Lattice-based Discovery of Hybrid Relaxed Functional Dependencies (Discussion Paper)

Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese

University of Salerno, Department of Computer Science
via Giovanni Paolo II n.132, 84084 Fisciano (SA), Italy
{lcaruccio,deufemia,gpolese}@unisa.it

Abstract. Relaxed functional dependencies (RFDs) are properties expressing important relationships among data. Thanks to the introduction of approximations in data comparison and/or validity, they can capture constraints useful for several purposes, such as the identification of data inconsistencies or patterns of semantically related data. Nevertheless, RFDs can provide benefits only if they can be automatically discovered from data. In this discussion paper we present an RFD discovery algorithm relying on a lattice structured search space, and a new candidate RFD validation method. An experimental evaluation demonstrates the discovery performances of the proposed algorithm on real datasets.

Keywords: Functional Dependency · Discovery Algorithm · Approximate Match · Constraint Mining.

1 Introduction

Functional dependencies (FDs) were originally used to verify database design and assess schema quality. In the last decades, they have been used for several new purposes, such as data profiling [1], query relaxation, data cleansing, and so forth. To this end, their definition has been often extended in order to express constraints in these and other emerging application domains [7].

Recent literature refers to extended FD definitions with the term *relaxed functional dependencies* (RFDs) [3]. In particular, there exist RFDs relaxing on the data comparison, which compare tuples by using data similarity rather than equality, those relaxing on the *extent*, which admit the possibility for the RFD to hold only on a subset of data, and finally, hybrid ones, which relax on both criteria. Thresholds might be used in all categories, either to specify the similarity degree or the minimum percentage of tuples on which the RFD should hold.

In order to exploit FDs in practical domains several algorithms to discover them from data have been proposed [2,16]. However, RFDs are much more complex to specify at design time, since they also require the specification of thresholds for evaluating the similarity between attribute values and/or for deriving the minimum extent.

In this discussion paper, we describe the algorithm proposed in [6], namely DiMε, which is able to mine many different types of RFDs, relaxing on the tuple comparison and/or on the extent by using several tuple comparison methods and coverage measures [3]. The discovery technique underlying DiMε generates candidate RFDs based on Difference Matrices created from input thresholds, and on a lattice structure, also used within FD discovery algorithms [2], in order to model the search space. Moreover, DiMε provides a new validation technique.

The paper is organized as follows. Section 2 reviews the RFD discovery algorithms existing in the literature. Section 3 provides some background definitions concerning RFDs. Section 4 presents the DiMε algorithm, whereas Section 5 shows the experimental results. Concluding remarks are included in Section 6.

2 Related Work

The automatic discovery of FDs is accomplished either through column-based or row-based methods [1]. The former start generating candidate FDs based on an attribute lattice, verifying their validity, and then using holding FDs to prune the search space for candidate FDs yet to be verified [2]; the latter compare attribute values for each pair of tuples, in order to generate two different sets of attribute subsets, namely *agree-sets* and *difference-sets*, from which candidate FDs are derived [8,16]. Finally, an hybrid algorithm has been proposed in [12]. With respect to these algorithms, DiMε discovers a broader class of dependencies beyond FDs, which entails much more a complex validation phase. In fact, the validation of FDs merely requires to compute cardinalities of equivalence classes, whereas the validation of RFDs requires more complex computations on intersecting classes induced by similarity functions.

Several discovery algorithms have been proposed for *Matching dependencies*, e.g., [15], and *Differential Dependencies*, e.g., [14], two examples of RFD relaxing on attribute comparison. While these discovery algorithms are each focused on a specific RFD, DiMε is designed for a more generic class of RFDs [4], including RFDs relaxing on the tuple comparison method, those relaxing on the extent, and finally, hybrid ones relaxing on both.

3 Relaxed Functional Dependencies

Informally, an RFD is a functional dependency that relaxes on the tuple comparison, by using constraints on the distance or similarity between attribute values, and/or that relaxes on the extent, by using a coverage measure to indicate the minimum number or percentage of tuples on which the RFD must hold, and/or by using conditions restricting the applicability domain of the RFD. In what follows, we recall a formal definition of RFD from our previous survey [3].

Definition 1 (Relaxed functional dependency). *Consider a relational database schema $\mathcal{R}$, and a relation schema $R = (A_1, \ldots, A_m)$ of $\mathcal{R}$. An* RFD *φ on $\mathcal{R}$ is denoted by*

$$X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2} \tag{1}$$

where

- $X = X_1, \ldots, X_h$ *and* $Y = Y_1, \ldots, Y_k$, *with* $X, Y \subseteq attr(R)$ *and* $X \cap Y = \varnothing$;
- $\Phi_1 = \bigwedge_{X_i \in X} \phi_i[X_i]$ ($\Phi_2 = \bigwedge_{Y_j \in Y} \phi_j[Y_j]$, *resp.), where* ϕ_i (ϕ_j, *resp.) is a conjunction of predicates on* X_i (Y_j, *resp.) with* $i = 1, \ldots, h$ ($j = 1, \ldots, k$, *resp.). For any pair of tuples* $(t_1, t_2) \in dom(R)$, *the constraint* Φ_1 (Φ_2, *resp.) is true if* $t_1[X_i]$ *and* $t_2[X_i]$ ($t_1[Y_j]$ *and* $t_2[Y_j]$, *resp.) satisfy the constraint* ϕ_i (ϕ_j, *resp.)* $\forall\ i \in [1, h]$ ($j \in [1, k]$, *resp.).*
- Ψ *is a coverage measure defined on* $dom(R)$, *quantifying the amount of tuples violating or satisfying* φ. *Among the most commonly used coverage measures there are the confidence, the* $g3$-error, *and the probability.*
- ε *is a threshold indicating the upper bound (or lower bound in case the comparison operator is* $\geq$*) for the result of the coverage measure.*

Given $r \subseteq dom(R)$ a relation instance on R, r *satisfies* the RFD φ, denoted by $r \vDash \varphi$, if and only if: $\forall\ t_1, t_2 \in r$, if Φ_1 indicates true, then *almost always* Φ_2 indicates true. Here, *almost always* is expressed by the constraint $\Psi \leq \varepsilon$. A more general definition of RFD is provided in [3].

As an example, in a database of scientific publications it is likely to have the same address and affiliation for authors with the same name. Thus, an FD $\{\mathsf{Author}\} \rightarrow \{\mathsf{Address}, \mathsf{Affiliation}\}$ might hold. However, these attributes might have been stored using different abbreviations, hence the following RFD might hold: $\mathsf{Author}_{\phi_1} \rightarrow \{\mathsf{Address}_{\phi_2}, \mathsf{Affiliation}_{\phi_3}\}$, where ϕ_1, ϕ_2, and ϕ_3 are constraints on a string similarity function. Moreover, since authors might change affiliation during their life, or there might be homonimies, the previous RFD should tolerate possible exceptions. This can be modeled by the following RFD:

$$\mathsf{Author}_{\phi_1} \xrightarrow{\psi(Author, Address, Affiliation) \leq 0.02} \{\mathsf{Address}_{\phi_2}, \mathsf{Affiliation}_{\phi_3}\}$$

4 The DiMε Discovery Algorithm

The discovery of RFDs is the problem of finding a set of *minimal* RFDs holding on a relation instance r. An RFD $X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2}$ is *minimal* if there not exists a subset $Z \subset X$ such that $Z_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2}$ holds on r.

For sake of simplicity and w.l.o.g, in the rest of the article we assume that Y consists of a single attribute.

The *Difference Matrix and* ε*-threshold* (DiMε) discovery algorithm presented in [6] starts from user-specified thresholds, and performs a level-by-level generation of candidate dependencies to be successively validated. Candidate RFD generation is accomplished by means of an attribute lattice built by considering all the possible attribute combinations. Thus, DiMε generates attribute sets X, and then formulates all the possible RFDs $X \backslash \{A\} \rightarrow A$, with $A \in X$, to be validated. With respect to FD discovery algorithms, DiMε exploits their candidate generation strategies, and generalizes their pruning strategies, but it provides a new validation technique dealing with similarity subsets, and taking into consideration the possibility that an RFD might hold for a subset of tuples.

Tuple number	Height	Weight	Shoe size
1	175	70	40
2	175	75	39
3	175	69	40
4	176	71	40
5	178	81	41
6	169	73	37
7	170	62	39

Table 1: A sample dataset.

	1	2	3	4	5	6	7
1	0	0	0	1	3	6	5
2	0	0	0	1	3	6	5
3	0	0	0	1	3	6	5
4	1	1	1	0	2	7	6
5	3	3	3	2	0	9	8
6	6	6	6	7	9	0	1
7	5	5	5	6	8	1	0

(a) $M_{\mathsf{Height}_{\mathsf{abs}}}$

	1	2	3	4	5	6	7
1	0,0	0,5	0,1	1,1	3,11	6,3	5,8
2	0,5	0,0	0,6	1,4	3,6	6,2	5,13
3	0,1	0,6	0,0	1,2	3,12	6,4	5,7
4	1,1	1,4	1,2	0,0	2,10	7,2	6,9
5	3,11	3,6	3,12	2,10	0,0	9,8	8,19
6	6,3	6,2	6,4	7,2	9,8	0,0	1,11
7	5,8	5,13	5,7	6,9	8,19	1,11	0,0

(b) $M_{\mathsf{Height}_{\mathsf{abs}},\mathsf{Weight}_{\mathsf{abs}}}$

Fig. 1: Difference Matrices for the dataset shown in Table 1.

Candidate RFD validation The validation of candidate RFDs entails verifying that two tuples are similar on the RHS attribute, whenever they are similar on the LHS ones, and the property holds for a significant portion of the database. This is done by generating similarity subsets of tuples and using them to validate candidate RFDs. As opposed to tuple partitions built in most FD discovery algorithms, similarity subsets might also intersect. To better explicate the generation process of similarity subsets, we introduce the concept of *difference matrix*.

Definition 2 (Difference matrix of an attribute). *Let r be an instance of a relation schema R, A an attribute of R, and δ a distance function defined on the domain of A. The* difference matrix *for A is a matrix M_A whose entry (i,j) contains the value $\delta(t_i[A], t_j[A])$ of the projections of tuples t_i and t_j on A.*

As an example, let us consider the sample dataset in Table 1. Since it has all numerical attributes, we use the absolute difference, denoted with *abs*, as a distance function to construct the difference matrix $M_{\mathsf{Height}_{\mathsf{abs}}}$ (Figure 1(a)).

The definition of difference matrix can be easily generalized to attribute sets, in which an entry will contain an n-tuple of distance values, one for each attribute in the set. The difference matrix for an attribute set X can be derived from the difference matrices for the single attributes composing it, by concatenating elements with the same coordinates. An example is provided in Figure 1(b).

We exploit the notion of difference matrix to generate the *similar pattern* of a tuple t or a matrix row, which represents sets of tuples or matrix columns satisfying a given constraint wrt t. Successively, we group the tuples sharing the same similar patterns into *similar pattern subsets*.

Definition 3 (Similar pattern subsets). *Let r be an instance of a relation schema R, $X = \{A_1, \ldots, A_n\}$ an attribute set of R, M_{X_Δ} a difference matrix for X, and $\phi = (\phi_1, \ldots, \phi_n)$ a sequence of constraints on the values of M_{X_Δ}. A similar pattern of tuple t_i of M_{X_Δ}, denoted as $\tau_X^{t_i}$, is the sequence $(j_1, \ldots, j_h)$ with $h \leq |r|$ and $M[i, j_k] = (d_1, \ldots, d_n)$, where d_q satisfies the constraints ϕ_q $\forall q \in [1, n]$ and $\forall k \in [1, h]$. A* similar pattern subset S_X *for X is defined as $S_X = \{j_1, \ldots, j_h\}_{\{i_1, \ldots, i_k\}}$ with $1 \leq k \leq h \leq |r|$, $\tau_X^{i_p} = (j_1, \ldots, j_h)$ $\forall p \in [1, k]$ and $\tau_X^{i_v} \neq (j_1, \ldots, j_h)$ $\forall v \notin [1, k]$. The set of different* similar pattern subsets *for X is denoted as I_X.*

Example 1. If in the dataset of Table 1 the user specifies constraints based on the *abs* function, the $\leq$ comparison operator, 1 as a threshold for both the attributes Height and Shoe Size, and 10 for the attribute Weight, then the following sets of similar pattern subsets are generated:

- $I_{\text{Height}} = \{\{1, 2, 3, 4\}_{\{1,2,3,4\}}, \{5\}_{\{5\}}, \{6, 7\}_{\{6,7\}}\}$;
- $I_{\text{Weight}} = \{\{1, 2, 3, 4, 6, 7\}_{\{1,3\}}, \{1, 2, 3, 4, 5, 6\}_{\{2,6\}}, \{1, 2, 3, 4, 5, 6, 7\}_{\{4\}}, \{2, 4, 5, 6\}_{\{5\}}, \{1, 3, 4, 7\}_{\{7\}}\}$;
- $I_{\text{Shoe Size}} = \{\{1, 2, 3, 4, 5, 7\}_{\{1,3,4\}}, \{1, 2, 3, 4, 7\}_{\{2,7\}}, \{1, 3, 4, 5\}_{\{5\}}, \{6\}_{\{6\}}\}$.

In order to validate candidate RFDs, DiMε reduces the number of similar pattern subsets for the attributes of the LHS, by eliminating singletons (since they correspond to the matrix diagonal entries that trivially compare each tuple with itself) obtaining *stripped similar pattern subsets.*

Validation of RFDs relaxing only on the tuple comparison. Stripped similar pattern subsets can be used to validate candidate RFDs based on the concept of *similar pattern subset refinement.*

Definition 4. *A set $\widehat{I}_X$ is said to* refine *another set $\widehat{I}_{X \cup A}$ if every similar pattern subset in $\widehat{I}_X$ is contained in one of the subsets in $\widehat{I}_{X \cup A}$.*

Based on the refinement notion, we derive the following lemma for RFDs.

Lemma 1. *A relaxed functional dependency $X_{\Phi_1} \rightarrow A_{\Phi_2}$ holds on an entire database instance if and only if $\widehat{I}_X$* refines *$\widehat{I}_A$.*

Example 2. Let us consider the database instance of Table 1. If we set the constraints as defined in Example 1, then according to Lemma 1 the RFD

$$\{\text{Height}_{\phi_1}, \text{Weight}_{\phi_2}\} \rightarrow \text{Shoe Size}_{\phi_3}$$

holds on the entire database. In fact, since $\widehat{I}_{\text{Height,Weight}} = \{\{1, 2, 3, 4\}_{\{1,2,3,4\}}\}$ and $\widehat{I}_{\text{Shoe\_Size}} = \{\{1, 2, 3, 4, 5, 7\}_{\{1,3,4\}}, \{1, 2, 3, 4, 7\}_{\{2,7\}}, \{1, 3, 4, 5\}_{\{5\}}\}$, each similar pattern in $\widehat{I}_{\text{Height,Weight}}$ is included in a similar pattern of $\widehat{I}_{\text{Shoe\_Size}}$.

Alternatively, the refinement property between sets of stripped similar pattern subsets can be verified by calculating $||\widehat{I}_Z||$, which is defined as:

$$||\widehat{I}_Z|| = \frac{\sum_{t_i \in r} (|s_{t_i}| - 1)}{2} \forall s_{t_i} \in \widehat{I}_Z \tag{2}$$

It represents the number of pairwise similar tuples in $\widehat{I}_Z$. Since for an attribute set $W \subset Z$, $\widehat{I}_Z$ always refines $\widehat{I}_W$, it means that $\widehat{I}_{X \cup A}$ always refines $\widehat{I}_X$. However, we know that an RFD $X_{\Phi_1} \rightarrow A_{\Phi_2}$ holds only if $\widehat{I}_X$ is a refinement of $\widehat{I}_A$. Thus, since $\widehat{I}_{X \cup A}$ cannot have similar pattern subsets of size greater than those in $\widehat{I}_X$, the RFD holds if $\widehat{I}_{X \cup A}$ and $\widehat{I}_X$ are equal, which yields the following lemma.

Lemma 2. *A relaxed functional dependency $X_{\Phi_1} \rightarrow A_{\Phi_2}$ holds on an entire database instance if and only if $\|\widehat{I}_{X \cup A}\| = \|\widehat{I}_X\|$.*

Validation of hybrid RFDs. In case of hybrid RFDs, DiMε should also consider as valid the RFDs holding on a subset of tuples, according to a coverage measure and a user-defined threshold. In this case, it is not possible to use the previously defined lemma to validate candidate RFDs. In particular, to accomplish the validation process it is necessary to calculate the satisfiability degree of the instance according to a coverage measure. In this paper, DiMε relies on the *g*3-*error* coverage measure [9], but it can also work with other coverage measures by overloading the function computing the measure from similar subsets.

The computation of the *g*3-*error* for RFDs with hybrid relaxation is an NP-complete problem [13], since the MINIMUM VERTEX COVER problem can be reduced to it. A solution to this problem can be found in polynomial time in case of disjoint similar pattern subsets, in which case, we can use the same strategy of AFD discovery [9]. However, since in case of intersecting similar pattern subsets each of these tuples can fall in more than one similar pattern subset of I_X, it is not possible to perform a computation local to each similar pattern subset, but more a global analysis is required. Thus, we provide an greedy solution for calculating the *g*3-*error* in polynomial time [10].

5 Evaluation

We implemented DiMε[1] in the Java, by using the Levenshtein distance for comparing textual attributes, the absolute difference for comparing numerical ones, and the *g*3-*error* as the coverage measure for extent evaluation [11]. The experiments have been performed on a machine with an Intel Xeon W 3.2GHz 8-core CPU, 64 GB RAM, with a 64-Bit Java environment. We considered six real-world datasets from the UCI Machine Learning repository[2]. During the experiment sessions we varied the similarity and the coverage measure thresholds simultaneously, in the range $[0, 4]$ with step 1, and in the range $[0, 0.4]$ with step 0.1. Only, for the Abalone dataset we considered the range $[0.0, 0.4]$.

Figure 2 shows the number of RFDs extracted by DiMε according to several tuple comparison thresholds, whereby each line represents a different extent threshold. We can observe that the highest number of RFDs is usually discovered with *g*3-*error* thresholds slightly above zero. Concerning the variation of tuple

[1] https://dastlab.github.io/dime/.

[2] https://archive.ics.uci.edu/ml/index.php

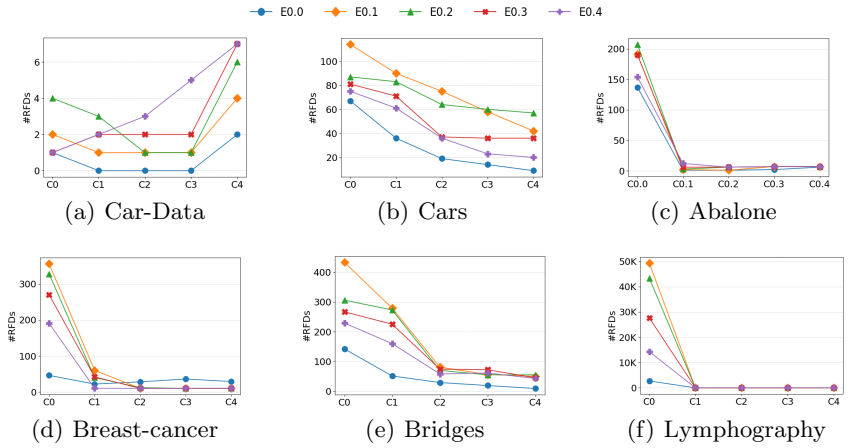

Fig. 2: Number of RFDs varying the similarity and the extent thresholds.

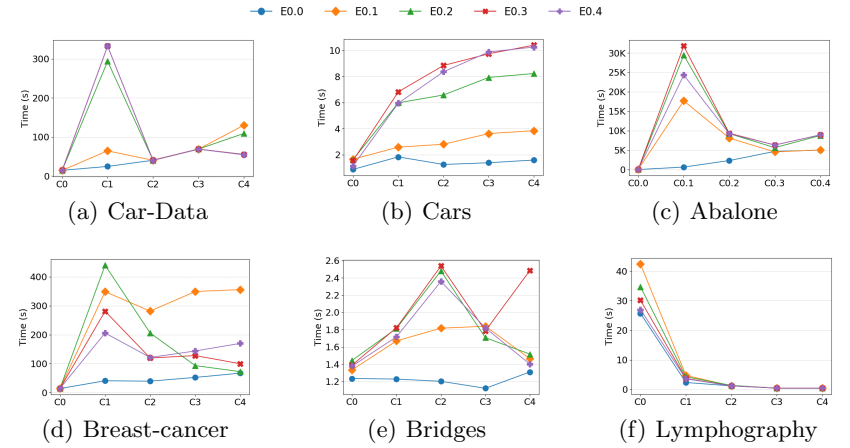

Fig. 3: Runtimes by varying the similarity and the extent thresholds.

comparison thresholds, in most cases the number of RFDs drastically drops when increasing them from 0 to 1. This is mainly due to the fact that with a zero threshold many key dependencies are obtained, which are likely to be invalidated when the tuple comparison threshold becomes greater than zero.

Figure 3 shows the execution times of DiMε. In general, although we expect higher execution times on datasets with more columns and/or rows, this is not necessarily true, because the attribute value distributions might heavily affect the computation steps to be performed, since they influence the number of tuple similarities, and hence the number of RFD candidates to be processed.

6 Conclusion and future work

The problem of discovering RFDs adds a considerable complexity to the dependency discovery process, since the relaxation criteria reduce the possibilities of pruning search paths, and prevent the possibility to exploit the properties of disjoint partitions during the validation phase. The performed evaluation highlights the effectiveness of DiMε in the discovery of RFDs relaxing on both the tuple comparison and the extent dimensions. In the future, we would like to further investigate the considered problem in order to derive an algorithm for RFD discovery capable of automatically inferring the threshold ranges of their validation, as done in [5], instead of requesting them to the user.

References

1. Abedjan, Z., Golab, L., Naumann, F.: Profiling relational data: A survey. The VLDB Journal **24**(4), 557–581 (2015)
2. Abedjan, Z., Schulze, P., Naumann, F.: DFD: Efficient functional dependency discovery. In: CIKM '14. pp. 949–958 (2014)
3. Caruccio, L., Deufemia, V., Polese, G.: Relaxed functional dependencies – A survey of approaches. IEEE TKDE **28**(1), 147–165 (2016)
4. Caruccio, L., Deufemia, V., Polese, G.: Evolutionary mining of relaxed dependencies from big data collections. In: WIMS '17. pp. 5:1–5:10 (2017)
5. Caruccio, L., Deufemia, V., Polese, G.: Discovering relaxed functional dependencies based on multi-attribute dominance. IEEE TKDE (2020), to appear
6. Caruccio, L., Deufemia, V., Polese, G.: Mining relaxed functional dependencies from data. Data Min. Knowl. Discov. **34**(2), 443–477 (2020)
7. Chang, S.K., Deufemia, V., Polese, G., Vacca, M.: A normalization framework for multimedia databases. IEEE TKDE **19**(12), 1666–1679 (2007)
8. Flach, P.A., Savnik, I.: Database dependency discovery: A machine learning approach. AI communications **12**(3), 139–160 (1999)
9. Huhtala, Y., Kärkkäinen, J., Porkka, P., Toivonen, H.: TANE: An efficient algorithm for discovering functional and approximate dependencies. The Computer Journal **42**(2), 100–111 (1999)
10. Johnson, D.S., Garey, M.R.: Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman & Co. (1979)
11. Kivinen, J., Mannila, H.: Approximate inference of functional dependencies from relations. Theoretical Computer Science **149**(1), 129–149 (1995)
12. Papenbrock, T., Naumann, F.: A hybrid approach to functional dependency discovery. In: SIGMOD '16. pp. 821–833 (2016)
13. Song, S.: Data Dependencies in the Presence of Difference. Ph.D. thesis, The Hong Kong University (2010)
14. Song, S., Chen, L.: Differential dependencies: Reasoning and discovery. ACM Transactions on Database Systems **36**, 16 (2011)
15. Song, S., Chen, L.: Efficient discovery of similarity constraints for matching dependencies. Data & Knowledge Engineering **87**, 146–166 (2013)
16. Wyss, C., Giannella, C., Robertson, E.: FastFDs: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract. In: DaWaK '01. pp. 101–110 (2001)