A Fast Algorithm Based on Apriori Algorithms to Explore the Set of Repetitive Items of Large Transaction Data

Javad Ghofrani
Software Engineering Group
Leibniz Universität Hannover
Hanover, Germany
javad.ghofrani@inf.uni-hannover.de

Arezoo Bozorgmehr
DZHW
Hanover, Germany
bozorgmehr@dzhw.eu

Amir Panah
Hadaf Institute of Higher Education
Sari, Iran
amir.panah2020@gmail.com

ABSTRACT
Parallel data mining is utilized to improve the performance of analyzing large databases within a reasonable time frame. Exploring associative rules is an important task in data mining with various practical applications that can be used to explore knowledge in the form of a set of repetitive items or associative rules. Parallel algorithms divide the data superficially and then using different distribution approach, like data distribution, candidate and numerical candidate distribution, extract the set of repetitive items, and ultimately explore strong association laws.

In this paper, a parallel algorithm is proposed to explore the collection of repetitive items from big and dense transaction databases. In this algorithm, the numerical candidate distribution together with the using of two processes at each level of parallel processing increases the speed of extraction of data patterns. Two steps, extracting the repetitive items, which include generating a set of candidate counts, and computing a set of repetitive items, are performed simultaneously using both processes at each level.

The proposed algorithm by us is compared in terms of runtime and accelerated with basic Apriori algorithms and numerical distribution on four datasets. Experimental results prove that the proposed algorithm has better optimization in terms of maximizing the performance and has better run time than the two other algorithms for transactional data in different scales and dense data.

CCS Concepts
• Information systems→Association rules • Theory of computation→Massively parallel algorithms.

Keywords
Big Data; Data Mining; Apriori algorithms; Association rules.

1. INTRODUCTION
Rapid advances in communication technologies, storage and data processing have led to an increase in recording and storage of data and thus an increase in the volume of data banks. Researchers are facing with more and more observations in various fields such as engineering, economics, astronomy and biology every day. Compared to older and smaller databases, today's databases create new challenges in data analysis. Because manual processing of these big data is time-consuming, over time, the automatic processing of these data was felt by statistical and exploratory methods.

The process of automating the processing and obtaining useful data from the databases is called the Knowledge Discovery in Databases (KDD), which is an interactive, repetitive process [1]. The process of Knowledge Discovery in Databases has various steps including the selection of data, pre-processing, data mining and evaluation. Data mining [2] is a fundamental step in this process and uses intelligent methods [3, 4, 5] for exploring data including exploring associative rules, clustering, categorization, and regression to create models and explore unknown patterns [6].

Data mining [7] extracts useful knowledge from a large amount of data. As the size of the data increases, the search algorithms for conventional association rules cannot deliver the results at acceptable times. Even in a worse case, a single processor may not have enough memory to hold all data. In the past years, among the techniques of data mining, particular attention has been paid to the algorithms for the discovery of frequent patterns. In the meantime, the discovery algorithms of the repetitive items set are further developed, which ultimately leads to establish of associative rules.

To explore the collection of repetitive items of large transaction data, parallel processing of known algorithms of data mining is designed. As mentioned, exploring the associative rules is one of the most important methods in data mining that is used to explore over-repeated patterns of databases.

The issue of exploring the associative rules on transaction databases was first proposed by Agrawal et al [8], which was presented to explore the knowledge and patterns of purchasing a store's users. In Agrowal's proposal, the total basket of users was reviewed in a specific time period, and then the association rules were extracted from it.

The Apriori algorithm [9] is one of the first and most popular algorithm used to explore the collection of repetitive items, however, it has some deficiencies. To overcome these, improvements have been made using various techniques on it. One of these is the parallelization techniques which were first proposed by Han et al [10]. In a comprehensive study, they found that most of the exploratory algorithms for parallel associative rules were based on the parallelization of the Apriori algorithm

ICCDA 2018, March 23–25, 2018, DeKalb, IL, USA

ACM ISBN 978-1-4503-6359-4/18/03…$15.00
https://doi.org/10.1145/3193077.3193089

that repeatedly generated and tested the set of candidate counts from 1 to k until the set of repeating sets was found.

The Apriori algorithm has a higher computational complexity due to surface-levelness than other algorithms exploring the repetitive patterns of the transaction database. FP-growth algorithms [11] and Eclat [12] use the first-depth search from all candidate sets. However, on one hand, Parallel algorithms based on FP-growth and Eclat, due to the nature of the pattern search, increase only the number of processor cores in order to increase the number of exploring processes of repetitive items. On the other hand, Apriori parallel algorithms use the distributed memory calculation model, the transmission of the message as well as breaking the pattern exploration process into several sub-processes to explore associative rules. In the distributed computing model based on the Apriori algorithm, the data are split horizontally into several parts, and in each section one process is used, a set of candidates of each section is merged until, and the set of repetitive items are extracted.

The process breakdown model is similar. However, implementing the process in each section in a dense transaction dataset that has a set of long-length items has a high computational complexity. Also, one of the other problems of these algorithms is the idle time of processes at the time of exploring and the calculation of the support value (Eq (1)) of repetitive items from a set of candidate counts.

In this paper, a parallel algorithm (HP-Apriori) has been proposed by us, based on the Apriori algorithm that divides the dataset horizontally into two parts, and in each part two processes are considered, means, in each section, there are two processes and they use four processes in total. In the first repetition, each process generates a set of single items, and in replications subsequent the candidate sets are generated and from these sets, a set of repetitive items is extracted by other processes of each section. Our approach increases the runtime method to explore the collection of repetitive items in each section.

The rest of this paper is structured as following: A short overview on existing algorithms presented in section 2. Section 3 analyzes the existing approaches and related works. We describe our approach in Section 4. Section 5 presents and discusses the results of evaluating our approach. Finally, Section 6 closes with a conclusion and an outlook of future work.

2. ALGORITHMS

2.1 Association Rules Mining

Association rules mining [13, 14] is a research area in the field of data mining, and is aimed to discover interesting and important links between frequently retrieved items or information patterns in large databases and transaction caches, which has recently attracted a lot of research efforts. Various algorithms have been developed to explore association rules that include Level wise algorithms and Pattern-Growth algorithms [15]. The Apriori algorithm is one of the most widely used algorithm for exploring repetitive patterns, which is an algorithm dependent on a two-stage level. In the process of exploring association rules, there are two criteria of support and confidence for the discovery of repetitive patterns, so all the search association algorithms operate within the Support-Confidence Framework [14].

In the transaction dataset, support of rules A and B represents the extent of the occurrence of these rules in the transaction databases. In other words, support represents the percentage of transactions that include A and B in the database. Also, the occurrence of rules A, B indicates that if A occurs, there is also a certain percentage of occurrence of B.

2.2 Multilevel Association Rules

In many cases, the association rules between data items occur at relatively high conceptual levels [16]. The strategy is generally used up and down, and the number of repetitions for the set of important items begins at each conceptual level, also it starts at the highest level [17]. Then work at lower levels and continues the same way. One important algorithm of this strategy is the Apriori algorithm [18]. This is a superficial search algorithm, which ends with the end of exploration of the level (k) to the next step, k + 1. This action is repeated until the condition or final condition is fulfilled. Hens, this strategy reduces the search space in the production of repeated items.

2.3 Depth-First Search Algorithms

Depth-first search algorithms [19] usually perform better than first-order algorithms, and they are more effective, especially when dealing with long patterns or the small frequency thresholds. Initially, the database is converted into a tree called FP-tree. After that, it directly deals with the extraction of frequent patterns from this tree. Among these algorithms, FP-Growth [20, 21] is one of the fastest and most popular methods.

2.4 Parallel Algorithms

Parallel algorithms [22, 23] in computer science, in contrast to traditional sequential algorithms, are algorithms in which every process in each part of the program is executed on different processors, and at the end the results will be put together for the desired result. In parallel processing [24], many operations are run simultaneously, unlike the serial processing, which computational steps are sequentially executed [25]. Repetitive numerical methods, such as the Newton method or the three-dimensional problem, are examples of Parallel algorithms [26].

3. RELATED WORK

Agrawal et al. [13] introduced question of exploring community rules on transaction databases, which were presented for discovering the knowledge and patterns of purchasing a store's users. In Agrowal's proposal, the total basket of users was reviewed in a specific time period, and then the association rules were extracted from it. In terms of efficiency, among D-CLUB search algorithms, the D-CLUB has the best results, and then FP-growth and Eclat and the Apriori algorithm have the worst results. They also have the same performance in parallel mode. However, each of them has advantages and disadvantages. A number of algorithms are designed to explore the collection of repetitive fractures from massive transaction databases, which are based on the Apriori algorithm.

Park et al. [14] have proposed surface-dependent algorithms that are one of the algorithms for discovery of associative rules including the model growth methods provided by Chang et al. [15]. The park attempted to use hacking techniques to improve the efficiency of the Apriori algorithm.

Zackie et al. [27] have proven that the distribution of the count is a suitable parallel of the Apriori algorithm and divides the data into horizontal components. Each processor generates a set of local K-candidate from a local database. At the end of each occurrence, local values are aggregated among all processes in general values, so that a set of repeatable items can be found. All of these efforts have led to the provision of methods for removing

the Apriori algorithm from the field of exploration, and pushing it towards parallel or distributed exploration.

4. OUR WORK

In this paper we suggested a parallel algorithm which uses shared memory. In that architecture, processors and disks have access to a shared memory, which is typically done through an internal network connection. Our new algorithm is based on Apriori algorithm that explores the set of repetitive items in parallel with large transaction data. Our algorithm divides the dataset horizontally in two parts and then divides each horizontal set into two sub-sets. For each subset data, the proposed algorithm uses a processor, so for each horizontal section there are two processors that run together, a processor generates a set of K-candidate from the set of items in dataset, and the next processor extracts the set of (K-1) repetitive items. The set of single items in the list consists of a set of single items to a set of (k-1) items and a generic transaction index. This index is generated in the first iteration, where the set of single items is produced by the processors, and in the subsequent replications of the list Updated.

To evaluate the proposed algorithm, three dense and dense standard databases have been used, and using other criteria such as the degree of parallelization, the efficiency of the algorithm compared with the two previous algorithms has been considered more. The simulation results show that the proposed algorithm has an average extraction speed of 4 times the Apriori algorithm and is about 2.4 times better than the numerical distribution algorithm.

Figure 1 depicts the communication model between processes in

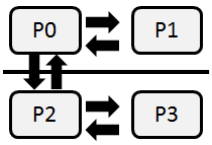

Figure 1. The communication model of processes in the HP-Apriori algorithm.

the proposed parallelized algorithm schematically, in which the processes of each section are interconnected, and the P0 and P2 processes also use the results of each other. In each repetition, the P1 and P3 processes generate a set of candidate counts (K + 1), the P2 process integrates and arranges the candidate set in the two P1 and P3 processes, and also The P0 process calculates the set of K-points, and extracts sets of repetitive items that have support beyond the minimum threshold. Table 1 and Table 2 illustrate the pseudo-code of the P0, P2, respectively. Table 3 shows the pseudo-code of P1 and P3 processes.

Table 1. Pseudo-code processes P0

Procedure_1 HP-Apriori (Dataset, Minsup)P0

C_1= {{i}|i ∈ I};
While ($c_1 \neq \emptyset$)
{
For each transaction T∈ partition do
For each candidate itemset $c_1 \neq C_1$ do
If($c_1 \subseteq$ T) then
$S(c_1)= S(c_1) +1$;
}
While (K>1)
{
Combine $S(c_K)$ from P0 and P2;
Calculate global count;
If (sup(c_K)>= α_{min})
S(F)= S(F)+ c_K;
}

In the first step, each process is applied to a section of the dataset; the number of records in the first part and samples in the second part is equal to N/2. Here N is the number of transaction databases. Also, the number of items in the sections related to the processes P0, P2, P1 and P3 is equal to L/2, where L is the largest transaction in the data set. In addition to the two symbols N and L, M denotes the number of items in the data set, K is the largest length of the repetitive items set, and h is the number of rows in the proposed algorithm, which is h = 2 in this algorithm.

Table 2. Pseudo-code processes P2

Procedure_1 HP-Apriori (Dataset, Minsup)P2

C_1= {{i}|i ∈ I};
While ($c_1 \neq \emptyset$)
{
For each transaction T∈ partition do
For each candidate itemset $c_1 \neq C_1$ do
If($c_1 \subseteq$ T) then
$S(c_1)= S(c_1) +1$;
}
While (K>1)
{
Sort and update $S(c_K)$;
Calculate Supp (C_K);
}

Table 3. Pseudo-code processes P1 and P3

Procedure_2 HP-Apriori (Dataset, Minsup)P1, P3

C_1= {{i}|i ∈ I};
While ($C_1 \neq \emptyset$)
{
For each transaction T∈ partition do
For each candidate itemset $c_1 \neq C_1$ do
If($c_1 \subseteq$ T) then
$S(c_1)= S(c_1) +1$;
}
K=2;
While ($C_K \neq \emptyset$)
{
For each C_K in index list;
K++;
}

The implementation of processes occurs at different times T0, T1, T2. At T0 time, all processes generate two lists; a list of first items set and an index list.

In the index list, for each item, a row is created that contains all the transactions in which the item occurred. At T1 time, the created lists in each row are combined, to be used to generate a set of next candidates. Combination all sets of first items and production a set of K candidates occur at T2 time. This technique increases the speed of database scanning, so that to calculate support value for a collection of items instead of scanning entire databases, only a few lines in the index list are scanned for the length of the set of the desired item.

5. EVALUATION

In this section, the efficiency of the HP-Apriori algorithm is evaluated and compared to the past works. Two criteria running time and speed are used to demonstrate the efficiency of the proposed algorithm. Some terms are used to evaluate the results such as:

5.1 Association Rules Support

Association Rules Support based on Equation Eq (1) indicates that, how much percentage of transactions of databases can find the X and Y sets together and its value expressed in percentage. It should be noted that minsup is considered the least support value.

Eq (1): $Support(X \rightarrow Y) = P(Y \cap Y)$

5.2 Set of Repetitive Items

Set of repetitive items are often found in the transactions and if they are also larger than the minsup threshold, they are called repetitive items. Eq (2).

Eq (2): $Confidence\ (X \rightarrow Y) = P(P \cap Y) = P(X \cap Y)/P(X)$

This evaluation was performed in OS 7 with one 8-core microprocessor and 6 GB of original memory. It also implements all algorithms using the .net and C\#. In this study, three sets of data were used to compare the Apriori algorithms and numerical distributions (CDs), which are a dense dataset and two spars datasets. These datasets are selected from the Irvine Machine Learning Database [11].

5.3 Measurement Results

The HP-Apriori algorithm has been evaluated using time and speed criteria with Apriori and CD algorithms and the results of this comparison are clearly illustrated. Figure 2 depicts the required runtime of the implementation of proposed algorithm in comparison with the other two methods on the Retail, Accidents and Kosarak datasets, in which the proposed algorithm extracts the set of repetitive items in less time. With increasing support value in Retail dataset, because the database is spars and the number of sets of repetitive items is very small for high support value, the difference in efficiency between the three algorithms is reduced. For example, for support value of 20%, the speed of the proposed algorithm is 3.7 times, but for the 60% it is 2.3 times.

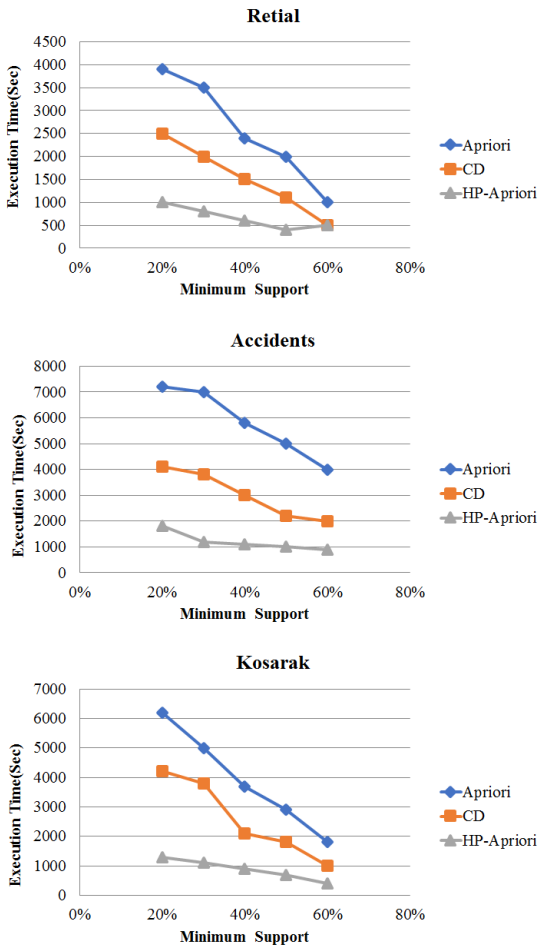

Figure 2. Runtime on the retail, accidents and kosarak datasets.

The Accidents dataset is a dense dataset that extracts the process of exploring the set of repetitive items with high execution time. Unlike the Retail dataset greatly reduces the number of sets of repetitive items by increasing the minimum support value, due to the congestion of the Accidents dataset, the difference between the implementation times of algorithms in various support value to the size of the dataset is not large. The Kosarak dataset is spars and even has an average length less than the Retail dataset, but still has a high number of transactions. However, the efficiency of the proposed algorithm is around 0.6 for the Apriori algorithm compared to the Retail dataset. It can be concluded that the effect of increasing the number of transactions on the efficiency of the algorithm is less than the effect of database densification; although, it is about 0.8 for Accidents dataset.

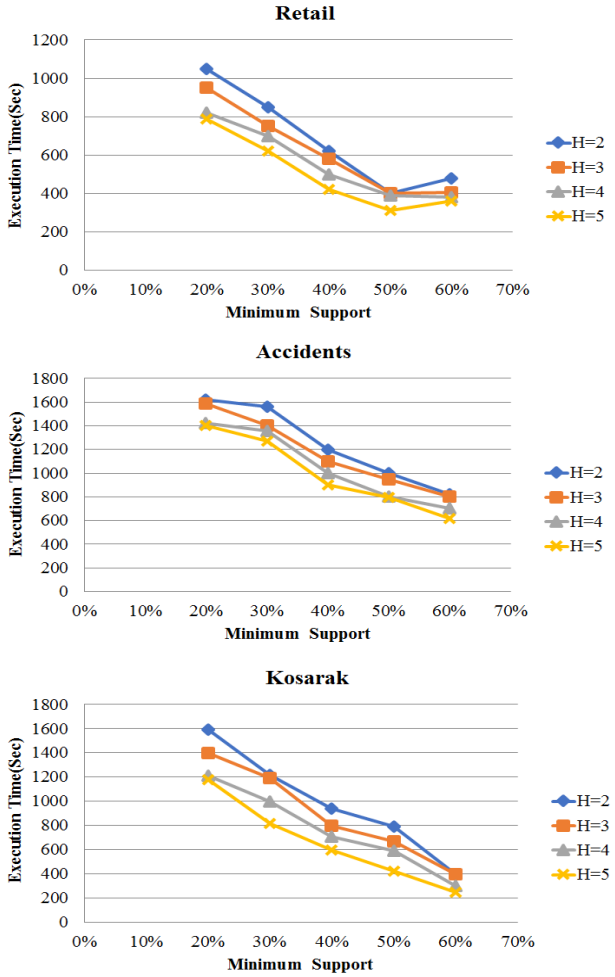

Figure 3. Runtime on three datasets based on different levels.

In another test, the proposed algorithm is based on different categorization, in which the databases were divided into several sections including 2, 3, 4, and 5, and in each section two processes

were used and the implementation time of the experiment was in different forms. Figure 3 illustrates the implementation time of the HP-Apriori algorithm in terms of different levels on the three datasets. With increasing number of levels, the speed of the algorithm does not increase significantly compared to the previous level. This issue is also apparent by results of Accidents and Kosarak datasets. The combination of local counting at different levels is time-consuming and causes waiting of the process at various levels during the overall counting process, which causes the unemployment of these processes. As a result, the degree of parallelism of the algorithm is reduced, because in the parallel strategy, the processes must always be processed.

The Accidents dataset is a dense database, so the number of candidate sets generated in each level is high, and as a result, the combination and total count of these set of items are longer than the spars datasets. As you can see, the speed of the algorithm is not different at different levels and is close to each other. The

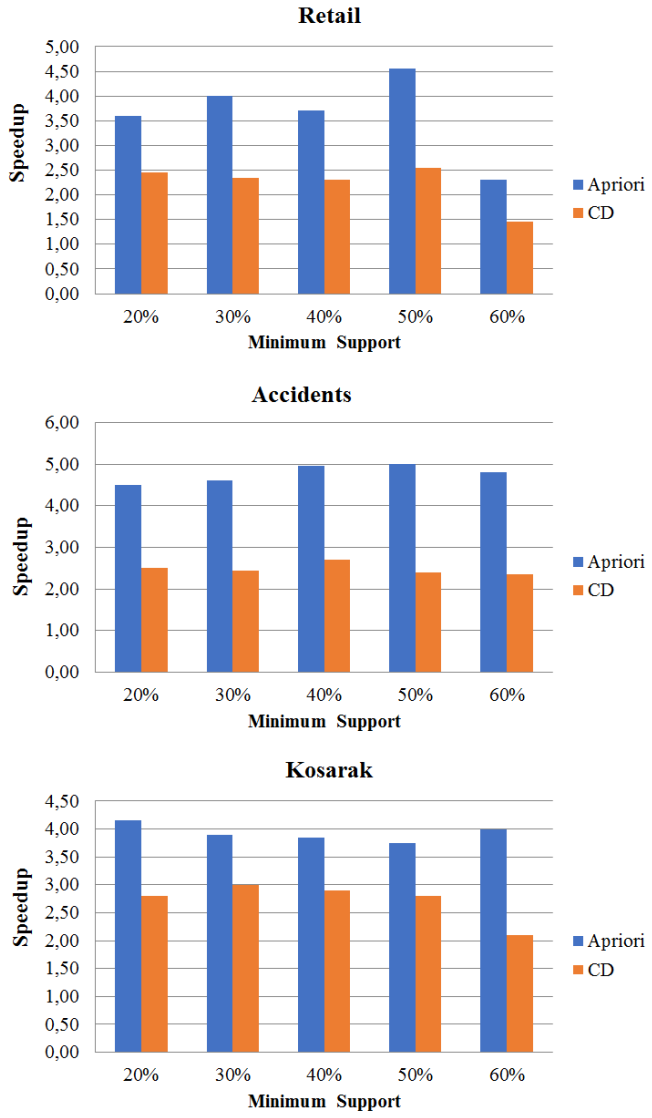

Figure 4. HP-Apriori speeds over other algorithms on three datasets.

algorithm's speed difference for the Kosarak dataset is due to the fact that it is more durable than the other two databases. The number of sets produced in each level is low and, as a result, the combination of these patterns requires less time and the degree of parallelism of the algorithm is greater.

Figure 4 illustrates the speed of the HP-Apriori algorithm compared to the Apriori and CD algorithms on the three datasets. The results of Retail dataset point to the fact that the HP-Apriori algorithm extracts considerably faster than the other two algorithms of the set of repetitive items. For the minimum support value of 50%, the proposed algorithm is as high as 4.7 and 2.8 for the Apriori and CD algorithms, respectively. The average speed of the HP-Apriori algorithm is 3.5 times as much as Apriori, and about 2.1 times the CD. In all cases, the HP-Apriori algorithm increases the speed of extraction of repetitive items set from the Accidents dataset. The proposed algorithm has the most improvement compared to the Apriori algorithm for at least 50% support value, and the extraction speed is 4.9 times, and for the support value of 20%, the least improvement is about 4.4 times. On the other hand, the HP-Apriori algorithm has the lowest rate of improvement (2.3x) compared to the CD for support value of 40% the highest improvement in speed (2.6 times) and for the support

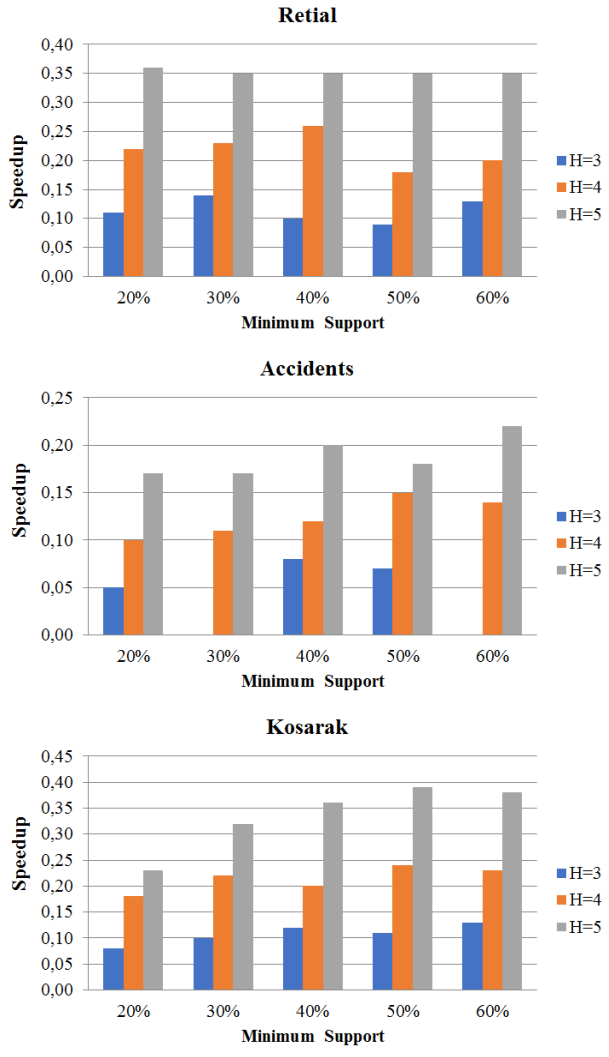

Figure 5. HP-Apriori speeds in different levels on the three datasets.

value of 60%. The average of the HP-Apriori algorithm, the speed of pattern extraction on the Accidents dataset, is 4.6 times the Apriori and about 2.4 times the CD.

The results of Kosarak dataset show that the HP-Apriori algorithm support values the Apriori and CD algorithms with the slightest improvement in speeds of 3.8 and 2.3, in order to support value 50% of the algorithm. Also, Apriori and CD algorithms have the best improvement in speed for support value of 20% (4.2 times and 2.8 times). The average proposed algorithm for extracting repetitive items set is about 3.8 times as much as Apriori and equals 2.5 times the CD. On the whole, the average proposed algorithm is 4 times the extraction speed compared to Apriori and about 2.3 times the CD.

Figure 5 illustrates the speed of the proposed algorithm, based on the various levels tested on the three datasets indicating the ratio of the proposed two-level algorithm. The proposed algorithm in Retail dataset speeds up for three levels of about 0.10, for four levels of 0.21, and also for a mode of 5 levels, about 0.36 compared to the two-level mode. As the number of levels increases, the unprocessed periods of processes increase, which results in a reduction in the degree of parallelism of the algorithm.

Increasing the number of levels in the proposed algorithm for the Accidents dataset has less effect on the speed of the algorithm than the Retail dataset. This is due to the density of the databases that reduces the speed of the algorithm. For H = 3, the speed of the algorithm is improved by about 0.7 compared to H = 2, while for H = 4 and H = 5 it is about 0.11 and 0.18, respectively.

Figure 6 depicts the average inaction rate for the three datasets, in which the lowest average unemployment rate is recorded for H = 2.

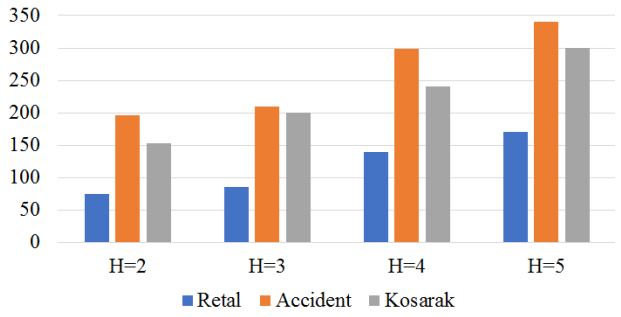

Figure 6. Average inaction processes for different levels on the three datasets.

In another test, the Parallelizable Rate (PR) algorithm has been investigated on a dataset using different levels. To calculate this rate, a relationship is defined in which the expectation value is obtained for each process, and then the average of waiting time for each level is calculated and, therefore, the PR is calculated based on the equation Eq (3).

$$\text{Eq (3)} \quad PR_H = \frac{\frac{W_{P_1}+W_{P_2}+W_{P_3}+W_{P_4}}{4}}{\text{Execution Time}}$$

Table 4 depicts the PR of the proposed algorithm with MST = 20% on a different dataset. As it can see, the Parallelizable Rate algorithm for two levels is higher than the PR-algorithm with higher levels, due to the fact that based on the proposed algorithm that one process generates a set of candidates at each level and another calculates support value of the generated candidates, P2 process combines the set of generated components, and also the process P0 extracts the set of repetitive items from the set of combinations. At this time of generating and extracting, other level processes that have the task of calculating support value for the candidate set are idle, thus it causes to reduce the Parallelizable Rate.

Table 4. The parallelism rate of the HP-Apriori Algorithm

	H=2	H=3	H=4	H=5
Retail	91.10	86.98	88.10	85.11
Accidents	92.75	91.90	92.00	89.99
Kosarak	94.87	95.1	93.87	94.02

As the number of levels increases, the number of generated lists for the candidate set increases, and as a result, the time of combining these lists increases. As the number of levels increases, the set of candidate decreases and, as a result, the time of list combination increases. Consequently, before the combination the K-candidate completes, the list (k+1)-candidates is generated and is waiting for the completion of the combination. This algorithm is optimal for up to four levels and has lower performance for higher levels.

6. CONCLUSIONS

Exploring the repetitive patterns in the form of a set of repetitive items or association rules from the transaction database is one of the main methods in data mining. By increasing the amount of transaction data, the speed of extraction of these patterns decreases; therefore, the common algorithms for these data are low performance. Distributed or parallel algorithms for these datasets can be useful. Among the exploring patterns algorithms, the Apriori-based algorithms due to surface-level dependencies have lower speed than FP-Growth-based algorithms. In this paper, initially, by examining the proposed algorithms and describing each of them, it was shown that the most important issue in massive transaction databases is the slowdown in the extraction of data patterns from these datasets. FP-Growth-based algorithms have tried to improve this speed by increasing the number of processor cores. Also, Apriori-based algorithms were provided by breaking the process or distributing data in this field.

In the following, an algorithm called HP-Apriori was proposed by us to increase the speed of extraction of repetitive items. HP-Apriori algorithm divides data into different levels using the process breakdown approach, and uses two processes at each level to increase speed, especially in dense transaction databases that have long transactions. In each step, the algorithm simultaneously extracts K-candidates set at each level and another process generates the set of (k + 1)-candidate.

The implementation time of the proposed algorithm was compared to the Apriori and CD algorithms and was compared to the Retail, Accidents, and Kosarak datasets. On average, the proposed algorithm improved the extraction speed by about 4 times compared to Apriori algorithm, and about by 2.6 times to CDs algorithm, respectively.

In addition, we are going to investigate more improvement work in this field. Increasing the number of processor cores in addition to breaking the processes or distributing the information horizontally in different systems and using two processes in each system can greatly increase the efficiency that is suggested for future work.

7. REFERENCES

[1] Piatetsky-Shapiro, G. (1994). An overview of knowledge discovery in databases: Recent progress and challenges.

In *Rough Sets, Fuzzy Sets and Knowledge Discovery* (pp. 1-10). Springer London.

[2] Maimon, O., & Rokach, L. (Eds.). (2005). *Data mining and knowledge discovery handbook* (Vol. 2). New York: Springer.

[3] Singh, Y., & Chauhan, A. S. (2009). NEURAL NETWORKS IN DATA MINING. *Journal of Theoretical & Applied Information Technology*, *5*(1).

[4] Linoff, G. S., & Berry, M. J. (2011). *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons.

[5] Kumar, K., Chauhan, B. K., & Pandey, J. P. Data Mining and Its Applications: A.

[6] Mariscal, G., Marban, O., & Fernandez, C. (2010). A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review*, *25*(2), 137-166.

[7] Polkowski, L. (Ed.). (2013). *Rough sets in knowledge discovery 2: applications, case studies and software systems* (Vol. 19). Physica.

[8] Agrawal, R., Imieliński, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. In *Acm sigmod record* (Vol. 22, No. 2, pp. 207-216). ACM.

[9] Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB* (Vol. 1215, pp. 487-499).

[10] Han, E. H., Karypis, G., & Kumar, V. (2000). Scalable parallel data mining for association rules. *IEEE Transactions on Knowledge and Data Engineering*, *12*(3), 337-352.

[11] Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, *8*(1), 53-87.

[12] Goethals, B. (2004, March). Memory issues in frequent itemset mining. In *Proceedings of the 2004 ACM symposium on Applied computing* (pp. 530-534). ACM.

[13] Park, J. S., Chen, M. S., & Yu, P. S. (1995). *An effective hash-based algorithm for mining association rules* (Vol. 24, No. 2, pp. 175-186). ACM.

[14] Agarwal, R. C., Aggarwal, C. C., & Prasad, V. V. V. (2001). A tree projection algorithm for generation of frequent item sets. *Journal of parallel and Distributed Computing*, *61*(3), 350-371.

[15] Li, Y. C., & Chang, C. C. (2004). A new FP-tree algorithm for mining frequent itemsets. In *Content Computing* (pp. 266-277). Springer, Berlin, Heidelberg.

[16] Han, J., & Fu, Y. (1995, September). Discovery of multiple-level association rules from large databases. In *VLDB* (Vol. 95, pp. 420-431).

[17] Harrington, P. (2012). *Machine learning in action* (Vol. 5). Greenwich, CT: Manning.

[18] Brown, M. S. (2014). *Data mining for dummies*. John Wiley & Sons.

[19] Tr émaux, C. P. (2010). École Polytechnique of Paris (X: 1876). In *French engineer of the telegraph in Public conference, December* (Vol. 2, pp. 0980-603).

[20] Johnson, J., Lodhi, M. K., Cheema, U., Stifter, J., Dunn-Lopez, K., Yao, Y., ... & Wilkie, D. J. (2017). Outcomes for end-of-life patients with anticipatory grieving: Insights from practice with standardized nursing terminologies within an interoperable Internet-based electronic health record. *Journal of Hospice & Palliative Nursing*, *19*(3), 223-231.

[21] Adamo, J. M. (2012). Data mining for association rules and sequential patterns: sequential and parallel algorithms. Springer Science & Business Media.

[22] Zaki, M. J. (1999). Parallel and distributed association mining: A survey. *IEEE concurrency*, *7*(4), 14-25.

[23] Agrawal, R., & Shafer, J. C. (1996). Parallel mining of association rules. *IEEE Transactions on knowledge and Data Engineering*, *8*(6), 962-969.

[24] Navarro, C. A., Hitschfeld-Kahler, N., & Mateu, L. (2014). A survey on parallel computing and its applications in data-parallel problems using GPU architectures. *Communications in Computational Physics*, *15*(2), 285-329.

[25] Garg, R., & Mishra, P. K. (2009, March). Some observations of sequential, parallel and distributed association rule mining algorithms. In *Computer and Automation Engineering, 2009. ICCAE'09. International Conference on* (pp. 336-342). IEEE.

[26] Khaldi, D., Jouvelot, P., Ancourt, C., & Irigoin, F. (2012, September). Task parallelism and data distribution: An overview of explicit parallel programming languages. In *International Workshop on Languages and Compilers for Parallel Computing*(pp. 174-189). Springer, Berlin, Heidelberg.

[27] Zaki, M. J., Parthasarathy, S., Ogihara, M., Li, W., Stolorz, P., & Musick, R. (1997). Parallel algorithms for discovery of association rules. In *Scalable High Performance Computing for Knowledge Discovery and Data Mining* (pp. 5-35). Springer, Boston, MA.