Applied Intelligence (2020) 50:2647–2662
https://doi.org/10.1007/s10489-020-01664-w

A general-purpose distributed pattern mining system

Asma Belhadi[1] · Youcef Djenouri[2,3] · Jerry Chun-Wei Lin[4] · Alberto Cano[5]

Published online: 18 March 2020

Abstract
This paper explores five pattern mining problems and proposes a new distributed framework called DT-DPM: Decomposition Transaction for Distributed Pattern Mining. DT-DPM addresses the limitations of the existing pattern mining problems by reducing the enumeration search space. Thus, it derives the relevant patterns by studying the different correlation among the transactions. It first decomposes the set of transactions into several clusters of different sizes, and then explores heterogeneous architectures, including MapReduce, single CPU, and multi CPU, based on the densities of each subset of transactions. To evaluate the DT-DPM framework, extensive experiments were carried out by solving five pattern mining problems (FIM: Frequent Itemset Mining, WIM: Weighted Itemset Mining, UIM: Uncertain Itemset Mining, HUIM: High Utility Itemset Mining, and SPM: Sequential Pattern Mining). Experimental results reveal that by using DT-DPM, the scalability of the pattern mining algorithms was improved on large databases. Results also reveal that DT-DPM outperforms the baseline parallel pattern mining algorithms on big databases.

Keywords Pattern mining · Decomposition · Distributed computing · Heterogeneous architecture

1 Introduction

Pattern mining is a data mining task that aims at studying the correlations within data and discovering relevant patterns from large databases. In practice, different database representation could be observed (from Boolean databases to sequence databases). The problem of pattern mining is to find an efficient approach to extract the relevant patterns in a database. It is used in many applications and domains such as ontology matching [1], process mining [2], decision making [3], and constraint programming [4]. The pattern mining is also called with "Big data" applications such as in frequent genes extractions from DNA in Bio-informatics [5], relevant hashtags from twitter streams in social network analysis [6], analysis of sensorial data from IoT devices in smart city applications [7]. This work mainly focuses on mining the information from big transactional databases.

1.1 Motivation

Solutions to pattern mining problems [8–12] are high time consuming when dealing with large and very large databases for pattern mining problems such as FIM and WIM, and they are totally inefficient when solving more complex problems such as UIM, HUIM, and SPM. To improve the runtime performance of the pattern mining approaches, many optimization and high performance computing techniques have been proposed [13–18]. However, these strategies are inefficient when dealing with big databases, where only few number of relevant patterns are useful and displayed to the end user. We contemplate that these algorithms are inefficient because they consider the whole database in the mining process. In our previous work [19], we proposed a new algorithm for pattern mining algorithm, where the aim is to study the correlation

✉ Asma Belhadi
abelhadi@usthb.dz

Youcef Djenouri
youcef.djenouri@sintef.no

Jerry Chun-Wei Lin
jerrylin@ieee.org

Alberto Cano
acano@vcu.edu

[1] Department of Computer Science, USTHB, Algiers, Algeria

[2] Department of Computer Science, NTNU, Trondheim, Norway

[3] SINTEF Digital, Forskningsveien 1, 0314 Oslo, Norway

[4] Department of Computing, Mathematics and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway

[5] Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA

between the input data to split the whole problem into many smaller sub-problems, but as being as independent as possible. We proposed a *k*-means algorithm to assign the transactions into different clusters. We also developed an efficient strategy to accurately explore the clusters of transactions. This approach gives good results compared to the baseline serial methods. However, it still suffers from the runtime and accuracy performance when dealing with big databases. This is due to the separator items between clusters, where the mining process of these items should be carried out by exploring the transactions of all clusters. This issue degrades the overall performance of such an approach. Motivated by the preliminary results reported in [19], we propose a new parallel framework, which addresses the following issues, i) minimizing the number of separator items, ii) improving the runtime and accuracy on big databases.

1.2 Contributions

In this research work, we propose a generic intelligent pattern mining algorithm for dealing pattern mining problems on big databases. It is a comprehensive extension of our previous work [19]. With this in mind, the main contributions of this work are as follows:

1. Propose a new framework called DT-DPM for improving pattern mining algorithms in a distributed environment.
2. Develop a decomposition approach to cluster the transactions set into smaller similar groups.
3. Extend the MapReduce computing framework to deal with the pattern mining algorithms by exploiting the different dependencies between the transactions of the clusters.
4. Five cases studies (FIM, WIM, UIM, HUIM, and SPM) have been analyzed on well-known pattern mining databases by considering five best pattern mining algorithms in terms of time complexity as baseline algorithms for the DT-DPM framework. Experimental results reveal that by using DT-DPM, the scalability of the pattern mining algorithms was improved on large databases. Results also reveal that DT-DPM outperforms the baseline parallel pattern mining algorithms on big databases.

1.3 Outline

The remainder of the paper is organized as follows: Section 2 introduces the basic concepts of pattern mining problems. Section 3 reviews existing pattern mining algorithms followed by a detailed explanation of our DT-DPM framework in Section 4. The performance evaluation is presented in Section 5 whereas Section 6 draws the conclusions.

2 Pattern mining problems

In this section, we first present a general formulation of pattern mining and then we present a few pattern mining problems according to the general formulation.

Definition 1 (pattern) Let us consider $I = \{1, 2, \ldots, n\}$ as a set of items where n is the number of items, and $T = \{t_1, t_2, dots, t_m\}$ as a set of transactions where m is the number of transactions. We define the function a, where for the item i in the transaction t_j, the corresponding pattern reads $p{=}\sigma(i,j)$.

Definition 2 (pattern mining) A pattern mining problem finds the set of all relevant patterns L, such as

$$L = \{p \mid Interestingness(T, I, p) \geq \gamma\} \tag{1}$$

where the $Interestigness(T, I, p)$ is the measure to evaluate a pattern p among the set of transactions T and the set of items I, where γ is the mining threshold.

From these two definitions, we present the existing pattern mining problems.

Definition 3 (Boolean database) We define a Boolean database by setting the function σ (see Def. 2) as

$$\sigma(i,j) = \begin{cases} 1 & \text{if } i \in t_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Definition 4 (frequent itemset mining (FIM)) We define a FIM problem as an extension of the pattern mining problem (see Def. 2) by

$$L = \{p \mid Support(T, I, p) \geq \gamma\} \tag{3}$$

with $Support(T, I, p) = \frac{|p|_{T,I}}{|T|}$ where T is the set of transactions in a Boolean database defined by Def. 1, γ is a minimum support threshold, and $|p|_{T,I}$ is the number of transactions in T containing the pattern p.

Definition 5 (weighted database) We define a weighted database by setting the function σ (see Def. 2) as

$$\sigma(i,j) = \begin{cases} w_{ij} & \text{if } i \in t_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Note that w_{ij} is the weight of the item i in the transaction t_j.

Definition 6 (weighted itemset mining (WIM)) We define a WIM problem as an extension of the pattern mining problem (see Def. 2) by

$$L = \{p \mid WS(T, I, p) \geq \gamma\} \tag{5}$$

with $WS(T,I,p) = \sum_{j=1}^{|T|} W(t_j,I,p)$ where T is the set of transactions in the weighted database defined by Def. 3, $W(t_j,I,p)$ is the minimum weight of the items of the pattern p in the transaction t_j, and γ is a minimum weighted support threshold.

Definition 7 (uncertain database) We define an uncertain database by setting the function σ (see Def. 2) as

$$\sigma(i,j) = \begin{cases} Prob_{ij} & \text{if } i \in t_j \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Note that $Prob_{ij}$ is the uncertainty value of i in the transaction t_j.

Definition 8 (uncertain itemset mining (UIM)) We define a UIM problem as an extension of the pattern mining problem (see Def. 2) by

$$L = \{p \mid US(T,I,p) \geq \gamma\} \quad (7)$$

with $US(T,I,p) = \sum_{j=1}^{|T|} \prod_{i \in p} Prob_{ij}$ where T is the set of transactions in the uncertain database defined by Def. 5 and γ is the minimum uncertain support threshold.

Definition 9 (utility database) We define an utility database by setting the function σ (see Def. 2) as

$$\sigma(i,j) = \begin{cases} iu_{ij} & \text{if } i \in t_j \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Note that iu_{ij} is the internal utility value of i in the transaction t_j, we also define external utility of each item i by *eu(i)*.

Definition 10 (high utility itemset mining (HUIM)) We define a HUIM problem as an extension of the pattern mining problem (see Def. 2) by

$$L = \{p \mid U(T,I,p) \geq \gamma\}, with\ U(T,I,p)$$
$$= \sum_{j=1}^{|T|} \sum_{i \in p} iu_{ij} \times eu(i) \quad (9)$$

where T is the set of transactions in the utility database defined by Def. 7 and γ is the minimum utility threshold.

Definition 11 (sequence database) We assume a total order on items $\prec$, such as $1 \prec 2 \prec 3 \ldots \prec n$. A sequence is an ordered list of itemsets $s = \{I_1, I_2, \ldots, I_{|s|}\}$. Each itemset I_i is defined by setting the function σ (see Def. 2) as $\sigma(i,j) = i$, if $i \in t_j$

Definition 12 (sequential pattern mining (SPM)) We define a SPM problem as an extension of the pattern mining problem (see Def. 2) by

$$L = \{p \mid Support(T,I,p) \geq \gamma\} \quad (10)$$

where T is the set of transactions in the sequence database defined by Def. 9 and γ is the minimum support threshold.

3 Related work

Pattern mining has been largely studied in the last three decades [8–11, 20, 21]. There are many variants of pattern mining problem such as FIM, WIM, HUIM, UIM and SPM.

FIM It aims at extracting all frequent itemsets that exceed the minimum support threshold. Apriori [22] and FP-Growth [23] are the most popular algorithms. Apriori applies a *generate and test* strategy to explore the itemset space. The candidate itemsets are generated incrementally and recursively. To generate *k*-sized itemsets as candidates, the algorithm calculates and combines the frequent *(k-1)*-sized itemsets. This process is repeated until no candidate itemsets are obtained in an iteration. However, FP-Growth adopts a *divide and conquer* strategy and compresses the transactional database in the volatile memory using an efficient tree structure. It then applies recursively the mining process to find the frequent itemsets. The main limitation of the traditional FIM algorithms is the database format where only binary items can be mined. A typical application of this problem is the market basket analysis, which a given item (product) may be present or absent in the given transaction (customer).

WIM To address the FIM limitation, WIM is introduced, where weights are associated to each item to indicate their relative importance in the given transaction [24]. The goal is to extract itemsets exceeding minimum weight threshold. The first WIM algorithm is called WFIM: Weighted Frequent Itemset Mining [25]. It defines a weight range and a minimum weight constraint into the FP-Growth algorithm. Both weight and support measures are considered to prunethe search space. Yun [26] proposed WIP: Weighted Interesting Pattern. It introduces an infinity measure that determines the correlation between the items of the same pattern. The integration of the WIM in both Apriori and FP-Growth is studied in [27]. The results showed that FP-Growth outperforms Apriori for mining weighted patterns. Le et al. [28] proposed a frequent subgraph algorithm on a weighted large graph. A novel strategy is developed which aims to compute the weight of all candidate subgraphs. An efficient pruning strategy aims at reducing both the processing time and the memory usage. Lee et al. [29] mine the frequent weighted itemsets by employing a novel type of prefix tree structures. This allows to retrieve the relevant patterns more accurately without saving the list of identification number of the different transactions.

UIM An extension of WIM, called UIM, explores uncertain transactional databases, where two models (expected-support and probabilistic itemsets) are defined to mine uncertain patterns. Li et al. [30] proposed the PFIMoS: Probabilistic Frequent Itemset Mining over Streams algorithm. It derives the probabilistic frequent itemsets in an incremental way by determining the upper and the lower bounds of the mining threshold. Lee et al. [31] introduced the U-WFI: Uncertainmining of Weighted Frequent Itemsets algorithm. It allows to discover from a given uncertain database relevant uncertain frequent itemsets with high probability values by focusing on item weights. Liaqat et al. [32] show the use of uncertain frequent patterns in the image retrieval process. It incorporates the fuzzy ontology and uncertain frequent pattern mining for finding the relevant images regarding the user query. Lee et al. [33] suggest novel data structures to guarantee the correctness of the mining outputs without any false positives. It allows to retrieve a complete set of uncertain relevant patterns in an reasonable amount of time.

HUIM High Utility Itemset Mining is an extension of WIM where both internal and external utilities of the items are involved. The aim is to find all high utility patterns from transactional database that exceed the minimum utility threshold. The utility of a pattern is the sum of the utility of all its items, where the utility of an item is defined by the product by its internal and external utility values. Chan et al. [34] proposed the first HUIM algorithm. It applies the Apriori-based algorithm to discover top k high utility patterns. This algorithm suffers from the runtime performance, because the search space is not well pruned using the closure downward property. Thus, the utility measure is neither monotone nor anti-monotone.To address this limitation the TWU: Transaction Weighted Utility property is defined to prune the high utility pattern space [35, 36]. It is an upper-bound monotone measure to reduce the search space. More efficient HUIM algorithms based on TWU have been recently proposed such as EFIM: EFficient high-utility Itemset Mining [37], and d^2HUP: Direct Discovery for High Utility Patterns [38]. The particularity of such approaches is that they use more efficient data structures to determine the TWU and the utility values. Singh et al. [39] address the problem of the minimum utility threshold tuning and derived the top k high utility patterns. It uses transaction merging and data projection techniques to reduce the data scanning cost. It also develops an intelligent strategy designed for top k high utility patterns to prune the enumeration search tree. Gan et al. [40] proposed a correlated high utility pattern miner algorithm. It considers the positive correlation, the profitable value concepts, and several strategies to prune the search space. Lee et al. [41] developed an efficient incremental approach for identifying high utility patterns. It adopts an accurate data structure to mine high utility patterns in an incremental way.

SPM Sequential Pattern Mining is an extension of FIM to discover a set of ordered patterns in a sequence database [42–44]. Salvemini et al. [42] find the completeset of the sequence patterns by reducing the candidates generation runtime by employing an efficient lexicographic tree structure. Fumarola et al. [43] discover closed sequential patterns using two main steps, i) Finding the closedof sequence patterns of size 1, and ii) Generating new sequences from the sequence patterns of size 1, already deduced in the first step. Van et al. [44] introduced the pattern-growth algorithm in solving the sequential pattern mining problem with itemset constraints. It proposed an incremental strategy to prune the enumeration search tree which allows to reduce the number of visited nodes. Aisal et al. [45] proposed a novel convoy pattern mining approach which can operateon a variety of operational data stores. It suggested a new heuristic to prune the objects which have no chance of forming a convoy. Wu et al. [46] solved the contrast sequential pattern mining problem, which is an extension of SPM, discovered all relevant patterns that figure out in one sequence data and not in the others. These patterns are highly used in some specified application such as analysing anomalous customers in the business intelligence or medical diagnosis in the smart healthcare [47–49].

High performance computing Regarding high performance computing, many algorithms have been developed for boosting the FIM performance [15, 50–54]. However, few algorithms have been proposed for the other pattern mining problem [16–18, 55]. In [52], some challenges in big data analytics are discussed, such as mining evolving data streams and the need to handle many exabytes of data across various application areas such as social network analysis. The BigFIM: Big Frequent Itemset Mining [56] algorithm is presented, which combines principles from both Apriori and Eclat. BigFIM is implemented using the MapReduce paradigm. The mappers are determined using Eclat algorithm, whereas, the reducers are computed using the Apriori algorithm. [57] develops two strategies for parallelizing both candidate itemsets generation and support counting on a GPU (Graphic Processing Unit). In the candidate generation, each thread is assigned two frequent *(k-1)*-sized itemsets, it compares them to make sure that they share the common *(k-2)* prefix and then generates a k-sized candidate itemset. In the evaluation, each thread is assigned one candidate itemset and counts its support by scanning the transactions simultaneously. The evaluation of frequent itemsets is improved in [58] by proposing mapping and sum reduction techniques to merge all counts ofthe given itemsets. It is also improved in [59] by developing three strategies for minimizing the impact of the graphical processor unit thread divergence. In [60], a multi-level layer data structure is proposed to enhance the support counting of the frequent itemsets. It divides vertical data into several layers, where each layer is an index table of the next

layer. This strategy can completely represent the original vertical structure. In a vertical structure, each item corresponds to a fixed-length binary vector. However, in this strategy, the length of each vector varies, which depends on the number of transactions included in the corresponding item. A Hadoop implementation based on MapReduce programming approach called FiDoop: Frequent itemset based on Decomposition is proposed in [61] for the frequent itemsets mining problem. It incorporates the concept of FIU-tree (Frequent Itemset Ultrametric tree) rather than traditional FP-tree of FP-Growth algorithm, for the purpose of improving the storage of the candidate itemsets. An improved version called FiDoop-DP is proposed in [15]. It develops an efficient strategy to partition data sets among the mappers. This allows better exploration of cluster hardware architecture by avoiding jobs redundancy. Andrzejewski et al. [62] introduce the concept of incremental co-location patterns, i.e., update the set of knowledge about the spatial features after inserting new spatial data to the original one. The authors develop a new parallel algorithm which combines effective update strategy and multi GPU co-location pattern mining [63] by designing an efficient enumeration tree on GPU. Since the proposed approach is memory-aware, i.e., the data is divided into several package to fit to GPU memories, it only achieves an speedup of six. Jiang et al. [64] adopt a parallel FP-Growth for mining world ocean atlas data. The whole data is partitioned among multiple CPU threads, where each thread explores 300,000 data points and derives correlation and regularities of oxygen, temperature, phosphate, nitrate, and silicate in the ocean. The experimental results reveal that the suggested adaptation only reachesan speedup of 1.2. Vanhalli et al. [65] developed a parallel row enumerated algorithm for mining frequent colossal closed patterns from high dimensional data. It first prunes the whole data by removing irrelevant items and transactions using rowset cardinality table, which determines the closeness of each subset of transactions. In addition, it uses a hybrid parallel bottom-up bitset based approach to enumerate the colossal frequent closed patterns. This approach is fast, howeverit suffers from the accuracy issue, which may ignore some relevant patterns due to the preprocessing phase. It also requires additional parameter to be fixed, represented by a cardinality threshold. Yu et al. [66] propose the parallel version of PrefixSpan on Spark: PrefixSpan-S. It optimizes the overhead by first loading the data from the Hadoop distributed file system into the RDDs: Resilient Distributed Datasets, and then reading the data from RDDs, and save the potential results back into the RDDs. This approach reaches a good performance with a wise choice of the minimum support threshold. However, it is very sensitive to the data distribution. Kuang et al. [67] proposed the parallel implementation of FP-Growth algorithm in Hadoop by removing the data redundancy between the different data partitions, which allows to handle the transactions in a single pass. Sumalatha et al. [68] introduces the concept of distributed temporal high utility sequential patterns, and propose an intelligent strategy by creating a time interval utility data structure for evaluating the candidate patterns. The authors also defined two utility upper bounds, remaining utility, and co-occurrence utility to prune the search space.

To improve the runtime performance of the pattern mining approaches, several strategies have been proposed using metaheuristics, specifically exploiting evolutionary and/or swarm intelligence approaches [13, 14, 69, 70]. However, these optimizations are inefficient when dealing with large and big transactional databases where only few number of interesting patterns are discovered. To deal with this challenging issue, the next section presents a new framework, which investigates both decomposition techniques and distributed computing in solving pattern mining algorithms.

4 DT-DPM: decomposition transaction for distributed pattern mining

This section presents the DT-DPM (Decomposition Transaction for Distributed Pattern Mining) framework, that integrates the DBSCAN: Density-Based Spatial Clustering of Applications with Noise algorithm and distributed computing represented by MapReduce, CPU multi-cores and Single CPU for solving pattern mining problems. As seen in Fig. 1, the DT-DPM framework uses heterogeneous distributed computing and decomposition techniques for solving pattern mining problems. Detailed explanation of the DT-DPM framework, step by step, is given in the following.

4.1 DBSCAN

The aim of this step is to divide a database into a collection of homogeneous groups using decomposition techniques, where each group shares entries highly correlated, i.e., the database entries of each group share maximum number of items compared to the entries of the other groups.

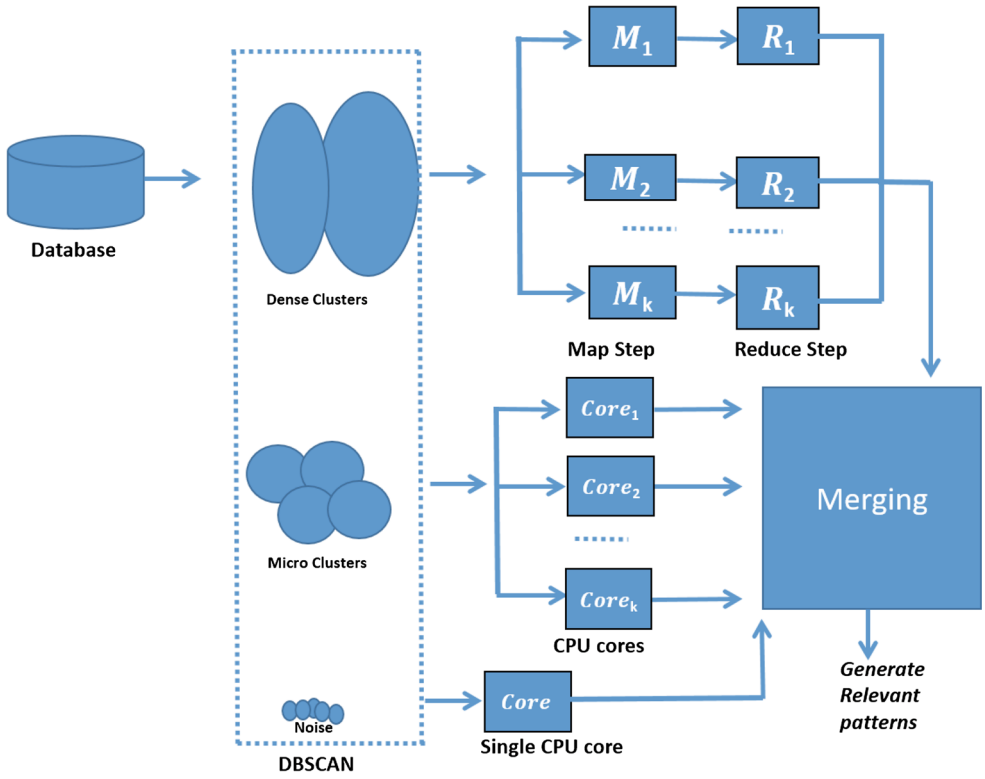
Definition 13 A database $\mathcal{D}$ is decomposed into several groups $G = \{G_i\}$, where each group G_i is subset of rows in $\mathcal{D}$ such as $G_i \cup G_j = \varnothing$. We define, $\mathcal{I}(G_i)$, the set of items of the group G_i by

$$\mathcal{I}(G_i) = \left\{ \cup \mathcal{I}\left(\mathcal{D}_j\right) / \mathcal{D}_j \in G_i \right\} \tag{11}$$

Proposition 1 Suppose that the groups in G share any items which means

$$\forall \left(G_i, G_j\right) \in G^2, \forall i \neq j, \mathcal{I}(G_i) \cap \mathcal{I}\left(G_j\right) = \varnothing \tag{12}$$

Fig. 1 DT-DPM framework



We have the following proposition

$$L = \left\{ \bigcup_{i=1}^{k} L_i \right\} \tag{13}$$

Note that $\mathcal{P}_i$ is the set of the relevant patterns of the group $\mathcal{G}_i$.

From the above proposition, one may argue that if the whole transactions in the database are decomposed in such a way, the independent groups will be derived. It means that, any group of transactions share items with any other group, and therefore, the groups could be solved separately. Unfortunately, such case is difficult to realize, as many dependencies may be observed between rows. The aim of the decomposition techniques is to minimize the separator items between the groups such as

$$G^* = G\arg\min \left| \cup \left(\mathcal{I}(G_i) \cap \mathcal{I}(G_j) \right) \right| \tag{14}$$

The aim of the decomposition step is to minimize the shared items between the different clusters, these shared items are called *Separator Items*. More formally, this decomposition generates a labeled non-directed graph weighted noted $G = < C, S >$ where C is the set of nodes formed by the clusters of G and S is the set of the separator items. Each element s^{ij} in S contains two components: i) s_l^{ij}, is the label of the element s^{ij}, it is represented by the set of items shared by the clusters C_i and C_j. ii)s_w^{ij} is the weight of the element s^{ij}, it is represented by the number of items shared by the cluster C_i and C_j. As a result, k disjoint partitions $P = \{P_1, P_2, \ldots, P_k\}$, where $P_i \cap P_j = \varnothing$, $\forall (i,j) \in [1, \ldots, k]^2$ and $\cup_{i=1}^{k} P_i = T$. The partitions are constructed by minimizing the following function

$$\sum_{i=1}^{k} \left(\sum_{j=1}^{k} \left(\sum_{l=1}^{|P_i|} \left(2^{\mathcal{G}_i^l} - 1 \right) - \sum_{l=1}^{|P_j|} \left(2^{\mathcal{G}_j^l} - 1 \right) \right) \right) \tag{15}$$

Solving this equation by exact solver requires high computational time. One way to solve this issue is to use clustering algorithms [71]. The adaptation of the DBSCAN [72] clustering algorithms has been investigated in this research. Before this, some definitions are given as follows.

Definition 14 (transaction-based similarity) We define Transaction-based similarity by an adopted Jaccard similarity [73] as

$$\mathcal{J}_{\mathcal{D}}(\mathcal{D}_i \mathcal{D}_j) = \frac{\sum_{x \in (\mathcal{D}_i \cap \mathcal{D}_j)} Sim(\mathcal{D}_i, \mathcal{D}_j, x)}{|\mathcal{D}_i| + |\mathcal{D}_j| + \sum_{x \in (\mathcal{D}_i \cap \mathcal{D}_j)} Sim(\mathcal{D}_i, \mathcal{D}_j, x)} \tag{16}$$

$$Sim(\mathcal{D}_i, \mathcal{D}_j, x) = \begin{cases} 1 & \text{if } x^i = x^j \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Note that x^i is the value of the variable x in the row data $\mathcal{D}_i$.

Definition 15 (centroids) We define the centroids of the group of rows data G_i, noted G_i by

$$G_i = \left\{ \cup \max_{x_l} \left(x_l^i\right) | x_l \in \mathcal{X}(G_i) \right\} \quad (18)$$

where $\max_{x_l} \left(x_l^i\right)$ is the most value of the variable x_l in the group G_i.

Definition 16 (clause neighborhoods) We define the neighborhoods of a row data $\mathcal{D}_i$ for a given threshold ϵ, noted $\mathcal{N}_{\mathcal{D}_i}$ by

$$\mathcal{N}_{\mathcal{D}_i} = \left\{ \mathcal{D}_j | \mathcal{J}_{\mathcal{D}}\left(\mathcal{D}_i \mathcal{D}_j\right) \geq \epsilon \vee j \neq i \right\} \quad (19)$$

Definition 17 (core data) A row data $\mathcal{D}_i$ is called core data if there is at least the minimum number of rows data σ_D such as $|\mathcal{N}_{\mathcal{D}_i}| \geq \sigma_D$

Algorithm 1 presents the pseudo-code of the decomposition for the rows data. The process starts by checking the ϵ-neighborhood of each transaction. The core transactions are determined and then iteratively collects density-reachable transactions from these core transactions directly, which may involve merging a few density-reachable clusters. The process terminates when no new transactions can be added to any cluster. The output of the decomposition step is a labeled non-directed graph weighted noted $O = < G, S >$ where G is the set of nodes formed by the groups of the rows data and S is the set of the separator items. Each element s^{ij} in S contains two components: i) s_l^{ij}, is the label of the element s^{ij}, it is represented by the set of variables by the groups G_i and G_j. ii)s_w^{ij} is the weight of the element s^{ij}, it is represented by the number of variables shared by the group G_i and G_j.

Algorithm 1 Database Decomposition via DBSCAN

```
1: Input:
   D = {D_1, D_2...D_m}: m rows data
   X = {x_1, x_2...x_n}: The set of n variables
   σ_D, ϵ: DBSCAN threshold.
2: Output:
   O =< G, S > such as
   G = {G_i}: The set of groups of rows data, and
   S: The set of separators variables.
3: G_outlier ← ∅
4: i ← 0
5: for each clause D_i do
6:    if |N_Di| ≤ σ_D then
7:       G_outlier ← G_outlier ∪ {D_i}
8:    end if
9:    i ← i + 1
10:   G_i ← {D_i}
11:   for D_j ∈ N_Di do
12:      Go to 25
13:   end for
14: end for
15: S ← ∅
16: for each G_i do
17:   for j>i do
18:      S ← S ∪ (X(G_i) ∩ X(G_j))
19:   end for
20: end for
21: return O
```

4.2 Mining process

After applying the DBSCAN algorithm on the whole data, the next step is to solve each resulted cluster independently using heterogeneous machines such as supercomputers, CPU cores, and single CPU core. The question that should be answered now is, which cluster should be assigned to which machine? We know that supercomputers are most powerful than CPU cores and single CPU. Obviously, clusters with high workload are assigned to supercomputer, the remaining clusters are assigned to CPU cores, and the noise data are assigned to a single CPU. This raises the following proposition,

Proposition 2 Consider a function Cost($\mathcal{A}$, m, n), that computes the complexity cost of the given pattern mining algorithm $\mathcal{A}$. Consider the output of the DBSCAN algorithm $O = < G, S >$, the mining process by $\mathcal{A}$ of the clusters could be performed in parallel by using heterogeneous machines according to the density of each cluster G_i represented by $(|G_i|, \mathcal{I}(G_i))$, and with a complexity cost threshold μ_{cost} as

1. If Cost ($\mathcal{A}$, $|G_i|$, $\mathcal{I}(G_i)$) $\geq \mu_{cost}$ then send G_i to MapReduce framework.

2. If Cost ($\mathcal{A}$, $|G_i|$, $\mathcal{I}(G_i)$) < μ_{cost} then send G_i to CPU cores.

Map and reduce Each mapper $\mathcal{M}_i$ first recuperates the assigned cluster G_i. It then processes the transactions of the cluster G_i, and applies the mining process for each transaction. Gradually, it creates a set of candidate patterns $\mathcal{C}_i$. When $\mathcal{M}_i$ scans all transactions of the cluster G_i, it sends $\mathcal{C}_i$ to the reducer $\mathcal{R}_i$. The reducer $\mathcal{R}_i$ scans the candidate patterns $\mathcal{C}_i$, and computes the local support of each pattern belong to $\mathcal{C}_i$. This allows to create the local hash table $\mathcal{LH}_{MR_i}$.

CPU cores The CPU cores solve clusters having complexity cost less than μ_{cost}, where each CPU core deal with one cluster of transactions, where it applies sequentially the given pattern mining algorithm $\mathcal{A}$ on the assigned cluster. The result is stored in the local hash table called $\mathcal{LH}_{CPU_i}$.

Single CPU The noise transactions resulted by applying DBSCAN are solved separately in a single CPU. The generated noises are merged into a small transactions, where the sequential process is applied on it. Again, as result, a local hash table called $\mathcal{LH}_{noise}$ is stored

Merging The merging step is then performed to determine the global support of all patterns and extract all relevant patterns from the global hash table $\mathcal{GH}$. This step considers the set of separator items S as wellas the clusters in the mining process. This allows to discover all relevant patterns from the whole transactions. The possible candidate patterns are then generated from the separator items. For each generated pattern, the aggregation function (see Definition 2) is then used to determine the interestingness of this pattern in the whole transactional database. Note that, the interestingness depends on the problem. For instance, if we are interested to deal with a frequent itemset mining problem, the interestingness function should be the support measure. The relevant patterns of the separator items are then concatenated with the local hash tables $\mathcal{LH}$ to derive the global relevant patterns of the whole transactional database, noted $\mathcal{GH}$.

Definition 18 Let us define an aggregation function of the pattern p in the clusters of the transactions C by

$$\mathcal{A}(p) = \sum_{i=1}^{k} Interestingness(G_i, \mathcal{I}(G_i), p) \tag{20}$$

DT-DPM improves the sequential version of the pattern mining algorithms by exploiting heterogeneous machines while generating and evaluating the candidate patterns. The load balancing is automatically conducted since the transactions are assigned to the workers using decomposition step. Workers including (mappers, CPU cores, and single CPU) can process transactions independently and stored the results into the local hash tables. The merging step is finally performed on the local hash tables and extract and extract the set of all frequent patterns by dealing with the separator items using the aggregation function.

5 Performance evaluation

Extensive experiments were carried out to evaluate the DT-DPM framework. Five case studies were investigated by considering the FIM, WIM, UIM, HUIM, and SPM problems. DT-DPM is integrated on the SPMF data mining library [74]. It offersmore than 150 pattern mining algorithms. DT-DPM java source code is integrated on the five best pattern mining algorithms in terms of time complexity [8]: i) frequent itemset mining: PrePost+ [75], ii) weighteditemset mining: WFIM [27], iii) uncertain high utility itemset mining: U-Apriori [76], iv) high utility itemset mining: EFIM [37], and v) sequential pattern mining: FAST [42]. All experiments were run on a computer with an Intel core i7 processor running Windows 10 and 16 GB of RAM.

5.1 Dataset description

Experiments were run on well-known pattern mining large and big databases. Table 1 presents the characteristics of the large databases used in the experiments. Moreover, three very large databases have been used to evaluate the patternmining approaches:

1. **Webdocs**[1] It is created from a set of links among *html* documents available on the Web. It contains 1,692,082 transactions with 526,765 different items. The maximum number of items per transactions of this database is 70,000 with 1.48 GB as database size [77].
2. **Twitter**[2] It is dataset related to user tweets. It contains 41.7 million user profiles, 1.47 billion social relations, 4,262 trending topics, and 106 million tweets [78].
3. **NewYorkTimes**[3] It consists of over 1.8 million newspaper articles published among twenty years.

In addition, an IBM Synthetic Data Generator for Itemsets and Sequences[4] is used to generate big databases of different number of items and different number of transactions.

[1] http://fimi.ua.ac.be/data/webdocs
[2] http://an.kaist.ac.kr/traces/WWW2010.html
[3] http://corpus.nytimes.com/
[4] https://github.com/zakimjz/IBMGenerator

Table 1 Large databases

Problem	Data set Type	Data set Name	$\|D\|$	$\|I\|$	Avg. Size/ Avg. Seq. Size
	Dense	Accident	340,183	468	33.8
	Dense	Chess	3196	75	37.0
	Dense	Connect	67,557	129	43.0
FIM, WIM,	Dense	Mushroom	8124	119	23.0
UIM, HUIM	Dense	Pumsb	49,046	2113	74.0
	Very Sparse	Korasak	990,000	41,270	8.1
	Sparse	Foodmart	4141	1559	4.4
	Very Sparse	Chainstore	1,112,949	46,086	7.2
	Book	Leviathan	5834	9025	33.81
SPM	Language	Sign	730	267	51.99
	Protein	Snack	163	20	60
	Web Stream	FIFA	20,450	2990	34.74

5.2 Decomposition performance

The first experiment aims at evaluating the clustering step. Several tests have been performed on the *k*-means algorithm to fix the number of clusters *k*. Figures 2 and 3 present both the quality of obtained clusters measures by the percentage of separator items and the runtime performance in seconds using the twelve databases mentioned above. By varying with the number of clusters from 2 to 20, the percentage of the separator items is reduced. Thus, when tk is set to 2, the percentage of the separator items exceeds 50%, while this percentage does not reach 1% when setting the number of clusters is set to 20 for almost databases. Moreover, by increasing the number of clusters, the runtimeincreases linearly, where it does not reach 2.50 seconds for all databases. As a result, we fix the number of clusters to 20 for the remaining of the experimentation. Given such results we can conclude that the preprocessing step does not influence on the overall performance of the DT-DPM framework, in particular for very large and big databases.

Figure 3 presents the runtime performance of the pattern mining algorithms with and without the DT-DPM framework for both strategies (approximate and exact) using different databases and with different mining thresholds. Experimental results reveal that by reducing the mining threshold and increasing the complexity of the problem solved, the pattern mining algorithms benefits from the DT-DPM framework. Therefore, for a small value of mining threshold and for more complex problem like UIM, HUIM or SPM, the approximate-based and exact-based strategies outperform the original pattern mining algorithms. For instance, when the minimum utility threshold is set to *1600 K*, the runtime of original EFIM and EFIM using the DT-DPM framework is 1 s on Connect database. However, when setting the minimum utility to *1000 K*, the runtime of original EFIM exceeds 8,000 seconds, and the runtime of EFIM with DT-DPM framework does not reach 1,500 seconds. These results are obtained thanks to many factors: i) the decomposition method applied in the DT-DPM framework by minimizing the number of separator items, ii) solving sub-problems with small number of transactions and small number of items, instead of dealing the whole transactional database with the whole distinct items, and iii) the ability of the pattern mining algorithms to be integrated with the DT-DPM framework.

5.3 Speedup of DT-DPM

Table 2 presents the speedup of the pattern mining algorithms with and without using the DT-DPM framework on the large databases. The results reveal that by increasing the number of mappers and the increasing the complexity of the problem solved, the speedup of the pattern mining algorithms benefits from the DT-DPM framework. Thus, for a large number of mappers, and for a more complex problem like UIM, HUIM or SPM, the mining process with DT-DPM outperform the

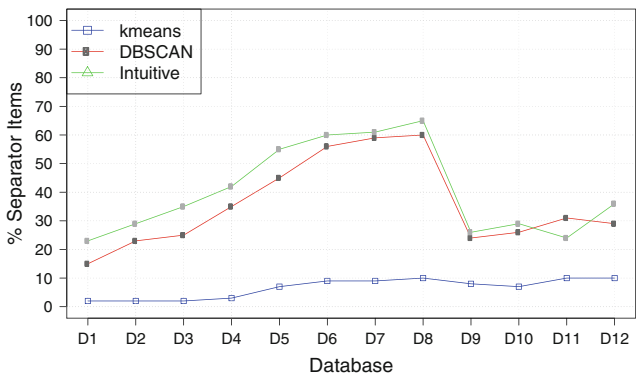


Fig. 2 Percentage(%) of separator items of the decomposition using different clustering algorithms on the following databases: D1: pumsb, D2: mushroom, D3: connect, D4: chess, D5: accident, D6: korasak, D7: foodmart, D8: chainstore, D9: leviathan, D10: sign, D11: snack, and D12: FIFA

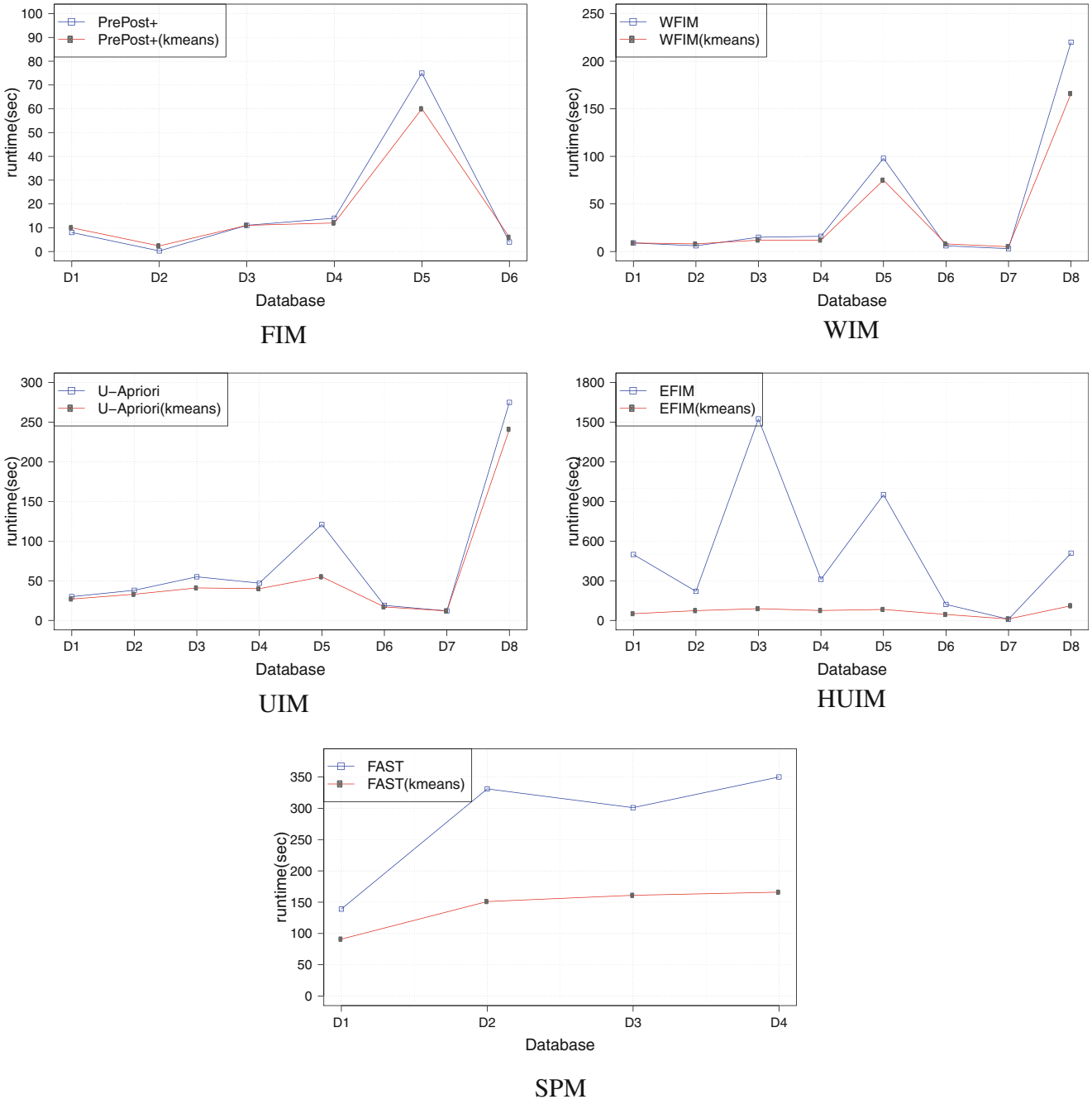


Fig. 3 Runtime (seconds) of pattern mining algorithm without and with decomposition step on the following databases: D1: pumsb, D2: mushroom, D3: connect, D4: chess, D5: accident, D6: korasak, D7: foodmart, D8: chainstore, D9: leviathan, D10: sign, D11:snack, and D12: FIFA

original pattern mining algorithms. For instance, when the number of mappers set to 2, the speedup of the original PrePost+ and PrePost using the DT-DPM framework is less than 5 in the pumsb database. However, by setting the number of mappers to 32, the speedup of the original EFIM does not reach 8630, and the speedup of the EFIM with DT-DPM framework exceeds 800 on Connect database. These results are achieved thanks to the following factors, i) The decomposition method applied to the DT-DPM framework by minimizing the number of theseparator items, and ii) Solving the sub-problems with small number of transactions and small number of items using the Mapreduce architecture.

5.4 DT-DPM Vs state-of-the-art algorithms

Figures 4 presents the speedup of DT-DPM against the baseline pattern mining algorithms (FiDoop-DP [15] for FIM, PAWI: Parallel Weighted Itemsets [16] for WIM,

Table 2 Speedup of pattern mining algorithms with and without using DT-DPM framework using different mappers (2, 4, 8, 16, 32)

Problem	Database	Without DT-DPM					With DT-DPM				
Mappers		*2*	*4*	*8*	*16*	*32*	*2*	*4*	*8*	*16*	*32*
	pumsb	2	3	7	11	34	**5**	**9**	**15**	**19**	**35**
	mushroom	3	8	10	13	36	**6**	**11**	**19**	**22**	**39**
FIM:	connect	5	10	12	15	39	**10**	**15**	**19**	**26**	**45**
PrePost+	chess	9	12	14	16	41	**13**	**18**	**23**	**29**	**52**
	accident	12	16	20	22	45	**19**	**23**	**29**	**36**	**66**
	korasak	2	2	5	9	11	**3**	**4**	**6**	**10**	**14**
	pumsb	4	8	11	14	37	**9**	**12**	**19**	**22**	**41**
	mushroom	7	10	12	15	39	**11**	**15**	**23**	**28**	**49**
	connect	11	14	18	21	45	**19**	**23**	**29**	**35**	**57**
WIM:	chess	13	18	23	29	49	**23**	**31**	**39**	**44**	**62**
WFIM	accident	29	35	52	59	68	**35**	**47**	**63**	**74**	**88**
	korasak	2	4	6	8	13	**5**	**8**	**10**	**13**	**19**
	foodmart	3	5	9	11	14	**6**	**9**	**12**	**15**	**22**
	chainstore	33	39	55	66	75	**36**	**51**	**66**	**79**	**95**
	pumsb	3	9	12	16	39	**11**	**13**	**18**	**23**	**43**
	mushroom	5	8	10	14	33	**7**	**8**	**15**	**23**	**42**
	connect	15	17	21	25	48	**23**	**29**	**36**	**45**	**59**
UIM:	chess	16	19	25	33	52	**18**	**26**	**31**	**46**	**72**
WFIM	accident	30	39	55	66	75	**42**	**52**	**69**	**79**	**90**
	korasak	3	7	10	12	18	**6**	**12**	**19**	**25**	**32**
	foodmart	4	7	12	16	20	**9**	**14**	**21**	**29**	**38**
	chainstore	41	55	67	81	99	**57**	**66**	**81**	**98**	**117**
	pumsb	51	76	91	105	133	**69**	**91**	**111**	**142**	**163**
	mushroom	32	49	60	71	82	**36**	**55**	**71**	**89**	**101**
	connect	161	245	367	415	629	**254**	**338**	**459**	**510**	**801**
HUIM:	chess	44	59	80	85	91	**49**	**60**	**88**	**96**	**104**
EFIM	accident	79	127	163	198	300	**121**	**201**	**293**	**368**	**411**
	korasak	23	35	44	60	68	**28**	**47**	**52**	**73**	**86**
	foodmart	8	15	27	33	43	**15**	**32**	**44**	**53**	**63**
	chainstore	48	72	88	94	100	**58**	**84**	**106**	**127**	**135**
	leviathan	19	36	52	82	103	**32**	**44**	**69**	**99**	**125**
SPM:	sign	41	78	93	105	119	**55**	**86**	**118**	**133**	**151**
FAST	snack	39	70	90	99	108	**50**	**78**	**110**	**145**	**173**
	FIFA	60	92	117	136	175	**90**	**109**	**142**	**178**	**213**

The values in bold represent the best values obtained during the the experiment

MRGrowth: MapReduce for FP-Growth [17] for UIM, PHI-Miner: Parallel High utility Itemset Miner [55] for HUIM, and MG-FSM: Mind the Gap for Frequent Seauence Mining [18] for SPM) using the big databases, and setting the mining threshold to10% for Webdocs, 5% for Twitter and 2% for NewYorkTimes database. The results reveal that by increasing the percentage of transactions from 5% to 100% and increasing the complexity of the problem solved, DT-DPM outperformed the baseline pattern mining algorithms and benefits from the intelligent partition of the transactional database and the intelligent mapping of the different clusters into the mapper nodes. For instance, the speedup of our framework is 701 on Webdocs whereas the speedup of FiDoop-DP is 521. On the other hand, for mining the speedup of our framework on NewYorkTimes is 1325 whereas the speedup of MG-FSM does not exceed 900.

5.5 Results on big databases

Figures 5 present the runtime of DT-DPM and both baseline MapReduce based models using big databases for solving both Itemset and Sequential Pattern Mining problems. The

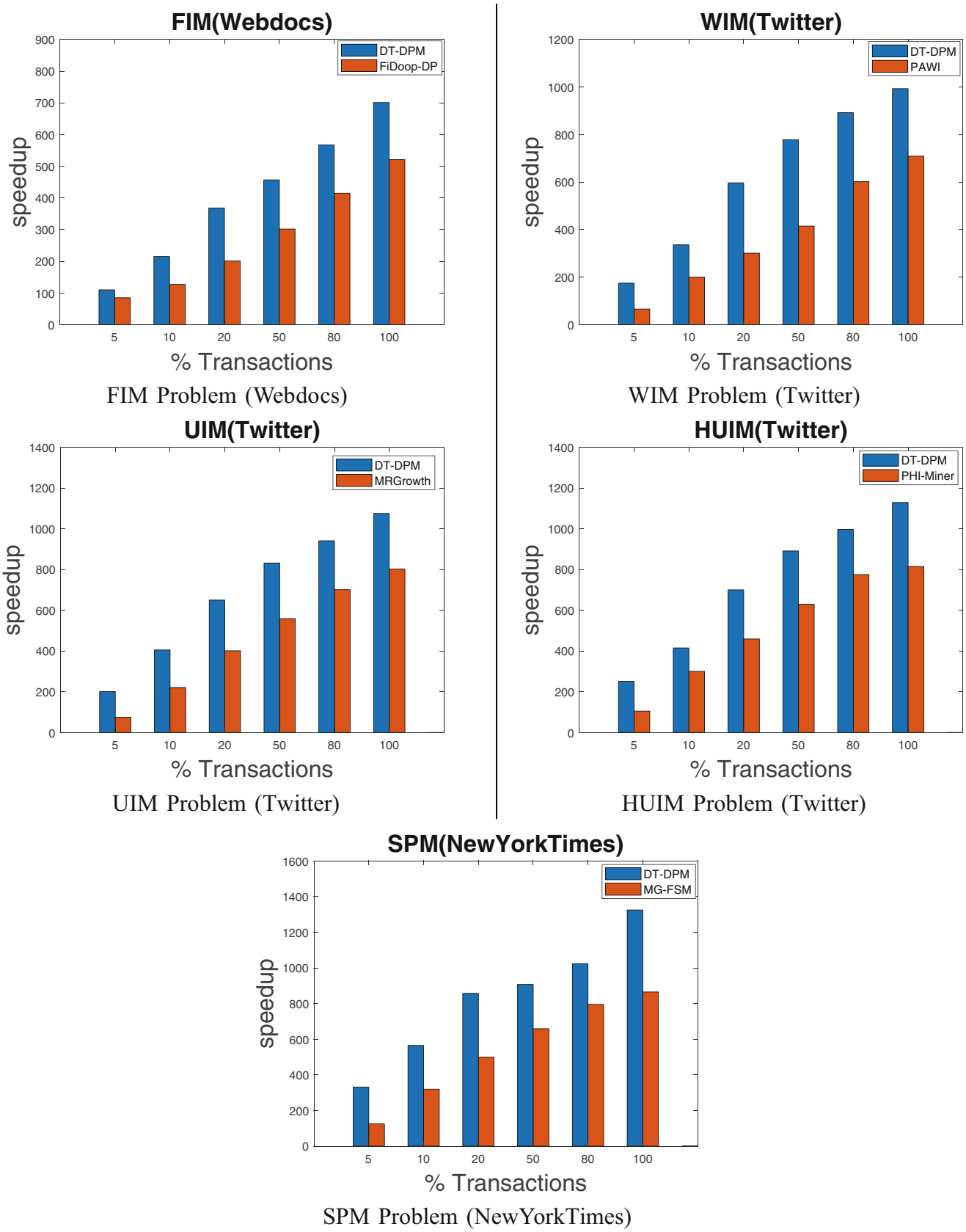
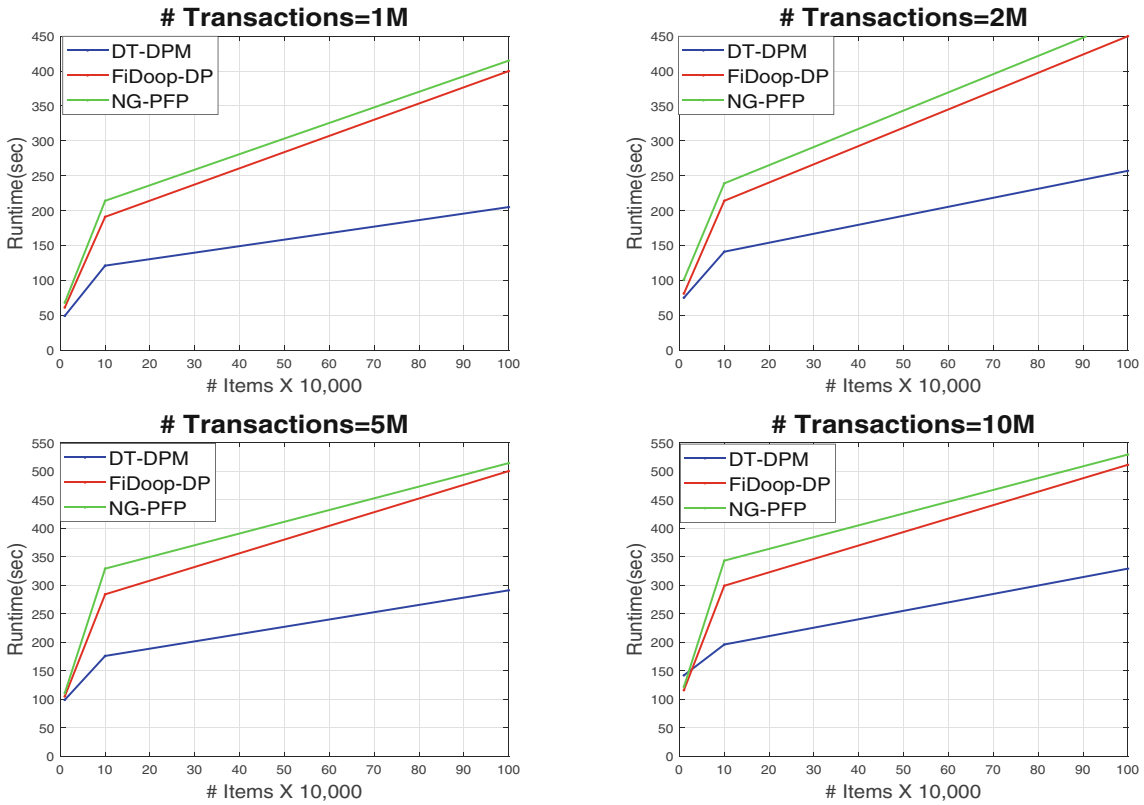


Fig. 4 Speedup of DT-DPM and the state-of-the-art parallel pattern mining algorithms using big databases



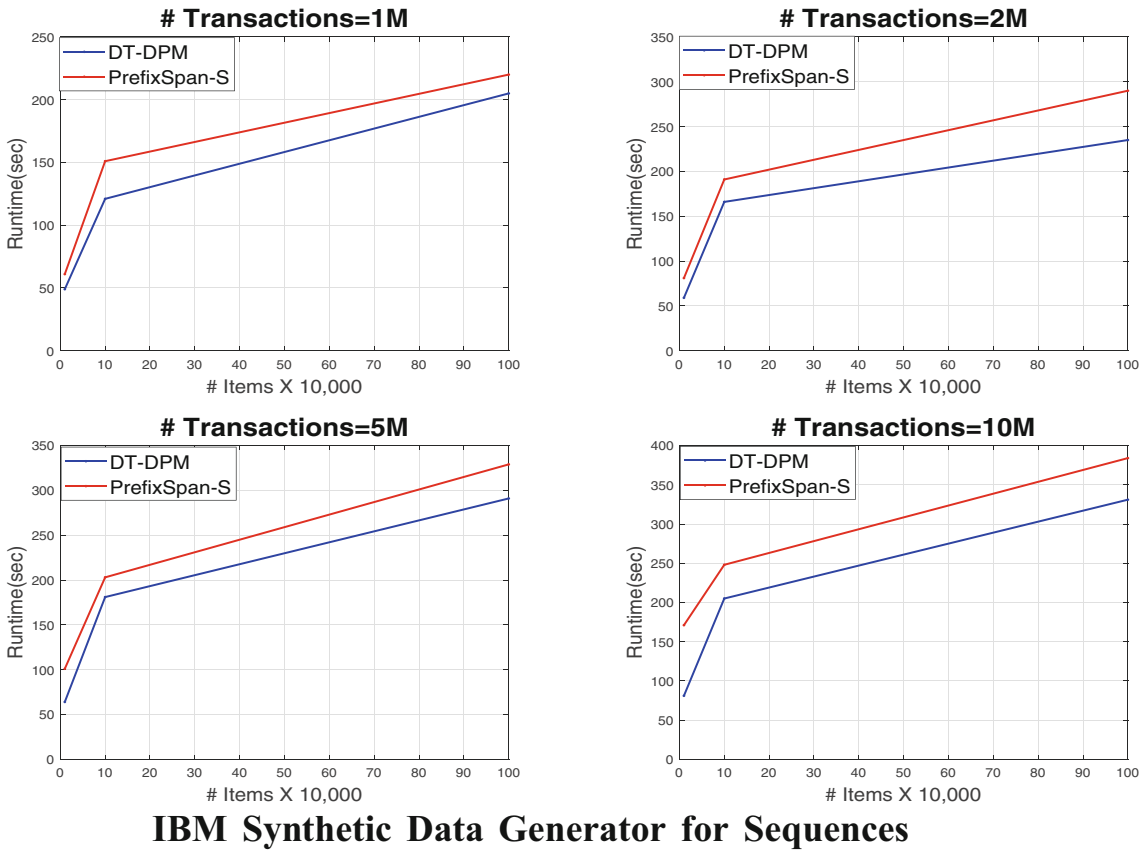


Fig. 5 Comparison of the Runtime of the DT-DPM and the baseline parallel-based pattern mining algorithms using big databases

baseline methods used is FiDoop-DP [15], and NG-PFP: NonGroup Parallel Frequent Pattern mining [67] for itemset mining, and PrefixSpan-S [66] for sequence mining. The results reveal that our model outperforms the baseline MapReduce based models in terms of computational time for both itemset and sequence mining. These results have been carried out whatever the number of items, and the number of transactions. Moreover, when the number of items varied from 10,000 to 1 million, and the number of transactions from 1 millionto 10 million, the ratio of runtimes between our model and the baseline models is increased. All these results are obtained thanks to, i) the *k*-means algorithm which minimizes the number of shared items, and ii) the efficient hybrid parallel processing of MapReduce, and the Multi CPU cores, which takes into account the information extracted during the decomposition step.

6 Conclusion

A novel distributed pattern mining framework called DT-DPM is proposed in this paper. DT-DPM aims to derive relevant patterns on big databases by studying the correlations within the transactional database and exploring heterogeneous architectures. The set of transactions are first grouped using a clustering approach, where transactions close to each other are assigned to the same cluster. For each cluster of transactions, the pattern mining algorithm is launched in order to discover the relevant patterns. DT-DPM solves the issue of cluster's size by incorporating heterogeneous computing such as single CPU, CPU cores, and MapReduce. Thus, the noise transactions are assigned to the single CPU core, the micro clusters are assigned to CPU cores, and dense clusters are assigned to the MapReduce reduce architecture. Experimental evaluation of DT-DPM was integrated on the SPMF tool, where five case studies have been shown (FIM, WIM, UIM, HUIM and SPM). The results reveal that by using DT-DPM, the scalability performance of pattern mining algorithms have been improved on large databases. Moreover, DT-DPM outperforms the baseline pattern mining algorithms on big databases. Motivated by the promising results shown in this paper, we plan to boost the performance ofDT-DPM and apply it on big data mining applications such as twitter analysis, smart building applications, and other large-scale applications.

Funding Information Open Access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital).

References

1. H. Belhadi, K. Akli-Astouati, Y. Djenouri, and J. C.-W. Lin Data mining-based approach for ontology matching problem. Appl Intell, pp. 1–18
2. Djenouri Y, Belhadi A, Fournier-Viger P (2018) Extracting useful knowledge from event logs: a frequent itemset mining approach. Knowl-Based Syst 139:132–148
3. Djenouri Y, Belhadi A, Belkebir R (2018) Bees swarm optimization guided by data mining techniques for document information retrieval. Expert Syst Appl 94:126–136
4. Djenouri Y, Djamel D, Djenoouri Z (2017) Data-mining-based decomposition for solving MAXSAT problem: towards a new approach. IEEE Intell Syst, vol. In press, pp. 1–15
5. He Z, Zhang S, Gu F, Wu J (2019) Mining conditional discriminative sequential patterns. Inf Sci 478:524–539
6. Choi H-J, Park CH (2019) Emerging topic detection in twitter stream based on high utility pattern mining. Expert Syst Appl 115:27–36
7. Djenouri D, Laidi R, Djenouri Y, Balasingham I (2019) Machine learning for smart building applications: Review and taxonomy. ACM Comput Surv (CSUR) 52(2):24
8. Fournier-Viger P, Lin JC-W, Vo B, Chi TT, Zhang J, Le HB (2017) A survey of itemset mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (4):7, e1207
9. Aggarwal CC, Han J (2014) Frequent pattern mining. Springer
10. Goethals B (2003) Survey on frequent pattern mining. Univ Hels 19:840–852
11. Mabroukeh NR, Ezeife CI (2010) A taxonomy of sequential pattern mining algorithms. ACM Comput Surv (CSUR) 43(1):3
12. Hsieh Y-H, Chen C-C, Shuai H-H, Chen M-S (2018) Highly parallel sequential pattern mining on a heterogeneous platform. in IEEE International Conference on Data Mining, pp. 1037–1042
13. Zhang L, Fu G, Cheng F, Qiu J, Su Y (2018) A multi-objective evolutionary approach for mining frequent and high utility itemsets. Appl Soft Comput 62:974–986
14. Djenouri Y, Comuzzi M (2017) Combining apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem. Inf Sci 420:1–15
15. Xun Y, Zhang J, Qin X, Zhao X (2017) FiDoop-DP: data partitioning in frequent itemset mining on hadoop clusters. IEEE Transactions on Parallel and Distributed Systems 28(1):101–114
16. Baralis E, Cagliero L, Garza P, Grimaudo L (2015) Pawi: Parallel weighted itemset mining by means of mapreduce, in IEEE International Congress on Big Data, pp. 25–32
17. Leung CK-S, Hayduk Y (2013) Mining frequent patterns from uncertain data with mapreduce for big data analytics, in International Conference on Database Systems for Advanced Applications, pp. 440–455
18. Miliaraki I, Berberich K, Gemulla R, Zoupanos S (2013) Mind the gap: Large-scale frequent sequence mining, in ACM SIGMOD International Conference on Management of Data, pp. 797–808
19. Djenouri Y, Lin JC-W, Nørvåg K, Ramampiaro H (2019) Highly efficient pattern mining based on transaction decomposition, in

IEEE International Conference on Data Engineering, pp. 1646–1649
20. Fournier-Viger P, Zhang Y, Lin JC-W, Fujita H, Koh YS (2019) Mining local and peak high utility itemsets. Inf Sci 481:344–367
21. Yun U, Kim D, Yoon E, Fujita H (2018) Damped window based high average utility pattern mining over data streams. Knowl-Based Syst 144:188–205
22. Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. ACM SIGMOD Rec 22(2):207–216
23. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. ACM SIGMOD Rec 29(2):1–12
24. Zhao X, Zhang X, Wang P, Chen S, Sun Z (2018) A weighted frequent itemset mining algorithm for intelligent decision in smart systems. IEEE Access 6:29 271–29 282
25. Yun U, Leggett JJ (2005) WFIM: weighted frequent itemset mining with a weight range and a minimum weight. in SIAM International Conference on Data Mining, pp. 636–640
26. Yun U (2007) Efficient mining of weighted interesting patterns with a strong weight and/or support affinity. Inf Sci 177(17):3477–3499
27. Yun U (2009) On pushing weight constraints deeply into frequent itemset mining. Intelligent Data Analysis 13(2):359–383
28. Le N-T, Vo B, Nguyen LB, Fujita H, Le B (2019) Mining weighted subgraphs in a single large graph. Inf Sci 514:149–165
29. Lee G, Yun U, Ryu KH (2017) Mining frequent weighted itemsets without storing transaction ids and generating candidates. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 25(01):111–144
30. Li H, Zhang N, Zhu J, Wang Y, Cao H (2018) Probabilistic frequent itemset mining over uncertain data streams. Expert Syst Appl 112: 274–287
31. Lee G, Yun U, Ryang H (2015) An uncertainty-based approach: frequent itemset mining from uncertain data with different item importance. Knowl-Based Syst 90:239–256
32. Liaqat M, Khan S, Younis MS, Majid M, Rajpoot K (2019) Applying uncertain frequent pattern mining to improve ranking of retrieved images. Appl Intell 49(8):2982–3001
33. Lee G, Yun U (2017) A new efficient approach for mining uncertain frequent patterns using minimum data structure without false positives. Futur Gener Comput Syst 68:89–110
34. Chan R, Yang Q, and Shen Y-D (2003) Mining high utility itemsets, in IEEE International Conference on Data mining, pp. 19–26
35. Liu Y, Liao W-k, and Choudhary A (2005) A two-phase algorithm for fast discovery of high utility itemsets, in Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 689–695
36. Lin C-W, Hong T-P, Lu W-H (2011) An effective tree structure for mining high utility itemsets. Expert Syst Appl 38(6):7419–7424
37. Zida S, Fournier-Viger P, Lin JC-W, Wu C-W, Tseng VS (2017) EFIM: a fast and memory efficient algorithm for high-utility itemset mining. Knowl Inf Syst 51(2):595–625
38. Liu J, Wang K, and Fung BC (2012) Direct discovery of high utility itemsets without candidate generation, in IEEE International Conference on Data Mining, pp. 984–989
39. Singh K, Singh SS, Kumar A, Biswas B (2019) TKEH: an efficient algorithm for mining top-k high utility itemsets. Appl Intell 49(3): 1078–1097
40. Gan W, Lin JC-W, Chao H-C, Fujita H, Philip SY (2019) Correlated utility-based pattern mining. Inf Sci 504:470–486
41. Lee J, Yun U, Lee G, Yoon E (2018) Efficient incremental high utility pattern mining based on pre-large concept. Eng Appl Artif Intell 72:111–123
42. Salvemini E, Fumarola F, Malerba D, and Han J (2011) Fast sequence mining based on sparse id-lists, in International Symposium on Methodologies for Intelligent Systems, pp. 316–325
43. Fumarola F, Lanotte PF, Ceci M, Malerba D (2016) CloFAST: closed sequential pattern mining using sparse and vertical id-lists. Knowl Inf Syst 48(2):429–463
44. Van T, Vo B, Le B (2018) Mining sequential patterns with itemset constraints. Knowl Inf Syst 57(2):311–330
45. Orakzai F, Calders T, Pedersen TB (2019) k/2-hop: fast mining of convoy patterns with effective pruning. Proceedings of the VLDB Endowment 12(9):948–960
46. Wu R, Li Q, Chen X (2019) Mining contrast sequential pattern based on subsequence time distribution variation with discreteness constraints. Appl Intell 49(12):4348–4360
47. Djenouri Y, Belhadi A, Lin J, Cano A (2019) Adapted k nearest neighbors for detecting anomalies on spatio-temporal traffic flow. IEEE Access 7:10 015–10 027
48. Belhadi A, Djenouri Y, Lin JC-W, Djenouri D, and Cano A (2020) A GPU-based two phase algorithm for identifying taxi frauds, IEEE Access, vol. In Press, pp. 1–14
49. Belhadi A, Djenouri Y, Lin JC-W, Zhang CC, Cano A (2020) Exploring pattern mining algorithms for hashtag retrieval problem. IEEE Access 8:10 569–10 583
50. Han E-H, Karypis G, Kumar V (2000) Scalable parallel data mining for association rules. IEEE Trans Knowl Data Eng 12(3):337–352
51. Zaki MJ (1999) Parallel and distributed association mining: a survey. IEEE Concurr 7(4):14–25
52. Wu X, Zhu X, Wu G-Q, Ding W (2014) Data mining with big data. IEEE Trans Knowl Data Eng 26(1):97–107
53. Cano A (2018) A survey on graphic processing unit computing for large-scale data mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8(1):e1232
54. Djenouri Y, Djenouri D, Belhadi A, Cano A (2019) Exploiting GPU and cluster parallelism in single scan frequent Itemset mining. Inf Sci 496:363–377
55. Chen Y, An A (2016) Approximate parallel high utility itemset mining. Big data research 6:26–42
56. Moens S, Aksehirli E, and Goethals B (2013) Frequent itemset mining for big data, in IEEE International Conference on Big Data, pp. 111–118
57. Jian L, Wang C, Liu Y, Liang S, Yi W, Shi Y (2013) Parallel data mining techniques on graphics processing unit with compute unified device architecture (CUDA). J Supercomput 64(3):942–967
58. Djenouri Y, Bendjoudi A, Mehdi M, Nouali-Taboudjemat N, Habbas Z (2015) GPU-based bees swarm optimization for association rules mining. J Supercomput 71(4):1318–1344
59. Djenouri Y, Bendjoudi A, Habbas Z, Mehdi M, Djenouri D (2017) Reducing thread divergence in gpu-based bees swarm optimization applied to association rule mining. Concurrency and Computation: Practice and Experience 29(9)
60. Li Y, Xu J, Yuan Y-H, and Chen L (2017) A new closed frequent itemset mining algorithm based on GPU and improved vertical structure. Concurrency and Computation: Practice and Experience, vol. 29, no. 6
61. Xun Y, Zhang J, Qin X (2016) FiDoop: parallel mining of frequent itemsets using mapreduce. IEEE Transactions on Systems, Man, and Cybernetics: systems 46(3):313–325
62. Andrzejewski W, Boinski P (2019) Parallel approach to incremental co-location pattern mining. Inf Sci 496:485–505
63. Andrzejewski W, Boinski P (2018) Efficient spatial co-location pattern mining on multiple GPUs. Expert Syst Appl 93:465–483
64. Jiang Y, Zhao M, Hu C, He L, Bai H, Wang J (2019) A parallel FP-growth algorithm on World Ocean Atlas data with multi-core CPU. J Supercomput 75(2):732–745
65. Vanahalli MK, Patil N (2019) An efficient parallel row enumerated algorithm for mining frequent colossal closed itemsets from high dimensional datasets. Inf Sci 496:343–362
66. Yu X, Li Q, Liu J (2019) Scalable and parallel sequential pattern mining using spark. World Wide Web 22(1):295–324

67. Kuang Z-j, Zhou H, Zhou J-p, Yang K et al (2019) A non-group parallel frequent pattern mining algorithm based on conditional patterns. Frontiers of Information Technology & Electronic Engineering 20(9):1234–1245
68. Sumalatha S, Subramanyam R (2020) Distributed mining of high utility time interval sequential patterns using mapreduce approach. Expert Syst Appl 141:112967
69. Djenouri Y, Djenouri D, Belhadi A, Fournier-Viger P, Lin JC-W, Bendjoudi A (2019) Exploiting GPU parallelism in improving bees swarm optimization for mining big transactional databases. Inf Sci 496:326–342
70. Djenouri Y, Djenouri D, Belhadi A, Lin JC-W, Bendjoudi A, and Fournier-Viger P (2019) A novel parallel framework for metaheuristic-based frequent itemset mining, in IEEE Congress on Evolutionary Computation, pp. 1439–1445
71. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv (CSUR) 31(3):264–323
72. Ester M, Kriegel H-P, Sander J, Xu X et al. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise, in International Conference on Knowledge Discovery and Data Mining, vol. 96, pp. 226–231
73. Seifoddini H, Djassemi M (1991) The production data-based similarity coefficient versus jaccard's similarity coefficient. Comput Ind Eng 21(1–4):263–266
74. Fournier-Viger P, Gomariz A, Gueniche T, Soltani A, Wu C-W, Tseng VS (2014) SPMF: a Java open-source pattern mining library. J Mach Learn Res 15(1):3389–3393
75. Deng Z-H, Lv S-L (2015) PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via children–parent equivalence pruning. Expert Syst Appl 42(13):5424–5432
76. Chui C-K, Kao B, and Hung E (2007) Mining frequent itemsets from uncertain data," in Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 47–58
77. Lucchese C, Orlando S, Perego R, and Silvestri F (2004) WebDocs: a real-life huge transactional dataset, in Frequent Itemset Mining Implementations, vol. 126
78. Kwak H, Lee C, Park H, and S. Moon (2010) What is Twitter, a social network or a news media? in International Conference on World Wide Web, pp. 591–600