Turkish Journal of Computer and Mathematics Education Vol.12 No. 7 (2021), 1788-1797

Research Article

Fast Frequent Item Mining From Big Datausing Map Reduce And Bit Vectors

Thirumaran. S, R. Nagarajan

Department of Computer Application, Alagappa Government Arts College, Karaikudi, India,
Department of Computer and Information Science, Annamalai University, India
Thirumaran.s@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 16 April 2021

Abstract: One of the most important areas that are constantly being focused recently is the big data and mining frequent patterns from them is an interesting vertical which is perpetually being evolved and gained plethora of attention among the research fraternities. Generally the data is mined with the aid of Apriori based algorithms, tree based algorithm and hash based algorithm but most of these existing algorithms suffer many snags and limitations. This paper proposes a new method that overrides and overcomes the most common problems related to speed, memory consumption and search space. The algorithm named Dual Mine employs binary vector representation and vertical data representations in the map reduce and then discover the most patterns from the large data sets. The Dual mine algorithm is then compared with some of the existing algorithms to determine the efficiency of the proposed algorithm and from the experimental results it is quite evident that the proposed algorithm "Dual Mine" outscored the other algorithms by a big magnitude with respect to speed and memory.

Keywords:

INTRODUCTION

The main purpose of the data mining is to unearth the previously unknown patterns hidden beneath the raw data [1]. The most common task that is hugely popular in the data mining vertical is frequent pattern mining where the most frequently occurring items are found (market basket analysis, frequently purchased commodities by the consumers, frequently visited web pages in a website). The pioneer in this frequent itemset mining is carried out by Srikanthagarwal who proposed the Apriori algorithm [2]. The Apriori algorithm employs the test and generate notion and then discovers the frequent patterns using level wise paradigm. But the most important drawback is excessive generation of candidate itemset especially the 2-itemset candidates which will increase the operational cost related to execution time and memory usage.

The FP-growth algorithm [3] is another popular frequent pattern mining algorithm that employs tree based structure to unearth the frequently co-occurring itemsets in the raw data. The important advantage of this algorithm is that it scans the database only two times unlike Apriori which scans k time where k is the maximum cardinality of the unearthed frequent patterns.

According to the author [Witten and Frank, 2000], the term Data mining is defined as a process discoveringhidden, anonymous, and putatively useful information from the given huge junk dataset. Data mining is one of the most exciting information based invention development created by us to ease out decision making.Data mining has become an essential service that can decode and unearth the cloakedpatterns and data presentclueless in the raw data into human readable and understandableinformation for a wider usage. It has a wide scope of usagein the field of marketing, bioengineering, gene technologies, finance, and engineering.

According to the authors David Hand, Mannila and Smyth [4] data mining is defined as,

"The analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner"

BIG DATA

Big data is a mind boggling term for immense data sets having enormous, progressively changed and complex structure with the difficulties of chronicling, investigating and imagining for additional methodology or results. Big data consolidates data from email, online life, content reports, images, sound, video files, and from plenty of sources that is absent in the customary social databases. Henceforth the big data will be unstructured, random and irregular which represent a bigger risk in the investigation. Coming up next are the primary qualities of the big data, and they are shown by 5V's,

Veracity-accentuations on the nature of the information crude data (e.g., suspiciousness, issue, and validity of the data)

Velocity – the speed at which data are filed or found (Speed and promptness).

Value - the handiness of data (Merit and worth).

Variety - different sorts, substance or configurations of data (Class, categories).

Volume-centers on the amount of data (Size and quantity).

Because of the 5Vs" characteristics of big data, new kinds of calculation are required for managing, questioning, and handling these big data so as to enable improved fundamental administration, comprehension, and procedure upgrade. This saturates and prods research and practices in data science, which plan to make organized or quantitative data scientific calculations to look at (e.g., evaluate, clean, change, and model) and mine big data.

BACKGROUND OF THE PAPER

The Knowledge Data Discovery process includes not many advances driving from crude data to some type of significant and valuable knowledge. The huge component of data present in a database regularly outperforms the ability to break down and mine it productively, in this manner bringing about a slack to comprehend the data totally in wording with the business needs like benefit, item infiltration in the market, and promotions.

Frequent itemset mining has increased tremendous significance among the examination society generally since the business houses have become globalized. It is basic for the business houses to tap the accessible data assets conveniently to the full degree to advance their items all inclusive. Numerous affiliation rule based frequent itemset mining calculations are created and proposed by various research researchers to improve and support the volume of business exchanges over the globe.

SCOPE OF THE PAPER

The proposed work mainly focuses on the discovery of frequent patterns by utilizing bit vectors and map reduce in very large databases and evades the time complexity which is considered as the main culprit in degrading the performance of the algorithm speed while mining the data required. The proposed algorithm uses new pruning technique to elude huge computation and unearth frequent patterns present in the very large databases. The primary scope of this paper is to improvise and alleviate the complicatedcomputations present in the state of the art existing algorithms and to come up with a simpleapproach without huge computational cost and overheads to discover frequent patterns.

CHALLENGES IN THE PAPER

Finding fascinating frequent itemset from the crude big data is an extreme undertaking as the whole procedure includes part of entangled computations identified with frequent count, pruning of unpromising things and memory related overheads bringing about expenses. So far numerous creators have proposed their methods to find frequent examples however the vast majority of the current strategies proposed endures the serious issue of delivering an enormous number of candidates and this confinement in fact decrease mining execution as far as speed and memory space. The first test present in this paper work is to uncover the frequent itemsets without compromising on the speed, memory and search space.

MOTIVATION

The inspiration of the proposed strategy depends on the assessment that for each kind of data and each sort of client inclination, it is basic to give the new methodology to produce ideal outcomes which limit the computational expense related to time as well as memory usage. The essential inspiration to complete this paper work is to give the clients a methodology which can proficiently deal with extremely big data with no difficulty and simplicity out the lumbering calculations required during the itemsetgeneration. The main reason which inspires adequate number of analysts in this vertical is that abundant volume online data that are promptly accessible over the globe and the majority of the firms are utilizing new and novel methods to pick up the bit of leeway and to draw in and hold the customers.

MAP REDUCE

MapReduce [16] is a synchronous and extendable programming architec-ture for data, thorough applications and specialized examination. MapReduce works just in Key/esteem sets. There are two phases of MapReduce work, alluded to as Map stage and Reduce stage. The information is separated into various sections by the Map stage. Each Map task gets a key/esteem set and creates a rundown of center key/esteem set. At that point the underline condition of MapReduce consolidations and mix all the center an incentive as indicated by the indistinguishable center key, the underline condition of MapReduce sends the center an incentive to the reducer. Every Reducer gets all the center records identified with a particular key and produces a final pair of key/esteem set.

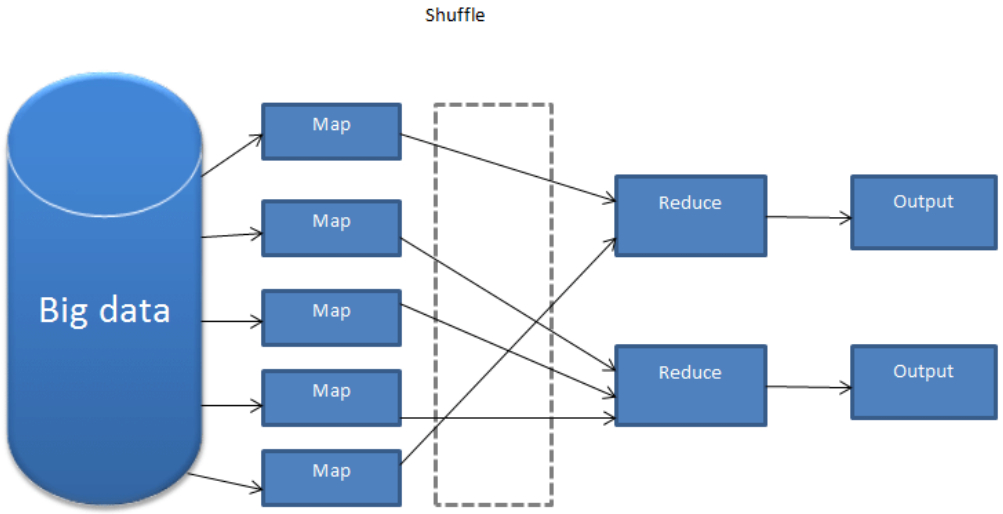

Figure 1: Map Reduce architecture

PROPOSED APPROACH

` The proposed approach involves lot of process and the important procedures are enumerated here under with some sample database. The sample database is shown in the table 1. The sample data shown in the table 1 is initially processed to find the unique items present in the transactions, and then the data is represented in the binary format as shown in the table 2. This binary table represents the bit vector representation and this minimizes the memory usage considerably to a greater extent. The map reduce phases are employed shrewdly to alleviate the overheads regarding the running time and the memory usage which is the major drawback in the existing algorithms.

TID	ITEM LIST
1	N, O
2	M, O, Q
3	M, Q
4	N, O ,Q
5	M, N, O, P
6	M, N, O, Q

Table 1: Sample database

PROCEDURE FINDUNIQUE

The procedure to find the unique items present in the table 1 is found using the following procedure shown in the figure 2.

Procedure DiscoverUniques(Input Data D)
INPUT: Input DataĐ **OUTPUT:** UniqueItemswith Count **BEGIN:** 1. Load the Input data Đ 2. ∀Row Ř ∈Đ do 3. Find UniqueItems U[]and let the initial Count → 0 4. Close For 5. ∀ Row Ř ∈Đ do 6. IF (U[] present in R) then 7. Increment UniqueItem's Count by 1 8. Close IF

```
    9. Close FOR
    10. Return the UniqueItem[ ] with its respective Count value
END PROCEDURE
```

Figure 2: Pseudo code to discover the unique items

The procedure shown in the figure 2 produces the following output with their respective count values as shown in then table 2.

ITEMS	COUNT	PRESENCE
M	4	2, 3, 5, 6
N	4	1, 4, 5, 6
O	5	1, 2, 4, 5, 6
P	1	5
Q	4	2, 3, 4, 5

Table 2: Unique items with their count

PROCEDURE TO GENERATE BITVECTOR

PROCEDURE CreateBinaryTable(data D, UniqueItems)

Input: Data D, UniqueItems

Output: Binary table with ones and zeroes

1. Load and scan the Data D
2. For each Transactional Row R ∈ D do
3. For Each Item I ∈ UniqueItems do
4. If [I present in R] then
5. Mark "1" in Binary table
6. Else
7. Mark "0" in Binary table
8. Close IF
9. Close For
10. Close For
11. Return Binary table

End

Figure 3: Pseudo code to create binary table for the sample data

The first unique item is fetched from the table 2 and then the depending upon the presence of the item in the sample transaction table (i.e.) if the item is present in the transaction row, then it is marked by "1" else it is marked by "0" as shown in the final binary table 3.

ITEMS	T1	T2	T3	T4	T5	T6	Count
M	0	1	1	0	1	1	**4**
N	1	0	0	1	1	1	**4**
O	1	1	0	1	1	1	**5**
P	0	0	0	0	1	0	**1**
Q	0	1	1	1	1	0	**4**

Table 3: Binary table created using the CreateBinaryTable procedure

Let us consider the first row as shown below and since the item "M" is present in the transactional rows T2, T3, T5 and T6 they are marked with 1's and the rest of the transactional rows are marked with 0's.

The first step of the map | **M** | 0 | 1 | 1 | 0 | 1 | 1 | reduce is applied to the binary table shown in the table 3 and the pseudo code is shown here, and the results of the first stage of the operation is showcased in this section.

The first Map function is applied as shown in the following procedure and the generalized mapping function is,

Function Map1:
⟨The presence of each ItemID in row, recent transaction ID⟩→ ⟨Trans ID, Item ID⟩ ∀Pair ⟨Trans ID, Item ID⟩∈ Transaction database **do** **Emit** ⟨Transaction ID, Item ID, 1⟩

Figure 4: First mapping function

This mapping function is applied to every transactional row present in the transaction database and the result of the mapped data is showcased here under,

Function Map1: **RESULT**
⟨1, N, 1⟩, ⟨1, Q, 1⟩, ⟨2, M, 1⟩, ⟨2, O, 1⟩, ⟨2, Q, 1⟩, ⟨3, M, 1⟩, ⟨3, Q, 1⟩, ⟨4, N, 1⟩, ⟨4, O, 1⟩, ⟨4, Q, 1⟩, ⟨5, M, 1⟩, ⟨5, N, 1⟩, ⟨5, O, 1⟩, ⟨5, P, 1⟩, ⟨6, M, 1⟩, ⟨6, N, 1⟩, ⟨6, O, 1⟩, ⟨6, Q, 1⟩.

Table 4: Initial mapping result

To make it simple the first reduce function is applied to the data and this reduces the unwanted items and prunes away the unpromising items from the database and there by reduces the memory space and the items discovered will be decreased with a perpetual decrease in the time taken for the execution of the algorithm. The first reduce function is showcased in the following figure 5.

Function Reduce1:
⟨item, transaction list⟩→ list of ⟨frequent item, transaction information⟩ ∀element in⟨_ , element, _⟩ emitted by map1 function **Initialize**Counter[element] = 0 **Initialize**List[element] = ϕ ∀transaction ∈⟨transaction ID, element, 1⟩ emitted by map1 **Increment** counter[element] by 1 List[element] = List[element] ∪ {transaction ID} **IF**counter[element] ≥ min support threshold **then** **Emit** ⟨element, counter [element], List [element] ⟩. Close **IF** Close **For** Close **For**

Figure 5: First Reduce function

The minimum support value is assumed to be 2 and the following result is produced by the first reduce function as shown in the table 5,

Function Reduce1: **RESULT**
⟨M, 4, {2, 3, 5, 6}⟩, ⟨N, 4, {1, 4, 5, 6}⟩, ⟨O, 4, {2, 4, 5, 6}⟩, ⟨Q, 5, {1, 2, 3, 4, 6}⟩.

Table 5: Initial Reduce result

The first reduce function prunes away the item or the element "P" as its count is one which is lesser than the user defined minimum support value. The items M,N,O and Q are retained as their counts are found to be (4,4,5,4).

The second mapping function is applied to the data and the pseudo code is shown in the following section,

Function Map2:
⟨frequent pattern p, its transaction information⟩→ ⟨transaction, item pair⟩ ∀p in⟨p,_ , List[p]⟩ emitted by reduce1 Function ∀transaction ∈List[p] **do** ∀⟨transaction ID, element ID⟩ in transaction database **do** **if** is Relevant(element, p) **then**

Emit ⟨transaction ID, {p} ∪ {element}, 1⟩.

Figure 6: Second Map function

The second mapping function produces the candidates that are formed from the map1 function (M, N, O, Q) are processed and the first element M is considered and all the relevant element related to the item M is emitted, they are,

⟨2, MO, 1⟩, ⟨2, MQ, 1⟩, ⟨3, MQ, 1⟩, ⟨5, MN, 1⟩, ⟨5, MO, 1⟩, ⟨6, MN, 1⟩, ⟨6, MO, 1⟩, and ⟨6, MQ, 1⟩. Note that (i) items present in the transaction row 4 is not emitted as element M is not present and ⟨5, MP, 1⟩ is not emitted as the item P is already pruned because its count is less than the user defined minimum support value.

Similarly the element N is considered and the following relevant elements are emitted, {⟨1, NQ, 1⟩, ⟨4, NO, 1⟩, ⟨4, NQ, 1⟩, ⟨5, NO, 1⟩, ⟨6, NO, 1⟩, ⟨6, NQ, 1⟩}.Note that

1. Items of the transaction row 3 are not emitted since that row does not contain the element N.
2. ⟨5, NP, 1⟩ is not emitted since the element P is pruned as its min_sup value is less than the user defined support count.
3. Patterns ⟨5, MN, 1⟩ and ⟨6, MN, 1⟩are irrelevant since they are already processed by the second map function.

Similarly the element O is considered and the following relevant elements are emitted, {⟨2, OQ, 1⟩, ⟨4, OQ, 1⟩, ⟨6, OQ, 1⟩}.

Finally for the element Q, there are no elements to emit since it does not have any relevant items.

Now the second Reduce function is applied and the result are shown in this section,

Function Reduce2:

⟨Element pair, list of common transactions⟩→ list of ⟨frequent element pair, transaction information⟩,

∀P in⟨_ , Elementgroup P,_⟩ emitted by map_2 Function
Initialize counter[P] = 0
InitializeList[P] = ϕ
∀transaction in⟨transaction, P, 1⟩ emitted by map_2 Function
Increment counter[P] by 1
List[P] = List[P] ∪ {transaction}
if counter[P] ≥min_Sup threshold **then**
Emit ⟨P, counter [P], List [P]⟩.
Close **IF**
Close **For**
Close **For**

Figure 7: Second Reduce function

The second reduce function produces the following results,

Function Reduce2: **RESULT**
⟨MN, 2, {5, 6}⟩, ⟨MO, 3, {2, 5, 6}⟩, ⟨MQ, 3, {2, 3, 6}⟩, ⟨NO, 3, {4, 5, 6}⟩, ⟨NQ, 3, {1, 4, 6}⟩, ⟨OQ, 3, {2, 4, 6}⟩.

Table 6: Second Reduce function result

The item pairs MN, MO, MQ, NO, NQ and OQ are frequent as the support of the corresponding pair is found to be 2,3,3,3,3,3 as these six pairs appears at least in two of the transaction rows.

The third mapping and the reduce functions are applied and the pseudo code is showcased in the following segment,

Function Map k-1:

⟨frequent item (k - 1) tuplet P, Transaction Information⟩→ ⟨Transaction, Element k-tuplet⟩,

∀P in ⟨P,_ , List[P]⟩ emitted by $Reduce_{k-1}$ function

∀Transaction in List[P] **do**
∀⟨Transaction, element⟩ in transaction database **do**
If is Relevant (element, P) **then**
Emit ⟨Transaction, P ∪ {element}, 1⟩.

Figure 8: K-1 Map function

Function Reduce K≥ 2:
⟨Element group, list of common transactions⟩→ List of ⟨frequent item group, transaction information⟩

∀P in ⟨ , Element group P, ⟩ emitted by Map_{k-1} Function
Initialize counter[P] = 0
Initialize List[P] = ϕ
∀Transaction in ⟨Transaction, P, 1⟩ emitted by Map_{k-1}Function
Increment counter[P] by 1
List[P] = List[P] ∪ {Transaction};
If counter[P] ≥min_sup threshold **then**
Emit ⟨P, counter[P], List[P]⟩.
Close **IF**
Close **For**
Close **For**

Figure 9: K-1 Reduce function

The elements/pair "MN" appears in two transactions T5 and T6 and it emits three relevant items {⟨5, MNO, 1⟩, ⟨6, MNO, 1⟩, ⟨6, MNQ, 1⟩}. Similarly the pair "MO" appears in three transactions T2, T5 and T6. This pair emits two relevant items {⟨2, MOQ, 1⟩, ⟨6, MOQ, 1⟩}. The pair "NO" appears in three transactions T4, T5 and T6. This pair emits two relevant items {⟨4, NOQ, 1⟩, ⟨6, NOQ, 1⟩}.

Now the reduce function is applied and the following frequent pattern result are found and they are, {M, N, O}, {M, O, Q} and {N, O, Q}: ⟨MNO, 2, {5, 6}⟩, ⟨MOQ, 2, {2, 6}⟩, and ⟨NOQ, 2, {4, 6}⟩.

The next mapping function is applied (i.e.) map4 but since there are no relevant items, the function returns nothing and algorithm ends after discovering the frequent itemsets.

EXPERIMENTAL EVALUATION

The proposed dual mine algorithm is evaluated with some synthetic datasets and compared with the existing algorithms with respect to running time and memory usage. The results are illustrated and it clearly indicates that the proposed DM algorithm outscores the existing algorithms by a good margin. The synthetic dataset are generated by the IBM Quest data mining code. The parameters of the dataset are shown in the table 7.

Parameters	Description of Parameter
D	Total number of transaction in the dataset
T	Average number of items per transaction
I	Average length of transactions within the maximal frequent itemset

Table 7: Parameters used in synthetic dataset generation

The proposed algorithm DM is compared with many existing algorithms to check the precise working against the available best algorithms accessible in the research world. The comparison is made on execution time or running time, memory utilization while execution, and on the volume of candidates produced during the execution and the results are exhibited. The existing algorithms compared in this paper are illustrated in this section,

The bigFIM [6] algorithm a famous algorithm to discover frequent itemsets from big data and the bigFIM algorithm combines the features of the Apriori algorithm and Eclat algorithm [7] to produce a hybrid approach and discovers the frequent itemsets.

The parEclat [8] algorithm developed by Zaki is a parallel Eclat algorithm which utilizes vertical representation of the data and then uses the concept of parallel computing to generate frequent itemsets from very large databases.

The Single pass counting SPC algorithm [9] is an implementation of Apriori algorithm in parallel using map reduce. Here in this algorithm the support count of the candidates is parallelized and the entire algorithm is classified into two phases and operates well by overcoming the shortfalls present in the classical Apriori algorithm.

The proposed dual mine algorithm as well as the other existing algorithms are executed on the synthetic dataset T4I2.5D1M and the resultant candidates that are generated are noted are shown in the table 8 and then compared with the graphical representations as shown in the figure 10.

CANDIDATE GENERATION					
SYNTHETIC DATASET NAME - T30I20D10M					
Algorithm Name	**Minimum Support Threshold Values**				
	0.30	**0.35**	**0.40**	**0.45**	**0.50**
bigFIM	15979306	13869626	11082983	8986831	7832207
parEclat	14977863	13737290	10168736	8562617	7856108
SPC	15885615	12050275	10155162	8050726	7857153
DM	13866342	1087526	9875638	7051636	6601887

Table 8: Experimental evaluation on synthetic T30I20D10Mdataset regarding the volume of candidate generated.

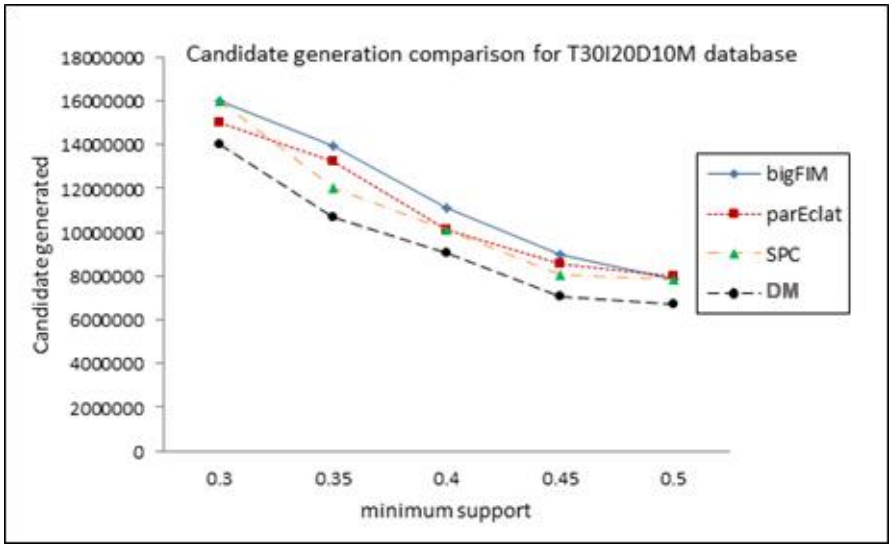

Figure 10: Candidate generation comparison for T30I20D10M synthetic database

As the density of the synthetic dataset is very large and transactions are long, the proposed DM algorithm performance was considerably good and executed most of the time without abnormal exit and outscored most of the other algorithms.

RUNNING TIME (SEC)	
SYNTHETIC DATASET NAME - T30I20D10M	
Algorithm	**Minimum Support Threshold Values**

Name	0.30	0.35	0.40	0.45	0.50
bigFIM	2989	2678	2188	1805	1665
parEclat	2676	2410	2108	1716	1587
SPC	2420	2219	1957	1692	1502
DM	1898	1778	1588	1403	1201

Table 9:Experimental evaluation on synthetic T30I20D10Mdataset regarding the running time.

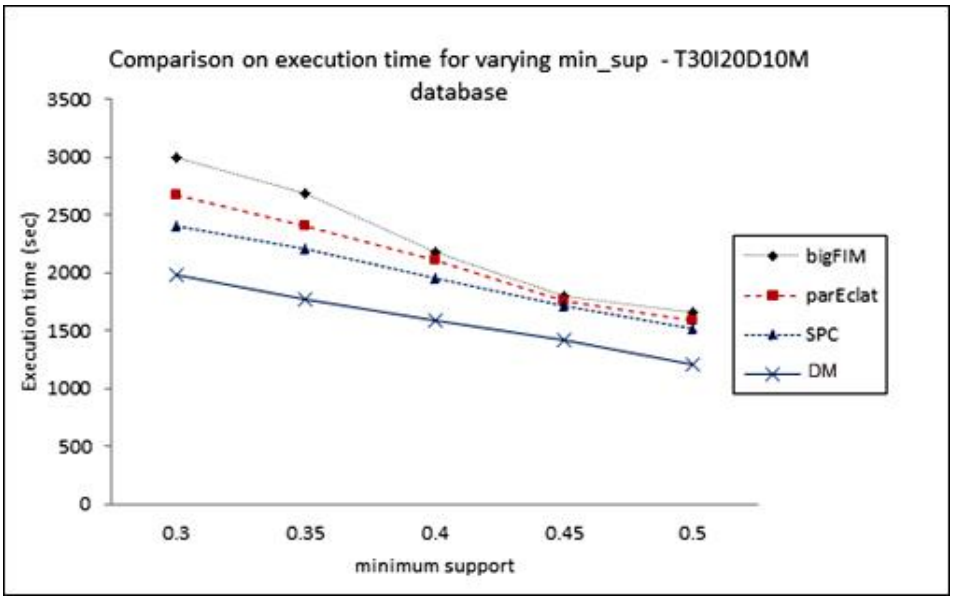

Figure 11: Run time comparison for T30I20D10M synthetic database

The proposed DM algorithm performed extremely well along with the SPC algorithm on the denser datasets and the proposed DM algorithm worked without out of memory error even for a very small minimum support count value (0.15). But BigFIM performed very badly and suffered many glitches. The memory based comparison was carried out in the next section.

MEMORY USAGE (MB)					
SYNTHETIC DATASET NAME - T30I20D10M					
Algorithm Name	**Minimum Support Threshold Values**				
	0.30	**0.35**	**0.40**	**0.45**	**0.50**
bigFIM	1880	1682	1481	1231	1003
parEclat	1610	1428	1150	1151	953
SPC	1418	1285	1187	1015	917
DM	1205	1187	1021	963	803

Table 10:Experimental evaluation on synthetic T30I20D10Mdataset regarding the memory usage.

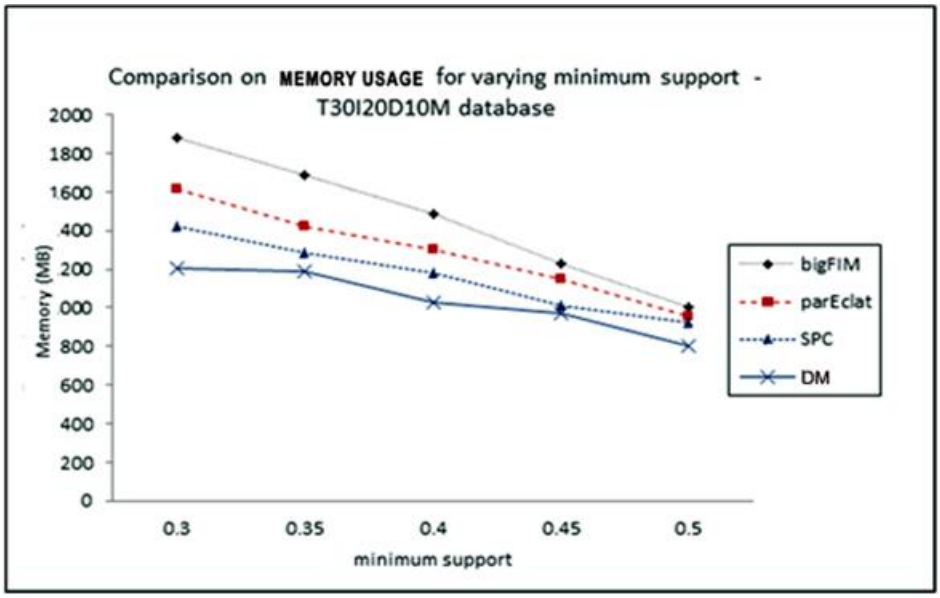

Figure 12: Memory usage comparison for T30I20D10M synthetic database

The memory consumption of the proposed dual mined algorithm was better than most of the existing algorithms and bigFIM algorithm performed the worst and the rest of the algorithm performed quite medially but lagged behind the DM algorithm.

CONCLUSION

The experimental evaluation of the three existing algorithms along with the proposed DM algorithm is carried out on dense synthetic dataset, the outcome of the experiments proved that the proposed algorithm fares better in consuming minimum memory, generates very less candidates and more importantly consumed very less time to complete the execution and the overall performance of the proposed algorithm was extremely good.

REFERENCES

1. RakeshAgrawal, Tomasz Imielinski, and Arun Swami."Mining association rules between sets of items in large databases". In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD 1993), Washington, DC, USA, pages 207-216. ACM, 1993.
2. RakeshAgrawal and RamakrishnanSrikant."Fast algorithms for mining association rules in large databases". In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB 1994), Santiago de Chile, Chile, pages 487-499. Morgan Kaufmann Publishers Inc., 1994.
3. Jiawei Han, Jian Pei, and Yiwen Yin."Mining frequent patterns without candidate generation". SIGMOD Records, 29(2):1-12, May 2000.
4. David Hand, HeikkiMannila, and Padhraic Smyth.*Principles of Data Mining*.Adaptive Computation and Machine Learning.MIT Press, Cambridge, Massachusetts, 2001.
5. Apache Hadoop (2013). MapReduce Tutorial, Hadoop 1.2.1 Documentation. https://hadoop.apache.org/docs/r1.2.1/mapred-tutorial.htm.
6. Sandy Moens, EminAksehirli, and Bart Goethals.Frequent itemset mining for big data. In 2013 IEEE International Conference on Big Data, pages 111{118. IEEE, 2013.
7. Mohammed J. Zaki. Spade: An efficient algorithm for mining frequent sequences. Machine Learning, 42(12):31-60, January 2001.
8. Mohammed J. Zaki, SrinivasanParthasarathy, MitsunoriOgihara, and Wei Li. Parallel algorithms for discovery of association rules. Data Mining and Knowledge Discovery, 1(4):343{373, 1997.
9. Ming-Yen Lin, Pei-Yu Lee, and Sue-Chen Hsueh. Apriori-based frequent itemset mining algorithms on mapreduce. In Proceedings of the Sixth International Conference on Ubiquitous Information Management and Communication, ICUIMC '12, pages 76:1-76:8, New York, NY, USA, 2012. ACM.