

# Multi-Sorted Inverse Frequent Itemsets Mining for Generating Realistic No-SQL Datasets

(Discussion Paper)

Domenico Saccà<sup>1</sup>, Edoardo Serra<sup>2</sup> and Antonino Rullo<sup>1</sup>

<sup>1</sup>*DIMES Department, University of Calabria, 87036 Rende, Italy*

<sup>2</sup>*Department of Computer Science, Boise State University, Boise, ID 83725, USA*

## Abstract

The development of novel platforms and techniques for emerging “Big Data” applications requires the availability of real-life datasets for data-driven experiments, which are however not accessible in most cases for various reasons, e.g., confidentiality, privacy or simply insufficient availability. An interesting solution to ensure high quality experimental findings is to synthesize datasets that reflect patterns of real ones. A promising approach is based on inverse mining techniques such as inverse frequent itemset mining (IFM), which consists of generating a transactional dataset satisfying given support constraints on the itemsets of an input set, that are typically the frequent and infrequent ones. This paper describes an extension of IFM that considers more structured schemes for the datasets to be generated, as required in emerging big data applications, e.g., social network analytics.

## Keywords

IFM, No-SQL, Itemset Mining

## 1. Introduction

Emerging “Big Data” platforms and applications call for the invention of novel data analysis techniques that are capable to effectively and efficiently handle large amount of data [1]. There is therefore an increasing need to use real-life datasets for data-driven experiments that are often not available for two main reasons: (i) *ownership* (i.e., they are often proprietary and out of reach for academic research), and (ii) *privacy* (they may include sensible data as it happens for many research areas, e.g., epidemiology, public health, social science, study of the behavior of large populations of individuals under natural scenarios, as well as under human interventions).

The two above limitations can be overcome by the design of “realistic” synthetic datasets that capture key attributes and activities of real ones without violating confidentiality [2].

In this paper we use an inverse data mining approach to generate an artificial NoSQL transactional dataset that reflects the patterns of a real one: the patterns can be thought of as a compressed representation of the original dataset and are first discovered by data mining techniques, and they are next used to generate a “realistic” pattern-preserving dataset.

---


*SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy*

✉ domenico.sacca@unical.it (D. Saccà); edoardoserra@boisestate.edu (E. Serra); n.rullo@dimes.unical.it (A. Rullo)

🆔 0000-0003-3584-5372 (D. Saccà); 0000-0003-0689-5063 (E. Serra); 0000-0002-6030-0027 (A. Rullo)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

A well-known inverse approach refers to the classical data mining problem of extracting the set of the *frequent/infrequent itemsets* from a transaction database. The related literature is rather rich and covers almost three decades: after the seminal papers in the nineties [3, 4], additional aspects have been studied in the last two decades. The perspective of the frequent itemset mining problem can be naturally inverted as follows: we are given in advance a set of itemsets together with their frequency as constraints and our goal is to compute a transactional database satisfying the above constraints. This problem, called the *inverse frequent itemset mining* problem (IFM), has been introduced in the context of defining generators for benchmarks of mining algorithms [5], and has been subsequently reconsidered in privacy preserving contexts [6, 7]. A reformulation of IFM in terms of frequencies instead of supports has been introduced in [8, 9] with the name FREQSAT and many variants of it have been proposed as well.

The original IFM formulation does not introduce any constraint on infrequency. A simple solution to exclude unexpected frequent itemset from a feasible solution is the formulation proposed in [10], which is called IFM<sub>S</sub>: only itemsets listed in the set  $S$  of constraints can be included as transactions in  $\mathcal{D}$ . A more general version of IFM with infrequency support constraint (IFM<sub>I</sub> for short), has been proposed in [11], where every infrequent itemset may occur a small number of times.

In order to enlarge the domain of IFM to NoSQL applications, an extension of IFM has been proposed in [12] that considers more structured schemes for the datasets to be generated, as required by emerging big data applications, e.g., social network analytics. A many-sorted extension of IFM called *ms-IFM* is defined by replacing the basic simple schema  $R(K, A)$  with a more general NoSQL schema  $R(K, A_1, \dots, A_p, A_1, \dots, A_q)$ , where  $K$  is the table key,  $A_1, \dots, A_p$  are single-valued attributes, and  $A_1, \dots, A_q$  are multi-valued attributes. This problem reduces to classical IFM when  $p = 0$  and  $q = 1$ , i.e., there is exactly one multi-valued attribute.

In this paper we describe the formal definition of *ms-IFM* of [12] and illustrate a method for solving *ms-IFM* that has been shown to be effective by means of a number of experiments. The solver is based on large-scale linear programming and generalizes the approach used in [11] to solve IFM<sub>I</sub>.

## 2. Many-Sorted IFM

Let a NoSQL *relation*  $R(K, A_1, \dots, A_p, A_{p+1}, \dots, A_{p+q})$  be given, where  $K$  is the table key,  $A_1, \dots, A_p$  are classical single-valued (SV) attributes and  $A_{p+1}, \dots, A_{p+q}$  are multi-valued (MV) attributes. We assume that the attributes are ordered, i.e.,  $K$  is the first attribute,  $A_1$  the second attribute,  $A_{p+1}$  the  $(1 + p + 1)$ -th attribute and so on. For each  $i$ ,  $1 \leq i \leq p + q$ , let  $\mathcal{A}_i$  be the finite domain for the attributes  $A_i$  and  $|\mathcal{A}_i| = n_i$ . We assume that the values of every domain  $\mathcal{A}_i$  (called *items*) are given in input – they are SV items or MV items depending on whether the attribute  $A_i$  is SV or MV. On the other hand, the domain  $\mathcal{K}$  of the key  $K$  is countably infinite and, then, its values are not listed.

Let  $\dot{n} = \sum_{i=1}^p n_i$ ,  $\ddot{n} = \sum_{i=p+1}^{p+q} n_i$  and  $n = \dot{n} + \ddot{n}$ . A NoSQL *tuple* on  $R$  is of the form  $t = [k, a_1, \dots, a_p, g_1, \dots, g_q]$ , where  $k \in \mathcal{K}$ , for each  $i$ ,  $1 \leq i \leq p$ ,  $a_i \in \mathcal{A}_i$  ( $a_i$  is an *item* of  $t$ ) and for each  $i$ ,  $1 \leq i \leq q$ ,  $g_i \subseteq \mathcal{A}_{p+i}$  ( $g_i$  is an *itemset* of  $t$ ). Items and itemsets are called *values* of  $t$ . A NoSQL *table* on  $R$  is a finite set of NoSQL tuples.

A *many-sorted transaction*  $T$  is a  $(p + q)$ -tuple  $[a_1, \dots, a_p, g_1, \dots, g_q]$  where the key attribute value is dropped out from a NoSQL tuple. It follows that a many-sorted transaction can be transformed into a tuple simply by inventing a key for it. Given any attribute  $A_i$  in  $R$  and a many-sorted transaction  $T$ ,  $T_{A_i}$  denotes the value of  $T$  for the attribute  $A_i$  (e.g.,  $a_i \in \mathcal{A}_i$  if  $i \leq p$  or  $g_{i-p} \subseteq \mathcal{A}_i$  if  $i > p$ ).

Let  $\mathcal{T}$  be the set of all many-sorted transactions. The cardinality of  $\mathcal{T}$  is  $n_{\mathcal{T}} = 2^{\dot{n}} \cdot \prod_{i=1}^p \dot{n}_i$ . A *many-sorted dataset*  $\mathcal{D}$  is set of pairs  $(T, \delta^{\mathcal{D}}(T))$ , where  $T$  is a many-sorted transaction and  $\delta^{\mathcal{D}}(T)$  is the number of occurrences of  $T$ .  $\mathcal{D}$  can be transformed into a NoSQL table  $T_{\mathcal{D}}$  by making  $\delta^{\mathcal{D}}(T)$  tuple duplicates of each many-sorted transaction  $T$ . The *cardinality* of  $\mathcal{D}$ , denoted as  $|\mathcal{D}|$ , is the number of pairs in  $\mathcal{D}$  and the *size* of  $\mathcal{D}$  is  $\delta^{\mathcal{D}} = |T_{\mathcal{D}}| = \sum_{T \in \mathcal{D}} \delta^{\mathcal{D}}(T)$ . We stress that in general  $\delta^{\mathcal{D}} \gg |\mathcal{D}|$ . From now on, we shall omit the term *many-sorted* whenever it is clear from the context.

A *sub-transaction*  $S$  is a  $(p + q)$ -tuple  $[a_1, \dots, a_p, g_1, \dots, g_q]$  on  $R$  for which the domain of each SV attribute is extended with  $\perp$ , which stands for a null value. Let  $\perp(S)$  denote the number of null values in  $S$  – a transaction  $T$  can be also seen as a sub-transaction type for which  $\perp(T) = 0$ . The *length* of  $S$ , denoted as  $l(S)$ , is the number of values different from  $\perp$  and  $\emptyset$ . A sub-transaction  $S$  *subsumes* a transaction  $T$  (written  $S \sqsubseteq T$ ) if for each SV attribute  $A_i$ , either  $S.A_i = \perp$  or  $S.A_i = T.A_i$ , and for each MV attribute  $A_i$ ,  $S.A_i \subseteq T.A_i$ . Observe that if  $S$  happens to be a transaction then every transaction  $T$  for which  $S \sqsubseteq T$  has the same SV items as  $S$ , whereas its MV itemsets are supersets of the corresponding itemsets in  $S$ . In the classical IFM setting, every transaction type  $S$  coincides with an itemset and the transactions  $T$  subsumed by  $S$  are all itemsets for which  $S \subseteq T$ . This analogy explains why we write  $S \sqsubseteq T$  for transaction subsumption.

Given a sub-transaction  $S$  for which  $l(S) > 0$  and two integers  $\sigma_1$  and  $\sigma_2$  for which  $0 \leq \sigma_1 \leq \sigma_2$ ,  $\gamma = \langle S, \sigma_1, \sigma_2 \rangle$  represents a *frequency support constraint* defined as: a database  $\mathcal{D}$  satisfies  $\gamma$  (written as  $\mathcal{D} \models \gamma$ ) if:  $\sigma_1 \leq \sum_{T \in \mathcal{D} \wedge S \sqsubseteq T} \delta^{\mathcal{D}}(T) \leq \sigma_2$ . An *infrequency support constraint*  $\gamma$  is a pair  $\langle S, \sigma \rangle$  and is actually a shorthand for the frequency support constraint  $\langle S, 0, \sigma \rangle$ . The upper bound  $\sigma$  will be simply referred to as  $\gamma_{\#}$ . A (frequency or infrequency) support constraint  $\gamma$  for which  $l(\gamma_S) = 1$  is called a *domain support constraint*. Given a set  $\Pi$  of support constraints, a database  $\mathcal{D}$  satisfies  $\Pi$  (written as  $\mathcal{D} \models \Pi$ ), if for each  $\gamma \in \Pi$ ,  $\mathcal{D} \models \gamma$ .

*Example.* Individuals are characterized by the SV attributes *Gender*, *Location* and *Age* and by the MV attributes *Groups* and *Events*: an individual may belong to various groups and may attend a number of events. A transaction  $I = [Male, Rome, 25, \{g_1, g_4\}, \{e_1, e_3\}]$  represents a 25-year old male individual located in Rome who belongs to the groups  $g_1$  and  $g_4$  and attends the events  $e_1$  and  $e_3$ . The transaction  $J = [Female, Rome, 20, \{g_1, g_2\}]$  represents an individual who does not attend any event. Note that, as the attributes do not define a key, there may exist several occurrences of the same individual. Examples of constraints are:

- $\langle [Male, Rome, \perp, \{g_1, g_2\}, \emptyset], 10000, 20000 \rangle$  fixes the range for the overall duplicate number of male individuals who are located in Rome and are participating to at least the groups  $g_1$  and  $g_2$ ;
- $\langle [Female, \perp, 25, \{g_1, g_2\}, \{e_1, e_3\}], 500, 1000 \rangle$  fixes the range for the overall duplicate number of 25-year old female individuals who are participating to at least the groups  $g_1$  and  $g_2$  and

attending at least the events  $e_1$  and  $e_3$ ;

*Infrequency constraints:*

- $\langle [\perp, \text{Cosenza}, \perp, \{g_1, g_2\}, \emptyset], 100 \rangle$  states that the number of individuals located at Cosenza in a feasible dataset who are participating to at least the groups  $g_1$  and  $g_2$  is at most 100;
- $\langle [\perp, \perp, \perp, \emptyset, \{e_1, e_2\}], 10000 \rangle$  states that the number of individuals attending at least the events  $e_1$  and  $e_2$  is at most 10000;
- $\langle [\perp, \perp, \perp, \{g_1\}, \emptyset], 100000 \rangle$  is a domain support constraint stating that the number of individuals participating to at least the group  $g_1$  is at most 100000.  $\square$

From now on, we assume that the following sets of constraints are given: (1) a set  $\Sigma$  of *frequency support constraints* with cardinality  $m = |\Sigma| > 0$  and (2) a set  $\widehat{\Sigma}$  of *infrequency support constraints* with cardinality  $\widehat{m} = |\widehat{\Sigma}| \geq 0$ . Let  $\check{\Sigma} = \Sigma \cup \widehat{\Sigma}$  and  $\check{m} = |\check{\Sigma}| = m + \widehat{m}$ .

Observe that, as the number of infrequency constraints could be very large, they are not induced as for the IFM<sub>T</sub> case [11]. but explicitly defined. The drawback of this formulation is that it is not anymore excluded to eventually obtain undesired frequent itemsets. But an accurate choice of explicit infrequent itemsets may reduce this risk.

**Definition.** Given  $R$ ,  $\check{\Sigma} = \Sigma \cup \widehat{\Sigma}$ , and an integer  $\text{size} > 0$ , the *multi-sorted inverse frequent itemset mining problem*, shortly denoted as *ms-IFM*, consists of finding a many-sorted dataset  $\mathcal{D}$  on  $R$  such that both  $\delta^{\mathcal{D}} = \text{size}$  and  $\mathcal{D} \models \check{\Sigma}$  (or of eventually stating that there is no such a dataset). If the integer constraint for the number of duplicates for a transaction of a database  $\mathcal{D}$  is relaxed (i.e., it is a rational number), the problem is called *relaxed ms-IFM*.  $\square$

By assuming that items, constraint bounds and  $\text{size}$  are stored using a constant amount of space, the input size of the problem is  $\mathcal{O}(n + \check{m}(p + q\check{n} + 1) + m)$ . It is easy to see that *ms-IFM* reduces to the IFM<sub>T</sub> problem if  $p = 0$  and  $q = 1$ , i.e., there exists exactly one non-key attribute in  $R$  and this attribute is of MV type, say  $G$ . A transaction is then any itemset on the domain  $\mathcal{G}$  of  $G$ . The next result shows that the complexity of *ms-IFM* and of *relaxed ms-IFM*.

**Proposition.** The decision version of *ms-IFM* is in PSPACE and NP-hard and the decision version of *relaxed ms-IFM* is NP-complete.  $\square$

Note that the higher complexity of the IFM<sub>T</sub> problem, which has been proved to be NEXP-complete in [11], derives from the task of discovering “minimal” itemsets that must be enforced to be infrequent: in IFM<sub>T</sub> minimal infrequent itemsets are computed by the resolution algorithm, whereas they are assumed to be part of the input for *ms-IFM*.

To solve the relaxed *ms-IFM*, in the next sections we define formulate *ms-IFM* as a linear program and present an extension of the classical simplex algorithm for its resolution.

## 2.1. Linear Program Formulation

We select an ordering of all *many-sorted transactions* in  $\mathcal{T}$ , say  $\{T_1, \dots, T_{n_{\mathcal{T}}}\}$ . We use the vector  $v = [1, \dots, n_{\mathcal{T}}]$  to list all possible non-empty *many-sorted transaction* indices. Let the vector  $s = [i_1, \dots, i_m, j_1, \dots, j_{\widehat{m}}]$  represent the indices of the frequency support constraints in  $\Sigma$  and

the infrequency support constraint  $\widehat{\Sigma}$ . Let  $l$  and  $u$  be two vectors of  $m$  integers such that for each  $j, 1 \leq j \leq m, l_j = \sigma_{min}^J$  and  $u_j = \sigma_{max}^J$ , where  $J = \gamma_{s_j}$  is the  $j$ -th frequency support constraints in  $\Sigma$  according to the ordering fixed by  $s$ . Let  $u'$  be the vector of  $\widehat{m}$  integers such that for each  $j, m \leq j \leq m + \widehat{m}, u'_j = \sigma_{max}^J$ , where  $J = \gamma_{s_j}$  is the  $j$ -th infrequency support constraints in  $\Sigma$  according to the ordering fixed by  $s$ . Let  $x$  be a vector of  $n_{\mathcal{T}}$  non-negative rational variables whose meaning is that its  $j$ -th coordinate,  $x_j$ , denotes the number of duplicates for the many-sorted transaction  $T_j$ . We consider a  $(m + \widehat{m}) \times n_{\mathcal{T}}$  matrix  $A$  where for each  $i, 1 \leq i \leq m + \widehat{m}$ , and for each  $j \in v, a_{ij} = 1$  if  $S \sqsubseteq T_j$  where  $\gamma_{s_i} = \langle S, \sigma_1, \sigma_2 \rangle$  or  $a_{ij} = 0$  otherwise. As usual, we introduce a vector  $w$  of  $2m + 1$  non-negative rational number artificial variables, whose values represent the costs of violating some support constraints.

The relaxed linear formulation of the  $ms$ -IFM is the following:

$$\begin{aligned} \text{LP : minimize } & \sum_{i=1}^{2m+1} w_i \quad \text{- subject to} \\ & w_i + \sum_{j \in v} a_{ij} x_j \geq l_i && 1 \leq i \leq m \\ & w_{m+i} - \sum_{j \in v} a_{ij} x_j \geq -u_i && 1 \leq i \leq m \\ & - \sum_{j \in v} a_{ij} x_j \geq -u'_j && m + 1 \leq i \leq m + m' \\ & w_{2m+1} + \sum_{j \in v} x_j \geq \text{size} && - \sum_{j \in v} x_j \geq \text{size} \end{aligned}$$

For solving the linear program we use the *column generation* approach (see e.g. [13]), which is an extension of the simplex method for dealing with linear programs with a large number of variables. The column generation simplex solves a linear program without explicitly including all columns (i.e., variables), in the coefficient matrix but only a subset of them with cardinality equal to the number of rows (i.e., constraints). Columns are dynamically generated by solving an auxiliary optimization problem called the *pricing problem*.

The linear program to be solved is denoted as the *master problem* (MP). In our case the MP problem consists of  $r = 2m + m' + 2$  rows and  $c = 2^n + 2m$  columns.

The linear program with only a subset of the MP columns with cardinality  $c'$  equal to the number  $r$  of rows is called the *restricted master problem* (RMP). Then, the restricted master problem is the master problem with a subset of columns w.r.t. the original master problem. From the theory of the simplex algorithm we know that a basic solution for a linear program is an assignment of the variables that satisfy the constraints and where only  $r$  (number of rows are activated) variables are assigned with a value greater or equal than zero (i.e., the variables in the basic solution), the remaining ones are zero. Starting from an initial basic solution, the simplex algorithm will iteratively find the optimal basic solution. Since each variable is assigned to a column, it turns out that if the RMP contains the columns associated to the variables in the optimal base solutions of the MP, then the RMP will return the same solution. Then the column

generation approach is an iterative algorithm that working on the restricted master problem will keep the number of columns bounded and iterates till the RMP does not contain the columns associated with optimal base solution of the MP.

The column generation method looks for an optimal basis as within the simplex algorithm. It starts from an initial basis and moves from a current basis to a new one by adding a new basic column with a negative reduced cost (*iteration step*). Primal feasibility is maintained and the objective function is non-increasing during this search. The reduced cost of a column can be computed by using the current dual variables. The task of providing a column with a negative reduced cost, or certifying that there is not such a column, is delegated to the pricing problem. If there is no column with a negative reduced cost, then the algorithm terminates and the current basis is optimal.

The crucial task is the formulation and resolution of the pricing problem that is done by using integer linear program. Due to page restrictions, we defer this issue to the full paper [12].

## 2.2. Empirical evaluation of Many-Sorted IFM

To evaluate the accuracy of the solutions computed by our *ms*-IFM approximated approach, we show that synthetic data can successfully replace original data in an analytics task, in particular *classification*, which is the problem of identifying to which of a set of categories (subpopulations), a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known [14]. To this end, we set up a classification problem by representing a transactional dataset with a standard dataset for classification, where each item corresponds to a binary attribute (i.e., 1 if present and 0 otherwise). To create several different classification tasks for each dataset used in the experiments, we selected the top-*k* most frequent items. For each of such items, we consider this item as the dependent feature (i.e., the binary attribute that we want to classify) and the remaining top-*k* items as the independent features (i.e., the input of the classification model). For each synthetic dataset generated by *ms*-IFM techniques, we train a classification model for each of the top-*k* most frequent items and we test the trained model on the original dataset.

In the experiments, we use 3 different classification models [14]: *Logistic Regression*, *Decision Tree*, and *Random Forest*. All of these models are trained with datasets and specific procedures to balance the training sets are executed. For each model and for each dataset, we compute the accuracy of the classification model, which was first trained and tested on the original dataset. The accuracy computed while testing classification on the original dataset (denoted as *original accuracy*) can be assumed to be the best classification accuracy value that any synthetic dataset generation approach can achieve.

To create instances for our *ms*-IFM, we used the Yelp dataset [15] to extract a No-SQL table. In particular, each transaction of the No-SQL table represents a business in Yelp (e.g., restaurant, hospital, etc.). In total there are 101,824 transactions. The table has 4 attributes: STARS (the average number of stars for each business – a SV attribute with 9 values), STATE (the state where the business is located – a SV attribute with 50 values), CATEGORIES (the categories to which a particular business belongs – a MV value attribute with 397 distinct values, e.g., garage, restaurant, space for kids, etc.). To create the frequency and infrequency support many-sorted constraints, we converted this NOSQL table in a transactional dataset.

**Table 1**

Accuracy for the 30-top frequent items with  $s(\%) \in \{1.2, 1.6\}$

classifier	Original	$S(\%) = 1.2$	$S(\%) = 1.6$
Logistic Regression	0.99	0.93	0.93
Decision Tree	0.99	0.85	0.84
Random Forest	0.99	0.92	0.92

In Table 1, we report the average classification results obtained for 30-top frequent items in the two different datasets obtained by means of two selected supports.

From Table 1 it is possible to see that the synthetic datasets generated are pretty close to the original dataset in term of accuracy. In addition, we did not observe a strong variation when the support changes. Further accuracy evaluations are reported in [12].

### 3. Conclusion

The *Inverse Frequent itemset Mining* (IFM) problem consists of generating a transactional database satisfying given support constraints on some itemsets, which are typically the frequent ones. In order to enlarge the application domain, an extension of IFM, called multi-sorted IFM (*ms-IFM*), has been introduced in [12] that considers more structured schemes for the datasets to be generated, as required in emerging big data applications, e.g., social network analytics.

This paper has given an overview of *ms-IFM* problem definition, linear program formulation, resolution using a column generation algorithm (an extension of the simplex method for dealing with linear programs with a large number of variables), and an experimental evaluation of the capability of the synthetic datasets to perform the data mining task of classification with an accuracy close to the one achieved with the original datasets.

### References

- [1] K. Michael, K. W. Miller, Big Data: New Opportunities and New Challenges [Guest editors' Introduction], *Computer* 46 (2013) 22–24. doi:<http://doi.ieeecomputersociety.org/10.1109/MC.2013.196>.
- [2] H. Wu, Y. Ning, P. Chakraborty, J. Vreeken, N. Tatti, N. Ramakrishnan, Generating Realistic Synthetic Population Datasets, *ACM Trans. Knowl. Discov. Data* 12 (2018) 45:1–45:22. URL: <http://doi.acm.org/10.1145/3182383>. doi:10.1145/3182383.
- [3] R. Agrawal, T. Imieliński, A. Swami, Mining Association Rules Between Sets of Items in Large Databases, in: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, SIGMOD '93*, ACM, New York, NY, USA, 1993, pp. 207–216.
- [4] D. Gunopulos, R. Khardon, H. Mannila, H. Toivonen, Data mining, Hypergraph Transversals, and Machine Learning, in: A. O. Mendelzon, Z. M. Özsoyoglu (Eds.), *Proceedings of the 16-th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '97*, ACM Press, 1997, pp. 209–216.

- [5] T. Mielikainen, On Inverse Frequent Set Mining, in: Proceedings of 2nd Workshop on Privacy Preserving Data Mining, PPDM '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 18–23.
- [6] R. Agrawal, R. Srikant, Privacy-preserving Data Mining, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00, ACM, New York, NY, USA, 2000, pp. 439–450. doi:10.1145/342009.335438.
- [7] X. Wu, Y. Wu, Y. Wang, Y. Li, Privacy Aware Market Basket Data Set Generation: A Feasible Approach for Inverse Frequent Set Mining, in: Proceedings of SIAM International Conference on Data Mining, SDM' 05, SIAM, Philadelphia, PA, USA, 2005, pp. 103–114.
- [8] T. Calders, Computational Complexity of Itemset Frequency Satisfiability, in: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '04, ACM, New York, NY, USA, 2004, pp. 143–154. doi:10.1145/1055558.1055580.
- [9] T. Calders, The Complexity of Satisfying Constraints on Databases of Transactions, *Acta Informatica* 44 (2007) 591–624.
- [10] A. Guzzo, D. Saccà, E. Serra, An Effective Approach to Inverse Frequent Set Mining, in: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, ICDM '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 806–811. URL: <http://dx.doi.org/10.1109/ICDM.2009.123>. doi:10.1109/ICDM.2009.123.
- [11] A. Guzzo, L. Moccia, D. Saccà, E. Serra, Solving Inverse Frequent Itemset Mining with Infrequency Constraints via Large-scale Linear Programs, *ACM Trans. Knowl. Discov. Data* 7 (2013) 18:1–18:39. URL: <http://doi.acm.org/10.1145/2541268.2541271>. doi:10.1145/2541268.2541271.
- [12] D. Saccà, E. Serra, A. Rullo, Extending Inverse Frequent Itemsets Mining to Generate Realistic Datasets: Complexity, Accuracy and Emerging Applications, *Data Min. Knowl. Discov.* 33 (2019) 1736–1774. URL: <https://doi.org/10.1007/s10618-019-00643-1>. doi:10.1007/s10618-019-00643-1.
- [13] P. C. Gilmore, R. E. Gomory, A Linear Programming Approach to the Cutting-Stock Problem, *Operations Research* 9 (1961) 849–859. URL: <http://or.journal.informs.org/cgi/doi/10.1287/opre.11.6.863>.
- [14] J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, Kaufmann, San Francisco, CA, USA, 2005. URL: <http://www.amazon.com/Data-Mining-Concepts-Techniques-Management/dp/1558604898>.
- [15] Yelp Challenge, 2017. URL: <https://www.yelp.com/dataset>.