Highly Efficient Pattern Mining Based on Transaction Decomposition

Youcef Djenouri[1], Jerry Chun-Wei Lin[2], Kjetil Nørvåg[1], and Heri Ramampiaro[1]
[1] Dept. of Computer Science, NTNU, Trondheim, Norway
[2] Dept. of Computing, Mathematics, and Physics, HVL, Bergen, Norway
{youcef.djenouri,noervaag,heri}@ntnu.no, jerrylin@ieee.org

***Abstract*—This paper introduces a highly efficient pattern mining technique called Clustering-Based Pattern Mining (CBPM). This technique discovers relevant patterns by studying the correlation between transactions in transaction databases using clustering techniques. The set of transactions are first clustered using the k-means algorithm, where highly correlated transactions are grouped together. Next, the relevant patterns are derived by applying a pattern mining algorithm to each cluster. We present two different pattern mining algorithms, one approximate and one exact. We demonstrate the efficiency and effectiveness of CBPM through a thorough experimental evaluation.**

***Index Terms*—Clustering, Pattern Mining, Decomposition, Scalability.**

1. Introduction

Frequent Pattern Mining is a data mining technique that finds frequently co-occurring items in a database, and accordingly provides relevant patterns. In practice, different database representation may be observed, ranging from Boolean databases to sequence databases. Various pattern mining techniques have been reported in the literature [1]. Frequent Itemset Mining (FIM) is the one of the most well-known of these techniques, which has been applied in several practical problem-solving applications. As an example, considering an information retrieval problem, the collection of documents is transformed into a transaction database, where each document is considered as a transaction and each term as an item. In this context, mining techniques, such as FIM, would allow to study different correlations between the terms of documents. For instance, if the pattern (*Data*, *Engineering*) is relevant, a high dependence exists between the terms *Data* and *Engineering*. Hence, if the user is looking for documents related to *Data*, it would be useful to also return documents related to *Engineering*. Unfortunately, pattern mining techniques for large databases, such as FIM, suffer from long processing time (runtime). To reduce the runtime of pattern mining, several optimization techniques have been proposed [2, 3]. However, these optimization techniques are incapable of dealing with databases containing a huge number of items, where only few of the relevant patterns are displayed to the end user. We contemplate that these techniques are inefficient because they consider the whole database in the mining process. In this paper, we propose a technique based on splitting the problem into several small sub-problems, as independent as possible, and then study and explore the correlation between them. Here, we introduce a new framework called clustering-based pattern mining (CBPM). In summary, the main contributions of this work are as follows: i) We use the k-means algorithm to decompose the transaction database into highly correlated clusters, where the aim is to minimize the number of shared items between clusters. ii) We propose two novel algorithms that use the clusters for pattern mining: an exact one that takes into account any shared items between clusters, and an approximate one that does not need to take into account the shared items. We then investigate the impacts of applying both the exact and the approximate algorithms on the mining effectiveness, as well as efficiency. iii) Finally, we integrate our approach with the well-known FIM PrePost+ algorithm [4]. The results on large datasets show that our approach outperforms the PrePost+ algorithm in terms of runtime. The results also demonstrate that the ratio of the satisfied patterns is more than 90% for the approximation-based algorithm.

The remainder of the paper is organized as follows. Section 2 gives an overview of related work on the most important pattern mining algorithms. A detailed explanation of our CBPM framework is given in Section 3. Section 4 presents the performance evaluation. Finally, Section 5 concludes the paper and outline our future work.

2. Related Work

Pattern mining problem has been largely studied over the past three decades. Various pattern mining techniques have been reported, including frequent, high utility and sequential patterns [5, 6, 7]. Frequent pattern mining is the first pattern mining problem which extracts all patterns that exceed the minimum support threshold. The main limitation of the conventional frequent pattern mining algorithms is that only binary cases could be mined. Several algorithms have been proposed for solving the frequent pattern mining problem [4, 8, 9, 10]. Agrawal et al. [8] proposed the Apriori algorithm, where candidate patterns are generated incrementally and recursively. To generate k-sized patterns as candidates, the algorithm calculates and combines the frequent (k-1)-sized patterns. This process is repeated until

no candidate patterns are obtained in an iteration. Han et al. [9] proposed the FP-Growth algorithm, which uses a compressed FP-tree structure for mining a complete set of frequent patterns without candidate generation. The algorithm consists of two phases: (i) construct an FP-tree that encodes the dataset by reading the database and mapping each transaction onto a path in the FP-tree, while simultaneously counting the support of each item and, (ii) extract frequent patterns directly from the FP-tree using a bottom-up approach to find all possible frequent patterns ending with a particular item. Deng and Lv [4] proposed PrePost+ algorithm, by employing a scalable data structure to represent itemsets; and adopting single path property of this structure to directly discovery frequent itemsets without generating candidate itemsets. All these algorithms suffer in terms of the runtime performance, as the search space is not well pruned, especially for low minimum support values. In order to improve the runtime performance of the pattern mining approaches, several techniques have been proposed, such as parallelization and metaheuristics. The parallelism operates based on exploiting local parallelism or distributed computing [2, 11], while the metaheuristics operates based on evolutionary and/or swarm intelligence approaches [3]. However, these optimization techniques are incapable of dealing with large transaction databases, where only a few number of interesting patterns may be discovered. To deal with this challenging issue, we present in this paper a new framework for pattern mining algorithms. This new framework explores decomposition techniques for determining relevant patterns. Similar ideas have been investigated in the database community, in particular in the areas of record linkage and entity resolution [12, 13, 14, 15]. The aim is to apply blocking-based techniques such as canopy clustering [13], suffix-blocking [12, 14], and Q-gram based indexing [15], to derive the dierent records that represent the same real-world object in a given database, and check if such a real-world object may be determined by a single record. These methods need domain specific knowledge and require complete redesign for pattern mining applications. In this paper, we attempt to follow these concepts by proposing a new framework for improving the runtime performance of the pattern mining algorithms.

3. Clustering-Based Pattern Mining (CBPM)

This section presents the principle of the CBPM framework, and describes its components in detail.

3.1. Overview

Here, we provide a general framework for the pattern mining for finding different dependencies between transactions, which will be used for efficient improvement of the mining process. This framework is illustrated in Figure 1, and is composed of two main steps: clustering and mining. **Clustering:** In this step, a transaction database is divided into a set of homogeneous clusters using clustering techniques, where a cluster may be viewed as a subset of transactions of the whole set of transactions. We take advantage of the clustering technique to extract the relevant knowledge, which will be used by the pattern mining algorithms. The patterns shared by two clusters constitute a shared set. The aim is to minimize the size of the shared sets, while having in the same cluster transactions that are highly correlated, i.e., transactions that share the maximum number of items. **Mining:** The mining process is applied on the clusters found in the previous step. In this context, two main approaches have been investigated, i.e., the approximation-based and exact. In the approximate approach, the clusters are mined to derive partial solutions, which are then merged into a global solution. In the exact approach, the mining process is applied on both the clusters and the shared sets, by aggregating these patterns on all clusters.

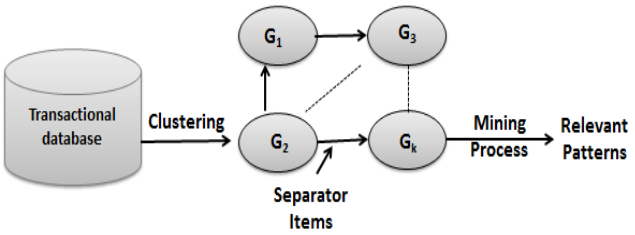

Figure 1. The CBPM framework.

3.2. Clustering

The set of transactions $T = \{t_1, t_2 ... t_m\}$ is partitioned into *k* disjoint clusters $C = \{C_1, C_2 ... C_k\}$, where each cluster C_i is a subset of the transactions in T. The set of items in C_i is denoted $I(C_i) = \bigcup_{t_j \in C_i} I(t_j)$, where $I(t_j)$ denotes the set of items in transaction t_j. The aim of clustering the transactions is to minimize the *shared items* (items occurring in more than one cluster) between the clusters. One way to solve this issue is to use a partitioning clustering algorithm. In this work, we adapt the k-means algorithm for clustering of the transaction database. In the following, this adaptation is presented in more detail.

Similarity computation. The similarity measure between two transactions t_i and t_j is computed as

$$D(t_i, t_j) = \max(|I(t_i)|, |I(t_j)|) - (|I(t_i) \cap I(t_j)|) \quad (1)$$

Centroid updating. Let us consider the set of transactions in cluster $C_i = \{t_1^{(i)}, t_2^{(i)}, ..., t_{|C_i|}^{(i)}\}$, the aim is to find the gravity center of this set, which is also a transaction. Inspired by the centroid formula developed in [16], we compute the centroid μ_i. The frequency of each item is calculated for all the transactions of the cluster C_i. The length of the transaction center, is denoted by l_i, and corresponds to the average number of items of all transactions in C_i as $l_i = \frac{\sum_{j=1}^{|C_i|} |I(t_j^{(i)})|}{|C_i|}$. Afterwards, the items of transactions in C_i are sorted according to their frequency, and only the l_i

most frequent items are assigned to μ_i, as $\mu_i = \{j | j \in \mathcal{F}_{l_i}\}$ where $\mathcal{F}_{l_i}$ denotes the set of the l_i frequent items of the cluster C_i.

Shared items determination. After constructing the clusters of transactions, we have to determine the set of items shared between the clusters. We define the set of shared items, denoted by S, as $S = \bigcup_{i=1,j>i}^{k} I(C_i) \cap I(C_j)$.

3.3. Mining Process

This step benefits from the knowledge extracted in the previous step. Instead of mining the whole transaction database, each cluster of transactions are handled separately. In this context, the two following approaches are proposed.

Approximate algorithm. In this approach, the clusters are handled separately without considering the shared items. The local relevant patterns are first extracted by applying the mining process on each cluster. The merging function is then used to derive the global relevant patterns. This function is constituted of the concatenation of all local relevant patterns. Such an approach returns partial relevant patterns from the whole transaction database. This is due to the fact that the shared items were not taken into account in the mining process.

Exact algorithm. This approach considers the shared items as well as the clusters in the mining process. This allows to discover all relevant patterns from the whole transactions. The mining process is first applied on each cluster of transactions, to extract the local relevant patterns. The possible candidate patterns are then generated from the shared items. For each generated pattern, the aggregation function (see Def. 3.1 below) is then used to evaluate this pattern in the whole transaction database.

Definition 3.1. We define an aggregation function of the pattern p in the clusters of the transactions C by

$$\mathcal{A}(p) = \sum_{i=1}^{k} Support(C_i, I(C_i), p)$$

Note that $Support(C_i, I(C_i), p) = \frac{|p|_{C_i, I(C_i)}}{|C_i|}$, where $|p|_{C_i, I(C_i)}$ is the number of transactions in C_i containing the pattern p. The relevant patterns of the shared items are then concatenated with the local relevant patterns of the clusters to derive the global relevant patterns of the whole transaction database.

4. Performance Evaluation

We carried out a series of experiments to evaluate the CBPM framework. The CBPM java source code is integrated in the frequent itemset mining algorithm [4]. All experiments have been performed on a computer with 64 bit core i7 processor running Windows 10 and having 16 GB of RAM. We ran the experiments using well-known

TABLE 1. DATASETS DESCRIPTION.

Name	Trans.Size	Item Size	Aver. Size
Accident	340,183	468	33.8
Chess	3,196	75	37.0
Connect	67,557	129	43.0
Mushroom	8,124	119	23.0
Pumsb	49,046	2,113	74.0

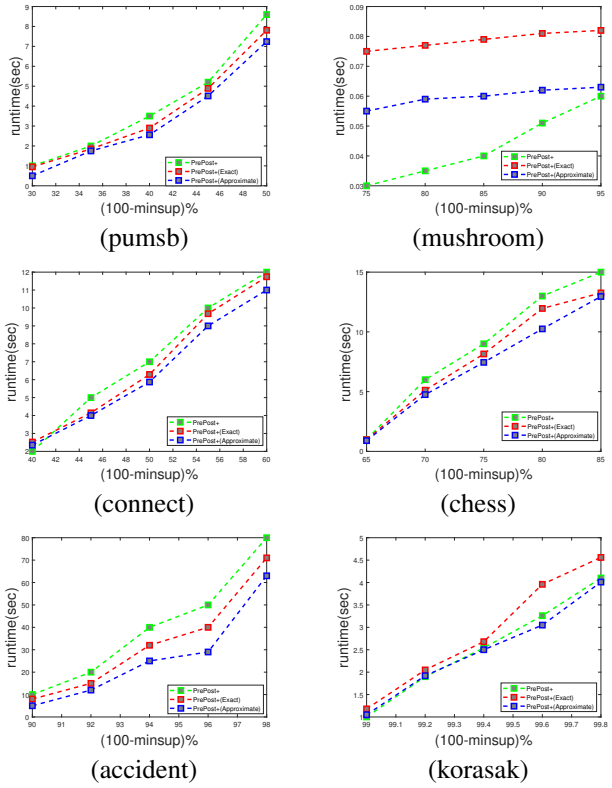

Figure 2. Runtime of the PrePost+ with and without the CBPM framework.

frequent itemset mining datasets[1]. The number of clusters are fixed to 20. Table 1 presents the characteristics of the datasets used in our experiments.

4.1. Runtime Performance

Figure 2 present the runtime performance of PrePost+ [4] with and without the CBPM framework for both the approximate and exact algorithms using different datasets and with different mining threshold. The results reveal that by reducing the mining threshold, PrePost+ benefits from the CBPM framework. These results are achieved thanks to the following factors: i) the decomposition method applied to the CBPM framework by minimizing the number of the shared items, and ii) solving the sub-problems with smaller number of transactions and smaller number of items,

1. http://fimi.uantwerpen.be/data/

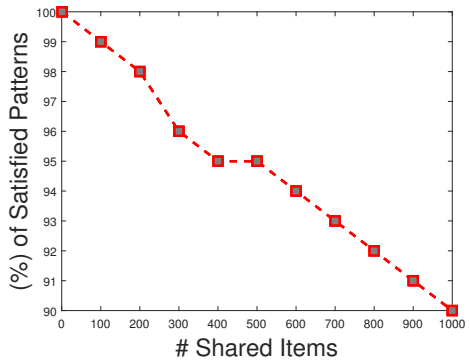

Figure 3. Ratio of the satisfied patterns using the approximate algorithm with the CBPM framework using IBM Synthetic Data Generator, and with different number of shared items

instead of dealing with the whole transaction database with all distinct items. An important side effect of this is also that memory consumption is significantly reduced.

4.2. Ratio of the Satisfied Patterns

This experiment evaluates the approximation-based algorithm proposed in this work. Note that in the pattern mining literature, there are many approximate algorithms exploiting the metaheuristics, including [3]. However, these approaches are not within the scope of this paper, since the main goal of this work is to show the effect of decomposition on the pattern mining algorithms. Figure 3 presents the ratio of the satisfied patterns using IBM Synthetic Data Generator for Itemsets[2]. Different transaction databases are generated. We varied the number of items from 100 to 100.000, and the number of transactions is fixed to 1.000.000. By varying the number of shared items from 1 to 1.000, the ratio of the satisfied patterns is reduced from 100% to 90%. These results reveal that the quality of the approximation-based algorithm depends on the clustering results, represented by the number of shared items. More efficient clustering algorithms could reduce the number of shared items, and consequently improve the accuracy of the approximate approach.

5. Conclusion

We have introduced a new intelligent pattern mining framework, called clustering-based pattern mining (CBPM). CBPM discovers relevant patterns by studying the correlation between the transaction database. The set of transactions are first partitioned using the k-means algorithm, where the high correlated transactions are grouped together. For each cluster of transactions, the pattern mining algorithm is launched in order to discover the relevant patterns. This is done by using one of two algorithms, one which is approximate and the other is exact. The experimental evaluation of the CBPM framework shows that the performance of compared to the basic PrePost+ algorithm is improved, and in particular when the search space is large. Motivated by the promising results shown in this paper, we plan to further improve the the performance of CBPM by exploiting potentials for parallelization in the approach, and developing better strategies to exploit the shared items.

2. https://github.com/zakimjz/IBMGenerator

Acknowledgment

This work was carried out at the Norwegian University of Science and Technology (NTNU), funded by a postdoctoral fellowship from the European Research Consortium for Informatics and Mathematics (ERCIM).

References

[1] B. Goethals, "Survey on frequent pattern mining," *Univ. of Helsinki*, vol. 19, pp. 840–852, 2003.

[2] Y. Xun, J. Zhang, X. Qin, and X. Zhao, "FiDoop-DP: data partitioning in frequent itemset mining on Hadoop clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 101–114, 2017.

[3] Y. Djenouri and M. Comuzzi, "Combining apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem," *Information Sciences*, vol. 420, pp. 1–15, 2017.

[4] Z.-H. Deng and S.-L. Lv, "PrePost+: an efficient n-lists-based algorithm for mining frequent itemsets via children–parent equivalence pruning," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5424–5432, 2015.

[5] C. C. Aggarwal and J. Han, *Frequent Pattern Mining*. Springer, 2014.

[6] P. Fournier-Viger, J. C.-W. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le, "A survey of itemset mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 4, p. e1207, 2017.

[7] Q.-H. Duong, P. Fournier-Viger, H. Ramampiaro, K. Norvaag, and T.-L. Dam, "Efficient high utility itemset mining using buffered utility-lists," *Applied Intelligence*, vol. 48, no. 7, pp. 1859–1877, 2017.

[8] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *ACM SIGMOD Record*, vol. 22, no. 2, 1993, pp. 207–216.

[9] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM SIGMOD Record*, vol. 29, no. 2, 2000, pp. 1–12.

[10] Y. Djenouri, D. Djenouri, J. C.-W. Lin, and A. Belhadi, "Frequent itemset mining in big data with effective single scan algorithms," *Ieee Access*, vol. 6, pp. 68 013–68 026, 2018.

[11] K.-W. Chon, S.-H. Hwang, and M.-S. Kim, "Gminer: A fast gpu-based frequent itemset mining method for large-scale data," *Information Sciences*, vol. 439, pp. 19–38, 2018.

[12] A. Allam, S. Skiadopoulos, and P. Kalnis, "Improved suffix blocking for record linkage and entity resolution," *Data & Knowledge Engineering*, vol. 117, pp. 98–113, 2018.

[13] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 169–178.

[14] T. De Vries, H. Ke, S. Chawla, and P. Christen, "Robust record linkage blocking using suffix arrays and bloom filters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 2, p. 9, 2011.

[15] M. Hadjieleftheriou, N. Koudas, and D. Srivastava, "Incremental maintenance of length normalized indexes for approximate string matching," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 429–440.

[16] Y. Djenouri, D. Djamel, and Z. Djenoouri, "Data-mining-based decomposition for solving MAXSAT problem: Towards a new approach," *IEEE Intelligent Systems*, 2017.