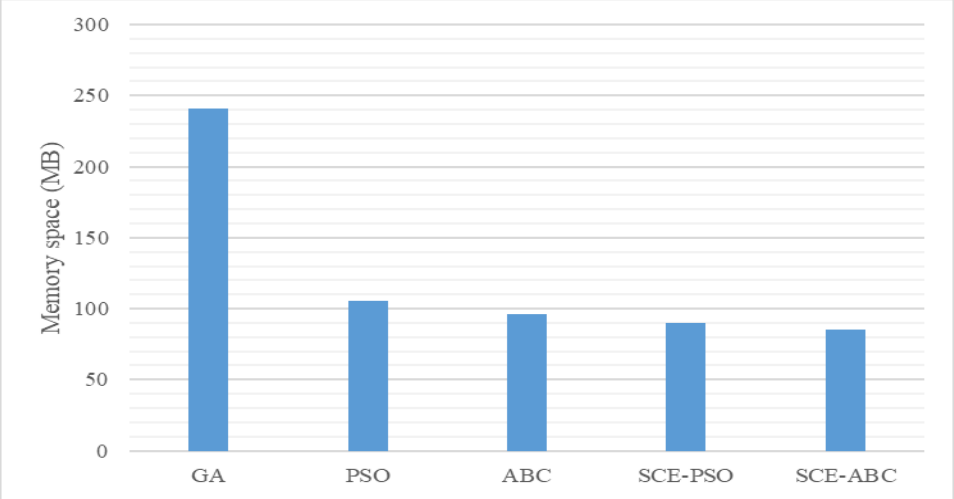

Figure 5 Comparison of memory consumption – Foodmart dataset

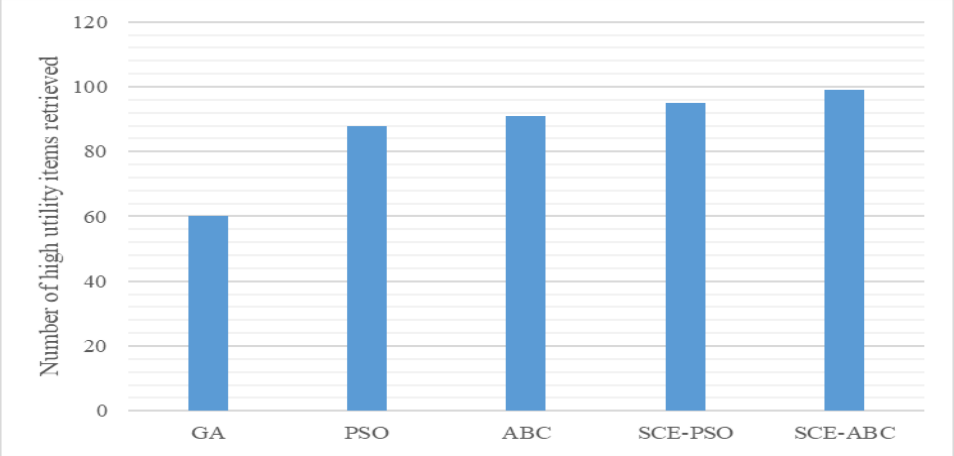

Figure 6 Comparison of number of high utility items retrieved – Foodmart dataset

7. Conclusion

Utility Mining considers utility factors of itemset, is an emerging topic in data mining. It is very beneficial in several real-life applications. Initially, the Genetic Algorithm was proposed to mine HUIs. But the drawback is the combinational problem of the generated itemsets a. Also it requires several parameters in the evolution process. Later to overcome these difficulties PSO was proposed. In this work, a modified PSO algorithm called SCE- PSO was used to mine high utility item sets from a transaction database. In SCE-PSO a population of points is sampled randomly in the feasible space. Then the population is partitioned into several complexes, which is made to evolve based on PSO. From the experiments, it was found that the proposed approach was better than the existing algorithms.

8. References

[1] C.-J. Chu, V. S. Tseng, and T. Liang, "An efficient algorithm for mining high utility itemsets with negative item values in large databases," *Appl. Math. Comput.*, vol. 215, no. 2, pp. 767–778, 2009.

[2] Y. Liu, W. Liao, and A. Choudhary, "A fast high utility itemsets mining algorithm," in *Proceedings of the 1st international workshop on Utility-based data mining*, 2005, pp. 90–99.

[3] J. Yan, H. Tiesong, H. Chongchao, W. Xianing, and G. Faling, "A shuffled complex evolution of particle swarm optimization algorithm," in *International conference on adaptive and natural computing algorithms*, 2007, pp. 341–349.

[4] D. Karaboga and others, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer …, 2005.

[5] V. Tereshko and A. Loengarov, "Collective decision making in honey-bee foraging dynamics," *Comput. Inf. Syst.*, vol. 9, no. 3, p. 1, 2005.

[6] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1001–1014, 2012.

[7] J. Liu, K. Wang, and B. C. Fung, "Mining high utility patterns in one phase without generating candidates," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1245–1257, 2015.

[8] S. Kannimuthu and K. Premalatha, "Discovery of high utility itemsets using genetic algorithm with ranked mutation," *Appl. Artif. Intell.*, vol. 28, no. 4, pp. 337–359, 2014.

[9] S. Kannimuthu and D. G. Chakravarthy, "Discovery of Interesting Itemsets for Web Service Composition Using Hybrid Genetic Algorithm," *Neural Process. Lett.*, pp. 1–27, 2022.

[10] J. M.-T. Wu, J. Zhan, and J. C.-W. Lin, "An ACO-based approach to mine high-utility itemsets," *Knowl.-Based Syst.*, vol. 116, pp. 102–113, 2017.

[11] C.-J. Chu, V. S. Tseng, and T. Liang, "An efficient algorithm for mining temporal high utility itemsets from data streams," *J. Syst. Softw.*, vol. 81, no. 7, pp. 1105–1117, 2008.

[12] G.-C. Lan, T.-P. Hong, V. S. Tseng, and others, "A projection-based approach for discovering high average-utility itemsets," *J. Inf. Sci. Eng.*, vol. 28, no. 1, pp. 193–209, 2012.

[13] C. Sivamathi and S. Vijayarani, "mining high utility itemsets using shuffled complex evolution of particle swarm optimization (SCE-PSO) optimization algorithm," in *2017 International Conference on Inventive Computing and Informatics (ICICI)*, 2017, pp. 640–644.