Mining Top-K High Utility Itemset Using Bio-Inspired Algorithms

Nam Ngoc Pham
Faculty of Applied Informatics
Tomas Bata University
Zlín, Czech Republic
npham@utb.cz

Zuzana Komínková Oplatková
Faculty of Applied Informatics
Tomas Bata University
Zlín, Czech Republic
oplatkova@utb.cz

Huy Minh Huynh
Faculty of Applied Informatics
Tomas Bata University
Zlín, Czech Republic.
huynh@utb.cz

Bay Vo*
HUTECH University
Ho Chi Minh City, Vietnam
vd.bay@hutech.edu.vn
*Corresponding author

2022 IEEE Workshop on Complexity in Engineering (COMPENG) | 978-1-7281-7124-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/COMPENG50184.2022.9905433

***Abstract*— High utility itemset (HUI) mining is a necessary research problem in the field of knowledge discovery and data mining. Many algorithms for Top-K HUI mining have been proposed. However, the principal issue with these algorithms is that they need to store potential top-k patterns in the memory anytime, and they request the minimum utility threshold to automatically rise when finding HUIs. Consequently, the performance of existing exact algorithms for Top-K HUIs mining tends to decrease when the database size and the number of distinct items in the databases rise. To address this issue, we suggest a Binary Particle Swarm Optimization (BPSO) based algorithm for mining Top-K HUIs effectively, namely TKO-BPSO (Top-K high utility itemset mining in One phase based on Binary Particle Swarm Optimization). The main idea of TKO-BPSO is not only to use a one-phase model and strategy Raising the threshold by the Utility of Candidates (RUC) to effectively increase the border thresholds for pruning the search space but also to adopt the sigmoid function in the updating process of the particles. This might significantly reduce the combinational problem in traditional HUIM when the database size and the number of distinct items in the databases rise. Consequently, its performance outperforms existing exact algorithms for mining Top-K HUIs because it efficiently overcomes the problem of the vast amount candidates. Substantial experiments conducted on publicly available several real and synthetic datasets show that the proposed algorithm has better results than existing state-of-the-art algorithms in terms of runtime, which can significantly reduce the combinational problem and memory usage.**

Keywords*— *bio-inspired algorithm; top-k high utility itemset mining; binary particle swarm optimization.

I. INTRODUCTION

High utility itemset (or pattern) mining (HUIM) is an extension of frequent itemset mining (FIM) [1] because the issue of FIM is to explore a set of itemsets whose occurrence frequency is greater or equal to a minimum support threshold specified by users, that is, frequent itemsets are determined by the occurring frequency in the database. Although the traditional FIM algorithms can mine the frequent itemsets in databases, they can cause other essential factors to be ignored, for example, the utility (profit) of items is not considered. To overcome this limitation, a utility-driven itemset mining approach [2], [3], which considers quantity and profit, also called high utility itemset mining [2], was designed to discover high utility patterns from transaction databases. In utility-driven itemset mining, each item in the transaction is associated with a unit profit and purchased quantity. The utility of an itemset in a transaction database is calculated by the sum of the utility of itemset appearing in all transactions of the database. An itemset is called a high utility itemset (HUI) if its utility value is no less than a minimum utility (minUtil) threshold specified by users. The principal target of HUIM is to discover HUIs whose utility value is greater or equal to a user-specified minimum utility threshold. High utility itemset mining is widely applied in numerous domains in real-world applications such as market analysis [4], click-stream analysis [22], [23]. Thus, it plays a critical role in knowledge discovery and data mining. Several studies have been devoted to HUI mining, such as Two-Phase [6], UP-Growth [8], FHM [9], and EFIM [10]. The above mentioned algorithms have suggested various upper bounds on the utility of itemsets, which have the downward closure property as FIM, i.e., anti-monotonic, like the transaction-weighted utilization (TWU) upper bound to reduce the search space. However, these upper bounds must face some issues, such as many low utility itemsets being evaluated to find the HUIs, and they might be loose. Therefore, the performance can deteriorate. To deal with these drawbacks, several algorithms based on utility-list such as HUP-Miner [7] and HUI-Miner [5], have been proposed. These algorithms proposed a novel utility-list structure that is used to store the utility information of itemsets in the database and is used to mine high utility itemsets in the memory in place of scanning the dataset many times. Thus, it efficiently overcomes the problem of the vast amount candidates. One of the most significant drawbacks in the field of HUIM is how to set an appropriate minimum utility threshold (minUtil). Hence, setting a suitable minimum utility threshold is the main task and very challenging because it affects the performance and result of mining HUIs algorithms, i.e., if a minUtil value is set too small, a huge number of patterns is found and the algorithm may have a very long execution time.

On the contrary, if minUtil threshold is set too high, only few patterns are found and the algorithm may miss some important patterns. To determine a suitable minUtil value, users must set a minimum utility threshold (minUtil) by trial and error tedious and time-consuming. To address this limitation, an efficient approach for mining HUIs such as TKU and TKO [21] have been proposed, where k is the desired number of HUIs to be mined. The main issue of top-k HUIM algorithms is that they need to store potential top-k patterns in the memory anytime, and they request the minimum utility threshold to automatically rise when finding HUIs. However, when the database size and the number of distinct items in the databases increasingly tend to rise, the performance of existing Top-K HUIM algorithms tends to drop. To tackle this issue, a stochastic optimization approach, i.e., evolutionary computation, has been proposed to discover large search spaces and find the optimal solutions using the principles of natural evolution. In this paper, we propose an efficient binary PSO-based approach to mine Top-K HUIs, called mining **T**op-**K** high utility itemsets in **O**ne phase based on **B**inary **P**article **S**warm **O**ptimization (TKO-BPSO). The key contributions of the TKO-BPSO algorithm are delineated as follows:

Firstly, a list-based structure named utility-list [5] is used to store the utility information of itemsets in the database, which is used to mine HUIs in the memory in place of scanning the dataset many times. It uses vertical data representation techniques to discover top-k HUIs in only one phase.

Secondly, this paper applies strategy RUC to effectively increase the border thresholds for pruning the search space.

Thirdly, a binary PSO-based algorithm to mine Top-K HUIs, namely TKO-BPSO, is designed to discover the HUIs by integrating the sigmoid function which can significantly reduce the combinational problem in the evolution process.

Finally, we conducted extensive experiments on real-life datasets to evaluate the performance of TKO-BPSO with state-of-the-art algorithms (TKU and TKO [21]) for mining HUIs. Results revealed that our proposed algorithm is efficient in terms of both memory usage and execute runtime.

The remained of this paper is organized as follows: related work is briefly reviewed in section 2. Preliminaries and the problem statement are described in section 3. The proposed TKO-BPSO algorithm is described in section 4. The experimental results are presented in section 5. Finally, our conclusions and future works are presented in section 6.

II. RELATED WORK

A. High Utility Itemset Mining

HUIM is an extension of FIM. Consequently, it has received significant attention from researchers. The goal of HUIM is to discover HUIs whose utility value is greater than or equal to a minimum utility threshold specified by users. Several studies have been devoted to HUIs mining, such as Two-Phase [6], UP-Growth [8], HUI-Miner [5], FHM [9], and EFIM [10]. All algorithms mentioned above for HUIM might be divided into two types that are two-phase and one-phase model algorithms. Their difference depends on whether the number of candidates is generated. For the two-phase algorithm, it has introduced various upper bounds on the utility of itemsets which has the downward closure property as FIM, i.e., anti-monotonic, such as the transaction-weighted utilization (TWU) upper bound to reduce the search space. As a result, it prunes the search space using this TWU model. However, the two-phase algorithm has two drawbacks. Firstly, it scans the database at least twice for HUI mining. Secondly, it generates vast amount candidates. Meanwhile, the one-phase algorithm such as HUI-Miner [5] and HUP-Miner [7]. They propose a novel utility-list structure that is used to mine HUIs in the memory in place of scanning the dataset many times. Thus, it efficiently overcomes the problem of the vast amount candidates. Although traditional HUIs mining algorithms can offer complete results, one of the most significant drawbacks of HUIM is setting an appropriate minUtil.

B. Top-K High Utility Itemset Mining

One of the most significant drawbacks in the field of traditional HUIM is setting an appropriate minUtil because it can affect the performance and the number of candidates generated. Thus, Top-K HUI mining is proposed. The goal target of top-k HUIM algorithms is that they need to store potential top-k patterns in the memory anytime, and they request the minimum utility threshold to automatically rise when finding HUIs. The first Top-K HUIM algorithm called TKU [21] was proposed. TKU is an algorithm that improves UP-Growth [8] from the HUIM algorithms. It designed efficient threshold-rising strategies for Top-K HUI mining. Besides, the TKO algorithm has been studied for Top-K HUI mining using a single-phase model. It is an extension of the HUI-Miner [5] algorithm. TKO utilized two strategies, PE and DGU, in which PE is to raise the threshold and DGU is to prune unpromising itemsets during dataset scanning. Meanwhile, KHMC [24] is an algorithm that improves FHM [9]. It suggested three effectively threshold-raising strategies (RIU, CUD, COV), in which the COV strategy not only raises the threshold but also prunes the effective search space. Although several studies have been devoted to top-k HUI mining, the database size, and the number of distinct items in the databases increasingly tend to rise, so the performance of existing Top-K HUIM algorithms tends to drop.

C. Bio-Inspired Algorithms for Itemset Mining

As the database size and the number of distinct items in the databases increasingly tend to rise, the performance of existing Top-K HUIM algorithms tends to drop. To address this issue, heuristic-based and evolutionary computation (EC) algorithms have been proposed [11]. The main idea of existing HUIs mining algorithms based on bio-inspired computing is to find the optimal values of one population that are maintained in the next population, i.e., follow the traditional roadmap of the original GA and PSO algorithms; for example, Kannimuthu et al [12] applied the genetic algorithm to find high utility itemsets using genetic algorithm with ranked mutation using minimum utility threshold. In addition to GA, Particle Swarm Optimization (PSO), a bio-inspired and population-based solution, is also proposed to find the optimal methods by applying the velocity to update the individuals (particles).

Kennedy et al [13] first introduced PSO and its basic idea was originally inspired by simulation of the social behavior of animals such as bird flocking, fish schooling and so on to solve the optimization problems. However, the traditional PSO algorithms cannot be utilized to optimize for a discrete-valued search space. Kennedy et al [14] designed a binary PSO algorithm (BPSO) to solve the limitation of continuous PSO. Several algorithms for mining HUIs based on PSO have been proposed such as HUIM-BPSOsig [15], HUIM-BPSO [16], and Bio-HUIF-PSO [17]. Besides, several algorithms for mining high average utility itemsets based on PSO are also suggested such as HAUI-BPSO [18], and HAUI-PSO [19].

Heuristic-based algorithms, which are inspired by biological and physical phenomena, have been proposed to discover the vast search space of HUIM. However, they are often time-consuming to mine all HUIs which satisfy the minimum utility threshold. To address this issue, we suggest a binary PSO-based algorithm for Top-K HUI mining effectively, where k is the desired number of HUIs to be mined. Our proposed TKO-BPSO algorithm is different from other PSO algorithms in that it not only lets users specify k but also uses a list-based structure to store the utility information of itemsets in the database to mine high utility itemsets in the memory instead of scanning the dataset many times.

III. PRELIMINARIES AND PROBLEM DEFINITION

A. Preliminaries

Let $\boldsymbol{DB} = \{T_1, T_2, .., T_n\}$ be a transaction database comprising a set of n distinct items I, I = {i$_1$, i$_2$,…, i$_m$}. Each transaction $T_d \in$ DB, 1≤ d ≤ n, is a finite set of items such that $T_d \subseteq$ I and has a unique identifier **tid**. Also, each item i_p in a transaction T_d has a purchase quantity (internal utility) denoted as q (i_p , T_d). Each unit profit (external utility) of an item $i_p \in I$ is denoted as p (i_p). The set X = $\{j_1, j_2, \dots, j_k\}$ ⊆ I, 1≤ k ≤ m, is called an itemset; an itemset containing k distinct items is said to be a k-itemset, where k is the length of an itemset. Consider the transaction database in table 1 and the profit value in table 2 as the running sample database.

Table 1: An example transaction database

Tid	Transaction (item, quantity)	Transaction Utility
T1	a:1, c:1, d:1	8
T2	a:2, c:6, e:2, g:5	27
T3	a:1, b:2, c:1, d:6, e:1, f:5	30
T4	b:4, c:3, d:3, e:1	20
T5	b:2, c:2, e:1, g:2	11

Table 2: Profit table

Item	a	b	c	d	e	f	g
Profit	5	2	1	2	3	1	1

Definition 1. The utility of an item i_p in a transaction T_d is the multiplication of the internal utility and the external utility of i_p in T_d, denoted by: $u(i_p, T_d) = q(i_p, T_d) \text{ x } p(i_p)$ (1)
For example, the utility of item (a) in transaction T_1 is calculated as u(a, T_1) = q(a, T_1) x p(a) = 1 x 5 =5.
Definition 2. The utility of an itemset X in transaction T_d is denoted as u(X, T_d), and defined as:

$$u(X, T_d) = \sum_{i_p \in X \wedge X \subseteq T_d} u(i_p, T_d) \quad (2)$$

For example, the utility of the itemset (ac) in transaction T_1 is calculated as u(ac, T_1) = u(a, T_1) + u(c, T_1) = 1x5 + 1x1= 6.
Definition 3. The utility of an itemset X in a database DB is the sum of utility of itemset X appearing in all transactions of DB, and denoted as: $u(X) = \sum_{X \subseteq T_d \wedge T_d \in DB} u(X, T_d)$ (3)

For example, the utility of itemset (ac) in DB is calculated as u(ac) = u(ac, T_1) + u(ac, T_2) + u(ac, T_3) = 6 + 16 + 6 = 28.
Definition 4. The transaction utility of a transaction T_d is the sum of the utility of all items in T_d, and denoted as:

$$tu(T_d) = \sum_{i_p \in T_d} u(i_p, T_d) \quad (4)$$

For example, tu(T_2) = u(a, T_2) + u(c, T_2) + u(e, T_2) + u(g, T_2) = 27.
Definition 5. The transaction-weighted utilization of an itemset X in a database DB is denoted as TWU(X), and defined as:

$$\text{TWU(X)} = \sum_{X \subseteq T_d \wedge T_d \in DB} tu(T_d) \quad (5)$$

For example, TWU(a) = tu(T$_1$) + tu(T$_2$) + tu(T$_3$) + u(g, T$_2$) = 65.
Definition 6. The utility of a transaction database DB (the total utility of a database DB) is the sum of the utility of all transactions in the DB denoted as $TU = \sum_{T_d \in DB} tu(T_d)$ (6)
For example, the total utility in DB is calculated as TU = tu(T_1) + tu(T_2) + tu(T_3) + tu(T_4) + tu(T_5) = 96.

B. Problem statement

Let the input be a transactional database DB and the parameter k as the desired number of HUIs. The TKO-BPSO solution can be used to solve the combinational problem in the evolution process and to effectively increase the border thresholds for pruning the search space to discover the HUIs.

IV. THE TKO-BPSO ALGORITHM

A. Baisc idea of Particle Swarm Optimization

The Particle Swarm Optimization (PSO) is a bio-inspired algorithm derived from the foraging behavior of birds or fish. In the PSO algorithm, numerous particles are random generated. Each particle moves toward the optimal value according to two equations as follows:

$$v_i^{t+1} = wv_i^t + c_1 r_1 (pbest_i - x_i^t) + c_2 r_2 (gbest - x_i^t) \quad (7)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (8)$$

In the equations (7) and (8), v_i^t and v_i^{t+1} are the velocities of the ith particle at iterations t and t + 1, i.e., t represents current number of iterations; x_i^t and x_i^{t+1} are represented the positions of the ith particle at iterations t and t + 1; r1, r2 are random numbers in range of (0, 1). Each particle has memories to keep its previous best particle (personal best, pbest), so $pbest_i$ is the previous best location of the ith particle; gbest is the current best location of all particles; The three constants w, c_1, c_2 are weighting coefficients in which c_1 and c_2 are selected such that c_1 = c_2 = 2.

B. The Sigmoid function for Updating Particles

According to the traditional PSO solution, the velocity of the particles is updated and is delineated in the equation (7).

However, the particles, which are updated based on sigmoid function used in a BPSO method [14], are delineated as follows:

$$x_i^j(t+1) = \begin{cases} 1 & rand() < sig\left(v_i^j(t+1)\right) = \frac{1}{1+e^{-v_i^j(t+1)}} \\ 0 & otherwise \end{cases} \quad (9)$$

This above equation (11) is used to determine the probability of the (j-th) location of a particle. If the created rand() function is less than $sig\left(v_i^j(t+1)\right)$, then the value of the corresponding jth position of a particle is set as 1; otherwise, it is set as 0, in which the rand() function is a uniform distribution in the range of (0, 1). Therefore, the sigmoid function is used for normalization.

C. *Top-K HUI Mining Based on the TKO-BPSO Method*

After the sigmoid function is adopted, which may significantly reduce the combinational problem in the evolution process. To discover the top-k HUIs, we use strategy RUC to effectively increase the border thresholds for pruning the search space and the utility-list structure to decrease the number of databases scanners effectively.

D. *Strategy to **R**aise the threshold by the **U**tilities of **C**andidates (RUC)*

The first minUtil threshold is set to 0. It applies a structure named TopK-CI-List to maintain top-k HUIs, where itemsets are arranged in descending order of utility. At first, TopK-CI-List is empty. When an itemset X is found by the search procedure more detail in [21] and its utility is greater or equal to minUtil threshold, then X is added to TopK-CI-List. If more than k itemsets are already in TopK-CI-List, minUtil threshold can be safely raised to the utility of the kth itemset in TopK-CI-List. Following this, itemsets, which have a utility lower than the raised min_util threshold, are deleted from TopK-CI-List.

E. *Utility-List structure*

In the section, a utility-list structure has been proposed to maintain the utility information of a database by two scans of the database. The utility-list structure and related properties are briefly introduced in this section, and more detail can read in [3]. Each item/itemset is associated with a utility-list, which are called initial utility-lists, by two scans of the database. The utility values and TWU of items are calculated in the first database scan. Items in each transaction are arranged in ascending order of TWU values and the utility-list of each item is constructed in the second database scan.

Table 3 below delineates an example database, where items in each transaction are sorted in ascending order of TWU values (twu(f) = 30, twu(g) =38, twu(d) = 58, twu(b) = 61, twu(a) = 65, twu(e) = (88), twu(c)=96). Figure 1 shows utility-lists of items for the database in Table 3. Each element in the utility-list of itemset X includes three fields: tid, iutil, and rutil.

- Fields Tid and iutil contain the identifier of T_d and the utility of itemset X in T_d, i.e., $u(X, T_d)$ respectively.
- Field rutil is the remaining utility of itemset X in T_d.

Definition 7. The remaining utility of an itemset X in transaction T_d is the sum of the utility of all the items after X in T_d, denoted as: $ru(X,T_d) = \sum_{i\in(T_d/X)} u(i_p, T_d)$ (10)

Table 3. Transactions for Constructing Utility-Lists

Tid	Transaction (item, quantity)	Transaction Utility
T_1	d: 1, a: 1, c: 1	8
T_2	g: 5, a: 2, e: 2, c: 6	27
T_3	f: 5, d: 6, b: 2, a: 1, e: 1, c:1	30
T_4	d: 3, b: 4, e:1, c: 3	20
T_5	g: 2, b: 2, e: 1, c:2	11

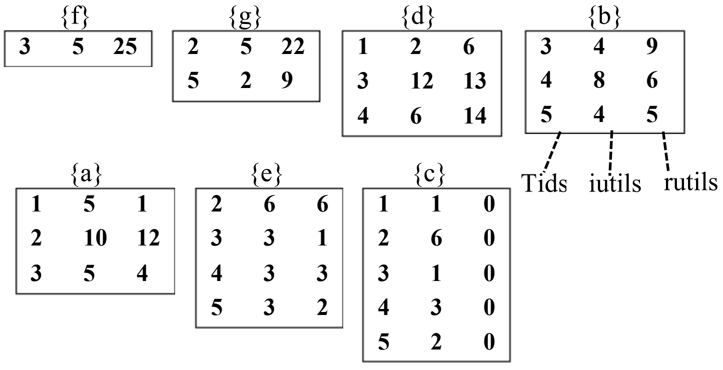

Figure.1. initial utility-lists

For example, the remaining utility of item {d} in transaction T_1 is calculated: $ru(d, T_1) = u(a, T_1) + u(c, T_1) = 5 + 1 = 6$

V. PERFORMANCE EVALUATION

Significant experiments were conducted to demonstrate the effectiveness and efficiency of the proposed TKO-BPSO algorithm. We conducted the experiment on a computer with an intel core (TM) i-7-4600U CPU and 8 GB of RAM, running the 64-bit Microsoft windows 10. The experiments of the algorithm were implemented in Java language and compared the performance of TKO-BPSO with the two exact algorithms, TKU and TKO [21]. The publicly available several real datasets were used to evaluate the performance of the proposed algorithm. These datasets were downloaded from the SPMF data mining library [20].

A. Experiments on Runtime

The execution time of the proposed algorithm is evaluated and compared to the two exact algorithms, TKU and TKO, under various k values and on different datasets, in second (s). The TKO-BPSO algorithm performs better than the TKU and TKO algorithms about runtime in many cases. In all the tested datasets, TKO-BPSO tends to become increasingly different as the k values rise. For example, in Chess datasets, the runtime of TKO-BPSO is 97s, 105s, 106s, 109s, 112s, and 130s when the k parameter is 1000, 1100, 1200, 1300, 1400, and 1500 respectively, while The runtime of TKO is 128s, 133s, 136s, 144s, 145s, and 149s with k similarly and respectively. Moreover, the runtime of TKU is very slow comparation the TKO. Therefore, TKO-BPSO is much more efficient than TKU.

B. Memory Evaluation

The memory usage of the designed TKO-BPSO algorithm is evaluated on all the given datasets. Our proposed algorithm utilizes less memory than the two TKU and TKO algorithms because it is designed to find the HUIs by integrating the sigmoid updating strategy and utility-list structure. Thus, it can significantly reduce the combinational problem in the evolution process while the database size and the number of distinct items in the databases increasingly rise. In all the tested datasets, TKO-BPSO tends to become increasingly different as the k values rise. For example, in Accidents_10% datasets, the memory usage of TKO-BPSO, in Megabyte (MB), is 396 MB, 398 MB, 378 MB, 400 MB, 420 MB, and 481 MB when the k parameter is 500, 1000, 1500, 2000, 2500, and 3000 respectively, while the memory usage of TKO is 405 MB, 402 MB, 413 MB, 448 MB, 455 MB, and 675 MB with k similarly and respectively. Moreover, the memory usage of TKU consumes much more comparation the TKO. Consequently, TKO-BPSO utilizes less memory than TKU.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed TKO-BPSO algorithm which uses the utility-list structure and RUC strategy to prune the search space to speed up the runtime. In addition, it adopts the sigmoid function which may significantly reduce the combinational problem in the evolution process to mine Top-K HUIs. Experimental results are presented that the proposed algorithm outperform the exact two algorithm, TKU and TKO, in terms of runtime and memory usage.

In the future, we will improve by designing compressed data structures, and it can also be used in the incremental mining of HUIs and mining of top-k HUIs from sequential datasets.

Acknowledgments

This work was supported by the Technology Agency of the Czech Republic, under the project no. FW01010381, by internal Grant Agency of Tomas Bata University under the project no. IGA/CebiaTech/022/001, and further by the resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin.

References

[1] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” in Proceedings of the 20th ACM International Conference on Very Large Data Bases, vol. 1215. Citeseer, 1994, pp. 487–499.

[2] W. Gan et al., “A survey of utility-oriented pattern mining,” IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 4, pp. 1306–1327, 2021.

[3] M. Liu and J. Qu, “Mining high utility itemsets without candidate generation,” in Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 55–64, 2012.

[4] H. Ryang, U. Yun, and K. Ryu, “Discovering high utility itemsets with multiple minimum supports,” Intell. Data Anal., vol. 18, no. 6, pp. 1027–1047, 2014.

[5] M. Liu and J. Qu, “Mining high utility itemsets without candidate generation,” in Proc. ACM Int. Conf. Inf. Knowl. Manag., 2012, pp. 55–64.

[6] Y. Liu, W. Liao, and A. N. Choudhary. 2005. A two-phase algorithm for fast discovery of high utility itemsets. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining. 689–695.

[7] S. Krishnamoorthy, “Pruning strategies for mining high utility itemsets,” Expert Systems with Applications, vol. 42, no. 5, pp. 2371–2381, 2015.

[8] V. S. Tseng et al. “An efficient algorithm for high utility itemset mining” In Proceedings of the International Conference on Knowledge Discovery and Data Mining. 253–262.

[9] P. Fournier-Viger, C. Wu, S. Zida, and V. S. Tseng. 2014. FHM: Faster high utility itemset mining using estimated utility co-occurrence pruning. In Proceedings of the International Symposium on Foundations of Intelligent Systems. 83–92.

[10] S. Zida et al., “A highly efficient algorithm for high utility itemset mining.” In Proceedings of the Mexican International Conference on Artificial Intelligence. 530–546, 2015.

[11] S. Ventura and J. M. Luna. 2016. Pattern Mining with Evolutionary Algorithms. Springer.

[12] Kannimuthu, S., Premalatha, K., 2014. Discovery of high utility itemsets using genetic algorithm with ranked mutation. Appl. Artif. Intell. 28 (4), 337–359.

[13] Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948.

[14] Kennedy, J., Eberhart, R., 1997. A discrete binary version of particle swarm algorithm. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 4104–4108.

[15] J. C.-W. Lin et al., ‘‘Mining high-utility itemsets based on particle swarm optimization,’’ Eng. Appl. Artif. Intell., vol. 55, pp. 320–330, Oct. 2016.

[16] V. S. Tseng et al., ‘‘Efficient algorithms for mining high utility itemsets from transactional databases,’’ IEEE Trans. Knowl. Data Eng., vol. 25, no. 8, pp. 1772–1786, Aug. 2013.

[17] W. Song and C. Huang, "Mining High Utility Itemsets Using Bio-Inspired Algorithms: A Diverse Optimal Value Framework," IEEE Access, vol. 6, pp. 19568-19582, 2018.

[18] N. N. Pham ‘‘Mining high average utility pattern using bio-inspired algorithm: student research abstract,’’ in Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, April 2022, pp 445–449.

[19] Wei Song, Chaomin Huang. Mining High Average-Utility Itemsets Based on Particle Swarm Optimization, UK, 2018.

[20] P. Fournier-Viger et al., “The SPMF open-source data mining library version 2.” in Proceedings of the 19th European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 36-40, 2016.

[21] V. S. Tseng et al., "Efficient Algorithms for Mining Top-K High Utility Itemsets," in IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 1, pp. 54-67, 1 Jan. 2016.

[22] Huynh, H.M et al, “Sequential Pattern Mining Using IDLists. In proceeding of the 12th International Conference, ICCCI 2020, Da Nang, Vietnam, Nov 30 – Dec3, 2020.

[23] Huynh, H.M., Nguyen, L.T.T., Pham, N.N. et al, “An efficient method for mining sequential patterns with indices,” Knowledge-Based Systems 239:107946, December 2021.

[24] Q. H. Duong, B. Liao, P. Fournier Viger, and T. L. Dam, “An efficient algorithm for mining the top-k high utility itemsets using novel threshold raising and pruning strategies,” Knowledge-Based Systems, vol. 104, pp. 106–122, 2016.