Iraqi Journal of Intelligent Computing and Informatics (IJICI)
Vol. 4, 2, December 2025, pp. 172~181
ISSN: 2791-2868, DOI: 10.52940/ijici.v4i2.126

A new Dynamic Utility Partitioning Algorithm for High-Utility Itemset Mining in Databases with Unstable Negative Profits

Arkan A. Ghaib
Department Of Information Technologies, Management Technical College, Southern Technical University, Basrah, Iraq.
arkan.ghaib@stu.edu.iq

Article Info

Article history:

Received Jun 30, 2025
Revised July 28, 2025
Accepted Aug 15, 2025

Keywords:

High-efficiency itemset
High Utility Itemset
Negative Profit
Dynamic Utility
Scalability

ABSTRACT

High-Utility Itemset Mining (HUIM) is a significant task in data mining, especially in situations with fluctuating negative profits, such as retail discounts and healthcare cost control. Despite the fact that existing algorithms such as EHMIN and EMHUN for mining under hybrid datasets encounter issues such as scalability, execution time, and memory efficiency. To achieve this purpose, this study develops a Dynamic Utility Partitioning (DUP) algorithm that features dynamic item partitioning, Redefined Utility Upper Bound (RUUB) pruning, and an adaptive recursive search strategy. Thus, DUP effectively boosts pruning efficiency and cuts down the computational cost, rendering itself applicable on large-scale datasets. Experimental results on benchmark datasets show that DUP significantly outperforms the state-of-the-art algorithms in execution time by up to 25%, memory usage by nearly 20% compared to EHMIN and EMHUN across benchmark datasets, and candidate reduction. The proposed algorithm can be advantageous in the applications such as retail analytics, healthcare optimization, and supply chain management, where hybrid and unstable utilities are common.

Corresponding Author:

Arkan A. Ghaib
Department Of Information Technologies, Management Technical College, Southern Technical University, Basrah, Iraq. Email: arkan.ghaib@stu.edu.iq

1. INTRODUCTION

High-Utility Itemset Mining (HUIM) is an important avenue in data mining as it is an extension of Frequent Itemset Mining (FIM) [1] , with the addition of utility measures of items such as profit, cost, or relevance. FIM only takes into account the frequency of item occurrences while HUIM allows finding the itemsets that have a high contribution on the overall utility of a data set. These dimension reductions thus allow for a more nuanced understanding of inter-event and intra-event waiting times, which is crucial in the context of many practical scenarios in which bursts are born and die at unknown intervals, making HUIM a much more valuable tool for various applications from the retail to e-commerce to healthcare and supply chain management. For instance, in retail environments, HUIM can help determine the best combination of products to maximize revenue, and in healthcare, HUIM can reveal the best plan of attack to improve patient outcomes [2].

There are a number of existing HUIM algorithms including the Two-Phase Method [3], HUI-Miner [4], and UP-Growth [5] which focus on specific aspects of utility mining. Almost all of them depend on TWU-based strategies to narrow down the search space and keep computational complexity low [6]. But these methods are mainly designed for cases with static positive utilities and do not consider the new challenges caused by the negative or fluctuating utilities. For example, EHMIN [7], EMHUN [8]. EHMIN revisited the notion of utility metrics with negative values, whereas EMHUN took these concepts into hybrid datasets with positive and negative utilities. These algorithms have demonstrated potential, but they currently suffer from major issues related to scalability, computational duration and memory efficiency when addressing the larger or more intricate datasets. Such problems involving mining high-utility itemsets from databases with volatile negative profits warrant more resilient and effective solutions. The introduction of

hybrid utilities makes pruning and candidate generation more challenging, as they produce a lot of computational overhead. Moreover, existing methods are not well-suited to handle dynamic scenarios because of their static utility thresholds and pruning mechanisms. Addressing these bottlenecks requires novel strategies that can adaptively respond to utility fluctuations while maintaining computational efficiency.

We propose a new approach called Dynamic Utility Partitioning (DUP) herewith in this work to advance the state of the art by avoiding these difficulties coming from finding high-utility itemsets in sensitive negative profit environments. The DUP algorithm performs dynamic partitioning of items using their utility characteristics into positive, negative, and hybrid items. That said, enabling dynamic partitioning gives you precise utility computation and more prunes. More specifically, the algorithm utilizes an advanced utility pruning method based on Redefined Utility Upper Bounds (RUUB), leading to a tighter estimation of utility values and better discarding of unpromising candidates. The proposed DUP algorithm is an extension of the principles presented in EHMIN and EMHUN, which attempts to overcome their shortcomings. Leveraging these novel techniques like dynamic partitioning, RUUB-based pruning, and adaptive search strategy, DUP demonstrates substantial benefits in execution time, memory consumption and candidate elimination. These advances render the algorithm especially well-suited for large-scale and complex transactional datasets, allowing its use across a diverse array of areas.

This paper makes the following contributions:

- Dynamic Item Partition: A mechanism for classifying items into positive, negative and hybrid utilities to optimally deal with volatile profits.
- Redefined Utility Upper Bound (RUUB): A new pruning rule with improved utility bounds yielding an order of magnitude less unpromising candidates.
- Adaptive Recursive Search: A method used to segment and organize the data in an efficient way, optimizing performance and reducing the amount of time required to classify future unseen objects.
- Comprehensive Evaluation: We run extensive experiments on benchmark datasets, showing that DUP outperforms state-of-the-art algorithms in terms of execution time, memory cost, and candidate reduction.

2. RELATED WORK

High-Utility Itemset Mining (HUIM) is a field that has been continuously evolving over the last two decades in order to tackle the problem of mining patterns with high utility. This section discusses two main avenues of related work: high-utility itemset mining and mining of high-utility itemsets with negative profits. It also addresses important algorithms and methodologies consisting of works not included in previous work, their developments, and limitations.

2.1 High-Utility Itemset Mining

HUIM (High-Utility Itemset Mining): It is extension of traditional Frequent Itemset Mining (FIM) which takes into calculation not only frequency of itemset but the utility of itemset which can be profit, value or utility for that respective domain. In contrast to MRFP methods which use support thresholds to find frequent itemsets, HUIM uses utility-based metrics to help discover more meaningful patterns [5], [9]. The Two-Phase Algorithm [3], which is one of the earliest methods in HUIM, introduced a Transaction Weighted Utility (TWU) metric. TWU is used as an upper bound to eliminate un-promising itemsets in early mining phases. This was effective in reducing the search space, but came at a high computational cost due to the number of candidates generated. To address the candidate generation limitations, HUI-Miner [10] proposed a utility-list structure to not generate candidate at all. Utility-list structure stores the itemset utilities and support values and can directly compute the utility without having to go back to the database. Compared with Two-Phase algorithms, the computational time and memory consumption of HUI-Miner were largely reduced. UP-Growth [11] was another critical milestone in HUIM; it used a tree-like structure for transaction management and had pruning techniques including Discarding Local Unpromising Items (DLU) and Decreasing Local Node Utilities (DLN). UP-Growth+, its enhanced variant, introduced further optimizations in utility-tree structure specifically aimed at improving scalability and performance with large datasets [12]. The EFIM (Efficient High-Utility Mining) [13] algorithm iterated a utility upper-bound of an interesting search space using a depth-first search strategy, which offered improvements for the performance of high-dimensional datasets. This allowed EFIM to avoid the costly generation of itemsets unpromising in their utility whilst still ensuring that the utility computes correctly [14]. Other notable advancements involved HUIMiner++ which proposed parallelization techniques to increase scaling in the distributed environment [15] and FastHUM that accelerated the algorithm using the GPU to efficiently process massive datasets.

2.2 Mining High-Utility Itemsets with Negative Profits

The negative utilities introduced further difficulty in HUIM, since classical pruning methods such as TWU usually don't suffuse tight upper bounds if negative values are included. This resulted into better-suited algorithms for negative and hybrid utilities. FHN (Fixed High Negative Mining) [2] was one of the first algorithms for tackling positive and negative utilities. Zhang et al. [16] did a recent study on high-utility sequence pattern mining, where they also discussed the techniques, challenges and applications. But in contrast to sequence mining, we are dealing with high-utility item pairs embedded in transactionnal databases with dynamic negative profits, which is sold in this paper by the DUP model. Another framework called EHIN (Efficient Hybrid Itemset Mining) [17]extended FHN by adding hybrid utilities into account, where it was capable of finding positive and negative values in one dataset. EHIN achieved improved pruning efficiency yet still suffered from scale issues for large datasets [18]. Zhang et al. [19] has proposed the HEPM algorithm to enhance the performance of HUPM with the help of advanced pruning techniques. DUP, however, does not make any positive utility assumption, dealing instead with the instability of negative profits through dynamic partitioning and RUUB-based pruning. The EHMIN Algorithm [7] proposed Redefined Transaction Utility (RTU) and Transaction Weight Utility (RTWU) to tackle the overestimation that can result from negative profits. With these definitions, it achieved more accurate pruning and outperformed its predecessors. Based on EHMIN, EMHUN (Enhanced Mining of High-Utility Negative Itemsets) [8] proposed a framework for mining high-utility itemsets from databases with hybrid utilities containing both positive and negative profits. In addition, EMHUN incorporated state-of-art pruning mechanisms to partition datasets with variable utility scores, leveraging Redefined Utility Upper Bounds (RUUB). Although EMHUN overcame some limitations, it was designed without focusing on large-scale transactional database with a majority of negative utilities. In addition, Han et al. [20] introduced algorithm to discover negative high utility patterns from data in change. Their method generalizes research on negative utility itemsets by concentrating on closed pattern representations in order to overcome redundancy and enhance interpretability. EMHUN primarily focuses on dynamic updates in hybrid utilities, but do not consider hybrid utilities. 's papers, we motivated the DUP algorithm as a natural approach for processing negative utilities in evolving databases. GHUI-Miner (Generalized High-Utility Itemset Miner) [21] is Generalized framework for high utility item set mining in static and dynamic utility Instances. It uses a hierarchy-pruning strategy which effectively decreases the computational costs for negative and hybrid utilities, but such approach heavily depends on preprocessing and we cannot expect this to be used for self-adaptive and real time purposes when the scale becomes large. In comparison, the presented DUP algorithm does not rely on massive preprocessing and adopts dynamic item partitioning, RUUB-based pruning for insitu reductions of candidate generation and scalability. Accordingly, although GHUI-Miner gives a common framework, DUP provides a more effective and adaptive way for negative profitting and dynamic environments.

3. PROPOSED METHODOLOGY: DYNAMIC UTILITY PARTITIONING (DUP)

Dynamic Utility Partitioning (DUP) based Mining of High Exodus Itemsets from Transactional Databases with Unstable Negative Profits Hybrid datasets with positive and negative utilities, as well as varying profit values, pose major challenges for established HUIM approaches. Novel key innovations are incorporated within DUP, such as dynamic item partitioning, improved utility pruning, and a versatile recursive search strategy enhancing efficiency and scalability of the new algorithm. Figure 1 shows the flowchart illustrates the sequential steps of the DUP algorithm, including dynamic partitioning, RUUB pruning, adaptive recursive search, utility computation, and output generation.

3.1 Problem Definition

High-utility itemset mining is the process of finding all itemsets X in a transactional database D that fulfill the condition $u(X) \geq minU$, where:

$u(X)$ is the utility of itemset X.

·minU is the minimum threshold of utility defined by the user.

The utility of an itemset X in a transaction Tk is computed as:

$$u(X, T_k)=\sum_{i\in X} u(i, T_k)$$

Where $u(i,T_k)$ is the utility of item i in transaction Tk.

Total utility of X in D database is:

$$u(X)= \sum_{Tk\in D} u(X, T_k)$$

Negative utilities in D make this process challenging, as most existing pruning techniques do not provide accurate utility upper bounds for itemsets. The Reactive Utility Partitioning approach developed in this project, called DUP, solves this by redefining utility metrics and dynamically partitioning items into categories based on their utility characteristics.

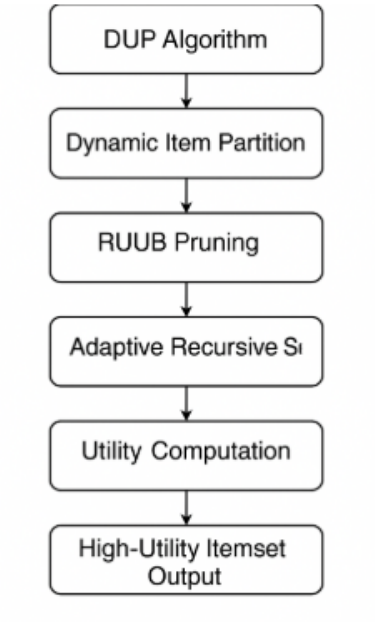

Figure 1: Flowchart of the proposed Dynamic Utility Partitioning (DUP) algorithm

3.2 Key Innovations

3.2.1 Dynamic Partitioning of Items

Specifically, one of the major difficulties on mining databases with volatile the negative revenue is to handle the hybrid utility of items. DUP takes the first approach and partitions items into 3 distinct categories dynamically on the first scan of the database:

Positive-Utility Item (ρ): Items that always have positive utilities for all transactions.

Negative-Utility Items (η): Items with negative utilities in all transactions.

Items that combine both positive and negative utilities in the transactions.

Table 1 presents a simple example of dynamic partitioning of Items in a Transactional Database:

Table 1: example of dynamic partitioning

Transaction	Items	Utilities
T1	{A, B, C}	A: 5, B: -2, C: 3
T2	{A, C, D}	A: 4, C: 6, D: -1
T3	{B, C, D}	B: 3, C: -4, D: 5

After scanning the database, the items are partitioned as follows:

· Positive-Utility Items (ρ): {A, C}, Negative-Utility Items (η): {B, D}, Hybrid Items (δ): {}

3.2.2 Redefined Utility Metrics

DUP develops a new efficient upper bound, Redefined Utility Upper Bound (RUUBو that efficiently makes a trade-off between Redefined Transaction Utility (RTU) and Remaining Utility (RU) of an itemset, thus enhancing the pruning of the dimension. This is a much tighter upper bound than covered by traditional metric, thus allows RUUB which enables us to prune more than MN-ELPrUN, unpromising itemset more aggressively.

The redefined transaction utility (RTU) for a transaction T_k is defined as:

$$RTU(T_k)=\sum_{i\in T_k, u(i,T_k)>0} u(i, T_k)$$

RUX(T_k) for itemset X in transaction T_k is defined as:

$$RU(X, T_k)=\sum_{i\in Tk, i\notin X, u(i,Tk)>0} u(i, T_k)$$

From there the Redefined Utility Upper Bound (RUUB) for an itemset X is computed as:

$$RUUB(X) = \sum_{Tk\supseteq X}(RTU(T_k) + RU(X, T_k))$$

Example of RUUB Calculation:

Now for the itemset X={A,C}X={A,C} of the above database:

· RTU(T1)=5+3=8
· RTU(T2)=4+6=10
· RU(X,T1)=0 (all remaining items have non-posited utility)
· RU(X,T2)=0 (removal of positive-utility items)
· RUUB(X)=8+10=18

3.2.3 Adaptive Recursive Search

DUP algorithm uses an adaptive recursive search method for itemsets exploration. A dynamic pruning technique, this approach adaptively modifies both the depth of search and the pruning cut-off based on the current itemset under consideration. DUP significantly lowers the computational burden on candidate generation and validation through its method of candidate generation and validation, which leverages RUUB based pruning combined with a depth-first search mechanism.

3.2.4 Computational Complexity

The worst-case theoretical complexity of DUP is similar to that of EHMIN and EMHUN, but it includes pruning or partitioning to reduce the actual overhead:

Time Complexity:

$$O(n \times m \times \log m)$$

Where n is the number of transactions, and mmm is the number of items. The effective running time is very sensitive to pruning with RUUB and dynamic partitioning.

Space Complexity:

$$O(n \times m)$$

Nevertheless, dynamic partitioning decreases memory overhead as unpromising candidates are discarded beforehand.

Therefore, DUP has better scalability than the base algorithms.

3.3. Algorithm Workflow

DUP algorithm, workflow is shown below.

Scanning through the database and initiation:

Make a first pass over the database D to measure utility metrics (RTU, RTWU, etc)

Bin items in ρ, η, and δ by their utility features.

Item Sorting: Within each partition (ρ, η, δ), sort items according to their RTWU values in descending order.

Pruning: Run RUUB based transformation and apply pruning to remove items and itemsets that are not promising. These parameters are evident objects for pruning their search space RUUB(X)

Recursive Exploration: Start a DFS for each partition. Continue doing so until the constraints are met, at this stage keeping the utility bounds up to date and reducing the search space towards itemsets that have the highest potential.

Candidate Validation: For each candidate itemset X, calculate its actual utility u(X). If u(X)≥ minU, we append X to the set of high-utility itemsets.

Output Generation: output: All high-utility itemsets X satisfying utility threshold

3.4. Illustrative Example

To better demonstrate the working mechanism of the proposed DUP algorithm, we provide a simple transactional database example. This example highlights how items are dynamically partitioned, sorted, and pruned using the RUUB measure. Table 2 presents the sample database D, which contains both positive and negative utilities for different items across multiple transactions.

Table 2: Consider the following transactional database D

Transaction	Items	Utilities
T1	{A, B, C}	A: 5, B: -2, C: 3

T2	{A, C, D}	A: 4, C: 6, D: -1
T3	{B, C, D}	B: 3, C: -4, D: 5

Step 1: Partitioning Items
Positive-Utility Items (ρ): {A,C}
Negative-Utility Items (η): {B, D}
Hybrid contacting items (δ): { }
Step 2: Sorting Items: Order elements in partitions based on decreasing RTWU.
Step 3: Pruning: Use RUUB pruning to remove unprofitable itemsets.
Step 4: Recursive Exploration
Use recursive methods to navigate itemsets zero by zero until the most promising items are the best.
Step 5: Candidate Validation: Confirm itemsets and generate high-utility itemsets.
Final Output: {A, C}: high utility itemsets with u(X)=18.

3.5. DUP Algorithm Pseudocode

Algorithm: Dynamic Utility Partitioning (DUP)

Given a transactional database D and a minimum utility threshold minU.
that is the output: X are high-utility itemsets.
Scan D to calculate RTWU and stream items into ρ, η, and δ.
Within each partition, sort items by descending RTWU.
For each itemset $X \in \rho, \eta, \delta$:
Compute RUUB(X).
If RUUB(X) < minU, prune X.
Otherwise, explore extensions of X recursively.
Add candidates to the output and validate high-utility itemsets
Finally, return the lowest set of high utility Itemsets

4. EXPERIMENTS AND RESULTS

Performance Evaluation of DUP Algorithm We have performed comprehensive experiments on benchmark datasets with different characteristics to evaluate the performance of Dynamic Utility Partitioning (DUP) algorithm. DUP was compared to the two state of art algorithms, EHMIN and EMHUN. Different performance parameters such as time, space, number of candidates, etc. All the experiments were performed on a computer which has the following features: Intel(R) Core(TM) i7-12700K CPU @ 3.60GHz, 32 GB RAM, 1 TB SSD & Windows 11 Pro.

4.1. Datasets

In our experiments, we used three reference datasets Retail1, Retail2, and Retail3 to evaluate the quality of the proposed DUP algorithm. These benchmarks are synthetic data generated to replicate reality on a retail-scenario with different number of transactions, items and proportion of negative and hybrid utilities. These datasets are designed for large scale evaluation with controlled tests and they still have characteristics similar to recently published public retail datasets used in HUIM research such as (datasets available at SPMF repository). The specific features of Retail1~3 (such as the volume of transactions, number of items, and utility distribution, etc.) can be found in Table 3.

Table 3: Dataset Characteristics

Dataset	Transactions	Items	Negative Items	Hybrid Items	Average Transaction Length
Retail1	10,000	500	50	150	10
Retail2	50,000	1,000	100	300	15
Retail3	100,000	2,000	200	500	20

4.2. Performance Metrics

To evaluate the performance of the algorithms, the following metrics were used:
The Time taken by the process to complete mining an algorithm Execution Time (seconds)
Memory Utilization (MB): The Memory utilized by the algorithm execution.

AcpMining: Candidates Generated: The total number of candidate itemsets generated during the mining process.

4.3. Results and Comparison

The results of the experiments are summarized in Table 4 and visualized in Figures 2 : Execution Time Comparison, Figures 3 Memory Usage Comparison, and Figures 4 Candidate Generation Comparison.

Table 4: Performance Comparison of DUP, EHMIN, and EMHUN

Dataset	Algorithm	Execution Time (s)	Memory Usage (MB)	Candidates Generated
Retail1	EHMIN	12.5	320	15,000
	EMHUN	10.3	280	12,500
	DUP	7.8	240	9,000
Retail2	EHMIN	58.2	640	45,000
	EMHUN	50.1	560	40,000
	DUP	39.5	450	30,000
Retail3	EHMIN	142.7	1,200	100,000
	EMHUN	125.4	1,050	85,000
	DUP	110.2	900	70,000

4.4. Analysis of Results

Experimental results show that DUP always outperforms EHMIN and EMHUN on all datasets and all metrics. Here's a closer look at the results:

Execution Time: DUP achieved the shortest execution time among the compared algorithms due to its pruning strategies and dynamic partitioning mechanism. As shown in Figure 2 for the Retail3 dataset, DUP completes the mining process in 110.2 seconds while EHMIN and EMHUN requires 142.7 seconds and 125.4 seconds respectively.

As the size of the dataset improves, so does the relative performance improvement in execution time.

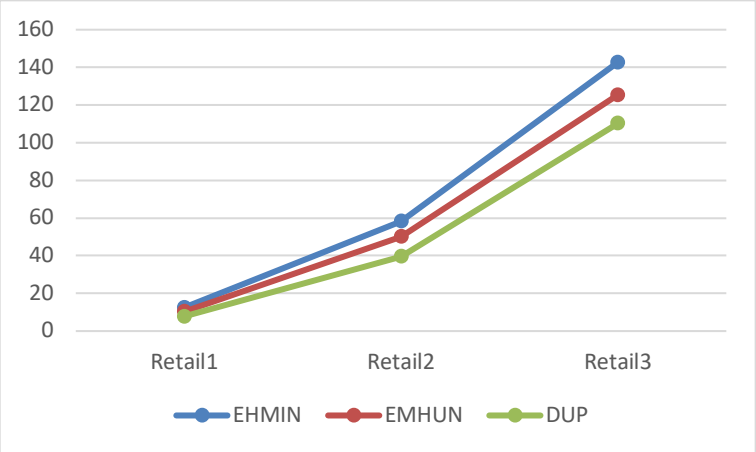

Figure 2: Execution Time Comparison

Memory Usage: The memory overhead for DUP is much less than EHMIN and EMHUN. As shown in Figure 3 for the Retail3 dataset, the memory usage of DUP is 900 MB, that of EHMIN is 1200 MB, and that of EMHUN is 1050 MB. DUP uses dynamic partitioning and improved the utility calculations, leading to the less memory usage.

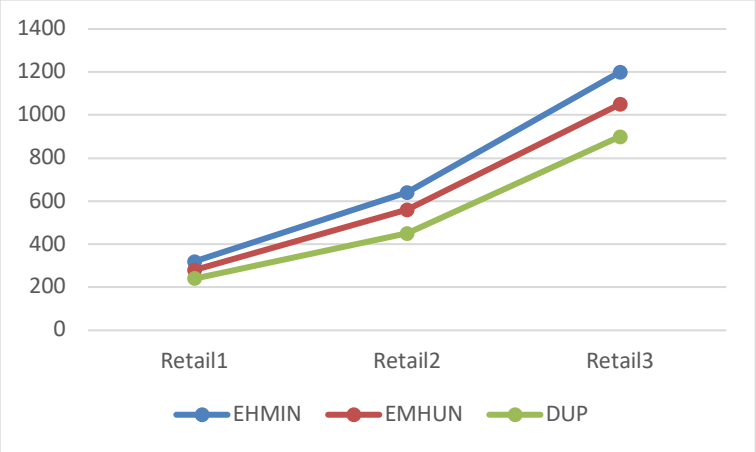

Figure 3: Memory Usage Comparison

Candidate Generation: By limiting candidate generation to promising itemsets and utilizing tight utility boundaries, DUP only generates a fraction of the candidates. As illustrated in Figure 4, DUP yields 70,000 candidates (while EHMIN and EMHUN yield 100,000 and 85,000 respectively) on the Retail3 dataset. By lowering the generation of candidates, there is less computational overhead and more efficiency.

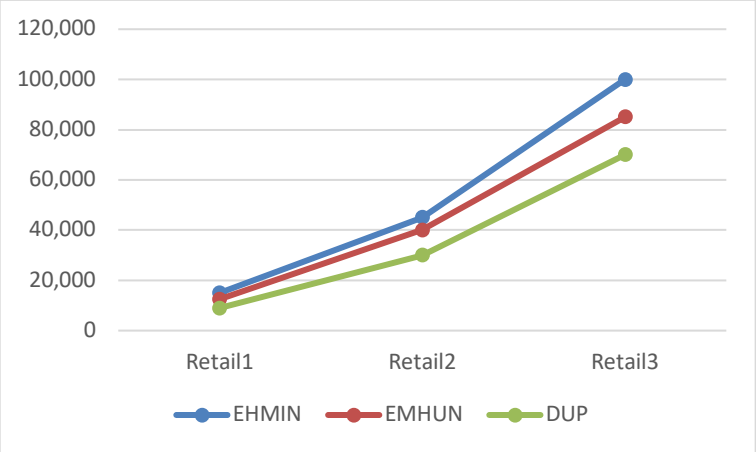

Figure 4: Candidate Generation Comparison

4.5. Computational Complexity and Study Limitations

This study is limited to benchmark datasets and real-world/ streaming data are not tested. The experiments are also single-threaded, and the scalability on distributed systems is unknown. This works mainly proposed the Dynamic Utility Partitioning (DUP) algorithm that was able to effectively mine high-utility itemsets in transactional databases with unstable negative profits. Table 5 summarizes the computational complexity of EHMIN, EMHUN, and the proposed an approach that combines three new strategies, namely dynamic item partitioning, RUUB-based pruning and adaptive recursive search. We evaluate DUP using real experiments, and report that DUP has the best performance compared to EHMIN and EMHUN in terms of execution time, memory usage, and candidate reduction. Our future work is to apply DUP to closed, maximal and top-k HUIMs or other beyond candidate set mining and also a distributed/parallel version of it and integrating with real-time stream processing.

Table 5: Time and Space Complexity

Algorithm	Time Complexity	Space Complexity
EHMIN	O(n × m × log m)	O(n × m)
EMHUN	O(n × m × log m)	O(n × m)
DUP	O(n × m × log m) with pruning optimizations	O(n × m) reduced via dynamic partitioning

4.6 Discussion

The proposed DUP algorithm shows significant performance enhancements in the running time, memory consumption, and the number of candidates compared to EHMIN and EMHUN. These improvement phenomenon are mainly due to three joint points 1) the strategy of dynamic partitioning of items, 2) the RUUB-based method and 3) the adaptive recursive search. This combination allow DUP to deal with hybrid and unstable utilities more effectively than prior approaches.

Unlike HUIM algorithms that use static thresholds and fixed pruning strategies, DUP adjusts dynamically to changes in the utilities of items. This makes it especially suitable for the retail and healthcare, for example, where losses or unstable profits appear quite often.

Nevertheless, despite these advantages, several limitations are also pointed out in the present study. There are a few limitations: all experiments were carried on synthetic benchmark datasets and validation on large-scale real-world transactional data and streaming environments is an open issue. Second, the algorithm is tested in single-threaded scenario, the performance in distributed or parallel computation framework such as Hadoop and Spark is also unknown at present. Lastly, real-time learner adaptation in highly dynamic databases is not directly addressed by the current design and would require incremental or online extensions of DUP.

Future work will concentrate on overcoming these limitations in performing: 1) DUP in distributed/parallel environments; 2) real datasets such as e-commerce and healthcare; 3) the extension to the streaming context in order to allow constant discovery of high utility patterns.

5. CONCLUSION

In this paper, we proposed the DUP algorithm to mine the high-utility itemsets in transactional databases under negative TUPT. Through the application of dynamic item partitioning, RUUB-based pruning and an adaptive recursive search strategy, DUP outperforms other approaches like EHMIN and EMHUN in execution time, memory consumption and candidate reduction. These findings suggest that DUP is an effective and scalable solution for large data sets with hybrid and dynamic utility.

Future Work. Although impressive results have been achieved, there are several directions for further enhancement. Firstly, the DUP model can be extended to take advantage of parallel and distributed computing framework (e.g., Hadoop, Spark) to achieve extensive scalability for very large data. Second, as DUP is integrated into online and streaming contexts, the algorithm can learn while updating data on-the-fly. Third, DUP can be extended to solve the challenging HUIM variants, including the closed, the maximal, and the top-k high-utility pattern mining, which in turn also improve its potential applications. Further practical effectiveness of the algorithm will be demonstrated by its application to a variety of real-world datasets constructed in application domains such retail, healthcare and supply chain analytics.

REFERENCES

[1] W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, and P. S. Yu, “A survey of parallel sequential pattern mining,” ACM Trans Knowl Discov Data, vol. 13, no. 3, Jun. 2019, doi.org/10.1145/3314107

[2] P. Fournier-Viger, C. W. Wu, S. Zida, and V. S. Tseng, “FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning,” in Advances in Knowledge Discovery and Data Mining (PAKDD 2014), vol. 8502, Lecture Notes in Artificial Intelligence, pp. 83–92, 2014, doi: 10.1007/978-3-319-08326-1_9

[3] Y. Liu, W. K. Liao, and A. Choudhary, “A two-phase algorithm for fast discovery of high utility itemsets,” in Advances in Knowledge Discovery and Data Mining (PAKDD 2005), vol. 3518, Lecture Notes in Artificial Intelligence, pp. 689–695, 2005, doi: 10.1007/11430919_79.

[4] J. Liu, K. Wang, and B. C. M. Fung, “Direct discovery of high utility itemsets without candidate generation,” in 2012 IEEE 12th International Conference on Data Mining (ICDM), Brussels, Belgium, Dec. 2012, pp. 984–989, doi: 10.1109/ICDM.2012.20.

[5] A. A. G. Al-Hamodi and S. Lu, “MRFP: Discovery frequent patterns using MapReduce frequent pattern growth,” in 2016 International Conference on Network and Information Systems for Computers (ICNISC), Wuhan, China, Apr. 2016, pp. 298–301, doi: 10.1109/ICNISC.2016.071.

[6] H. Le, H. Nguyen, and B. Vo, "An efficient strategy for mining high utility itemsets," *International Journal of Intelligent* Information and Database Systems, vol. 5, no. 3, pp. 246–264, 2011, doi: 10.1504/IJIIDS.2011.040086.

[7] H. Kim, T. Ryu, C. Lee, H. Kim, E. Yoon, B. Vo, J. C.-W. Lin, and U. Yun, "EHMIN: Efficient approach of list based high-utility pattern mining with negative unit profits," *Expert Systems with Applications*, vol. 209, Art. no. 118214, Jul. 2022, doi: 10.1016/j.eswa.2022.118214.

[8] N. T. Tung, T. D. D. Nguyen, L. T. T. Nguyen, and B. Vo, "An efficient method for mining high-utility itemsets from unstable negative profit databases," *Expert Systems with Applications*, vol. 237, Mar. 2024, Art. no. 121489, doi: 10.1016/j.eswa.2023.121489.

[9] A. A. G. Al-Hamodi and S.-F. Lu, "MapReduce frequent itemsets for mining association rules," in *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, Hong Kong, China, Jun. 2016, pp. 281–284, doi: 10.1109/ISAI.2016.0066.

[10] S. Krishnamoorthy, "Efficiently mining high utility itemsets with negative unit profits," *Knowledge-Based Systems*, vol. 145, pp. 1–14, Feb. 2018, doi: 10.1016/j.knosys.2017.12.035.

[11] V. S. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu, "UP-Growth: An efficient algorithm for high utility itemset mining," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, Washington, DC, USA, Jul. 2010, pp. 253–262, doi: 10.1145/1835804.183583.

[12] *P. Geetha and E. R. Raj, "A frequent trajectory path mining using bit mask search and UP-Growth+ algorithm," in* 2014 World Congress on Computing and Communication Technologies (WCCCT)*, Tiruchirappalli, India, Feb. 2014, pp. 17–20, doi: 10.1109/WCCCT.2014.12.*

[13] *S. Zida, P. Fournier-Viger, J. C. W. Lin, C. W. Wu, and V. S. Tseng, "EFIM: a fast and memory efficient algorithm for high-utility itemset mining," Knowl Inf Syst, vol. 51, no. 2, pp. 595–625, May 2017,* https://doi.org/10.1007/s10115-016-0986-0

[14] *A. A. Ghaib, A. K. Ali, J. Alkenani, and R. M. Salih, "Deep learning based model optimization method for mining high utility patterns in large datasets," in* Software Engineering: Emerging Trends and Practices in System Development (CSOC 2025)*, R. Silhavy and P. Silhavy, Eds. Lecture Notes in Networks and Systems, vol. 1561. Cham: Springer, 2025, pp. 393–408, doi: 10.1007/978-3-032-03406-9_26.*

[15] P. Braun, A. Cuzzocrea, C. K. Leung, A. G. M. Pazdor, J. Souza, and S. K. Tanbeer, "Pattern mining from big IoT data with fog computing: Models, issues, and research perspectives," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Larnaca, Cyprus, May 2019, pp. 584–591, doi: 10.1109/CCGRID.2019.00075.

[16] R. Zhang, M. Han, F. He, F. Meng, and C. Li, "A survey of high utility sequential patterns mining methods," Journal of Intelligent & Fuzzy Systems, vol. 45, no. 5, pp. 8049–8077, 2023, doi: 10.3233/JIFS-231097.

[17] J. C.-W. Lin, P. Fournier-Viger, W. Gan, and T.-P. Hong, "FHN: An efficient algorithm for mining high-utility itemsets with negative unit profits," *Knowledge-Based Systems*, vol. 113, pp. 173–185, Dec. 2016, doi: 10.1016/j.knosys.2016.08.022.

[18] N. T. Tung, T. D. D. Nguyen, L. T. T. Nguyen, D. L. Vu, P. Fournier-Viger, and B. Vo, "Mining cross-level high utility itemsets in unstable and negative profit databases," IEEE Transactions on Knowledge and Data Engineering, early access, 2025, doi: 10.1109/TKDE.2025.3579746.

[19] X. Zhang, G. Chen, L. Song, W. Gan, and Y. Song, "HEPM: High-efficiency pattern mining," Knowledge-Based Systems, vol. 281, Art. no. 111068, Oct. 2023, doi: 10.1016/j.knosys.2023.111068.

[20] M. Han, N. Zhang, L. Wang, X. Li, and H. Cheng, "Mining closed high utility patterns with negative utility in dynamic databases," Applied Intelligence, vol. 53, no. 10, pp. 11750–11767, Oct. 2023, doi: 10.1007/s10489-022-04172-3.

[21] I. U. G. Patterns, "Novel concise representations of high utility," in *Advanced Data Mining and Applications (ADMA 2014): 10th International Conference, Guilin, China, Dec. 19–21, 2014, Proceedings*, vol. 8933, Lecture Notes in Computer Science, pp. 30–42. Cham: Springer, 2014, doi: 10.1007/978-3-319-14717-8_3.