An effective approach to inverse frequent set mining

Antonella Guzzo  
DEIS  
University of Calabria  
Rende, CS, Italy  
guzzo@deis.unical.it

Domenico Saccà  
DEIS  
University of Calabria  
Rende, CS, Italy  
sacca@unical.it

Edoardo Serra  
DEIS  
University of Calabria  
Rende, CS, Italy  
eserra@deis.unical.it

Abstract—The inverse frequent set mining problem is the problem of computing a database on which a given collection of itemsets must emerge to be frequent. Earlier studies focused on investigating computational and approximability properties of this problem. In this paper, we face it under the pragmatic perspective of defining heuristic solution approaches that are effective and scalable in real scenarios. In particular, a general formulation of the problem is considered where minimum and maximum support constraints can be defined on each itemset, and where no bound is given beforehand on the size of the resulting output database. Within this setting, an algorithm is proposed that always satisfies the maximum support constraints, but which treats minimum support constraints as soft ones that are enforced as long as possible. A thorough experimentation evidences that minimum support constraints are hardly violated in practice, and that such negligible degradation in accuracy (which is unavoidable due to the theoretical intractability of the problem) is compensated by very good scaling performances.

Keywords—Inverse Frequent Mining, Data Reconstruction, Complexity.

I. INTRODUCTION

The inverse frequent set mining problem (short: IFM problem in our framework) is the problem of computing a database (or deciding whether such database exists or not) on which a given collection of itemsets must be “frequent” [1]. This problem attracted much attention in the recent years, due to its applications in privacy preserving contexts [2], [3] and in defining generators for benchmark data [4]. In particular, earlier studies mainly focused on investigating its computational properties, by charting a precise picture of the conditions under which it becomes intractable [5], [6], [1], and by observing that in its general formulation it is NP-hard even if one looks for approximate solutions [2]. In this paper, the IFM problem is reconsidered from a pragmatic point of view instead. Indeed, we concentrate on defining heuristic approaches that are able to efficiently and efficaciously solve the problem in real-world scenarios. In particular, we consider the original formulation of the problem in [1], where the “frequency” of any itemset in the database is measured in terms of its support, i.e., as the number of the transactions in which it occurs. Other approaches to the IFM problem (e.g., [5], [6]) considered the actual frequency (i.e. support divided by the number of transactions), however, as discussed in [1], supports convey more information than frequencies and hence the perspective of [1] is adopted here. In fact, while keeping this perspective, we investigate a more general setting obtained by relaxing the simplifying assumption in [1] that the size of the output database must be known beforehand, and by furthermore considering a minimum and a maximum support constraint (over each itemset required to occur in the database) in place of a single support value. As a matter of fact, the resulting framework was not investigated in earlier literature either from a theoretical point of view or from an algorithmic perspective. Within this framework, our contributions are:

- We first study the computational complexity of the inverse frequent set mining problem. It turns out that even when the size of the database is not fixed beforehand, the problem is still NP-hard and, hence, unlikely to be efficiently solvable.
- Given that an important source of intractability of IFM is in the fact that we do not known the “structure” of the various transactions into the database, we define and analyze the computational properties of a variant of IFM, called IFM$_S$, where the transactions in $D$ are fixed from the given set $S$—to have fixed the set of the possible itemsets that can be inserted into the output database is a scenario that may arise in several application domains. Surprisingly, this variant turns out to be NP-hard as well.
- Motivated by the above intractability result, we propose a heuristic approach to solve the IFM$_S$ problem. Then, we extend it in order to handle the basic IFM problem, by using a preprocessing step that transforms any instance of IFM into an instance of IFM$_S$. Notably, the heuristic approach (over both) is designed in a way that guarantees to satisfy the maximum support constraints issued over the itemsets, but that treats minimum support constraints as soft ones that are enforced as long as it possible.
- Finally, we conduct a thorough experimental activity over both syntactic and real data. Results evidence that the output of our technique will hardly violate minimum support constraints (though they are treated as soft ones). Thus, our solution approach emerged to be effective for real application scenarios. In addition, the results show that our approach enjoys very good scaling, thereby paving the way for its application over real scenarios.
Organization. The paper is organized as follows. Section II formalizes the inverse frequent itemset mining problem and analyzes its computational complexity. The solution approaches to face IFM and IFM$\subseteq$ are discussed in Section III and Section IV, respectively. Experimental results are illustrated in Section V. Finally, related works and conclusions are discussed in Section VI and VII, respectively.

II. PROBLEM STATEMENT AND COMPLEXITY ISSUES

Let $I$ be a finite domain of elements, also called items. Any subset $I \subseteq I$ is called an itemset over $I$. A database $D$ over $I$ is a bag of itemsets, each one usually called a transaction. The support of an itemset $I \subseteq I$ in $D$, denoted as $\sigma^D(I)$, is defined as the number of itemsets $J$ in $D$ containing $I$, i.e., such that $I \subseteq J$ holds. We say that $I$ is a frequent itemset in $D$ if $I$’s support is no less than a given support threshold minsupp. Finding all the frequent itemsets in $D$ is the well-known frequent itemset mining problem. In the paper, we consider the inverse frequent itemset mining problem: Given a set $S$ of itemsets, finding a database in which each element of $S$ is frequent.

Definition 1 (IFM Problem): Let $S = \{I_1, \ldots, I_m\}$ be a set of itemsets over the items in $I$. Let $\sigma_{\min} : S \rightarrow \mathbb{N}$ and $\sigma_{\max} : S \rightarrow \mathbb{N}$ be two functions associating to each itemset $I_i \in S$ the natural numbers $\sigma_{\min}(I_i)$ and $\sigma_{\max}(I_i)$ with $\sigma_{\min}(I_i) \leq \sigma_{\max}(I_i)$, called the minimum and the maximum support of $I_i$, respectively. Then, the Inverse Frequent Itemset Mining Problem on $I$, $\{I_1, \ldots, I_m\}$, $\sigma_{\min}$ and $\sigma_{\max}$ (short: IFM($I$, $\{I_1, \ldots, I_m\}$, $\sigma_{\min}$, $\sigma_{\max}$)) consists of finding a database $D$ over $I$ such that $\sigma_{\min}(I_i) \leq \sigma^D(I_i) \leq \sigma_{\max}(I_i)$, for each $I_i \in S$.

It is worthwhile noticing that the inverse frequent mining problem was mainly formulated and analyzed in the literature within contexts where the frequency of an itemset (i.e. its support $\sigma^D$ divided by the total number of transactions in $D$) is considered in place of the support in the problem formulation. When using this perspective, IFM is generally referred to as the FREQSAT problem, and various complexity results are known for it [6].

Considering the support of the itemsets rather than their frequency received considerably less attention instead. In particular, in [1], a slight variation of IFM was studied where the size $|D|$ of the output database $D$ is fixed beforehand (short: IFM$_{|D|}$). This variation was observed to be NP-hard, even if $\sigma_{\min} = \sigma_{\max}$. However, the complexity of the main problem IFM (i.e., in absence of such an additional constraint on the size of $D$) was not derived in earlier literature. Our first contribution is to complete the picture of the complexity issues arising with the decision version of the inverse frequent mining problem. Indeed, we shall evidence that IFM is computationally intractable too, thereby calling for (heuristic) solution approaches that are efficient in actual scenarios. In particular, the result can be proven by exhibiting a reduction from the IFM$_{|D|}$ problem, which is NP-hard even if $\sigma_{\min} = \sigma_{\max}$.

Theorem 2: IFM is NP-hard, also when $\sigma_{\min} = \sigma_{\max}$.

Note that there are several sources of intractability in the formulation of IFM. A major one is that the “structure” of the various transactions to be inserted into $D$ is not known beforehand. Therefore, when building such database we are uncertain on which kinds of transaction to exploit. Towards devising heuristic approaches for IFM, a natural idea is then to consider a simplification of IFM where all the possible transactions are taken from the set $S$ of itemsets provided in input. Let IFM$_S$ denote this variation of IFM. Rather surprisingly, we can show that even IFM$_S$ is intractable in general (and feasible in polynomial time, if $\sigma_{\min} = \sigma_{\max}$), by exhibiting a reduction from the graph 3-colorability problem and by encoding the solutions of IFM$_S$ in terms of a system of linear equations over polynomially many variables and constraints, when $\sigma_{\min} = \sigma_{\max}$.

Theorem 3: IFM$_S$ is NP-complete, in general; and, feasible in polynomial time, if $\sigma_{\min} = \sigma_{\max}$.

For the sake of completeness, note that the above result does not entail the one in Theorem 2 and viceversa; indeed, IFM$_S$ coincides with IFM if and only if $S$ is the set of all the possible itemsets that can be built over $I$.

III. A LEVEL-WISE SOLUTION APPROACH FOR IFM$_S$

In order to devise an efficient and effective algorithm for IFM, we find convenient to firstly tackle the special case where the output database must be constructed by using as transactions the input itemsets only (cf. IFM$_S$ problem), and then to build the general algorithm for IFM on top of the solution approach for IFM$_S$.

In fact, given that it is not possible to efficiently enforce both the minimum and the maximum support on each itemset in $S$ (cf. Theorem 3), we shall pragmatically propose in this section an approach to face IFM$_S$ which takes care of the function $\sigma_{\max}$ only, while treating $\sigma_{\min}$ as a soft constraint that must be satisfied as long as it is possible. Here, it is worthwhile anticipating that a thorough experimental activity conducted on our solution approach evidenced that the output of our technique will hardly violate constraints on $\sigma_{\min}$ (though they are treated as soft ones).

A. Description of the Algorithm

Recall that IFM$_S$(I, S, $\sigma_{\min}$, $\sigma_{\max}$) amounts to finding a database $D$ that is built over the itemsets in $S$ (each one possibly occurring with multiple repetitions) and where constraints associated with the functions $\sigma_{\min}$ and $\sigma_{\max}$ are satisfied. Thus, a natural approach to face IFM$_S$ is to iterate over the elements in $S$ and decide how many copies have to

Due to space constraints, proofs are reported in the full version available at http://www.info.deis.unical.it/guzzol.
be added in the output database for each of them. However, various strategies can be used to process the elements in \( S \) and to decide about the number of copies to be added for each of them. Our strategy is based on two key ideas.

Firstly, we propose to process the itemsets of \( S \) that are candidates for being added to the output database \( \mathcal{D} \) by means of a level-wise approach, where larger (w.r.t. set containment) itemsets are processed first. Formally, let \( S_1, S_2, ..., S_k \) be a succession of subsets of \( S \) such that: (i) \( S_i \cap S_j = \emptyset \) for each \( i \neq j \); (ii) \( \bigcup_{i=1}^{k} S_i = S \); and, (iii) for each pair of subsets \( S_i \) and \( S_j \) with \( i < j \), and for each itemset \( J \in S_j \), there is an itemset \( I \in S_i \) s.t. \( I \supset J \) and there is no itemset \( \bar{I} \in S_i \) s.t. \( \bar{I} \subseteq J \). Note that the succession \( S_1, S_2, ..., S_k \) can efficiently be computed from \( S \), and that by processing itemsets according to their order of occurrence in the succession, we can enforce that each itemset is processed prior to its subsets.

Secondly, whenever processing the \( i \)-th element \( S_i \) of the above succession and for each itemset \( I \in S_i \), we propose to add in \( \mathcal{D} \) the minimum possible number of copies of \( I \) that suffices to satisfy \( \sigma_{\min}(I) \) and that do not lead to violate the maximum support constraints on the subsets of \( I \), which have in fact to be still processed. Formally, let \( \Delta^\mathcal{D}(I) = \min_{J \in S_j \subseteq I}(\sigma_{\max}(J) - \sigma^\mathcal{D}(J)) \). Then, such number of copies is given by the expression \( \Delta^\mathcal{D}_{\min}(I) = \min(\sigma_{\min}(I) - \sigma^\mathcal{D}(I), \Delta^\mathcal{D}(I)) \). Note that one may obtain \( \Delta^\mathcal{D}_{\min}(I) \leq 0 \), thereby implying that the algorithm fails in providing the minimum support for \( I \).

An algorithm implementing these ideas is shown in Fig. 1. The input of the algorithm is a set of itemsets \( S \) and two functions \( \sigma_{\min} \) and \( \sigma_{\max} \) denoting the support constraints associated with such itemsets. The output is a database with itemsets as transactions, which is meant as a heuristic solution to \( \text{IFM}_S(I, S, \sigma_{\min}, \sigma_{\max}) \).

The algorithm starts by setting the database \( \mathcal{D} \) to the empty set, and then applies the level-wise exploration of the itemsets in \( S \) in order to add elements into \( \mathcal{D} \)—for the moment let us get rid of step 4 that implements an optimization discussed in Section III-B. In fact, given that each update on \( \mathcal{D} \) preserves the maximum support constraint on all the subsets of the processed itemset and given that itemsets are processed according to their set inclusion, it is immediate to check that the resulting database \( \mathcal{D} \) is such that for each itemset \( I \in S \), \( \sigma^\mathcal{D}(I) \leq \sigma_{\max}(I) \) holds. However, we have no theoretical guarantee that the minimum support constraint is satisfied over each itemset. In an extreme case, no copy of some itemset might be added to \( \mathcal{D} \) even though a certain number of them are required.

B. Optimization Issues

In the algorithmic scheme we have discussed, at each level \( i \) of the search, itemsets from \( S_i \) only are added into \( \mathcal{D} \). In practice this might be too restrictive. Indeed, we might think of enforcing the support of each itemset \( I \in S_i \) by adding an itemset \( J \supset I \) contained in some set \( S_j \) with \( j < i \) rather than by directly adding \( I \). In fact, \( S_j \) is guaranteed to exists (except for \( i = 1 \)), by construction of the succession \( S_1, ..., S_k \).

This optimization founds on the idea that enforcing the support of \( I \) by including copies of one of its supersets has the side-effect of incrementing the support of other itemsets included in \( J \) and, hence, belonging to same level subsequent to \( S_j \). This is very relevant in our approach, given that this effect goes in the direction of amplifying the chances of ending up with a database satisfying the minimum support constraints over all itemsets, which is a critical issue as we discussed above. In practice, to implement this strategy, the algorithm in Fig. 1 accounts for an optimization step that is performed before that itemsets at the current level \( S_i \) are analyzed. This optimization is next discussed in detail.

Let \( \Delta^\mathcal{D}_{\max}(I) = \min(\sigma_{\max}(I) - \sigma^\mathcal{D}(I), \Delta^\mathcal{D}(I)) \) be the maximum number of copies of \( I \) that can be added to \( \mathcal{D} \) while still satisfying \( \sigma_{\max}(I) \) and while not violating the maximum support constraints on the subsets of \( I \). For any itemset \( I \in S_j \) and for any element \( S_i \) with \( i > j \), define then \( \text{inc}(I, S_i, \mathcal{D}) \) as the value:

\[
\min \left( \Delta^\mathcal{D}_{\max}(I), \min_{I' \in S_i \cup \mathcal{D} \cup \mathcal{D}} \frac{(\sigma_{\min}(I') - \sigma^\mathcal{D}(I'))}{\sigma_{\min}(I')} \right).
\]

Intuitively, this is the maximum increment allowed on the support of \( I \) computed by also considering as a bound the minimum support which suffices to satisfy \( \sigma_{\min} \) on any of its subsets that have still to be satisfied (i.e., for which \( \sigma_{\min}(I_i) > \sigma^\mathcal{D}(I_i) \) currently holds). In practice, we do want to increment the support of \( I \) by affecting as less as possible the support of its subsets. Based on the increment
values computed at the level associated with $S_i$, we want to compute the set of all the itemsets in some level below $S_i$ that leads to reduce as much as possible the number of itemsets to be added to $D$. To this end, define first $gainSet(I, S, J, D)$ as the itemsets whose minimum support is not yet satisfied and that are subsets of $I$ and supersets of $J$, i.e., $gainSet(I, S, J, D) = \{I' \in S \mid I' \subset I \land I' \supset J \land \sigma_{\min}(I') < \sigma^D(I')\}$. Define then $gain(I, S_i, J, D)$ as:

$$inc(I, S_i, D) \ast (\lfloor gainSet(I, S, J, D) \rfloor - 1) / \Delta \sigma^D_{\max}(I).$$

Intuitively, this is a normalized value that is meant to denote the advantage of adding $inc(I, S_i, D)$ copies of $I$ to $D$ w.r.t. an itemset $J$ that has still to be processed. By averaging this gain over all the itemsets that have to be processed, we define the value $avgGain(I, S_i, D)$ as:

$$\sum_{j=1}^{k} \left[ \sum_{j=1}^{k} \frac{gain(I, S_i, J, D)}{|S_j|} \right].$$

Finally, let $bestCandidates(S_i, D)$ be the set of all itemsets in $\bigcup_{j=1}^{k} S_j$ on which the maximum value (not equals to zero) of the average gain is achieved. These itemsets are those that we consider the most promising for being added to $D$. These itemsets are computed in step P1 of the optimization procedure and updated in P5, after that $inc(I, S_i, D)$ copies of an itemset $I$ (arbitrarily picked from $bestCandidates(S_i, D)$) have been actually added to $D$.

As for the running time, note that while without optimization we need to iterate only over the sets in $|S|$ (as to compute $\Delta \sigma^D_{\min}(I)$), we now need to compute the set $bestCandidates(S_i, D)$, which is feasible in $O(|S|^2)$. In total, the complexity is now $O(|S|^3 \times |I|)$.

IV. SOLVING THE GENERAL CASE OF IFM

Now that a solution approach for $IFM_S$ has been described, we can move to discussing a solution approach for the more general $IFM$ problem. In fact, it is worthwhile observing that the algorithm in Fig. 1 can already be seen as a heuristic approach to face $IFM$. Indeed, in addition to the specific heuristic used to deal with the support constraints, this algorithm heuristically solves $IFM$ by restricting the set of all the possible transactions to those in $S$.

In practice, focusing on the set $S$ might be too restrictive to solve $IFM$. Thus, we propose to enlarge the original set $S$ by including various novel itemsets (built from those in $S$), whose exploitation is envisaged to be beneficial for improving the effectiveness of the algorithm in Fig. 1.

The approach, illustrated in Fig. 2, constructs a novel set $S'$ to be used as input for the algorithm in Fig. 1 by merging $k$ itemsets at most from $S$. The merging function is carried out recursively, by picking an itemset at time from a set of candidates itemsets ($S_{can}$).

In particular, it is worthwhile observing that itemsets are merged together if and only if their intersection is not empty (which motivates the initialization in step 2 and the check in F5). Indeed, this is in line with the approach discussed in Section III-B, where the gain of two itemsets having no subset in common is equals to zero. In addition, note that the merging process is restricted to maximal itemsets only, i.e., to those which are included in the first level $S_1$.

We conclude the section by noticing that as for the support constraints, for each itemset $I \in S'$, we set $\sigma'_{\min}(I)$ as the value:

$$\begin{cases} \sigma_{\min}(I) & I \in S \\ \max(0, \max(\sigma'_{\min}(Y)|Y \in S' \land I \subset Y)), & I \notin S \\ \end{cases}$$

and we set $\sigma'_{\max}(I)$ as the value:

$$\begin{cases} \sigma_{\max}(I) & I \in S \\ \min(st, \min(\sigma'_{\max}(Y)|Y \in S' \land I \subset Y)), & I \notin S \\ \end{cases}$$

where $st$ is a user bound on the support of the new itemset.

Given these novel input specifications, the algorithm for $IFM_S'$ is then applied and used as a heuristic approach to solve $IFM$ on $S$. Eventually, note that for any fixed natural number $k$, the above preprocessing step is not an overhead for the running time of the algorithm in Fig. 1.

V. EXPERIMENTS

The above solution approach for $IFM$ has been implemented, and a thorough experiential activity has been conducted to assess its efficiency and effectiveness. Details on this activity are discussed in the remaining of the section.
A. Data Preparation

Experimentation is carried out over three distinct datasets [7], which have been often used as reference benchmarks for frequent itemsets discovery algorithms: The artificial dataset T10I4D100K, and the two real datasets BMS-WebView-1 and BMS-WebView-2. These latter databases contain clickstream data from two e-commerce web sites (each transaction represents a web session and each item in a page viewed in that session). Summary information on these three datasets are illustrated in Fig. 3.

For each of the above databases, the idea of the experimentation is to firstly extract the set $S$ of the itemsets that are actually frequent (by standard itemsets discovery algorithms). For each itemset $I \in S$ whose frequency is $\sigma$, we define $\sigma_{min}(I) = \sigma - \alpha \ast \sigma$ and $\sigma_{max} = \sigma + \alpha \ast \sigma$, where $\alpha$ is a normalized real number (that will be varied in the experimentations). Then, the whole set $S$ with the support constraints constructed as above will be supplied as input to our algorithm for the IFM problem.

To evaluate scalability, we shall just refer to the running time. Instead, to evaluate its effectiveness in solving IFM, we shall take care of the following relative error index:

$$er(\%) = \frac{1}{|S|} \ast \sum_{I \in S: \sigma_{D}(I) < \sigma_{min}(I)} \frac{\sigma_{min}(I) - \sigma_{D}(I)}{\sigma_{min}(I)}$$

. It is worthwhile observing that the above index accounts for how many itemsets from $S$ occur in the output database with a support that is below the required constraint. In fact, recall from Section III-A that the maximum support constraint is always satisfied with our approach. Clearly enough, values of $er(\%)$ close to 0 are desirable.

All the results discussed below have been obtained by experimenting with an Intel dual core with 1.8GB memory, running windows XP Professional.

B. Results

A first series of experiment was aimed at assessing the scalability of our approach w.r.t. some key input parameters. In particular, we considered the real datasets by varying the parameter $\alpha$ (from 0 to 0.3). Execution times over BMS-WebView1 are reported in Fig. 4, for increasingly larger support$^2$. Note that time is not affected by the size of the interval over the support constraints.

$^2$Results on the other datasets (and further experiments) are discussed in the full version.

In a second series of experiments, we assessed the effectiveness of the approach by computing the relative index error over the three datasets, by varying the support and the $\alpha$ parameter. Results, reported in Fig. 5, evidence that the error rate is below 2% for BMS-View-1 and T10I4D100K dataset, while is below 10% for BMS-View-2. It comes with no surprise that accuracy improves by increasing $\alpha$. Moreover, the error is always equals to 0 when $\alpha > 0.3$,.
i.e., when a large support constraint window is defined.

In a further set of experiments, we considered the experimentation perspective discussed in [3]. There, it is argued that the effectiveness of inverse frequent itemsets mining algorithms has to be assessed (1) by comparing the itemsets that can be actually rediscovered on the synthetic database with those occurring in the original one, and (2) by comparing the performances of a mining algorithm (e.g., Apriori) over the syntactic and the original dataset.

In order to deal with (1) above, we used the Jaccard, Dice, and Overlap [3] indices to compare similarities between original frequent itemsets and those occurring in the syntactic output database. Results for BMS-WebView2, reported in Fig. 6, evidence that very high accuracy measures are obtained, and that support threshold values are greater or equal to the support threshold used for data generation.

Finally, as for (2), we report in Fig. 7 the difference between execution times of Apriori when running on the original dataset and when running on the syntetic one (build on T10I4D100K by using the supports values: 0.5, 0.6 and 0.7). Note that the lower is the support used in the generation of dataset, the smaller is the difference of performances.

VI. RELATED WORK

The IFM problem has been firstly introduced in [1] from the computational complexity point of view, by showing that deciding whether there is a dataset compatible with the given frequent sets is NP-hard. However, the size of the database was assumed to be fixed beforehand, which is an assumption that we removed in our research. This problem was subsequently considered in [5], [6] within a novel formulation (namely \texttt{FREQSAT}) where the support of the itemsets is measured in terms of their frequency. Here several variants of \texttt{FREQSAT} have been studied and a complete parameterizations of their intrinsic difficulty was provided. In the light of the intractability of the IFM problem, approximation strategies have been discussed in [2]. In particular, the authors asked whether it is possible to satisfy the various support constraints in an approximate fashion. The result was again bad news, since it is shown that the problem is not approximable unless \( P = NP \). This strongly motivates looking for heuristic approaches, which is precisely the perspective we adopted in this paper. In fact, a heuristic approach to generate a database satisfying the given frequency constraints was firstly proposed in [3]. The approach founds on the Iterative Proportional Fitting (IPF) method to estimate contingency tables, and on graphical decomposition techniques to decompose these tables into subsets and apply IPF algorithms only on the irreducible components. As a consequence, the feasibility of this approach depends on the assumption that many of the items are (conditionally) independent. Another heuristic approach was discussed in [4], where a method to generate basket datasets for benchmarking activities is discussed which is applicable when the length distributions of frequent and maximal frequent itemset collections is available. These two earlier approaches deal with the \texttt{FREQSAT} problem, and hence they are not applicable to our case where the support measure is used in place of the frequency.

VII. CONCLUSIONS

The inverse frequent set mining problem IFM has been considered together with the variant IFM\(_\#\) in which the transactions to be added in the output database must be picked from the given set \( S \). Motivated by their intractability, we proposed two heuristic approaches that have been tested over both syntactic and real data. Results show that they are effective and scalable, thereby paving the way for applications over real scenarios.

REFERENCES


