A Prime Number Based Approach for Closed Frequent Itemset Mining in Big Data

Mehdi Zitouni, Reza Akbarinia, Sadok Ben Yahia, Florent Masseglia

A Prime Number Based Approach for Closed Frequent Itemset Mining in Big Data

Mehdi Zitouni[1,2], Reza Akbarinia[2], Sadok Ben Yahia[1,3], and Florent Masseglia[2]

[1] University of Tunis ElManar, Faculty of Sciences of Tunis, LIPAH-LR 11ES14, 2092 Tunis, TUNISIA
[2] INRIA & LIRMM , Montpellier, FRANCE
[3] Institut Telecom, Telecom SudParis, umr 5157, cnrs Samovar, FRANCE
{Mehdi.Zitouni,Reza.Akbarinia,Florent.Masseglia}@inria.fr
Sadok.Benyahia@fst.rnu.tn

Abstract. Mining big datasets poses a number of challenges which are not easily addressed by traditional mining methods, since both memory and computational requirements are hard to satisfy. One solution for dealing with such requirements is to take advantage of parallel frameworks, such as MapReduce, that allow to make powerful computing and storage units on top of ordinary machines. In this paper, we address the issue of mining closed frequent itemsets (CFI) from big datasets in such environments. We introduce a new parallel algorithm, called CloPN, for CFI mining. One of the worth of cite features of CloPN is that it uses a prime number based approach to transform the data into numerical form, and then to mine closed frequent itemsets by using only multiplication and division operations. We carried out exhaustive experiments over big real world datasets to assess the performance of CloPN. The obtained results highlight that our algorithm is very efficient in CFI mining from large real world datasets with up to 53 million articles.

Keywords: Data Mining, Closed frequent itemset, MapReduce, Big data, parallel algorithm, CloPN

1 Introduction

In the past few years, advances in hardware and software technologies have made it possible for the users of information systems to produce large amounts of transactional data. Such data, which rapidly grows over time and needs more and more space and computing, is referred to as big data. Unfortunately, mining only frequent itemsets in big data generates an overwhelming large number of itemsets. The latter has the disadvantage to make their interpretation almost impossible and affect the reliability of the expected results.

Several studies were conducted to define and generate condensed representations of frequent itemsets. In particular, closed frequent itemsets have received much attention with very general proposals. Since its introduction in [1], closed

frequent itemsets (CFI) have been actively studied. However, existing algorithms for mining CFI flag out good performance as far as dealing with small datasets or when the support threshold is high. But, when the scale of dataset grows or the support threshold turns to be low, both memory usage and communication costs become hard to bear. Some early efforts tried to speed up the mining algorithms by running them in parallel [2, 3]. Even though, they could improve the mining performance, but promote other issues such as data partitioning, minimization of communication costs, and potential errors caused by node failures.

To deal with the above mentioned issues, we can take advantage of parallel frameworks, such as MapReduce or Spark, that allow to make powerful computing and storage units on top of ordinary machines. In this paper, we introduce a parallel algorithm, called CloPN (CLOsed itemset as Prime Numbers), in the aim of an efficient CFI mining by using such frameworks. In CloPN, we develop a new approach based on mathematical techniques. The items from the dataset are transformed into prime numbers, and closed frequent itemsets are generated by using only division and multiplication operations in the MapReduce framework. The main contributions of this paper are as follows :

- We propose a numerical representation of transactional datasets using a new transformation technique.
- We design an efficient MapReduce-based parallel algorithm for CFI mining by taking advantage of the mathematical properties of our numerical representation.
- We carried out exhaustive experiments on real world datasets to evaluate the performance of CloPN. The results show that our algorithm sharply outperforms the pioneering algorithms in CFI mining of large real world datasets with up to 53 millions articles.

2 Related Work

Many research efforts [4, 5] have been introduced to design parallel FIM algorithms capable of working under multiple threads under a shared memory environment. Unfortunately, these approaches do not address the problem of heavy memory requirement when processing large scale databases.

Since the introduction of closed frequent itemset in [1], numerous algorithms for mining it were proposed [6]. In fact, these algorithms tried to reduce the problem of finding frequent itemsets to the problem of mining closed frequent itemsets by limiting the search space to only closed frequent itemsets rather than the the whole powerset lattice. Furthermore, they have good performance whenever the size of dataset is small or the support threshold is high. However, as far as the size of the datasets become large, both memory use and communication cost are unacceptable. Thus, parallel solutions are of a compelling need. But, research works on parallel mining of closed frequent itemset are few. Authors in [7] introduce a new algorithm based on the parallel FP-Growth algorithm Pfp [8] that divides an entire mining task into independent parallel subtasks and achieves quasi-linear speedups. The algorithm mines closed frequent itemsets in

four MapReduce jobs and introduces a redundancy filtering approach to deal with the problem of generating redundant itemsets. However, experiments on algorithm were on a small-scale dataset.

3 Preliminary Notions

In this section, we give the definition of some concepts used throughout the paper.

Definition 1. (Extraction context) *An extraction context is a triplet* $\mathcal{K} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$*, where* $\mathcal{O}$ *represents a finite set of objects,* $\mathcal{I}$ *is a finite set of items and* $\mathcal{R}$ *is a binary (incidence)relation (i.e.,* $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{I}$*). Each couple* $(o, i) \in \mathcal{R}$ *expresses that the object* $o \in \mathcal{O}$ *contains the item* $i \in \mathcal{I}$*.*

The closure operator ($''$) denotes the closure operator $\phi \circ \psi$ *s.t.* (ϕ, ψ) represents a couple of operators defined by $\psi : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{O})$ *s.t.* $\psi(I) = \{o \in \mathcal{O} | \forall\ i \in I, (o, i) \in \mathcal{R}\}$ and $\phi : \mathcal{P}(\mathcal{O}) \rightarrow \mathcal{P}(\mathcal{I})$ *s.t.* $\phi(O) = \{i \in \mathcal{I} | \forall\ o \in O, (o, i) \in \mathcal{R}\}$. It is worth of cite, the closure operator induces an equivalence relation on the power set of items partitioning it into disjoint subsets called *equivalence classes* [9]. The largest element (*w.r.t.* the number of items) in each equivalence class is called a *closed itemset* (CI) and is defined as follows:

Definition 2. (Closed itemset) *An itemset* $I \subseteq \mathcal{I}$ *is said to be closed if and only if* $I'' = I$ *[1]. The support of I, denoted by Supp(I), is equal to the number of objects in* $\mathcal{K}$ *that contain I. I is said to be frequent if Supp(I) is greater than minsupp.*

4 Frequent Closed Itemset Mining

In this section, we introduce our CloPN algorithm for CFI mining. It operates in two steps. The first step is dedicated to transform the data and generates a list of transformed frequent items sorted in a descending order of their supports using a MapReduce pass. We denote this list as *F-List*. Using a second MapReduce pass, CloPN mines the complete set of closed frequent itemsets. Both steps are developed in the following.

4.1 First Step : Prime Number Transformation

In this subsection, we present some definitions and theorems used to transform and manipulate the input data by our algorithm.

Definition 3. (Prime Number) *An integer* $X > 1$ *is a prime number, if it is divided evenly only by 1, and itself, e.g.,* $X = 193$ *is a prime number.*

Definition 4. (PRIME FACTORS) *A non-prime integer $X > 1$ is a composed integer. So, X can be presented by a set of prime numbers multiplied together to make the original number. Formally, X can be represented as $X = p_1^{m1} p_2^{m2} ... p_r^{mr}$, having p_i as a prime number, and m_i a positive integer called multiplicity* [4] *of p_i, e.g., The prime factors of $X = 330$ are 2, 3, 5 and 11, since we have $330 = 2 \times 3 \times 5 \times 11$.*

Let us now define our data transformation technique.

Definition 5. (TRANSACTION TRANSFORMATION) *Let $\mathcal{T} = \{i_j ... i_k\}$ be a transaction, where i_r items from $\mathcal{T}$. The transformation process assign a prime number p_r to each item $i_r \in \mathcal{T}$. Then, the new numeric value of the transaction, denoted as $V_{\mathcal{T}}$, is computed by applying the following equation:*

$$V_{\mathcal{T}} = \prod_{j}^{k} p_r \tag{1}$$

Figure 1 illustrates the transformation phase for a sample dataset.

	A	B	C	D	E
1	×		×	×	
2		×	×		×
3	×	×	×		×
4		×			×
5	×	×	×		×

Item	PN	Supp
B	2	4
C	3	4
E	5	4
A	7	3

ID	$\mathcal{T}$	Primes	$V_{\mathcal{T}}$
1	C, A	3,7	21
2	B, C, E	2,3,5	30
3	B, C, E, A	2,3,5,7	210
4	B, E	2,5	10
5	B, C, E, A	2,3,5,7	210

Fig. 1: (**Left**) An extraction context $\mathcal{K}$. (**Middle**) The Items from $\mathcal{K}$ sorted in descending order with their corresponding prime numbers and supports. (**Right**) The context $\mathcal{K}$ transformed and reduced.

4.2 Second Step : Closed Frequent Itemset Mining

After the transformation step, CLOPN starts the second MapReduce pass to extract the complete set of frequent closed itemsets. Below, we give a description of the Map and Reduce phases for mining CFIs of our algorithm.

Map phase : During this step, the CloPN algorithm splits the context into sets of minimized contexts, denoted as Conditional-context. Bellow we give the definition of a conditional-context cited in [10].

[4] There is no duplicated items in a transaction from transactional dataset, so we will suppose that the multiplicity will be $m_i = 1$, without any loss of information.

Definition 6. (Conditional-context) [11] *Given an extraction context $\mathcal{K}$. Let i be a frequent item in $\mathcal{K}$. The i-Conditional-context is the subset of transactions containing i, while all infrequent items, item i and items following i in the F-list are omitted.*
Let j be a frequent item in X-conditional-context, where X is a frequent itemset. The jX-conditional-context is the subset of transactions in X-conditional-context containing j, while all infrequent items, item j, and items following j in local F-list are omitted.

Having frequent items sorted in descending order of their respective supports, for each item i the map function of CloPN checks its conditional-context from transactions containing only items that collocate with i. To facilitate the exploration of these sub contexts, we could use the technique proposed in [10] that defines a header table which is associated to each context. This table lists items contained in the corresponding conditional-context, sorted in descending order of their supports. However, in our approach, extracting closed frequent itemset wont need the use of this table (cf following section). Each of our map tasks emits a key-value pair, where the key is an item from the F-list generated in step one, and the value is a transaction which will be part of the conditional-context of the item.

Reduce Phase : Before describing the Reduce phase of our approach, for an itemset X, the following properties holds.

1. Closed itemset X extracted from a conditional-context is discovered by concatenating the items which are as frequent as X (in the conditional-context).
2. There is no need to develop a conditional-context of an itemset Y included in a closed frequent itemset already discovered X such that Supp(X) is equal to Supp(Y).

Now, having the following definition,

Definition 7. (Greatest Common Divisor (Gcd)) *The greatest common divisor of two or more integers, when at least one of them is not zero, is the largest positive integer that divides the numbers without a remainder. e.g. the* Gcd *of 8 and 12 is 4.*

We introduce the following theorem to extract frequent closed itemsets.

Theorem 1. (Gcd-Closure) *Let X-conditional-context be the subset of transactions contained X. The greatest common divisor in X-conditional-context represents the closure between all transactions in the conditional-context.*

Proof. As shown in Definition 2, the closure of an itemset X is produced from the intersection between all transactions containing X. With definition 7 and manipulating the prime numbers, the Gcd between integers is unique. Thus, having all VT_{ID}, extracting the closure from a set of transactions amounts to calculate the Gcd between them. Thus, the Gcd in X-Conditional-context is the closure between transactions composing $X-$Conditional-context.

After the shuffling process, each reducer will receive an item with its conditional-context as input. In fact, having the prime number representing the item and its transactions as a set of $V_{\mathcal{T}}$s, computing the closure from the conditional-context is straightforward by computing the Gcd of all transactions of the conditional-context. Doing so, there is no further need to store supports of items contained in the conditional-context. Indeed, if the closure exists, then will have inevitably the same support as that of the item. By concatenating the closure to the candidate item multiplying the prime number of the item and the number representing the closure, the result will be a closed frequent itemset that is represented as a number which is added to the set of final results. The steps of CloPN algorithm are as follows.

- **Step 0** : A preprocessing step to transform our data into numerical form.
- **Step 1** : Parallel Counting F-list: a MapReduce job is executed in order to count the frequency of all items that appear in $\mathcal{K}$. Each mapper is given one part of $\mathcal{K}$. The result is stored in one F-list for each shard of $\mathcal{K}$ in mappers.
- **Step 2** : Parallel mining closed frequent itemset: in this step CloPN mines the locally closed frequent itemset using the Gcd approach in parallel.
- **Step 3** : Collecting Results.

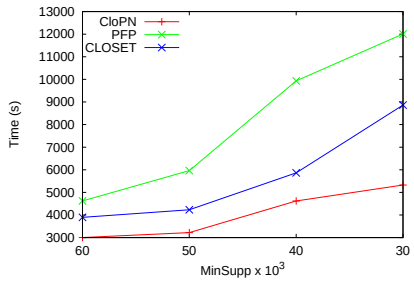
With the above four steps, we can mine the complete set of closed frequent itemsets correctly. Indeed, CloPN starts after the preprocessing phase, to count supports of items of the ground set. For that, it uses a classical MapReduce counting job, where Map is fed with fragments from $\mathcal{K}$, and the output of Reduce would be the list of items (primes) with their supports sorted in descending order and pruned from those who are not frequent. In the second step, CloPN starts to mine the set of closed frequent itemsets. CloPN algorithm uses a second MapReduce pass where it splits the context into a new conditional-contexts, having the dataset as a Map input. It tests the inclusion of the item by dividing the product of the transaction in dataset by the prime number representing the item. This latter is token as key and its conditional-context as value to start the reduce phase. At this point, the CloPN algorithm tries to extract the closure by applying our new closure operation that calculates the Gcd between all transactions in the conditional-context.

5 Experimental Evaluation

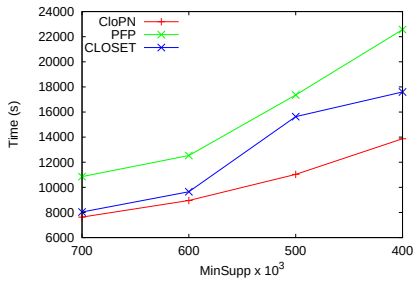
In order to assess effectiveness of the proposed approach, we carried out thorough tests on two real-life datasets. The first one, called "Wikipedia Articles", represents a transformed set of Wikipedia articles into a transactional dataset, each line mimics a research article. It contains 7,892,123 transactions with 6,853,616 distinct items, in which the maximal length of a transaction is 153,953, and the size of the whole dataset is 4.7 Gigabytes. The second dataset, called "ClueWeb", consists of about one billion web pages in ten languages that were collected in January and February 2009 and it is used by several tracks of the TREC conference. During our experiments, we used a part of "Clue Web" dataset that contains 53,268,952 transactions including 11,153,752 items with a maximal length

of a transaction equal to 689,153. The size of the considered "Clue Web" is 24.9 Gigabytes.

To perform our experiments, we used one of the clusters of Grid5000 [12] which is a large-scale and versatile test-bed for experiment-driven research on parallel and distributed computing. Our experiments were performed on a cluster with 10 nodes equipped with Hadoop 1.3.0 version. One node was designed as master, which was responsible for scheduling tasks execution among different nodes, and other nodes were set as workers. Our algorithm was implemented using java and the JDK version is openjdk-7-jdk. We compared our algorithm to a basic implementation of CLOSET algorithm in MapReduce and the parallel FP-Growth algorithm (PFP in short), which is described briefly in the Related Work section.



(a) The results on "Wikipedia Articles"



(b) The results on "Clue Web"

Fig. 2: Experimental Results of CLOPN

Figure 2a illustrates the experimental results tested on "Wikipedia Articles" of the three algorithms under different values of minimum support (Minsupp) less then 1% of the overall size of the dataset. CLOPN sharply outperforms both other algorithms. The reason is that,"Wikipedia articles" presents a large number of items which is almost equal to transactions numbers. Thus, as far as the Minsupp value is low, PFP and CLOSET generate too many candidates, and a lot of long conditional-contexts for each one. So, the inclusion tests and evaluations under the pruning methods used in these two algorithms causes lead as expected to poor performances. CLOPN overcomes these problems by using prime numbers to generate the conditional-contexts through division operations. Furthermore, the GCD in each conditional-context has eliminated the check of supports between the candidate and its deduced closure. Thus, the phase of computing the frequency of the closure is no longer of need.

Figure 2b shows the experimental results tested on "ClouеWeb". By reducing Minsupp, the number of frequent closed items does not change significantly. However, the number of matching transactions for candidate itemsets increases quickly, and this has the disadvantage of generating big conditional-contexts for itemset candidates. Interestingly enough, our algorithm performs much better, thanks to the efficient strategies which we developed, particularly the GCD ap-

proach that avoids redundant computations for producing the closure from each conditional-context.

6 Conclusion

In this paper, we revisited the closed frequent itemset mining problem and proposed a new algorithm for mining closed frequent itemsets by using prime numbers and processing big integers expressed in MapReduce model. Experimental results on two large scale datasets show that our algorithm achieves good scalability. The results illustrate that treating big data with string operations could cause many memory problems and a huge computational cost. The proposed method is able to solve this problems efficiently.

References

1. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT Conf.*, pages 398–416, 1999.
2. Mohammad El-hajj and Osmar R. Zaïane. Parallel leap: Large-scale maximal pattern mining in a distributed environment. In *In Conf. on Parallel and Distributed Systems*, pages 135–142, 2006.
3. Ke Chen, Lijun Zhang, Sansi Li, and Wende Ke. Research on association rules parallel algorithm based on fp-growth. In *ICICA Conf.*, pages 249–256, 2011.
4. Osmar R. Zaïane, Mohammad El-Hajj, and Paul Lu. Fast parallel association rule mining without candidacy generation. In *ICDM Conf.*, pages 665–668, 2001.
5. Li Liu, Eric Li, Yimin Zhang, and Zhizhong Tang. Optimization of frequent itemset mining on multiple-core processor. In *VLDB Conf.*, pages 1275–1285, 2007.
6. Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *KDD Conf.*, pages 236–245, 2003.
7. Su-Qi Wang, Yu-Bin Yang, Yang Gao 0001, Guang-Peng Chen, and Yao Zhang. Mapreduce-based closed frequent itemset mining with efficient redundancy filtering. In *ICDM Workshops*, pages 449–453. IEEE Computer Society, 2012.
8. Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y. Chang. Pfp: Parallel fp-growth for query recommendation. In *ACM Conference on Recommender Systems (RecSys)*, pages 107–114, 2008.
9. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. In *KDD Conf.*, pages 66–75, 2000.
10. Christian Borgelt. An implementation of the fp-growth algorithm. In *OSDM Workshop*, pages 1–5, 2005.
11. Jian Pei, Jiawei Han, and Runying Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.
12. `https://www.grid5000.fr/`.