International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST229618

# MapReduce Framework to Improve the Efficiency of Large Scale Item Sets in IoT Using Parallel Mining of Representative Patterns in Big Data

**A Geetha[1], Ravindra Changala[2], Goda Gangaram[3], Dr. Mahesh Kotha[4]**

[1]Assistant Professor, CSIT Department, CVR College of Engineering, Hyderabad, Telangana, India
[2]Assistant Professor, IT Department, Guru Nanak Institutions Technical Campus, Hyderabad, Telangana, India
[3]Assistant Professor, Department of CSE (AI&ML), CMR Technical Campus, Hyderabad, Telangana, India
[4]Assistant Professor, Department of CSE (AI&ML), CMR Technical Campus, Hyderabad, Telangana, India

**Article Info**
Volume 9, Issue 6
Page Number : 151-161

**Publication Issue**
November-December-2022

**Article History**
Accepted : 05 Nov 2022
Published : 18 Nov 2022

**ABSTRACT**

With the coming of the time of huge information, individuals can gather rich and various information from a wide assortment of assortment gadgets, like those of the Internet of Things. Information concealed in enormous information is extremely helpful and important. Frequent pattern mining, as a basic method of data mining, is applied to every aspect of society. However, the application of traditional frequent pattern mining methods to big data involves bottlenecks due to the large number of result sets. Such bottlenecks make it challenging to deliver commonsense worth underway and life. We proposed a new approach which involved representative patterns using mining technique. This framework can make the runtime difficult to evaluate in large data environments. Our approach gives best solution for the above gaps with the help of online representative pattern-set parallel-mining algorithm. We used MapReduce framework which performs horizontal segmentation over data bases. Finally, several performance optimization strategies are proposed. As shown by numerous experiments on the actual dataset, the algorithm proposed in this paper improves the time efficiency by one order of magnitude. Several optimization strategies reduce the execution time to varying degrees.

**Keywords :** Big data, IoT data analysis, MapReduce, online parallel mining, representative pattern.

## I. INTRODUCTION

With the development of various portable devices, the Internet of Things, cloud computing, cloud storage and other technologies, all tracking data of objects can be uploaded to the cloud, and users can collect rich and diverse data from a wide variety of collection devices and mine hidden yet useful and valuable knowledge from big data. The pattern-mining process is shown in Fig. 1. Data sources have become increasingly diverse, and applications cover all aspects of life. In the medical field, sensor devices can be used to implement remote health monitoring.

The sensor sends the monitored patient's status, such as heartbeat, body temperature, blood pressure, and respiration rate, to the cloud's application in real time. The objective is to analyse hidden trends in patients' Internet of Things data, try to identify patterns that could cause complications, and send timely alerts to patients and physicians. In the retail domain, shopping can be facilitated
by using the Internet of Things and big data analysis technology. For instance, Amazon.com, a well-known retailer, has launched Amazon Go. Shopping uses the Internet of Things and big data analysis techniques: customers shop in stores, and the Internet of Things implements tracking goods on shelves or in shopping carts. When a customer leaves the store, the system charges the customer. In the banking sector, the Internet of Things technology is used to facilitate payments. Bank of America, for instance, is working with FitPay to promote the wearable payment technology. Through this collaboration, cardholders will be able to pay directly from their smart watches and other wearable devices. Banks will be able to identify customer behavior and preferences. The above examples illustrate that the big data collected by the Internet of Things exist in all aspects of life. These data are stored in the cloud; mining and analyzing these data efficiently to extract useful and valuable information hidden in the large datasets will become a major research hotspot of the current ``information-based'' big data era. Frequent pattern mining, as a major data-mining technology, has been widely applied in many fields [1], such as analysis of real-world supermarket shopping baskets, the associated program analysis of documents or Internet pages, plagiarism analysis of documents, biomarker analysis (of the relationship between a disease and a person's bio-physiological information), etc. Many algorithms exist for solving the frequent pattern mining problem [2].

Data compression is one of the most important methods in data mining. Unlike other methods, pattern compression can be achieved by clustering, which is an automatic process of aggregating certain patterns with the same characteristics into a group [4]. Assuming that objects are frequent patterns, they are compressed by means of - clustering. By selecting a representative pattern in each cluster, a representative pattern set that can represent all frequent patterns is obtained. At the present stage, there are many algorithms for this kind of data compression. However, most algorithms compress the frequent patterns that have been mined, and there are few related algorithms for data compression performed while mining [5].

The main contributions of this paper are as follows:

This paper presents an online representative pattern-set mining algorithm. The algorithm enumerates all possible frequent patterns through the process of depth first search and selects the representative pattern set to cover all frequent patterns in the enumeration process, making the representative pattern set as small as possible to achieve the compression effect of frequent patterns. The algorithm can be applied to data mining in the field of the Internet of Things.

In this paper, the MapReduce framework is used to parallelize the mining algorithm of representative pattern sets, and PRP (parallel mining of representative patterns) is proposed to improve the efficiency of the mining algorithm.

Several performance optimization strategies are proposed. Due to the possible load imbalance among nodes, the load balance at each node can be achieved by static estimation of the load volume of each cloud node.

## II. RELATED WORK

In recent years, many data-compression algorithms, such as generator frequent closed pattern, maximal pattern and redundancy-aware top-k pattern have been described. These algorithms have been proposed to reduce the size of pattern sets. Among these algorithms, frequent closed patterns are not only small in number but also lossless. All frequent patterns and their support can be obtained by all subsets of the exhaustive closed pattern. However, compared to the maximum mode, the number of closed modes is greater. The maximum pattern has an information loss, and although all patterns can be recovered from the maximum pattern, support information of such patterns is missing. In some applications, it may not be necessary to obtain the specific support of frequent patterns; if so, the maximum pattern will be preferred. Certain approaches exist presently to obtaining compression patterns by balancing the accuracy of the pattern set size with the pattern support. The pattern sets obtained by these methods are usually relatively small and representative.

No corresponding improved algorithm to solve this problem exists in the context of research and implementation of the online representative pattern-set parallel-mining algorithm on large-scale data. However, several algorithms for parallel mining frequent item sets have been proposed. Agrawal and Shafer [20] put forward a method of count distribution to parallelize single-machine algorithms. In the researchers' algorithm, the list of transaction datasets is distributed to different nodes. Calculating the support of candidate sets on each node is used to establish a local hash tree for each node to replace the global hash tree. The one-phase algorithm [2] is also a parallel algorithm that requires only one MapReduce task to mine complete frequent item sets. All possible subsets of transactions are generated in the map phase, and the global support is counted in the reduce phase. The frequent item sets are obtained by comparing to the minimum support min_sup. This paper presents an online representative pattern-set parallel-mining algorithm based on the high efficiency of the parallel framework.

## III. ALGORITHMICAL FRAMEWORK

In this section, the algorithm for online parallel mining of representative patterns (PRP) is described in detail. First, the complete framework of the algorithm based on Hadoop is introduced. Accordingly, the overall framework will be introduced according to the flow of the MapReduce computing framework, mainly including two MapReduce processes. Second, the new algorithm called PRP is introduced in detail. Finally, the parallelization algorithm considering load balancing and the implementation of the reduction strategy are brief introduced.

The whole framework is divided into three parts: obtaining statistics of frequent 1-itemsets, grouping of frequent 1-itemsets considering load balancing, and online parallel mining of the representative pattern set. These three parts will be handled using two MapReduce frameworks.

PARALLEL STATISTICAL COUNTING

In this part, the frequent 1-itemset is counted by a MapReduce process. The first step is partitioning: for a given transaction database, the database is divided into contiguous parts, and each part is stored on P different computers. Accordingly, each computer stores a part of the database data. Each part is called a slice. The next step is to count the frequent 1-itemset [2]: this part uses a MapReduce procedure that produces the frequency of each item that appears in the database. Each mapper process processes a slice of the database; this step can determine the vocabulary I of the item, which is unknowable for a large database; the frequent 1-itemsets produced by the results of this step are stored in an F-list.

FREQUENT ITEMSET GROUPING

This part is completed by a computer. In this part, the possible depth of each frequent 1-itemset is estimated by a sampling method; the frequent itemsets are grouped by considering load balancing, and the results of this section are applied to the next part that handles the partitioning aspect of representative pattern set mining. The $|I|$frequent 1-items in the *F*-list generated in the previous step are grouped, and the quantity $Q$ to be grouped is determined according to the number of computers in the reduce process of the mining part of the pattern set. The grouped table is called a G-list; each group is given a unique group number gid. Both the F-list and the *G-list* are very small, and the time complexity is consistent with $O(|I|)$ ; hence, this step can be done on a computer in a few seconds. The data generated by the result of this step are stored in a *G-list.*

PDRP is an algorithm for the online mining of representative patterns. This algorithm adopts the depth-first method while mining frequent patterns, uses the longest pattern to cover frequent patterns without knowing the complete coverage information, and generates the representative pattern set online. The depth-first search process scans each pattern twice: first, to visit the pattern while descending and, second, to visit the pattern after visiting its children. It can be seen that after the second visit to a pattern, all patterns that may cover it have already been enumerated, and no other patterns covering the pattern can be observed among the patterns enumerated later. A pattern is output the second time it is visited. Otherwise, a new data structure called RP-tree is used in this paper. The structure of RP-tree is similar to that of FP-tree. In an FP-tree, each node modifies the state of the node by aggregating the support of all new itemsets. However, an RP-tree modifies the state of the node by calculating the maximum support of the new itemsets. For instance, a representative schema set is ACDEF: 2, ACD: 3, D: 4, BD: 1, B:
1, and BCF: 1; its RP-tree is shown in Fig. 1.

Finally, the PDRP algorithm scans all output patterns, and if any pattern is determined to not be covered (such pattern is called an exploration pattern), the algorithm finds a current maximum set (a representative pattern set) that covers the pattern. The current maximum set mentioned here means that the items contained in the set are the largest, and the pattern is one of representative patterns that are to be mined last. Using this algorithm, representative pattern sets on the projection database can be obtained.

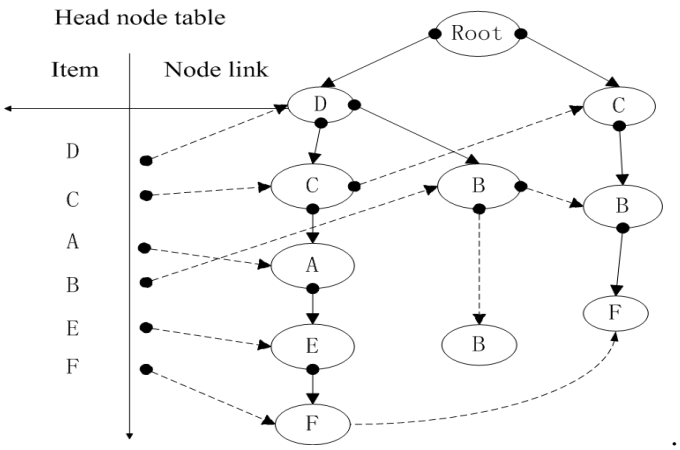



Fig 1. Sample RP-tree.

## IV. LOAD BALANCING

Load adjusting is a vital innovation for accomplishing an equilibrium in a group framework. Its guideline is to make the heap on every hub arrive at a fair standard with the goal that the handling season of every hub is something very similar, to further develop the general handling productivity. In the past segment, a delegate design set equal mining calculation was presented, however load adjusting was not thought of. The calculation proposed in this paper extends the first data set through the comparing connection between the gathering id in the G-list table and the regular 1-itemset and afterward processes it in the guide capability of every hub by projecting to various hubs.

Then, we present the computation of each heap unit. The heap of the whole agent design set equal mining calculation process is equivalent to the amount of the heaps on every hub independently addressing the example set mining, and the heap on every hub is the amount of the heaps of all the incessant 1-itemset projection information bases on every hub. Be that as it may, the heap of each incessant 1-thing projection data set is altogether different from the info information, and it is difficult to acquire the precise estimation of its heap results.

## V. ALGORITHM EXAMPLE

In view of the above depiction, we can figure out a fundamental web-based delegate design set equal mining calculation. To all the more likely comprehend the calculation work process, this segment presents a model informational index for which each step of the proposed equal web based mining agent design calculation is portrayed exhaustively. This segment utilizes the exchange data set with the information displayed in Table 1 to represent the execution aftereffects of each step of the calculation. At first, the initial segment of the calculation is executed, including in equal. This part utilizes MapReduce to produce all the continuous 1-itemsets and their comparing support. For the information in Table 1, the base help is set to 2, and a successive 1-itemset called a F-list is created, as displayed in Table 2.

Then, the second piece of the calculation is executed to bunch the successive thing sets. The static burden adjusting technique is utilized to bunch the incessant 1-itemsets, and the heap state of each continuous thing is acquired through estimation.

| Item | Sup |
|---|---|
| D | 7 |
| C | 7 |
| F | 6 |
| A | 6 |
| G | 4 |
| D | 3 |
| E | 2 |

Table 1. F-list.

| Map input key Value | Ordered transaction set (reduce infrequent items) | Mapper output key : Value |
|---|---|---|
| A,B,C,D | B,C,A,D | 1: B,C,A,D 2: A,D 3: D |
| B,D,F,G | B,F,G,D | 1: B,F,G,D 2: F,G,D 3: G,D |
| B,C,F | B,C,F | 1: B,C,F 2: F |
| A,B,C,D,F ,G | B,C,F,A,G,D | 1: B,C,F,A,G,D 2: F,A,G,D 3: G,D |
| A,C,G | C,A,G | 1: C,A,G 2: A,G 3: G |
| B,C,G | B,C,G | 1: C,A,G 3: G |
| A,C,F | C,F,A | 1: C,F,A 2: F,A |
| A,B,C,E,F | B,C,F,A,E | 1: B,C,F,A,E 2: F,A,E 3: E |
| A,B,E,F | B,F,A,E | 1: B,F,A,E 2: F,A,E 3: E |

Table 2. Mapping

By calculating the total load, the load is evenly distributed to each node. The load situation and the final grouping result are shown in Table 3.

The strategy embraced in this step can effectively ensure that the regular example is packed at every free hub based on not losing the continuous example. This step really utilizes the idea of identicalness class. Since Hadoop sends the outcome delivered by the mapper to a similar hub as per a similar key worth, the calculation proposed in this paper achieves the target of not losing the outcome utilizing equal projection. As the example that produces B and C will just exist on the main hub, the example containing B and C won't show up on the second and third hubs; hence, the outcome is that all incessant examples are right. This step guarantees that the chose delegate example can likewise be chosen quite well. Because of equal handling, a few repetitive delegate designs are as yet chosen, and the critical improvement in time effectiveness compensates for this weakness. The calculation proposed in this paper performs effectively. The mapper aftereffects of this step are displayed in Table 4.

| Reducer Input (projection data base) key: value | Local frequent items(projection data base) |
|---|---|
| 1:{BCAD/BFGD/BCF/BCFAGD/CAG/BCG/CFA/BCFAE/BFAE} | {(B:7,C:5,F:5,A:3,G:3,D:3,E:2)}\|B<br>{(C:7,F:4,A:5,G:3,D:2)} \|C |
| 2: D/FGD/F/FAGD/AG/FA/FAE/} | {(F:6,A:4,G:2,E:2)} \|F<br>{(G:2,D:2,E:2)} \|A |
| 3:{D/GD/GD/G/G/E/E} | {(G:4,D:2)} \|G<br>{(D:3)} \|D<br>{(E:2)} \|$E$ |

Table 3. The mapper results

Regarding the key-esteem matches created by the mapper, for the Hadoop framework, the information will be sent through the correspondence interaction. This step is called mix. This step is additionally the essential justification for the equal speed increase proportion decrease. Because of the correspondence cost, the calculation can't ensure the speed increase proportion of the equal calculation. By yielding the worth of a similar key worth to a similar minimizer, every minimizer is identical to a nearby informational index. On every minimizer, the nearby agent design set equal mining calculation is controlled by counting the neighborhood continuous 1-itemsets. The

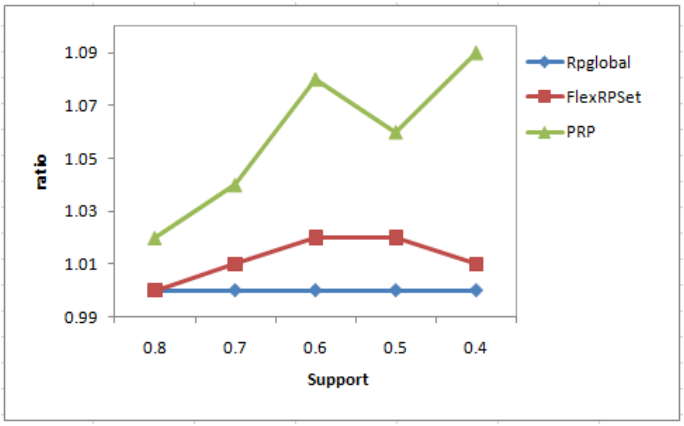
consequence of producing nearby successive things on the minimizer is displayed in Table 4.

## VI. EXPERIMENTAL ENVIRONMENT AND DATA SETS

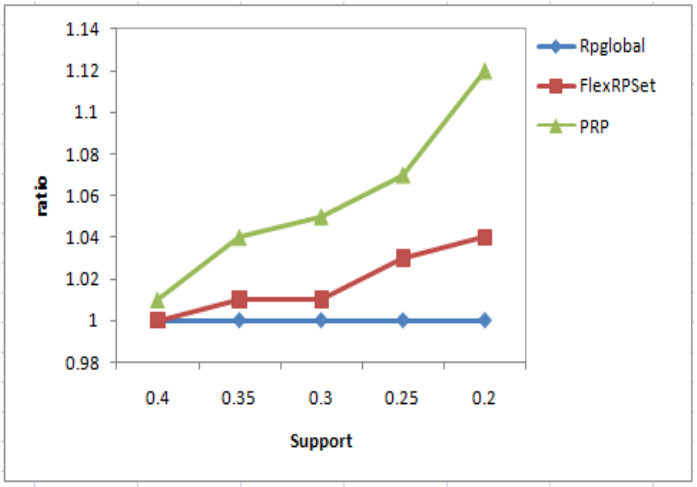
The calculation utilized in the trial is written in the Java language, and the running climate is Hadoop 1.2.1. The bunch utilized in this paper has a sum of 5 machines, one of which works as an expert hub, and the other four are working hubs. The central processor recurrence of the machine is 2.33 GHz, the Smash size is 4 GB, and the working framework is RedHat. The particular data of the informational collections [5] is displayed in Table 4.

The size of information utilized in this paper shifts. A few informational collections don't arrive at the predetermined size; hence, the informational collection of the predefined size is gotten by extending the informational collection, and another informational index is framed by separating the informational collection in an irregular way. Probes informational collections of different sizes have shown that our calculation has widespread pertinence.

The first experiment is to compare the algorithm PFP proposed in this paper with the existing algorithms RPglobal and FlexRPset. First, set N to be the number of representative patterns generated by RPglobal. Fig. 2 shows the ratio of the number of patterns generated by other algorithms to the




(a)




(b)

Fig 2. Ratio of representative patterns for various values of min_sup. (a) accidents. (b) mushroom.

number of representative patterns generated by RPglobal for various values of the minimum support. Clearly, the ratio of the number of representative pattern sets for the RPglobal algorithm is always 1. The parallel nodes of the experiment are set to 2, the parameter is set to 0.2, and the experiments will be performed on the data sets accidents and mushroom.

Fig. 4 shows that the number of representative pattern sets is slightly higher because the lgorithm proposed in this paper generates representative pattern sets in the process of mining frequent patterns. This is a method for covering frequent patterns by

selecting the longest pattern, while the existing algorithms RPglobal and FlexRPSet mine representative pattern sets by selecting a greedy algorithm that can cover the most frequent patterns. The experimental results show that the number of representative pattern sets generated by the proposed algorithm is similar to that of the existing algorithms, showing that the proposed method is very effective.

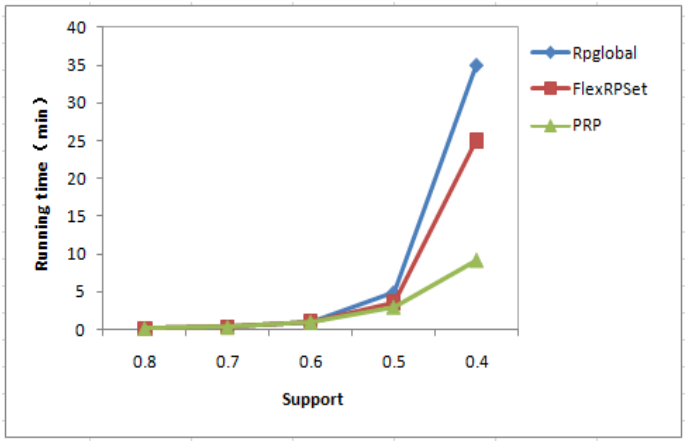
Further, Fig. 5 shows the comparison of the ratio of the number of representative patterns generated by several algorithms at various values of parameter $\delta$ - The parallel nodes of the experiment are set to 2, the parameter support is set to 0.4. The algorithm proposed in this paper and the existing algorithms still have little difference in the number of result sets. Moreover, the number of result sets of the proposed.

| Data Set | Number of transactions | Number of items | Maximum number of columns in an itemset | Average Number of columns in an itemset |
|---|---|---|---|---|
| Mushrooms | 8214 | 119 | 23 | 23 |
| Accidents | 340183 | 468 | 52 | 33 |
| Chess | 3196 | 75 | 37 | 37 |
| Connect | 67557 | 119 | 43 | 43 |
| Pumsb | 49046 | 2113 | 74 | 74 |
| Pumsb_star | 49046 | 2088 | 63 | 50 |

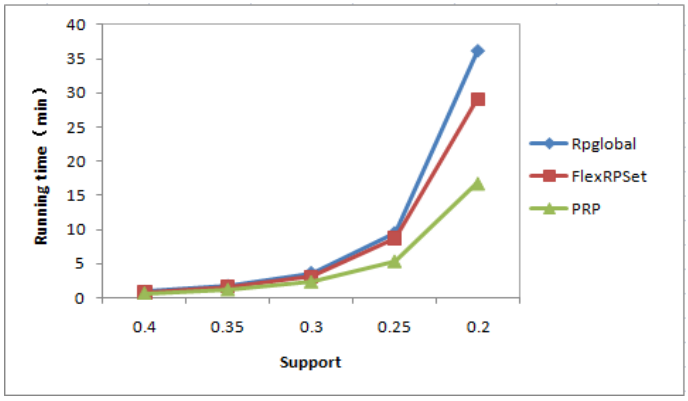
Table 4 Data sets used in the experiments.

## VII. ALGORITHM EFFICIENCY ANALYSIS

The second set of experiments compares the efficiency of the proposed PFP algorithm with the existing algorithms RPglobal and FlexRPset. The experimental results are shown in Fig. 6. The parameter  is set to 0.2. The algorithm presented in this paper reduces the running time quite significantly. Moreover, it is observed that the algorithm proposed in this paper is more efficient when the support is smaller, which is because the smaller the support is, the larger the number of frequent patterns generated. The current calculations are carried out by compacting successive examples, and the calculation proposed in this paper is to mine delegate design sets during the time spent regular example mining. Subsequently, the quantity of successive example sets is huge in the event that the help is more modest, and the calculation proposed in the paper accomplishes an extraordinary improvement in the general proficiency. Fig. 3 shows the correlation of the running season of the calculation with the difference in boundary δ The boundary support is set to 0.4. As displayed in Fig. 3, the calculation introduced in this paper performs well at various upsides of δ Contrasted with the other two calculations; the proposed calculation altogether lessens the running season of the general program. As the worth of δ is slowly expanded, it is seen that the proposed PRP calculation has a steady running time on the grounds that the PRP calculation creates a delegate design set during the mining system; in this manner, the co proficient δ makes little difference.




(a)



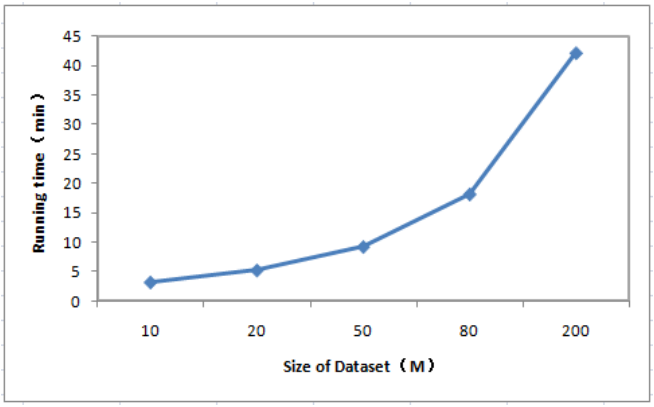

(b)

Fig 3. Running time for various values of min_sup. (a) Accidents. (b) Mushroom.

## SCALABILITY ANALYSIS

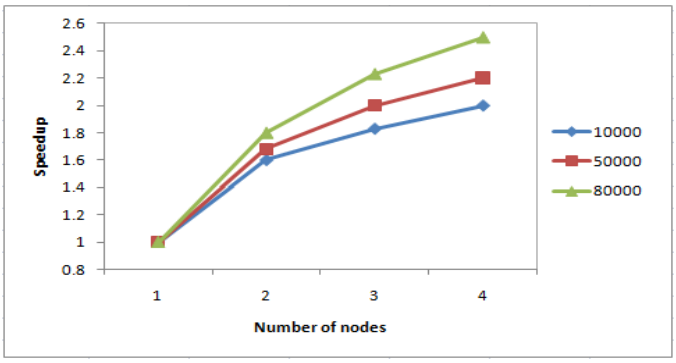
In this algorithm, parallelization is realized using Hadoop, and the communication cost will be a major factor affecting the running time of MapReduce processing. In the entire process of unning the program, the data are first sliced, and each node is allocated a certain amount of data. Therefore, the scalability of the algorithm is observed by increasing the size of data. The result is shown in Fig. 8. The parallel nodes of the experiment are set to 2, the parameter $\delta$ is set to 0.2, and the parameter support is set to 0.4.

The trial results show that the running season of the equal calculation increments dramatically, matching the guess in the paper. The intricacy of the calculation results for the most part from the list phase of the calculation, and the identification activity itself is outstanding; thus, the paper executes equal handling in the calculation. By disseminating the identification cycle to various hubs, the running season of the program is diminished enormously, and at the calls for remarkable investment; accordingly, there is no straight change with the expansion in the quantity of groups. Second, the correspondence between MapReduce processes is tedious; thus, the more noteworthy the quantity of hubs is, the more drawn out the transmission time. Adding hubs just diminishes the running season of the calculation and doesn't lessen the transmission time. The over two focuses make sense of why the incline isn't a bend of 1.




(a)

Fig 8. Running time for different data set sizes.




(b)

Fig 4. Speedup for different data sets and nodes.

Same time, it is proven that the program in this paper scales well.

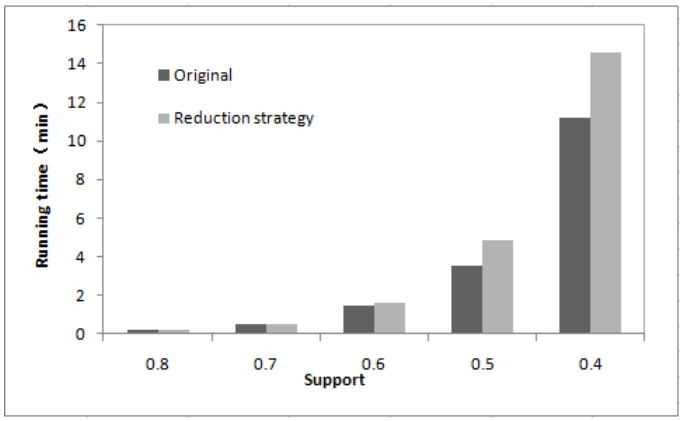



Fig 5. Results of different optimal strategies.

PERFORMANCE OPTIMIZATION ANALYSIS

The paper presents two procedures for enhancing execution: one depends on shut design decrease, and the other is on surmised decrease. The equal hubs of the investigation are set to 2, the boundary δ is set to 0.2, and the boundary surmised decrease factor is set to 0.7. As the trial results show, both of these two procedures significantly affect the exploratory outcomes and diminish the running season of the program. As displayed in Fig. 5, with the expansion in help, the decline in the running season of the proposed calculation is more evident, which is because of the decrease techniques proposed in this paper, i.e., the shut example decrease and the rough decrease. That's what the explanation is, as help diminishes, a rising number of continuous examples will be produced, and the calculation proposed in this paper can actually end the crossing of regular examples ahead of time by both shut design and estimated decrease. In this manner, it is extremely viable for shortening the running time.

## VIII. CONCLUSIONS

Most of the proposed representative pattern-set parallel mining algorithms select representative pattern sets after mining frequent patterns. This framework makes the running time difficult to evaluate in the context of big data. The online representative pattern-set parallel-mining algorithm based on the MapReduce parallel framework proposed in the paper improves the time efficiency by an order of magnitude. Although the running time is reduced by the parallel representative pattern-set mining algorithm, the number of result sets is increased. Therefore, it is necessary to minimize the result sets in the process of parallelization. In addition, the algorithm proposed in the paper is applied to the item sets, and the frequent pattern compression can be performed on the sequence in the future research. These problems are left by this paper for further studies.

## IX.REFERENCES

[1] N. G. Semaltianos. 2010. Critical Reviews in Solid State and Materials Sciences, 35, 2, 105-124. DOI: 10.1080/10408431003788233

[2] T. Iqbal, S. Tufail, and S. Ghazal. 2017. International Journal Nanoscience Nanotechnoogy. 13, 1, 19-52.

[3] C.-C. Chou, C.-H. Liu., and B.-H. Chen. 2014. Energy, 70, 231-238. DOI: 10.1016/j.energy.2014.03.118

[4] S. C. Mali, A. Dhaka, C. K. Githala, and R. Trivedi. 2020. Biotechnology Reports, 27, e00518. DOI: 10.1016/j.btre.2020.e00518

[5] M. Ganjali, P. Vahdatkhah, and S. M. B. Marashi. 2015. Procedia Materials Science, 11, 359–363. DOI:10.1016/j.mspro.2015.11.127

[6] I. M. Budiati, F. Sa'adah, N. D. Rifani, and A. Khumaeni. 2019. AIP Conf. Proc., 2202. DOI:10.1063/1.5141616

[7] Satriyani, C. M., dan Khumaeni, A., 2019, Synthesis of Colloidal Copper Nanoparticles Using Pulse Laser Ablation Method, Journal of Physics: Conference Series, 1217, 012019.

[8] R. M. Tilaki, A. I. Zad, and S. M. Mahdavi. 2007. Appl. Phys. A 88, 415–419. DOI: 10.1007/s00339-007-4000-2

[9] T. Begildayeva, S. J. Lee, Y. Yu, J. Park, T. H. Kim, J. Theerthagiri, A. Ahn, H. J. Jung, M. Y. Choi. 2021. Journal of Hazardous Materials 409, 124412. DOI: 10.1016/j.jhazmat.2020.124412

[10] H. Khalid, S. Shamaila, N. Zafar, R. Sharif, J. Nazir, M. Rafique, S. Ghani, and H. Saba. 2016. Acta Metall. Sin. (Engl. Lett.), 29, 8, 748–754. DOI: 10.1007/s40195-016-0450-x

[11] M. Fernández-Arias, M. Boutinguiza, J. D. Val, C. Covarrubias, F. Bastias, L. Gómez, M. Maureira, F. Arias-González, A. Riveiro, J. Pou. 2020. Applied Surface Science 507, 145032. DOI: 10.1016/j.apsusc.2019.145032

[12] J. Xiao, P. Liu, C. X.Wang, and G. W. Yang. 2017. Progress in Materials Science, 87, 140-220. DOI: 10.1016/j.pmatsci.2017.02.004

[13] L. Fedele, L. Colla, S. Bobbo, S. Barison, and F. Agresti. 2011. Nanoscale Research Letters, 6, 300. DOI: doi.org/10.1186/1556-276X-6-300

[14] S. Moniri, M. Ghoranneviss, M. R. Hantehzadeh, and M. A. Asadabad. 2017. Bull. Mater. Sci., 40, 1, 37–43. DOI: 10.1007/s12034-016-1348-y

[15] M. Alhajj, M. S. A. Aziz, F. Huyop, A. A. Salim, S. Sharma, and S. K. Ghoshal. 2022. Biomaterials Advances, 142, 213136. DOI: 10.1016/j.bioadv.2022.213136

**Cite this Article :**

A Geetha, Ravindra Changala, Goda Gangaram, Dr. Mahesh Kotha, "MapReduce Framework to Improve the Efficiency of Large Scale Item Sets in IoT Using Parallel Mining of Representative Patterns in Big Data", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 9 Issue 6, pp. 151-161, November-December 2022. Available at doi : https://doi.org/10.32628/IJSRST229618

Journal URL : https://ijsrst.com/IJSRST229618