Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Improving the Efficiency of Distributed Utility Item Sets Mining in Relation to Big Data

Arkan A. Ghaib[1] ✉ Yahya Eneid Abdulridha Alsalhi[2], Israa M. Hayder[3], Hussain A. Younis[4] and Abdullah A. Nahi[5]
[15]Department of Information Technology, Management Technical College, Southern Technical University, Basrah, Iraq
[2]General Directorate of Education Dhi Qar, Ministry of Education, Baghdad, Iraq
[3]Dept. Of Computer Systems Techniques, Qurna Technique Institute, STU, Basrah, Iraq
[4]College of Education for Women, University of Basrah, Iraq, School of Computer Sciences, University. Sains Malaysia, 11800 USM, Penang, Malaysia
Corresponding Author: Arkan A. Ghaib, **E-mail**: arkan.ghaib@stu.edu.iq

| ABSTRACT
High utility pattern mining is an analytical approach used to identify sets of items that exceed a specific threshold of utility values. Unlike traditional frequency-based analysis, this method considers user-specific constraints like the number of units and benefits. In recent years, the importance of making informed decisions based on utility patterns has grown significantly. While several utility-based frequent pattern extraction techniques have been proposed, they often face limitations in handling large datasets. To address this challenge, we propose an optimized method called improving the efficiency of Distributed Utility itemsets mining in relation to big data (IDUIM). This technique improves upon the Distributed Utility item sets Mining (DUIM) algorithm by incorporating various refinements. IDUIM effectively mines item sets of big datasets and provides useful insights as the basis for information management and nearly real-time decision-making systems. According to experimental investigation, the method is being compared to IDUIM and other state algorithms like DUIM, PHUI-Miner, and EFIM-Par. The results demonstrate the IDUIM algorithm is more efficient and performs better than different cutting-edge algorithms.

| KEYWORDS
High utility item sets, Item sets mining, High utility pattern, Parallel computing. Big data

| ARTICLE INFORMATION
ACCEPTED: 02 November 2023 **PUBLISHED:** 24 November 2023 **DOI:** 10.32996/jcsts.2023.5.4.12

1. Introduction

Patterns and correlations in large datasets. Its objective is to identify interesting associations and connections between variables or item sets, which can be used for prediction and decision-making purposes. With the growth of big data [Agrawal et al. 2003; Li et al. 2021], the analysis of such data and extracting meaningful patterns through frequent item set mining techniques have become significant research areas in data analysis.

High utility pattern mining (HUPM) is a specific data mining technique in which item sets or patterns are extracted from massive databases based on their high-utility values. This approach seeks to identify item sets with significant utility, highlighting their importance or usefulness. In other words, HUPM involves the process of uncovering patterns [Ahmed et al. 2009; Yao et al. 2004]. (HUPM) is a technique widely used in various domains like business and healthcare to extract the most valuable or useful patterns from large datasets [Chan et al. 2003]. Multiplying the profit and the number of items in an item set yields the utility value of that item set. For example, in retail, HUPM can identify profitable combinations of products customers frequently purchase items from the internet, while in healthcare, it can uncover effective treatment combinations for specific diseases [Liu et al. 2005; Baek et al. 2021].

Efficiently extracting high utility patterns from large datasets is a challenging task due to the exponential growth of potential item sets [Agrawal et al. 1994]. To address this challenge, several algorithms have been developed, including Apriori-based algorithms, FP-growth, and Eclat. These algorithms employ techniques like pruning and efficient data structures to mine high-utility patterns effectively. PHUI-Miner [Chen et al. 2016] has been proposed by Chen Yan. It is being developed in Spark because of its many advantages over Hadoop, which raise the algorithm's overall effectiveness. The mining process involves a trade-off between accuracy and performance because it is based on compression and testing that produces estimated HUIs. However, this method uses fewer resources and performs better with sampling. EFIM-Par [Tamrakar et al. 2017], a comparable version of the effective methodology for small datasets (EFIM), was proposed by Ashish Tamrakar.

In recent years, HUPM has gained significant popularity due to its relevance in various application domains [Han et al. 2000]. High utility item set mining (HUIM) has become increasingly common, where an item sets is considered a unit of utility that meets or exceeds a minimum utility threshold. However, utility measurements introduce complexities as they don't follow monotonic or anti-monotonic properties, leading to difficulties in comparing superset or subset item sets. To address this, a transaction-weighted utility measure has been developed [Nguyen et al. 2019]. Handling large datasets in HUPM poses challenges due to a single machine's limited processing and memory capabilities. Parallel computing algorithms have been developed to overcome these limitations and enable the efficient processing of large data [Gan et al. 2017]. The proposed Improving Distributed Utility Item sets Mining (IDUIM) algorithm aims to improve upon the shortcomings of the DUIM algorithm by integrating several enhancements.

The proposal of a scalable approach, the IDUIM algorithm, for mining item sets from massive data. It Improves and advances the DUIM algorithm [Gan et al. 2018]. Introducing a fairer concept for distributing the search space between compute nodes, taking into account both the transaction-weighted utility (TWUI) and the item's length in the subspace. This neutral division approach improves upon previous methods that only considered TWUI values [Vo et al. 2013]. Keeping portions of messages rather than complete transactions reduces the complexity of the process's storage [Yun et al. 2017]. By incorporating these contributions, the IDUIM algorithm aims to Improve the efficiency and effectiveness of high-utility item sets mining for large datasets.

2. Literature Review

Data mining is a widely used technique for extracting valuable information from large datasets [Krishnamoorthy 2015]. It has gained popularity in various domains, such as business, healthcare, finance, and social sciences. One popular data mining tool is MFP (Frequent Item sets Mining), which identifies sets of objects or item sets that appear frequently in transactions or exceed a specified threshold. The Apriori algorithm was one of the initial significant developments in frequent item sets mining, followed by the introduction of the FP-Growth algorithm that utilizes an efficient tree structure called the FP-tree. However, MFP does not consider the price or output of the products, leading to the emergence of weighted frequent pattern mining techniques that incorporate relative item values [Dalal et al. 2020].

Frequent item sets analysis serves as the foundation for various data mining tasks, including sequential mining and episode mining [Zhang et al. 2015],[Mustafa et al. 2022]. By considering both the internal utility (item quantity) and external utility (profit value), high utility pattern mining (HUPM) extends frequent item sets mining. HUPM has applications in a variety of industries, including marketing, click-stream analysis, biomedical technologies, and gene control. [Kumar et al. 2022]. To address the combinatorial explosion problem, researchers have proposed methods like the two-phase Apriori-based approach for discovering High Utility item sets (HUIMs) collections across different dataset scans [Wu et al. 2013].

High utility pattern mining has become a prominent research area within data mining, focusing on discovering frequent patterns in transactional datasets based on their utility values [Sundari et al. 2022]. Utility values go beyond traditional support and frequency measures by incorporating profits or costs associated with item sets. This mining technique finds applications in marketing, bioinformatics, finance, and other domains [Milhem et al 2022]. The UP-Tree, a data structure that efficiently stores high-utility item sets, is commonly used in various high utility pattern mining algorithms [Hassan et al 2022]. It leverages a tree structure to represent item sets and their utility values, enabling efficient pruning and search operations.

"Distributed Utility Item sets mining on Spark" by S. Althomali et al. (2015): A distributed utility item sets mining technique built on Apache Spark is presented in this paper. It concentrates on using Spark's in-memory processing capabilities to increase the effectiveness of extracting high utility item sets from huge datasets. [Sethi et al. 2018]. Hence, "Parallel Mining of High Utility Item sets on Big Data" by J. Zhang et al. (2016): The study suggests a MapReduce-based parallel technique for mining high utility item sets from massive data. It presents unique methods for scaling utility mining on massive datasets while lowering the computing cost. [Rathee et al. 2018]. While "Efficient Mining of High Utility Item sets from Large-Scale Data" by R. Nayak and D. P. Acharjya (2018): This work presents an efficient distributed algorithm for mining high utility item sets from large-scale data using Apache Hadoop. It employs pruning techniques and data partitioning strategies to improve mining performance and reduce the computational overhead [Lee et al. 2018]. As well "Efficient Utility Pattern Mining in Big Data using Apache Flink" by S. El-Menshawy

and K. El-Bahnasy (2020): The paper proposes an efficient utility pattern mining algorithm implemented on Apache Flink, a distributed stream processing framework. It focuses on leveraging the features of Flink, such as event time processing and state management, on achieving faster and scalable utility pattern mining on big data [Al-Bana et al. 2022]. Finally, "Efficient Mining of High Utility Item sets in Big Data using Spark and Shark" by K. C. Santosh and R. T. Goswami (2020): This work introduces an optimized approach for mining high utility item sets using Spark and Shark, an in-memory data analysis system built on top of Spark. It leverages Spark's distributed computing capabilities and Shark's columnar storage format to improve the efficiency of utility item sets mining on big data [Al-Rabeeah et al. 2019; Dalal et al. 2018].

These related works offer insights and techniques to Improve the efficiency of Distributed Utility Item sets mining in relation to big data. They explore various distributed processing frameworks and optimization strategies to improve mining performance, scalability, and resource utilization.

3. Methodology

To improve the efficiency of Distributed Utility Item sets mining in relation to big data, the following methodology can be adopted:

1. Problem Analysis: Begin by thoroughly understanding the requirements, constraints, and challenges associated with distributed utility item sets mining in relation to big data. Identify the specific performance bottlenecks and areas for improvement.
2. Data Partitioning: Divide the incoming data into more manageable groupings using suitable criteria, including horizontal or vertical partitioning. Parallel processing is made possible by this partitioning, which also speeds up total processing. Load Balancing: Ensure even distribution of workload among the processing nodes to prevent any nodes from becoming overloaded. Implement load balancing techniques to optimize resource utilization and ensure efficient processing.
3. Algorithm Optimization: Analyze and optimize the distributed utility item sets mining algorithms. Identify any computational or memory-intensive tasks and explore algorithmic enhancements, such as pruning techniques, efficient data structures, or parallel algorithm designs, to improve the performance.
4. Caching and Memory Management: Implement caching mechanisms to store intermediate results or frequently accessed data structures, reducing the need for redundant computations. Optimize memory management techniques to minimize disk I/O operations and enhance processing speed.
5. Scalability Considerations: Design the system to handle scalability requirements. Ensure that the distributed utility item sets mining process can scale horizontally by adding more processing nodes or vertically by utilizing more powerful hardware resources as the volume of data increases.
6. Profiling and Monitoring: Continuously monitor and profile the system performance to identify any performance bottlenecks or areas for optimization. Collect and analyze performance metrics to understand the system behavior and make informed decisions for further improvements.
7. Experimental Evaluation: Conduct experiments using benchmark datasets to evaluate the efficiency and effectiveness of the improving distributed utility item sets mining methodology. Compare the performance with existing approaches and algorithms to validate the improvements achieved.

As will be shown in the following figure, we can see that the methods were discussed and shown in how the steps were defined clearly. In summary, including a well-defined methodology in research papers promotes transparency, reproducibility, validity, and reliability of research findings. It also allows for advancing knowledge within a field and facilitates learning for researchers and students.

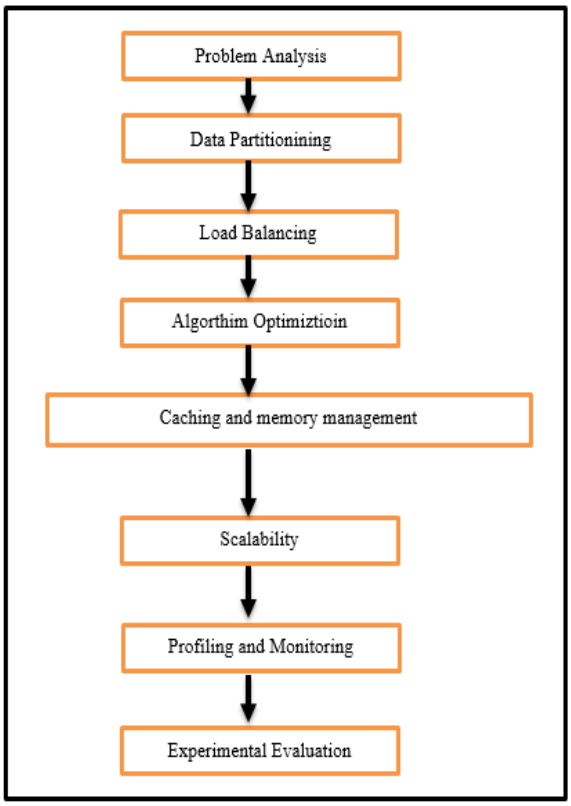


Figure 1. Methodology Steps

4. Proposed Work

The distributed IDUIM algorithm, which is used in the proposed work to extract item sets from enormous volumes of data, has been improved. In this method, the mining dataset is divided among the worker nodes, and each node independently computes high utility item sets. The model's numerous phases are succinctly described here.

4.1 IDUIM Algorithm

This section will provide an explanation of the DUIM algorithm, which is used to discover DHUIs through parallel loops in mining operations. The proposed concept involves performing a deep scan of the dataset twice, with the first scan identifying the TWUI and the second cycle producing a revised dataset. All low TWUI 1-item sets are eliminated from the dataset, and the remaining components are arranged in ascending order according to their TWUI in the updated transaction operation. The algorithm does not require the independent computation of TWUI since it is equivalent to the local utility of an item x. These items are known as secondary elements and should be included in transactions. Finally, the items are arranged in ascending order based on their TWUI.

Pseudocode of IDUIM(Distributed Utility item sets Mining) Algorithm:

Input: TD: transaction dataset, muT: minimum_utility_threshold
Output: a set of Improving Distributed utility item sets mining IDUIM
1: scanning Database to Compute the TD for item
2: Secondary(y) = {ei / ei ∈ EI ∧ TWUI (y,ei) ≥ muT}
3: Calculate the total utility value of TD
4: for all transaction t ∈ TD, do
5: Sort items E form utility u in secondary list
6: if t.utility ≠ NULL then
7: u= t.utility
8: end if
9: sort items in a sending order TWUI value in TD
10: if item's utility value < muT
11: stop processing items
12: end if
13: end for
14: Add the item to the current item sets
15: Generate all possible non-empty subsets of the current item sets
16: For each subset of the current item sets:
17: Calculate the total utility value (tuv) of the subset
18: If tuv of the subset ≥ muT
19: Add the subset to the list of DUIM
20: Otherwise, remove the subset from the list of high utility item sets
21: end if
22: end for
23: Remove the last item from the current item sets
24: Return the list of IDUIM

4.2 Data distribution and computation of DTWUI

As a component of the distributed file system, the high transactional data is equally split across the active nodes. Numerous worker nodes operate in parallel to complete the job. This is accomplished by utilizing the flat map and reducing the key functions. A value is mapped to more key-value pairs or zero using the flat map operation. Each node forms pairs of "item, utility" to handle the data that has been assigned to it. At the master node, the utility values of each item are grouped. Items that have high transactional weighted utility DTWUI below the utility threshold are removed from the transactions and dropped as questionable items. When updating the transactional database, only the items whose DTWUI is equal to or exceeds the threshold value are considered.

4.3 Modified dataset mining with sorted items

Item sets are simply discarded here if their DTWUI values fall less than the utility threshold since they are deemed unpromising item sets. A transaction that is empty is also removed. The transactions are sorted using Quicksort, and the remaining items are categorized using ascending DTWUI values. Since transactions contain items with rising DTWUI values, the revised version of the original dataset constitutes the entire dataset.

4.4 DUIM Mining at Nodes

The DUIM-Search technique is currently used to mine the item sets individually at each node [27]. Using ASU and ALU pruning techniques, each worker node computes the DUIs locally for the targeted subspace. The aggregated collection of results is produced by combining the findings of the worker nodes. DUIs. These DUIs make association rules for mining by revealing relationships between the components. The suggested work's whole process is shown in Fig. 2.

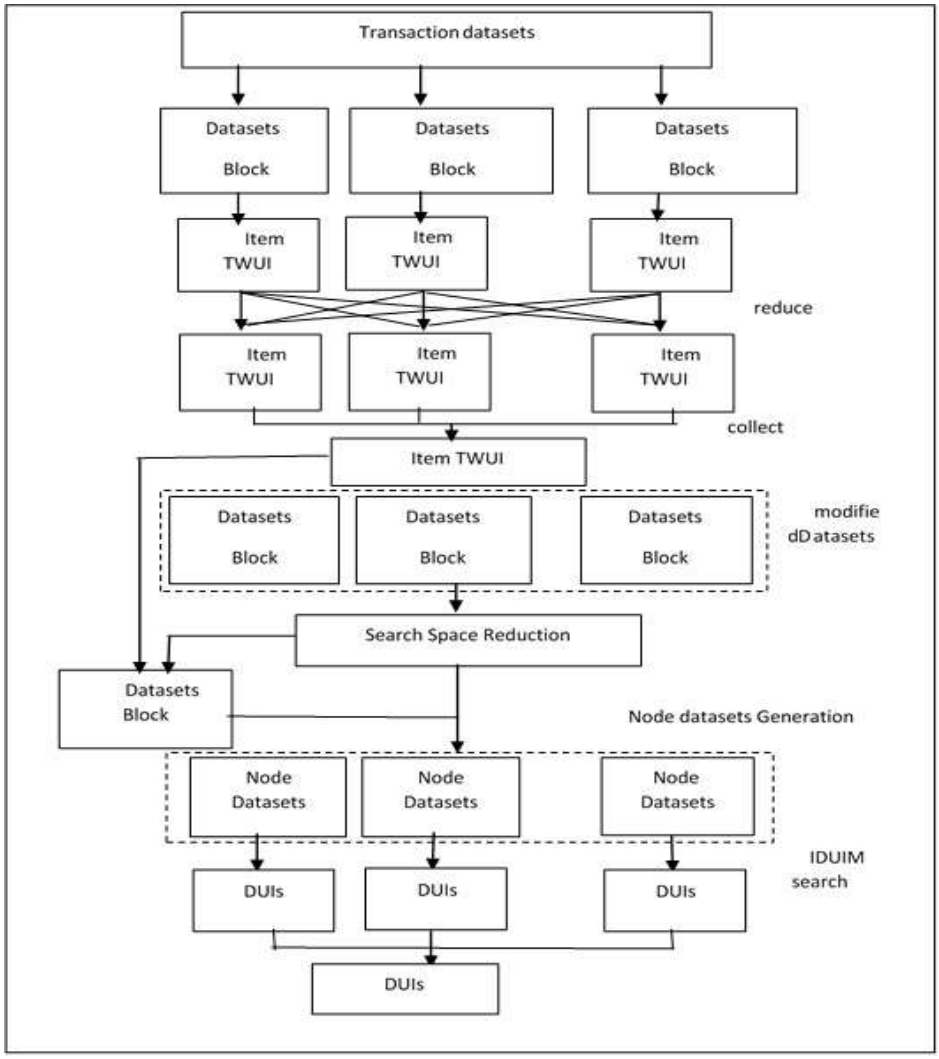


Figure 2. IDUIM Flow Graph

5. Experiments and Results

In this section, the results of the Improving Distributed Utility Mining IDUIM algorithm are discussed. All experiments were performed on a desktop computer with 8 GB of RAM and an Intel®Core(TM) i7 processor running at 2.80 GHz with Windows 10 OS.

5.1 Datasets

The SPMF library's real-world mining datasets Chainstore, Kosarak, Accident, and Connect are used to assess the performance of the IDUIM algorithm. The characteristics of these datasets are shown in Table 1, including the total number of transactions (#transaction), distinct items (#item), and the average number of item sets per transaction (#AvgItem). The Chainstore dataset contains grocery store data, while the Kosarak dataset consists of click-streams from a news portal. The Accident dataset is derived from actual events, and Connect features movement data from games.

Table 1 Characteristics of datasets.

Dataset	***#transaction***	***#item***	***#AvgItem***
Chainstore	*1112,949*	*46,086*	*7.23*
Kosarak	*990,002*	*41,270*	*8.1*
Accident	*340,183*	*468*	*33.8*
Connect	*67,557*	*129*	*43*

As will be shown in the next figure IDUIM Flow Graph starting with reducing the dataset and collecting the important data required for the process and then will be modified for the algorithm used for processing the data and generating the required nodes.

5.2 Performance assessment

The evaluation of the IDUIM algorithm is carried out based on three parameters, namely, execution time, memory requirements, and scalability, which are assessed in the following:

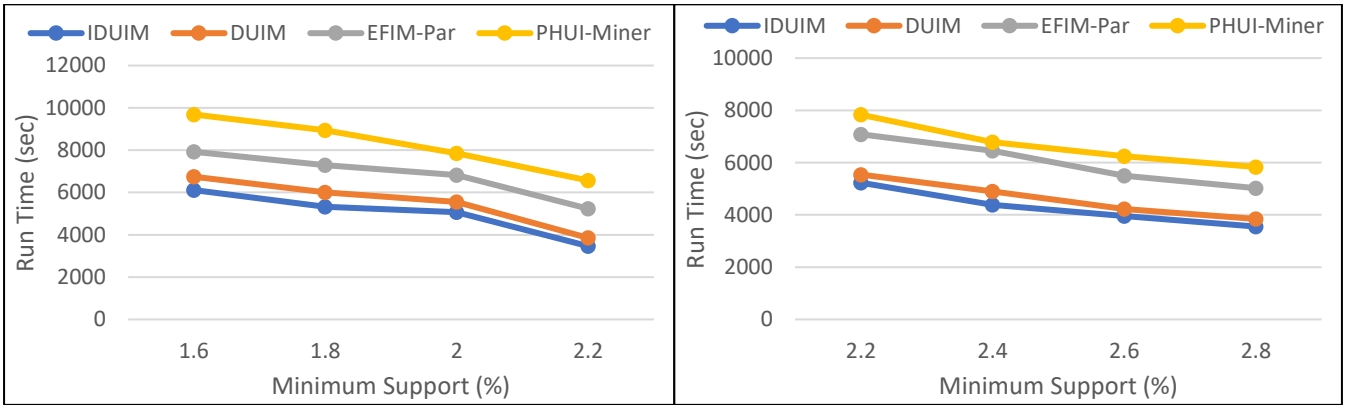
5.2.1. Execution time

The IDUIM algorithm is evaluated against other algorithms, DUIM, PHUI-Miner and EFIM-Par, in terms of runtime performance using four datasets and various thresholds. Table 2 displays the outcome of the Run Time results (seconds) tests. Results for the Chainstore dataset, displayed in Fig.3 (a), indicate that as threshold values increase, runtime decreases. The functionality of the algorithms is demonstrated on the Kosarak dataset in Fig.3 (b), and experiments on smaller datasets of Accident and Connect are shown in Fig.3 (c) and Fig.3 (d). The improved version of IDUIM exhibits superior performance when compared to the original version of DUIM and other algorithms, especially when dealing with large datasets containing a high volume of item sets per transaction.

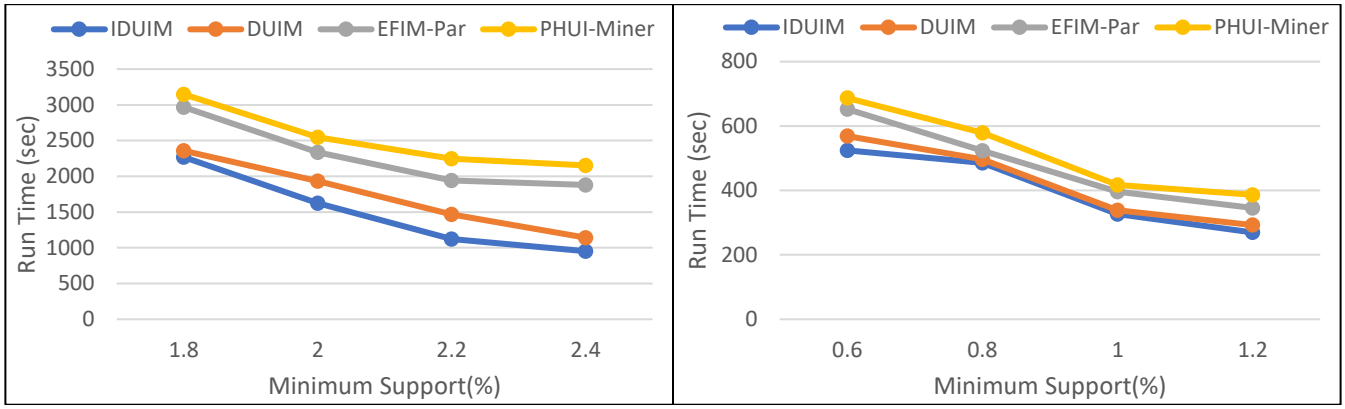
Table 2: Run Time (sec) on Various datasets

Chainstore			
Threshold%	*1.6*	*1.8*	*2*
IDUIM	*6112.54*	*5321.86*	*5072.65*
DUIM	*6744.63*	*6005.35*	*5547.66*
EFIM-Par	*7923.32*	*7287.76*	*6823.51*
PHUI-Miner	*9686.64*	*8932.83*	*7853.65*
Kosarak			
Threshold%	*2.2*	*2.4*	*2.6*
IDUIM	*5232.45*	*4386.75*	*3956.43*
DUIM	*5542.46*	*4894.23*	*4221.63*
EFIM-Par	*7084.73*	*6454.91*	*5498.34*
PHUI-Miner	*7834.85*	*6789.64*	*6245.75*
Accident			
Threshold%	*1.8*	*2*	*2.2*
IDUIM	*524.54*	*485.73*	*326.43*
DUIM	*568.73*	*496.41*	*338.65*
EFIM-Par	*652.62*	*523.69*	*396.42*
PHUI-Miner	*687.24*	*579.23*	*416.68*
Connect			
Threshold%	*0.6*	*0.8*	*1*
IDUIM	*2267.75*	*1623.84*	*1123.79*
DUIM	*2356.85*	*1932.56*	*1467.82*
EFIM-Par	*2967.37*	*2335.78*	*1943.24*
PHUI-Miner	*3147.23*	*2545.64*	*2246.31*

This improved processing speed is attributed to integrating binary recursive search for items and storing only crucial parts of transactions, which differs from the approach used in the previous version. As we can see in the following figures as mentioned above, the running time.



(a) (b)

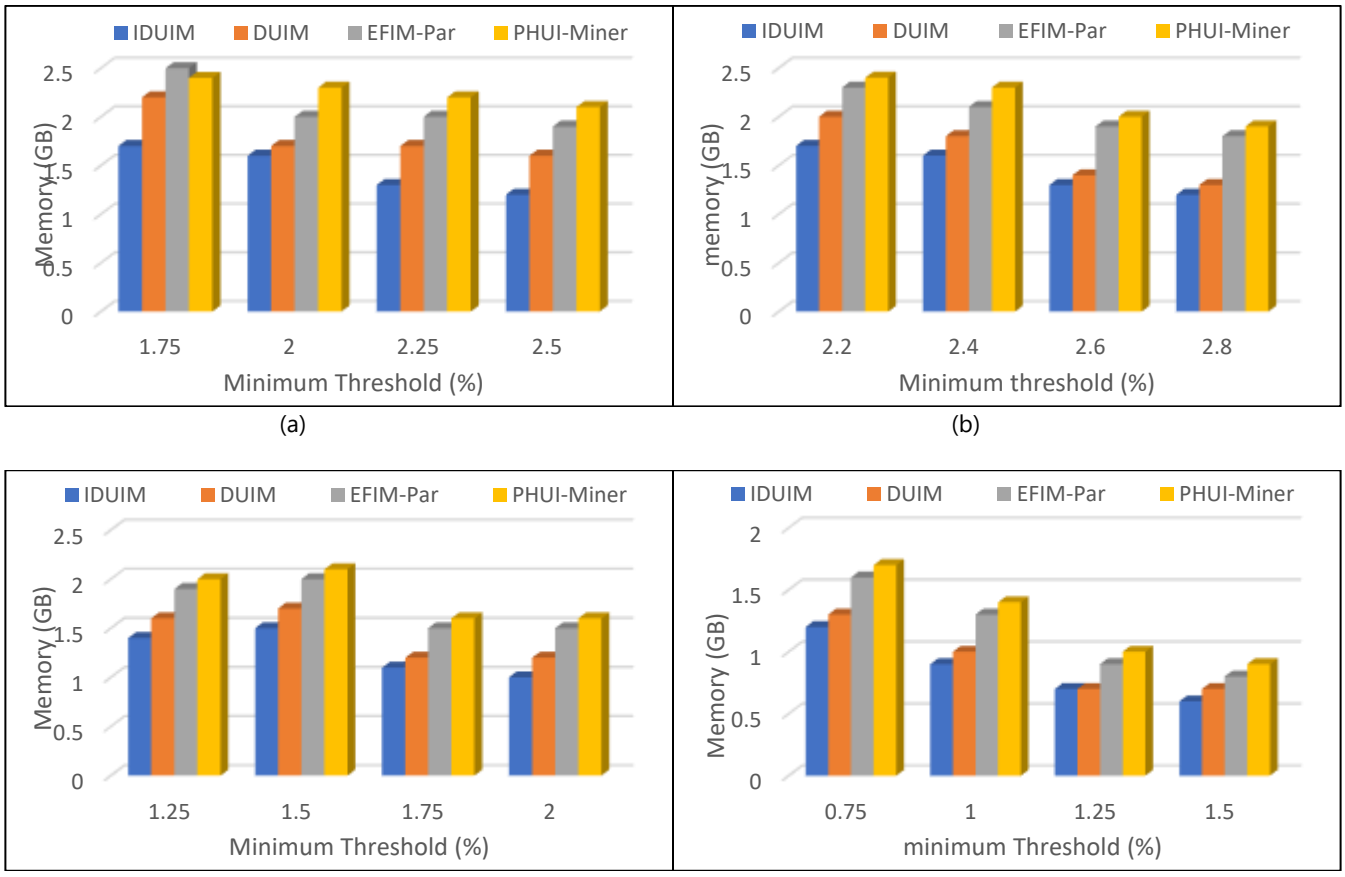


(c) (d)

Figure 3. Running time of the (a) Chainstore (b) Kosarak (c) Accident (d) Connect

5.2.2. Memory requirements

Calculations are made to determine the algorithm's memory usage based on the amount of primary memory assigned to it during execution. The memory needs of the IDUIM algorithms for various datasets are displayed in Fig. 4. The data in Fig.4 indicates that the proposed method's memory requirements are not significantly altered for the dense Connect dataset, as the subset of transactions is comparable to the original transactions, and there are few null transactions in the modified dataset. As a result, its memory requirements are relatively higher than those of the smaller, sparse Connect dataset. It can be inferred from the tests that were conducted that IDUIM's memory consumption is lower since it only saves a portion of the necessary transaction.



(a) (b)

(c) (d)

Figure 4. Memory consumption of the (a) Chainstore (b) Kosarak (c) Accident (d) Connect

5.2.3. Scalability

To assess the scalability of the IDUIM algorithms, the number of available nodes is adjusted using the Chainstore dataset for evaluation. The algorithms display nearly linear speedup due to their distributed nature, indicating effective scalability as the number of nodes increases. However, beyond a certain number of processors, communication and synchronization expenses become more significant than computing overhead, and the added processors have little effect on computation time. Hence, attaining a linear speedup is unrealistic as communication costs dominate the total running time. Additionally, scalability can be done by varying the size of the database, adjusting the number of transactions, and measuring the algorithm's runtime performance. Compared to other standard methods, IDUIM demonstrates shorter runtime performance.

6. Limitations of the Study

The limitations of this study include potential algorithm dependency, reliance on benchmark datasets, practical implementation challenges, optimal threshold determination issues, and computational overhead due to repeated executions. The summary of The Limitations of the Study Following :

1. Algorithm Dependency: The proposed IDUIM algorithm is compared to existing algorithms like DUIM, PHUI-Miner, and EFIM-Par. The limitations of these benchmark algorithms may influence the overall assessment of the proposed method. Variations in algorithm design, parameter settings, and dataset characteristics can impact the comparative results.
2. Dataset Specificity: The experimental evaluations are conducted using benchmark datasets, which may not fully represent the diversity and complexity of real-world big data scenarios. The performance of the IDUIM algorithm may vary when applied to different types of datasets, and its generalizability to various domains should be further investigated.
3. Practical Implementation Challenges: While the study demonstrates the effectiveness of IDUIM in terms of time and space complexity, the practical implementation of the proposed algorithm in real-world scenarios might encounter challenges such as hardware constraints, compatibility issues, and integration complexities with existing systems.
4. Optimal Threshold Determination: Setting an optimal minimum utility threshold is crucial for the success of the algorithm. The study acknowledges the challenge of determining the appropriate threshold, and the sensitivity of the algorithm to this parameter might lead to difficulties in obtaining meaningful results consistently across diverse datasets.
5. Repeated Execution for Threshold Tuning: The necessity of running the algorithm multiple times with different thresholds to achieve desired outcomes introduces an additional computational overhead. This iterative process may not be practical in situations where quick and accurate decision-making is essential.
6. Scalability Concerns: The study claims improved efficiency for mining item sets in large datasets, but the scalability of the IDUIM algorithm to extremely massive datasets or distributed computing environments is not explicitly addressed. The algorithm's performance under varying scales of data should be thoroughly examined.
7. Real-time Decision-making Assumption: The assertion that IDUIM provides insights for nearly real-time decision-making systems assumes a certain level of responsiveness that may not be achieved in all practical scenarios.

7. Conclusion

This work proposes enhancements to the DUIM algorithm for effectively extracting high utility item sets (HUIMs) by employing various optimization techniques. One such optimization strategy involves a more neutral search space division approach that considers both the subspace size and the transaction-weighted utility item sets (TWUI) of items. Additionally, storage efficiency is increased by preserving only specific subsets of transactions. In terms of time and space complexity, experimental assessments using benchmark datasets show that the IDUIM approach outperforms DUIM and other techniques. The method works well for organizing data for systems like recommendation engines and other data management systems and for deriving important insights from massive datasets. To further improve the robustness of the model, future research can explore pruning techniques that reduce the search space while maintaining model accuracy. Setting an optimal minimum utility threshold presents a challenge as the algorithm's output varies based on the chosen threshold. If the threshold is set too low, an excessive number of HUIMs may be generated, while setting it too high may result in no HUIMs being detected. Hence, running the algorithm multiple times with different thresholds may be necessary to obtain the desired outcomes. Future research should focus on dynamic threshold optimization, advanced pruning techniques, scalability studies, real-time decision-making validation, domain-specific evaluations, parallelization strategies, user interaction, benchmarking, visualization, and system integration for IDUIM algorithm enhancement.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
Acknowledgements: The authors would like to express their thanks to Southern Technical University (https://www.stu.edu.iq) Basrah, Iraq, for its support in the present study.

References

[1] Agrawal, R. & Srikant, R. (2003). Fast algorithms for mining association rules in large databases. In International Conference on Very Large Databases (487–499).

[2] Ahmed, C.F., Tanbeer, S.K., Jeong, B.-S., Lee, Y.K., (2009). An efficient candidate pruning technique for high utility pattern mining. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 749–756.

[3] Agrawal, R., Srikant, R., et al., (1994). Fast algorithms for mining association rules, in Proc. 20th int. conf. very large data bases, VLDB, 1215, Citeseer, 1994, 487–499.

[4] Al-Bana, M. R., Farhan, M. S., & Othman, N. A. (2022). An Efficient Spark-Based Hybrid Frequent Item sets Mining Algorithm for Big Data. Data, 7(1), 11. https://doi.org/10.3390/data7010011

[5] Al-Rabeeah, A., & Hashim, M. (2019). Social Network Privacy Models. *Cihan University-Erbil Scientific Journal,* 3(2), 92-101. https://doi.org/10.24086/cuesj.v3n2y2019.pp92-101

[6] Baek, Y., Yun, U., Kim, H., Kim, J., Vo, B., Truong, T., Deng, Z.-H., (2021) Approximate high utility item sets mining in noisy environments. Knowl.-Based Syst. 212, 106596.

[7] Chan, R., Yang, Q., & Shen, Y.-D. (2003). Mining high utility Item sets. In Proceedings of the Third International Conference on Data Mining (1–19). 10.1109/ICDM.2003.1250893.

[8] Chen, Y, and Aijun A. (2016) Approximate parallel high utility itemset mining. Big data research 6 (2016): 26-42.

[9] Dalal, S., & Dahiya, V. (2020). A novel technique - absolute high utility item sets mining (ahuim) algorithm for big data. *International Journal of Advanced Trends in Computer Science and Engineering, 9* (5), 7451–7460.

[10] Dalal, S. , & Dahiya, V. (2018). Review of high utility item sets mining algorithms for big data. *Journal of Advanced Research in Dynamical and Control Systems JARDCS, 10* (3), 274–283.

[11] Gan, W., Lin, J.C.-W., Chao, H.-C., and Zhan, J. (2017). Data mining in distributed environment: a survey, Wiley Interdisciplinary Reviews. Data Min. Knowl. Disc. 7, (6) e1216.

[12] Gan, W., Lin, J.C.-W., Fournier-Viger, P., Chao, H.-C., Hong, T.-P., and Fujita, H., (2018) A survey of incremental high-utility item sets mining, Wiley Interdisciplinary Reviews. Data Min. Knowl. Disc. 8, (2) e1242.

[13] Han, J., Pei, J. and Yin, Y., (2000) Mining frequent patterns without candidate generation. ACM sigmod record 29 (2), 1–12.

[14] Hassan, A. Y., Said, M., & Salem, S. M. S. (2022). Estimation of the transformer parameters from nameplate data using turbulent flow of water optimization technique. Indonesian J Electr Eng Comput Sci, 25(2), 639-647.

[15] Krishnamoorthy, S., (2015) Pruning strategies for mining high utility item sets. Expert Syst. Appl. 42 (5), 2371–2381.

[16] Kumar, S., & Mohbey, K. K. (2022). A review on big data based parallel and distributed approaches of pattern mining. *Journal of King Saud University-Computer and Information Sciences, 34*(5), 1639-1662.

[17] Lee, J., Yun, U., Lee, G., & Yoon, E. (2018). Efficient incremental high utility pattern mining based on pre-large concept. Engineering Applications of Artificial Intelligence, 72, 111-123. https://doi.org/10.1016/j.engappai.2018.03.020

[18] Li, H., & Sheu, P. C. Y. (2021). A scalable association rule mining heuristic for large datasets. *Journal of Big Data, 8* (86), 1–32. 10.1186/s40537-021-00473-3.

[19] Liu, Y., Liao, W., & Choudhary, A. (2005). A two phase algorithm for fast discov- ery of high utility item sets. Advances in Knowledge and Data Mining, 351, 689–695. 10.1007/11430919_79.

[20] Mustafa, R. A., Chyad, H. S., & Mutar, J. R. (2022). Enhancement in privacy preservation in cloud computing using apriori algorithm. *Indonesian Journal of Electrical Engineering and Computer Science, 26*(3), 1747-1757.

[21] Milhem, N., Abualigah, L., Nadimi-Shahraki, M. H., Jia, H., Ezugwu, A. E., & Hussien, A. G. (2022). Enhanced MapReduce performance for the distributed parallel computing: application of the big data. In Classification applications with deep learning and machine learning technologies (191-203). Cham: Springer International Publishing.

[22] Nguyen, T. D., Nguyen, L. T., Vo, B. (2019) A parallel algorithm for mining high utility item sets. In: Information Systems Architecture and Technology: Proceedings of 39th International Conference on Information Systems Architecture and Technology – ISAT 2018. Springer, Cham, 286–295.

[23] Rathee, S., Kashyap, A. (2018) Adaptive-Miner: an efficient distributed association rule mining algorithm on Spark. J Big Data 5, 6. https://doi.org/10.1186/s40537-018-0112-0

[24] Sethi, K. K., Ramesh, D., & Edla, D. R. (2018). P-FHM+: Parallel high utility item sets mining algorithm for big data processing. Procedia computer science, 132, 918-927.

[25] Sundari, P. S., & Subaji, M. (2022). An improved hidden behavioral pattern mining approach to enhance the performance of recommendation system in a big data environment. *Journal of King Saud University-Computer and Information Sciences, 34*(10), 8390-8400.

[26] Tamrakar, A. (2017) High utility itemsets identification in big data. PhD diss., University of Nevada, Las Vegas.

[27] Vo, B., Coenen, F., Le, B., (2013). A new method for mining frequent weighted item sets based on wit-trees. Expert Syst. Appl. 40 (4), 1256–1264.

[28] Wu, X., Zhu, X., Wu, G. Q., & Ding, W. (2013). Data mining with big data. IEEE transactions on knowledge and data engineering, 26(1), 97-107.

[29] Yao, H., Hamilton, H.J., Butz, C.J., (2004) A foundational approach to mining item sets utilities from databases. In: Proceedings of the 2004 SIAM International Conference on Data Mining. SIAM, 482–486.

[30] Yun, U., Lee, G., Yoon, E., (2017). Efficient high utility pattern mining for establishing manufacturing plans with sliding window control. IEEE Trans. Industr. Electron. 64 (9), 7239–7249.

[31] Zhang, F., Liu, M., Gui, F., Shen, W., Shami, A., & Ma, Y. (2015). A distributed frequent item sets mining algorithm using Spark for Big Data analytics. Cluster Computing, 18, 1493-1501.