Certus: An Effective Entity Resolution Approach with Graph Differential Dependencies (GDDs)

Selasi Kwashie Jixue Liu Jiuyong Li
Lin Liu Markus Stumptner Lujing Yang
School of ITMS, University of South Australia, Adelaide, Australia
selasi.kwashie@mymail.unisa.edu.au, firstname.lastname@unisa.edu.au

ABSTRACT

Entity resolution (ER) is the problem of accurately identifying multiple, differing, and possibly contradicting representations of unique real-world entities in data. It is a challenging and fundamental task in data cleansing and data integration. In this work, we propose graph differential dependencies (GDDs) as an extension of the recently developed graph entity dependencies (which are formal constraints for graph data) to enable approximate matching of values. Furthermore, we investigate a special discovery of GDDs for ER by designing an algorithm for generating a non-redundant set of GDDs in labelled data. Then, we develop an effective ER technique, Certus, that employs the learned GDDs for improving the accuracy of ER results. We perform extensive empirical evaluation of our proposals on five real-world ER benchmark datasets and a proprietary database to test their effectiveness and efficiency. The results from the experiments show the discovery algorithm and Certus are efficient; and more importantly, GDDs significantly improve the precision of ER without considerable trade-off of recall.

PVLDB Reference Format:
Selasi Kwashie, Jixue Liu, Jiuyong Li, Lin Liu, Markus Stumptner, and Lujing Yang. Certus: An Effective Entity Resolution Approach with Graph Differential Dependencies (GDDs). *PVLDB*, 12(6): 653-666, 2019.
DOI: https://doi.org/10.14778/3311880.3311883

1. INTRODUCTION

Identifying different records/objects in data that refer to the same real-world entity is an inherent task in many research fields, particularly, in data cleansing, data integration, and information retrieval. In data integration for instance, one would like to find different records (with possible contradictions) in one or multiple databases that refer to the same person in the real-world; or to detect different citations that refer to the same research paper. This problem is well-known in many more research communities, and ironically studied under different names, viz.: entity resolution, merge/purge, deduplication, record linkage, etc.

Proceedings of the VLDB Endowment, Vol. 12, No. 6
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3311880.3311883

In general, the causes of multiple representations of distinct real-world entities in data can be categorised into two groups [38]: (i) the inadvertent creation of multiple records for a unique entity in the same database; and (ii) the integration of different representations of the same real-world entity from different sources into the same database. These oversights are recurrent in data due to the existence of noise and inconsistencies. Noise here means 'small errors' (e.g., a typo 'Peter' for 'Peters'); inconsistency means conflicting and/or different values for properties/relations of an object.

Since the above-mentioned causes are almost unavoidable, it is crucial to design techniques capable of detecting duplicates in data. In fact, the need for entity resolution cannot be overemphasised as its applications are in numerous areas including law enforcement, e-commerce, and government. In law enforcement, for example, linking different representations of entities from different sources like the traditional structured databases and non-structured sources (e.g., entities mentioned in investigation/complaint reports) can be invaluable to an officer, and help to apprehend suspects.

A common approach to solving the ER problem in the database community is through the use of the so-called *record-matching rules*. Record-matching rules are constraints that simply state: "if any two records are *similar on certain properties*, then they *refer to the same entity*". This strategy is known as *rule-based* ER. In data-centric systems, rule-based ER solutions are often preferable to the non-rule-based counterparts [46], although the latter usually outperform the former (cf. a recent survey in [14]). A major reason for the preference of rule-based ER systems is their interpretability, which permits explicit encoding of domain knowledge [10] and interactive debugging of results [39].

The rule-based ER approach, however, has the following three problems indicated in [53]: (a) it is difficult to generate record-matching rules; (b) it is challenging to define the bounds of similarity for approximate-match conditions; and (c) efficient support for approximate-match conditions is non-trivial. To address (a), matching dependencies (MDs) [17,19] and conditional MDs (CMDs) [54] have been proposed as new classes of dependencies for ER in relational data. To resolve (b), the works in [53] and [46] provide optimization techniques based on disjunctive normal forms and global boolean formulae respectively, whereas dependency discovery alrigthms are used in [48,50,54]. And to tackle (c), indexes are used in [7,15] for efficient approximate matching.

In this work, we tackle the problem of ER over both structured and semi-structured data using the rule-based approach. Consider the profiles/representations of entities

Entities from semi-structured sources

pid: 1	**pid**: 2	**pid**: 3	**pid**: 4
name: John Wil. Smith	**name**: J Green-Smeeth	**name:** Bill J. Schmidt	**name**: Jane Green
sex: male	**gender**: f	**eye-colour:** blue	**dob**: 3/12/92
height: 1.82m	**dob**: 3/12/92	**born_on:** 25-02-1978	**eye-colour**: brown
dob: 2/25/87	**wife_of**: [1]	**tall:** 183cm	**nationality**: AUS
husband_of: [2]		**friends:** [4]	**friends**: [3]

Entities from structured sources

pid: 5	**pid**: 6	**pid**: 7	**pid**: 8	**pid**: 9
str. #: 121	**surname**: Williams	**str. #**: 67	**surname**: Billy	**surname**: Jennifer
str. name: George Rd	**middle name**: -	**str. name**: Main Str	**middle name**: Jon	**middle name**: Mary
city: Adelaide	**first name**: John	**city**: Sydney	**first name**: Smythe	**first name**: Green
state: SA	**dob**: 2000-12-18	**state**: NSW	**dob**: 1978-02-25	**dob**: 1992-12-03
country: Australia	**height**: 159	**country**: Australia	**height**: 178	**height**: 156
address_of: [8]	**sex**: 0		**sex**: 0	**sex**: 1
address_of: [9]			**lives_at**: [5]	**lives_at**: [5]
			husband_of: [9]	**wife_of**: [8]

Figure 1: Entity profiles from two sources

in Figure 1 from two sources, for example. The data in the first source have less regular properties/names and values, while the data in the second source have regular properties except for relationships. Our goal is to identify all profile pairs that refer to the same entities in the real-world, across both sources, irrespective of structure and with no assumed schema. In this scenario, current record-matching rules, e.g., MDs [17, 19] and CMDs [54], cannot be employed as they are limited to relational (structured) data.

We adopt a graph model to represent the profiles of entities which enables formal representation of profiles in even non-structured sources. Then, we investigate a new class of dependencies for graph data to allow the generation and use of rules for ER in graph data. Indeed, we extend the recently proposed *graph entity dependency* (GED) [24] to include the semantics of similarity and matching for use as declarative matching rules. Next, we developed a discovery algorithm to learn matching rules in duplicate-labelled graph data, thus, finding various bounds of similarity for attributes and relations for linking profiles. Moreover, we design an elaborate method for using the learned rules for effective ER.

Our contributions in this paper are summarized as follows.
1) We propose a new class of dependencies for graphs, namely graph differential dependencies (GDDs). GDDs extend GEDs by incorporating distance and matching functions *instead of* equality functions. GDDs subsume GEDs and their relational counterparts (e.g., MDs, CMDs). We present basic inference rules for the implication analysis of GDDs.
2) We define the set of non-redundant GDDs and propose an algorithm for the discovery of a class of GDDs for entity resolution, namely Linking GDDs, $\mathsf{GDD_L}$s. This discovery problem is more challenging and complex than previously studied discovery problems, involving the discovery of graph patterns, distance/matching constraints and constant values. We design an efficient algorithm to mine a non-redundant set of $\mathsf{GDD_L}$s in graph data with known duplicate labels.
3) We develop a GDD-based ER method, called Certus, to find different representations of unique entities in graph data. The approach leverages several pruning strategies to avoid the computational cost of applying GDDs to all profiles pairs. Certus allows the use of a low similarity score (for high recall) and GDDs to eliminate false-positives (for high precision).
4) Finally, we perform and report the experimental evaluations of the GDD discovery and ER algorithms using five real-world ER benchmark datasets and a proprietary dataset. The results show the efficiency and scalability of our proposals. More importantly, the results show GDDs improve precision of ER without significant sacrifice of recall.

The rest of the paper is organised as follows. Section 2: preliminary concepts and definitions. Section 3: the ER problem. Section 4: the formal definition, syntax, and semantics of GDDs. Section 5: a special GDD discovery for ER.

Table 1: List of frequently used notations

Symbol	Description
$G, Q[\bar{z}]$	entity profiles graph, graph pattern resp.
X, Y	sets of attributes/relations
Φ_X, Φ_Y	sets of distance functions on X, Y resp.
ER	entity resolution
GED	graph entity dependency
GDD, $\mathsf{GDD_L}$	graph differential dependency, linking GDD resp.

Section 6: our GDD-based ER solution. Section 7: evaluation of all proposals. Section 8: a summary of related works in the literature. Section 9: concluding remarks.

2. PRELIMINARIES

This section presents key definitions. We use $\mathbf{A}$ and $\mathbf{L}$ to denote the finite sets of *attributes* and *labels* respectively; and Table 1 lists frequently used notations in this work.

2.1 Entity Profiles, Graphs & Graph Patterns

Entity Profile. An *entity profile* for a real-world entity is a tuple $p = \langle \mathsf{pid}, \mathsf{eid}, \mathsf{type}, P, R \rangle$ where pid is the identity of the profile, eid is the identity of the real-world entity represented by the profile (often unknown/unavailable), type is the type of the entity (e.g., person or location), P is a list of attribute A and value c pairs, (A, c), of the entity, and R is a list of relation-label and pid pairs, $(\mathsf{rela}, \mathsf{pid}')$, describing the relation rela of p with another entity represented by pid'. For any pair (A, c), $A \in \mathbf{A}$ and $c \in dom(A)$ – domain of A.

If two entity profiles p_1 and p_2 represent the same real-world entity, then $p_1.\mathsf{eid} = p_2.\mathsf{eid}$.

Example 1 (Profile)**.** Figure 1 presents entity profiles from two sources (semi-structured and structured) with unknown eids. The first profile with $\mathsf{pid} = 1$ can be represented as a tuple $p = \langle \mathsf{pid}_1, \mathsf{eid}_1, \mathsf{type}_1, P_1, R_1 \rangle$, where $\mathsf{pid}_1 = 1$, $\mathsf{eid}_1 = \mathtt{unknown}$, $\mathsf{type}_1 = $ person, $P_1 = [(\mathsf{name}, \text{'John Wil. Smith'}), (\mathsf{sex}, \text{male}), (\mathsf{height}, 1.82\text{m}), (\mathsf{dob}, \text{'2/25/87'})]$ and $R_1 = [(\mathsf{husband\_of}, 2)]$. □

Entity Profiles Graph. The definitions of entity profiles graph and graph pattern follow those in [24, 25].

An *entity profiles graph* (a profiles graph) is a directed graph $G = (V, E, L, F_A)$, where: (i) V is a finite set of nodes; (ii) E is a finite set of edges, given by $E \subseteq V \times V$; (iii) each node $v \in V$ (resp. edge $e \in E$) has a label $L(v)$ (resp. $L(e)$) drawn from $\mathbf{L}$; (iv) each node $v \in V$ has an associated list $F_A(v) = [(A_1, c_1), \cdots, (A_n, c_n)]$ of attribute-value pairs, where $A_i \in \mathbf{A}, c_i \in dom(A_i)$ and $A_i \neq A_j$ if $i \neq j$. For an attribute $A \in \mathbf{A}$ and a node $v \in V$, $v.A$ may not exist.

An entity profile $p_i = \langle \mathsf{pid}_i, \mathsf{eid}_i, \mathsf{type}_i, P_i = [(A, c), \cdots], R_i = [(\mathsf{rela}, \mathsf{pid}_j), \cdots] \rangle$ is encoded as a node v_i in a profiles graph G as the following: $v_i = \mathsf{pid}_i$, $L(v_i) = \mathsf{type}_i$, $F_A(v_i) = P_i$, and for each $(\mathsf{rela}, \mathsf{pid}_j) \in R_i$, there exists node v_j representing profile $p_j = \langle \mathsf{pid}_j, \mathsf{eid}_j, \mathsf{type}_j, P_j, R_j \rangle$ and the edge $e_{ij} = (v_i, v_j)$ exists in E such that $\mathsf{rela} = L(e_{ij})$.

Informally, a profiles graph is a collection of entity profiles.

Example 2 (Profiles Graph)**.** Figure 2(a) shows an entity profiles graph, G, for the 9 (i.e., 7 `person`, 2 `location`) profiles and their relationships from Figure 1. A node in the graph represents a profile and edges from a node are relationships of the profile. For instance, nodes $\{6, 7\}$ have no relationship; nodes $\{1, 2, 3, 4\}$ have one relationship each; and nodes $\{5, 8, 9\}$ have 2 relationships each. □

Next, we define the concept of graph pattern: a constraint that aims to select/match a sub-graph from/to a graph.

Graph Pattern. A *graph pattern* is a directed graph $Q[\bar{z}, \mathcal{C}] = (V_Q, E_Q, L_Q)$, where: (i) V_Q (resp. E_Q) is a finite set of pattern nodes (resp. pattern edges); (ii) L_Q is a function that assigns a label to each node $v \in V_Q$ and to each edge $e \in E_Q$; (iii) $\bar{z}$ is all the nodes, called (pattern) variables, in V_Q; and (iv) $\mathcal{C}$ is a list of conditions on V_Q and E_Q, often omitted when the conditions are presented with a diagram. All labels are drawn from $\mathbf{L}$, including the wildcard, '$*$', as a special label.

Two labels $l_1, l_2 \in \mathbf{L}$ are said to *match*, denoted by $l_1 \asymp l_2$, iff: (a) $l_1 = l_2$, or (b) $l_1 =$ '$*$', or (c) $l_2 =$ '$*$'.

A *match* of pattern $Q[\bar{z}]$ in graph G is a homomorphism h from Q to G such that: (i) for each node $v \in V_Q$, $L_Q(v) \asymp L(h(v))$; and (ii) for each edge $e_1 = (u, v)$ in Q, there exists an edge $e_2 = (h(u), h(v))$ in G such that $L(e_1) \asymp L(e_2)$.

Example 3 (Graph Pattern)**.** Figure 2(b) is an illustration of four graph patterns Q_1, Q_2, Q_3 and Q_4 specified over entity profiles. The matches of these patterns in the profiles graph G (i.e., Figure 2(a)) are described below. Q_1 aims to match any pair of entities to z_1, z_2 without type restriction, because of '$*$', and no requirement on relations. Q_2 seeks to match two `person` entities without requirement on relations. Q_3 aims to match two pairs of `person` entities if each pair has the relation l_1 between them. Q_4 seeks to match any two `person` entities and a `location` entity if the two `person` entities have the relation l_2 with the `location` entity. In Figure 2(a), any two nodes match Q_1. Any pair of `person` nodes are a *match* of Q_2. A match for Q_3 is $\{(1, 2, 8, 9)\}$ if $l_1 \asymp l_h$. The only match for Q_4 is $\{(8, 5, 9)\}$ if $l_2 \asymp l_l$. ☐

2.2 Graph Entity Dependencies (GEDs)

A new class of dependencies for graphs, namely, *graph entity dependencies* (GEDs) was proposed in [24] recently, which subsumes graph functional dependencies (GFDs) [25] and graph keys (GKeys) [18], with useful applications in capturing inconsistencies and errors in graph data.

In this section, we recall the definition of GEDs, and in Section 4, extend them for use in entity resolution.

GEDs. [24] A GED ψ is a pair $(Q[\bar{z}], X \to Y)$, where $Q[\bar{z}]$ is a graph pattern, and X and Y are two (possibly empty) sets of constraints on $\bar{z}$. $Q[\bar{z}]$ and $X \to Y$ are referred to as the pattern and FD of ψ, respectively. A *constraint* on $\bar{z}$ in X and Y is one of: $\{x.A = c,\ x.A_1 = x'.A_2,\ x.\mathsf{eid} = x'.\mathsf{eid}\}$, where $x, x' \in \bar{z}$ are pattern variables, $A, A_1, A_2 \in \mathbf{A}$ are attributes.

Given a GED $\psi = (Q[\bar{z}], X \to Y)$ and a match $h(\bar{z})$ of $Q[\bar{z}]$ in a graph G, $h(\bar{z})$ satisfies a constraint $w \in X$, denoted by $h \models w$, iff for $x, x' \in \bar{z}$: (i) when w is $x.A = c$, then the attribute $h(x).A$ exists, and $h(x).A = c$; (ii) when w is $x.A_1 = x'.A_2$, then the attributes $h(x).A_1$ and $h(x').A_2$ exist and have the same value; and (iii) when w is $x.\mathsf{eid} = x'.\mathsf{eid}$, then $h(x).\mathsf{eid} = h(x').\mathsf{eid}$.

A match $h(\bar{z})$ satisfies X if $h(\bar{z})$ satisfies every $w \in X$. In the same way, $h(\bar{z}) \models Y$ is defined.

A match $h(\bar{z})$ satisfies $X \to Y$, denoted by $h(\bar{z}) \models X \to Y$, if $h(\bar{z}) \models X$ implies $h(\bar{z}) \models Y$.

Let $H(\bar{z})$ be all the matches of $Q[\bar{z}]$ in G. $H(\bar{z})$ satisfies ψ, denoted by $H(\bar{z}) \models \psi$, if for every $h(\bar{z}) \in H(\bar{z})$, $h(\bar{z}) \models X \to Y$. If $H(\bar{z}) \models \psi$, then $G \models \psi$. G satisfies a set Ψ of GEDs if G satisfies every $\psi \in \Psi$.

Example 4 (GEDs)**.** Consider the patterns defined in Figure 2(b). The GED $\psi_1 = (Q_2[x_1, x_2], X_1 \to Y_1)$ where $X_1 = \{x_1.\mathsf{name} = x_2.\mathsf{name}, x_1.\mathsf{dob} = x_2.\mathsf{dob}, \cdots\}$ and $Y_1 = \{x_1.\mathsf{eid} = x_2.\mathsf{eid}\}$ states the constraint that for any pair of `person` profiles (i.e., $Q_2[x_1, x_2]$), if they have the same values on attributes name, dob, etc., they must have the same eid, implying that they refer to the same real-world person. ☐

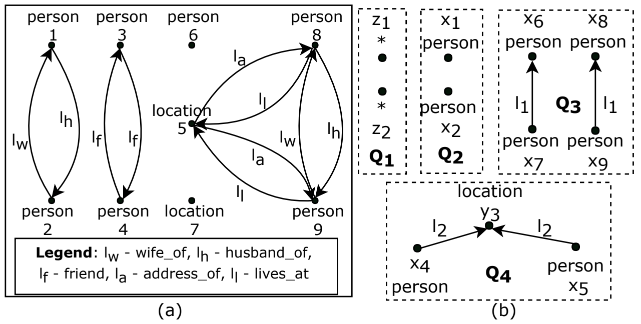

person 1
person 3
person 6
person 8
location 5
person 2
person 4
location 7
person 9
l_w
l_h
l_f
l_a
l_l
Legend: l_w - wife_of, l_h - husband_of, l_f - friend, l_a - address_of, l_l - lives_at
(a)
z_1 * z_2 Q_1
x_1 person x_2 person Q_2
x_6 person x_8 person x_7 person x_9 person l_1 Q_3
location y_3 l_2 x_4 person x_5 person Q_4
(b)

Figure 2: Profiles graph G - (a), Graph patterns $Q[\bar{z}]$ - (b)

GED Limitations. GEDs subsume some dependencies as special cases. For instance, GFDs [25], GKeys [18], and even relational FDs [4], CFDs [21] and equality-generating dependencies [9] can be expressed by GEDs.

However, GEDs employ exact matching of values and cannot capture the semantics of approximate matches in data. In many applications, data values have typos and variations. GEDs are therefore not suitable in such situations.

The need for dependencies to express approximate matches in data for data quality applications is well-known (cf. [19, 20, 49]). Examples of such dependencies in the relational data include matching [19] and differential [49] dependencies. These are useful in many applications in data quality management and data cleansing, in particular, entity resolution. In Section 4, we propose Graph Differential Dependencies (GDDs) which are extended from GEDs. GDDs capture similarity based matches in data and can serve our entity linking task in graph data.

3. PROBLEM FORMULATION

This section presents the ER problem as well as the motivation and overview of our solution.

The problem of entity resolution is to find all the maximal clusters, called *linked clusters*, of nodes in a given graph of entity profiles such that all the profiles in each cluster refer to the same real-world entity. That is, for any two profiles p_1 and p_2 in a cluster, $p_1.\mathsf{eid} = p_2.\mathsf{eid}$.

An algorithm for finding linked clusters is called a linking algorithm. Unfortunately, a linking algorithm does not work perfectly, resulting in some incorrect outcomes. When the algorithm concludes that two profiles satisfy $p_1.\mathsf{eid} = p_2.\mathsf{eid}$, the conclusion may be true (true-positive, TP) if the equivalence is confirmed by fact. The conclusion may also be false (false-positive, FP). Furthermore, the algorithm may fail to find some clusters and/or some profiles that should belong to a cluster. The missed ones are called false-negatives (FN). A good algorithm aims to achieve high precision (TP/(TP+FP)) and high recall (TP/(TP+FN)). Our aim is to find a good algorithm for ER in graph data.

Attaining both high precision and high recall is a challenging task for a linking algorithm. For instance, a linking algorithm can achieve a higher precision by using exact match of attribute values. However, because of noise in data (like spelling differences/errors) and missing attributes, the recall of linking in this case will be low. On the other hand, if approximate match of attribute values are used, the recall may be high, but the precision will diminish.

To overcome this struggle between precision and recall, we propose a solution that does *not* require a user to define specific thresholds for various value matches, but set a single reasonably low overall threshold of similarity, to ensure a high recall. Then, our solution uses GDDs to optimize and balance the precision and recall. More specifically, our linking algorithm relies on the discovery of GDDs to learn critical attributes, relations and best thresholds for approximate matches in ER. Formally, we will learn GDDs of the form $(Q[\{x, x'\}], \{\delta_{A_iA_j}(x.A_i, x'.A_j) \leq t_{A_iA_j}\} \rightarrow \delta_{\equiv}(x.\mathsf{eid}, x'.\mathsf{eid}) = 0)$ – *see next section for explanation to notations* – including the learning of the attributes A_i, A_j and their distance threshold $t_{A_iA_j}$ from an eid-labelled graph in Section 5. Then, we apply the learned GDDs to find linked clusters in entity profiles graph with *unknown* eid-labels.

4. GDD - AN EXTENSION OF GED

We now extend GEDs to capture the semantics of distance-based match as *graph differential dependency.*

4.1 Graph Differential Dependencies (GDDs)

GDD. A GDD σ is a pair $(Q[\bar{z}], \Phi_X \rightarrow \Phi_Y)$, where: $Q[\bar{z}]$ is a graph pattern called the *scope*, $\Phi_X \rightarrow \Phi_Y$ is called the *dependency*, Φ_X and Φ_Y are two (possibly empty) sets of distance constraints on the pattern variables $\bar{z}$. A *distance constraint* in Φ_X and Φ_Y on $\bar{z}$ is one of the following:

$$\delta_A(x.A, c) \leq t_A; \qquad \delta_{A_1A_2}(x.A_1, x'.A_2) \leq t_{A_1A_2};$$

$$\delta_{\equiv}(x.\mathsf{eid}, c_e) = 0; \qquad \delta_{\equiv}(x.\mathsf{eid}, x'.\mathsf{eid}) = 0;$$

$$\delta_{\equiv}(x.\mathsf{rela}, c_r) = 0; \qquad \delta_{\equiv}(x.\mathsf{rela}, x'.\mathsf{rela}) = 0;$$

where $x, x' \in \bar{z}$, A, A_1, A_2 are attributes in $\mathbf{A}$, c is a value of A, $\delta_{A_iA_j}(x.A_i, x'.A_j)$ (or $\delta_{A_i}(x, x')$ if $A_i = A_j$) is a user specified distance function for values of (A_i, A_j), $t_{A_1A_2}$ is a threshold for (A_i, A_j), $\delta_{\equiv}(\cdot, \cdot)$ are functions on eid and relations and they return 0 or 1. $\delta_{\equiv}(x.\mathsf{eid}, c_e) = 0$ if the eid value of x is c_e, $\delta_{\equiv}(x.\mathsf{eid}, x'.\mathsf{eid}) = 0$ if both x and x' have the same eid value, $\delta_{\equiv}(x.\mathsf{rela}, c_r) = 0$ if x has a relation named rela and ended with the profile/node c_r, $\delta_{\equiv}(x.\mathsf{rela}, x'.\mathsf{rela}) = 0$ if both x and x' have the relation named rela and ended with the same profile/node. □

The user-specified distance function $\delta_{A_1A_2}(x.A_1, x'.A_2)$ is dependent on the types of A_1 and A_2. It can be an arithmetic operation of interval values, an edit distance of string values or the distance of two categorical values in a taxonomy, etc. The functions handle the wildcard value '*' for any domain by returning the 0 distance.

We call Φ_X and Φ_Y the LHS and the RHS functions of the dependency respectively.

GDD satisfaction. (i) Given a GDD $\sigma = (Q[\bar{z}], \Phi_X \rightarrow \Phi_Y)$ and a match h of $Q[\bar{z}]$ in a graph G, h satisfies Φ_X, denoted by $h \vDash \Phi_X$, if h satisfies every distance constraint in Φ_X. $h \vDash \Phi_Y$ is defined in the same way. (ii) Let $H(\bar{z})$ be all the matches of $Q[\bar{z}]$ in a graph G. $H(\bar{z}) \vDash \sigma$ if $h \vDash \Phi_Y$ is true for every $h \in H(\bar{z})$ that $h \vDash \Phi_X$. (iii) The graph G satisfies σ, denoted by $G \vDash \sigma$ if $H(\bar{z}) \vDash \sigma$. □

A GDD is a constraint on graph data and can be used to enforce consistency. It can also be used to represent latent knowledge in data from a discovery point of view. In addition, it can be used to infer properties and relations of entities. This last point will be used in later sections for the inference of entity matches (in entity resolution).

Below are some cases where GDDs can be used to enforce consistency. A GDD requires that for a match h of $Q[\bar{z}]$ in a graph G, if h satisfies Φ_X, it should also satisfies Φ_Y.

1. Consider an example where h has two location (loc) profiles x_1 and x_2, Φ_X requires an approximate match on suburb, subb, allowing one-character difference, and Φ_Y requires an exact zip-code (zip) match. Then the GDD for the example is: $(Q[\{x_1, x_2\}, L(x_1) = \mathsf{loc}, L(x_2) = \mathsf{loc}], \Phi_X = \{\delta_{\mathsf{subb}}(x_1, x_2) \leq 1\} \rightarrow \Phi_Y = \{\delta_{\mathsf{zip}}(x_1, x_2) = 0\})$. If a graph does not satisfy these requirements, the graph is not valid.

2. Another GDD relating to Q_4 in Figure 2 is $\sigma_4 = (Q_4[\{x_4, x_5, y_3\}], \Phi_X \rightarrow \Phi_Y)$, where $\Phi_X = \{\emptyset\}$ and $\Phi_Y = \{\delta_{\equiv}(x_4.\mathsf{friend}, x_5.\mathsf{pid}) = 0, \delta_{\equiv}(x_5.\mathsf{friend}, x_4.\mathsf{pid}) = 0\}$. σ_4 specifies a constraint that for any match of Q_4, two mutual 'friend' relations between x_4 and x_5 must exist. This means, if two persons live at same location, they must be friends. Otherwise, the constraint is violated. Note that if we exchange Φ_X and Φ_Y in σ_4, the rule loses its power.

Example 5 (GDDs in ER)**.** Consider the entity profiles graph G, the graph pattern Q_2 in Figure 2, and a GDD $\sigma_1 = (Q_2[\{x_1, x_2\}], \Phi_{X1} \rightarrow \Phi_{Y1})$, where $\Phi_{X1} = \{\delta_{\mathsf{name}}(x_1, x_2) \leq 2, \delta_{\mathsf{dob}}(x_1, x_2) \leq 2\}$, and $\Phi_{Y1} = \{\delta_{\equiv}(x_1.\mathsf{eid}, x_2.\mathsf{eid}) = 0\}$. This GDD states that for any pair of `person` profiles, if their name and dob values are similar, then they refer to the same real-world person. □

GDDs & other Dependencies. The introduction of distance semantics in GDDs is non-trivial as it allows superior expressivity and wider application, with consequences of more challenging axiomatization and reasoning problems. Thus, the inference axioms and reasoning results of GEDs do not directly hold for GDDs and require full investigation.

The major difference between our proposed GDDs and GEDs in [24] is twofold. First, the constraints in our definition allow errors in value matching. This opens up GDDs' suitability to many real-world applications where data is noisy. In contrast, GEDs use exact match. Secondly, our GDDs unlike GEDs allow constraints on relations within the LHS and RHS functions of the dependency. Note that both GDDs and GEDs allow relations in the scope pattern $Q[\bar{z}(\mathcal{C})]$. However, having constraints like $\delta_{\equiv}(x.\mathsf{rela}, c_r) = 0$ and $\delta_{\equiv}(x.\mathsf{rela}, x'.\mathsf{rela}) = 0$ in the dependency $\Phi_X \rightarrow \Phi_Y$ is different from having them in the pattern $Q[\bar{z}]$. This is because, having more constraints in $Q[\bar{z}(\mathcal{C})]$ restricts the scope to which dependencies apply; and not all matches of a pattern need to satisfy a dependency. These flexibilities make GDDs much more expressive and useful in more applications.

GDDs subsume dis/similarity-based dependencies in relational data, e.g., differential dependencies (DDs) [49], conditional DDs [33], matching dependencies (MDs) [19] and conditional MDs [54]. In fact, unlike the distance-based dependencies (in relational data), GDDs are capable of expressing and enforcing constraints on relationships in data, and can be used in both structured and semi-structured data.

4.2 Inferring GDDs

In the following, we discuss some implication results of GDDs and introduce the concept of *irreducible* GDDs.

First, we define an order relation, namely subjugation, for distance constraints. This relation indicates which distance constraints are more restrictive.

Table 2: Inference rules for GDDs

$\mathcal{I}_1$	If $\Phi_Y \succcurlyeq \Phi_X$, then $G \models (Q[\bar{z}], \Phi_X \rightarrow \Phi_Y)$, for any G.
$\mathcal{I}_2$	If $G \models (Q[\bar{z}], \Phi_X \rightarrow \Phi_Y)$, then $G \models (Q[\bar{z}], \Phi_X \cup \Phi_Z \rightarrow \Phi_Y \cup \Phi_Z)$, for any non-empty set of constraints, Φ_Z.
$\mathcal{I}_3$	If $G \models (Q[\bar{z}], \Phi_X \rightarrow \Phi_{Z_1})$, $G \models (Q[\bar{z}], \Phi_{Z_2} \rightarrow \Phi_Y)$, and $\Phi_{Z_2} \succcurlyeq \Phi_{Z_1}$, then $G \models (Q[\bar{z}], \Phi_X \rightarrow \Phi_Y)$ holds.

Subjugation. Given two sets, Φ_{X_1} and Φ_{X_2}, of distance constraints on the same set of pattern variables $\bar{z}$ in $Q[\bar{z}]$, Φ_{X_1} *subjugates* Φ_{X_2}, denoted by $\Phi_{X_1} \succcurlyeq \Phi_{X_2}$ iff (i) for every constraint $\delta_{A_iA_j}(x.A_i, y.A_j) \leq t^{(1)}_{A_iA_j} \in \Phi_{X_1}$, there exists $\delta_{A_iA_j}(x.A_i, y.A_j) \leq t^{(2)}_{A_iA_j} \in \Phi_{X_2}$ and $t^{(1)}_{A_iA_j} \geq t^{(2)}_{A_iA_j}$. Other forms of distance constraints with a threshold are defined in the same way. (ii) for every match constraint $\delta_{\equiv}(\cdot,\cdot) = 0$ in Φ_{X_1}, it must also be in Φ_{X_2}.

The intuition of subjugation is that if a match satisfies Φ_{X_2}, then it *also* satisfies Φ_{X_1}. That is, Φ_{X_2} has more and tighter constraints while Φ_{X_1} has less and looser constraints. For example, $\Phi_{X_1} \succcurlyeq \Phi_{X_2}$ is true if $\Phi_{X_1} = \{\delta_{\mathsf{age}}(x, x') < 2\}$ and $\Phi_{X_2} = \{\delta_{\mathsf{age}}(x, x') < 1, \delta_{\mathsf{height}}(x, x') < 5\}$.

Implication of GDD. A GDD σ implies another GDD σ' if any match satisfying σ also satisfies σ'. If $\sigma = (Q[\bar{z}], \Phi_{X_1} \rightarrow \Phi_{Y_1})$ implies $\sigma' = (Q[\bar{z}], \Phi_{X_2} \rightarrow \Phi_{Y_2})$, then $\Phi_{X_1} \succcurlyeq \Phi_{X_2}$ and $\Phi_{Y_2} \succcurlyeq \Phi_{Y_1}$. That is, σ has a looser LHS Φ_{X_1} but a tighter RHS Φ_{Y_1} compared to those of σ'. For example, σ implies σ' if $\Phi_{X_1} = \{\delta_{\mathsf{age}}(x, x') < 2\}$ and $\Phi_{Y_1} = \{\delta_{\mathsf{weight}}(x, x') < 3\}$ while $\Phi_{X_2} = \{\delta_{\mathsf{age}}(x, x') < 1, \delta_{\mathsf{height}}(x, x') < 5\}$ and $\Phi_{Y_2} = \{\delta_{\mathsf{weight}}(x, x') < 5\}$.

Inference of GDDs. The general *implication problem* aims to investigate whether a given set Σ of GDDs *implies* a single GDD σ. A full investigation of the implication problem is out of the scope of this paper. We only present the results needed for the ER context. In Table 2, we derive three sound inference rules for GDD implication over *a given patern* $Q[\bar{z}]$, along the same lines as Armstrong's Axioms [8] for FDs.

The rules in Table 2 allow the pruning of implied dependencies during GDD discovery. The proofs of *soundness* of $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$ are straight-forward following the properties of *reflexivity, augmentation* and *transitivity* respectively, using the subjugation order. We omit the proofs for lack of space.

Irreducible GDD. Given a set Σ of GDDs, a GDD $\sigma = (Q[\bar{z}], \Phi_X \rightarrow \Phi_Y) \in \Sigma$ is irreducible if and only if there *does not exist* another GDD $\sigma_1 = (Q[\bar{z}], \Phi_{X_1} \rightarrow \Phi_{Y_1}) \in \Sigma$ such that: (i) $\Phi_{X_1} \succcurlyeq \Phi_X$ and (ii) $\Phi_Y \succcurlyeq \Phi_{Y_2}$. A set Σ of GDDs is non-redundant if every GDD $\sigma \in \Sigma$ is irreducible.

Irreducible GDDs are the important ones, and they are those to be discovered in data later on.

Example 6 (Implied GDDs). Let σ_2, σ_3 be two GDDs defined over the pattern $Q_3[x_6, \cdots, x_9]$ in Figure 2 as: $\sigma_2 = (Q_3[\bar{z}], \Phi_{X_2} \rightarrow \delta_{\equiv}(x_7.\mathsf{eid}, x_9.\mathsf{eid}) = 0)$, $\sigma_3 = (Q_3[\bar{z}], \Phi_{X_3} \rightarrow \delta_{\equiv}(x_7.\mathsf{eid}, x_9.\mathsf{eid}) = 0)$ where $\Phi_{X_2} = \{\delta_{\mathsf{name}}(x_7, x_9) \leq 3\}$ and $\Phi_{X_3} = \{\delta_{\mathsf{name}}(x_7, x_9) \leq 2, \delta(x_7.\mathsf{sex}, \mathsf{f}) < 1, \delta(x_9.\mathsf{sex}, \mathsf{f}) < 1\}$. Then, σ_3 is *not* irreducible if $\Sigma = \{\sigma_2\}$, since $\Phi_{X_2} \succcurlyeq \Phi_{X_3}$ and $\Phi_{Y_3} \succcurlyeq \Phi_{Y_2}$ are true, (here, Φ_{Y_3}, Φ_{Y_2} are same). And, σ_2 implies σ_3, and can be proven by using $\mathcal{I}_1, \mathcal{I}_3$. $\square$

5. DISCOVERY OF GDDS

For GDDs to be useful in any data quality/management application, (e.g., for ER in Section 6), there is the need for techniques that can learn GDDs from data automatically. This is because, although dependencies can be specified by domain experts, relying on experts is often infeasible/unrealistic as the process can be manually-complex, tedious and expensive [22, 33, 45, 54].

In this section, we investigate the discovery of a special category of GDDs, called Linking GDD, denoted by $\mathsf{GDD_L}$. A $\mathsf{GDD_L}$ is a special GDD of the form $\sigma = (Q[\{x, x'\}, x.\mathsf{type} = x'.\mathsf{type}], \Phi_X \rightarrow \Phi_{\mathsf{eid}})$, where $\Phi_{\mathsf{eid}} = \{\delta_{\equiv}(x.\mathsf{eid}, x'.\mathsf{eid}) = 0\}$ is a constraint requiring an eid match.

A $\mathsf{GDD_L}$ can be interpreted as: for any profiles that match the graph pattern $Q[\bar{z}]$, if they agree on Φ_X, then they must agree on Φ_{eid} (meaning they must have the same eid) implying that they refer to the same real-world entity.

We present a technique for finding valid $\mathsf{GDD_L}$s in an **eid-labelled** profiles graph. A profiles graph is eid-labelled, if for every node v in the graph, $v.\mathsf{eid}$ exists and has a value. The discovery work here is in the same direction as the discovery of MDs [48, 50, 51] and CMDs [54] in relational data.

5.1 $\mathsf{GDD_L}$ Discovery Problem

In practice, the set of all valid dependencies in data can be very large. Dependency discovery is to find a cover set of dependencies that *imply* all other valid dependencies. We seek frequent irreducible $\mathsf{GDD_L}$s. A $\mathsf{GDD_L}$ σ is **frequent** if the number $|H_X(\bar{z})|$ of matches satisfying both sides of the dependency of σ is more than a minimal support parameter k. When $k = 1$, all frequent irreducible $\mathsf{GDD_L}$s form a cover set. The minimal support parameter helps to control the number of $\mathsf{GDD_L}$s we find. $|H_X(\bar{z})|$ is called the **support** of σ and is denoted by $sup(\sigma)$.

Definition 1 ($\mathsf{GDD_L}$ Discovery). Given an eid-labelled entity profiles graph G, and a minimum support threshold k, the discovery problem is to find a set Σ of $\mathsf{GDD_L}$s such that $\forall\ \sigma = (Q[\{x, x'\}, x.\mathsf{type} = x'.\mathsf{type}], \Phi_X \rightarrow \Phi_{eid}) \in \Sigma$, $G \models \sigma$, σ is irreducible, and $sup(\sigma) \geq k$. $\square$

From the eid-labelled profiles graph G, we can derive the satisfaction set for the RHS $\Phi_{\mathsf{eid}} = \{\delta_{\equiv}(x, \mathsf{eid}, x'.\mathsf{eid}) = 0\}$. The satisfaction set of Φ_{eid}, $sat(\Phi_{\mathsf{eid}})$, is the set of *all* node pairs such that every pair of nodes have the same eid. That is, $sat(\Phi_{\mathsf{eid}}) = \{\forall (v_1, v_2) \in G \mid v_1.\mathsf{eid} = v_2.\mathsf{eid}\}$.

The strategy of our discovery method is to find a LHS Φ_X such that its satisfaction set $sat(\Phi_X)$, also a set of *all* node pairs that agree on Φ_X, is a subset of the RHS satisfaction set $sat(\Phi_{\mathsf{eid}})$. More specifically, let $sat(\Phi_X)$ be the satisfaction set of Φ_X, our aim is to find all LHS Φ_X such that *every* pair of node $(v_i, v_j) \in sat(\Phi_X)$ is in $sat(\Phi_{\mathsf{eid}})$. The condition for a dependency over a given pattern $Q[\bar{z}]$ is given in Property 1.

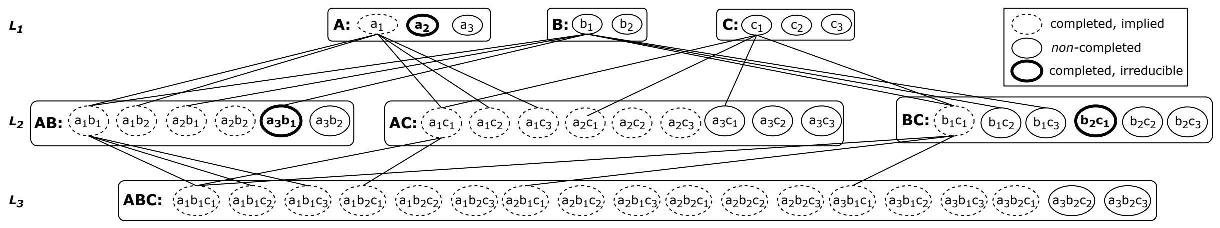
Property 1. *If* $sat(\Phi_X) \subseteq sat(\Phi_{\mathsf{eid}})$, *then* $\Phi_X \rightarrow \Phi_{\mathsf{eid}}$.

Our discovery method uses the Apriori [5] lattice to model the search space of LHSs. We now define a concept called an *itemset* that can be uniquely mapped onto a LHS Φ_X.

Definition 2 (Itemset & Itemset satisfaction set). Let an item-name be an attribute or a relation, denoted by ρ. An item is a (item-name ρ, threshold τ) pair denoted by $\rho[\tau]$. An itemset is a set of such items, denoted by $\mathcal{L} = \{\cdots, \rho_i[\tau_{ij}], \cdots\}$ where every item-name is distinct.

The satisfaction set of an itemset $\mathcal{L}$, denoted by $sat(\mathcal{L})$, is a set of all node pairs $sat(\mathcal{L}) = \{\cdots, (v_1, v_2), \cdots\}$ such that every pair of nodes has values whose distance is within the threshold τ_{ij} on the item-name ρ_i, for all $\rho_i[\tau_{ij}] \in \mathcal{L}$. $\square$

Property 2. *An item* $\rho[\tau_j]$ *can be uniquely mapped to a distance constraint* $\delta_\rho(x, x') \leq \tau_j$, *where* (x, x') *is a pair of pattern nodes of* $Q[\bar{z}]$.



L1
A: a1 a2 a3
B: b1 b2
C: c1 c2 c3
completed, implied
non-completed
completed, irreducible
L2
AB: a1b1 a1b2 a2b1 a2b2 a3b1 a3b2
AC: a1c1 a1c2 a1c3 a2c1 a2c2 a2c3 a3c1 a3c2 a3c3
BC: b1c1 b1c2 b1c3 b2c1 b2c2 b2c3
L3
ABC: a1b1c1 a1b1c2 a1b1c3 a1b2c1 a1b2c2 a1b2c3 a2b1c1 a2b1c2 a2b1c3 a2b2c1 a2b2c2 a2b2c3 a3b1c1 a3b1c2 a3b1c3 a3b2c1 a3b2c2 a3b2c3

Figure 3: Itemsets Lattice Construction & Pruning

Property 2 implies that if we find an itemset, $\mathcal{L}$, whose satisfaction set, $sat(\mathcal{L})$, meets the stipulation of Property 1, we find a dependency $\Phi_X \to \Phi_{eid}$ over $Q[\bar{z}]$, where Φ_X maps onto $\mathcal{L}$. In the following, we present our approach for the discovery of irreducible $\mathsf{GDD_L}$s via frequent itemsets mining.

5.2 The Proposed GDD$_L$ Discovery Method

The key steps of our $\mathsf{GDD_L}$ discovery technique are: *first level itemsets generation*, *itemset generation for higher levels*, and *detection of* $\mathsf{GDD_L}$s. We discuss these steps, then present the overall description of the discovery algorithm.

Construct Level 1 Itemsets. We start by constructing single itemsets: each itemset containing only one (item-name, threshold) pair. The main task is to determine possible and meaningful threshold levels $[\tau_1, ..., \tau_h]$ for an attribute ρ. These threshold levels are application-specific and are determined by users. For example, in policing applications, a difference of 5yrs can be a good scale for suspects' age. That is, the thresholds could be $[0, 5, 10, 15, 20]$; and comparing two profiles with age difference more than 20 years may not make sense. In the same way, the thresholds for the name differences of two people can be $[0, 1, 2, 3, 4]$. If two names have more than 5 character differences, they are hardly the same except for prefix (e.g., John for Jonathan) and synonym (e.g., Dick for Richard) cases which are resolved by special processes. The threshold for any relation is 0.

The next step is to compute the satisfaction sets of itemsets. For each distinct item-name, ρ, we compute the satisfaction set of every itemset on ρ with different thresholds, e.g., $\mathsf{name}[0], \cdots, \mathsf{name}[4]$. Given an item-name ρ_i with its list of possible thresholds $[\tau_{i1}, ..., \tau_{ih}]$ in ascending order, we first derive the set $S_{\rho_i} = \{(a_1, v_1), ..., (a_n, v_n)\}$ of all (ρ-value, node-pid) pairs in the graph G. For any two elements $(a_1, v_1), (a_2, v_2) \in S_{\rho_i}$, let $\delta_{\rho_i}(a_1, a_2) = d^i_{12}$. We add (v_1, v_2) into every $sat(\rho_i[\tau_{ij}])$ for all $\tau_{ij} \geq d^i_{12}$. In this way, we construct all single items $\rho_i[\tau_{i1}], \cdots, \rho_i[\tau_{ih}]$ for the item-name ρ_i and their corresponding satisfaction sets. Each single item becomes a Level-1 itemset in the lattice.

To aid the calculation of the distances of itemsets $sat(\rho[\tau])$ ($\tau = t_0, t_1, ..., t_k$; $t_0 = 0 < t_1 < \cdots < t_k$) at this level, we create an index for ρ: $idx(\rho) = \{\cdots, (val : \bar{V}), \cdots\}$ where $\bar{V}$ is a set of nodes whose ρ exists and its value is val. For any two index entries $(val_1 : \bar{V}_1)$ and $(val_2 : \bar{V}_2)$ (the two entries can be the same), let $\delta_\rho(val_1, val_2) = d$. We add the set of node pairs in $\bar{V}_1 \times \bar{V}_2$ to all $sat(\rho[t_j])(i \leq j \leq k)$ if $t_j \geq d$.

Derive Itemsets for Higher Levels. We further derive the itemsets and their satisfaction sets for the second level. A second level itemset is of the form $\{\rho_1[\tau_{1j}], \rho_2[\tau_{2l}]\}$ where $\rho_1 \neq \rho_2$; and its satisfaction set is derived from the satisfaction sets of the corresponding first level itemsets as $sat(\{\rho_1[\tau_{1j}], \rho_2[\tau_{2l}]\}) = sat(\rho_1[\tau_{1j}]) \cap sat(\rho_2[\tau_{2l}])$. This step does not need the calculation of distances; and because the satisfaction sets are sorted, this step is efficient.

Table 3: Example graph data for discovery

pid	**attributes & relations**	eid
v_1	name:john, sex:m, height:1.7, weight:80	e1
v_2	name:john, sex:m, height:1.7	e1
v_3	name:peter, sex:f, height:1.7	e2
v_4	name:peter, sex:f, height:1.5, race:asian	e2
v_5	name:peter, sex:f, weight:80, friend:v_4	e3
v_6	name:peter, weight:75, friend:v_4	e3
v_7	name:rob, birthdate:1998	e4
v_8	name:rob, birthdate:1998	e4

The lattice search space develops to higher levels following the same procedure above. Itemsets for Level-$(i+1)$ are derived from itemsets of Level-i. A Level-$(i+1)$ itemset must contain $(i+1)$ and only $(i+1)$ distinct items.

Detect GDD$_L$s. Let $\mathcal{L}$ be an itemset in the lattice search space. We say $\mathcal{L}$ is **completed** if $sat(\mathcal{L}) \subseteq sat(\Phi_{eid})$, i.e., Property 1 is met. When an itemset $\mathcal{L}$ is completed, it does not appear in any other itemset $\mathcal{L}'$ at higher levels. That is, no super itermset should contain a completed itemset.

When an itemset $\mathcal{L}$ is completed, we derive the $\mathsf{GDD_L}$ $(Q[\{x, x'\}, x.\mathsf{type} = x'.\mathsf{type}], \Phi_X \to \Phi_{eid})$ where $\mathcal{L}$ maps to Φ_X according to Properties 1 and 2. The lattice stops developing (the discovery ends) if no more than one itemset with non-empty satisfaction set exists, or the last level is complete.

Figure 3 is an exemplar diagram of the lattice for the item-names A, B, C and their respective distance thresholds $[a_1, a_2, a_3]$, $[b_1, b_2]$, and $[c_1, c_2, c_3]$. For brevity, due to space limits, an itemset, e.g., $\{A[a_1], B[b_1]\}$ is shown as $AB : a_1b_1$; and edges are shown from *only one* itemset for each attribute set at every level. The diagram illustrates how completed and irreducible itemsets at a level can be used to prune implied itemsets at the current and subsequent levels based on the inference rules in Table 2.

Example 7 (Discovery Approach)**.** We now show how the discovery method works with an example. Consider the eid-labelled graph described in Table 3. The RHS $sat(\Phi_{eid}) = \{(v_1, v_2), (v_3, v_4), (v_5, v_6), (v_7, v_8)\}$. Thus, any LHS Φ_X whose satisfaction set $sat(\Phi_X)$ is *completely contained* in $sat(\Phi_{eid})$ derives a dependency over the graph pattern $Q[\{x, x'\}]$.

We use compact notations: nm for name; ht for height; wt for weight; frd for friend; bd for birthdate. At the same time, to save space, we use $\mathsf{nm}[0]\mathsf{sex}[0][v_1v_2, v_3v_4]$ to mean the itemset $\{\mathsf{name}[0], \mathsf{sex}[0]\}$ with the satisfaction set $\{(v_1, v_2), (v_3, v_4)\}$ as established in Definition 2. The itemsets with empty satisfaction set are omitted. We also omit the type constraint in the pattern, $Q[\{x, x'\}, x.\mathsf{type} = x'\mathsf{type}]$. The Level-1 itemsets are the following.

Level 1: $\mathsf{nm}[0][v_1v_2, v_3v_4, v_3v_5, v_3v_6, v_4v_5, v_4v_6, v_5v_6, v_7v_8]$,
$\mathsf{sex}[0][v_1v_2, v_3v_4, v_3v_5, v_4v_5]$, $\mathsf{ht}[0][v_1v_2, v_1v_3, v_2v_3]$,
$\mathsf{ht}[0.2][v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4, v_3v_4]$,
$\mathsf{wt}[10][v_5v_6]$, $\mathsf{frd}[0][v_5v_6]$, $\mathsf{bd}[0][v_7v_8]$

Up to this point, the itemsets wt[10], frd[0], bd[0] are *completed* because their satisfaction sets is fully contained in $sat(\Phi_{eid})$. For example, $sat(\mathsf{wt}[10]) = \{(v_5, v_6)\} \subset sat(\Phi_{eid})$.

From the three completed nodes, we discover the following $\mathsf{GDD_{L}s}$ each with a support of 1:

$\sigma_1=(Q[\{x,x'\}],\{\delta_{wt}(x,x')\leq 10\}\rightarrow\Phi_{eid})$;
$\sigma_2=(Q[\{x,x'\}],\{\delta_{frd}(x,x')=0\}\rightarrow\Phi_{eid})$;
$\sigma_3=(Q[\{x,x'\}],\{\delta_{bd}(x,x')=0\}\rightarrow\Phi_{eid})$.

Level 2: $\mathsf{nm}[0]\mathsf{sex}[0][v_1v_2,v_3v_4,v_3v_5,v_4v_5]$, $\mathsf{nm}[0]\mathsf{ht}[0][v_1v_2]$, $\mathsf{nm}[0]\mathsf{ht}[0.2][v_1v_2,v_3v_4]$, $\mathsf{sex}[0]\mathsf{ht}[0][v_1v_2]$, $\mathsf{sex}[0]\mathsf{ht}[0.2][v_1v_2,v_3v_4]$

The itemsets $\mathsf{nm}[0]\mathsf{ht}[0]$, $\mathsf{nm}[0]\mathsf{ht}[0.2]$, $\mathsf{sex}[0]\mathsf{ht}[0]$, and $\mathsf{sex}[0]\mathsf{ht}[0.2]$ are *completed* at this level. The following $\mathsf{GDD_{L}s}$ are therefore discovered:

$\sigma_4=(Q[\{x,x'\}],\{\delta_{nm}(x,x')\leq 0,\delta_{ht}(x,x')\leq 0\}\rightarrow\Phi_{eid})$
$\sigma_5=(Q[\{x,x'\}],\{\delta_{nm}(x,x')\leq 0,\delta_{ht}(x,x')\leq 0.2\}\rightarrow\Phi_{eid})$
$\sigma_6=(Q[\{x,x'\}],\{\delta_{sex}(x,x')\leq 0,\delta_{ht}(x,x')\leq 0\}\rightarrow\Phi_{eid})$
$\sigma_7=(Q[\{x,x'\}],\{\delta_{sex}(x,x')\leq 0,\delta_{ht}(x,x')\leq 0.2\}\rightarrow\Phi_{eid})$

However, comparing σ_4 and σ_5, we see that $\mathsf{sex}[0]\mathsf{ht}[0.2]\succcurlyeq\mathsf{sex}[0]\mathsf{ht}[0]$, so σ_4 is implied (Sec 4.2, Implication) and not irreducible. Similarly, σ_6 is also implied and not irreducible.

Level 3: No itemset is possible. The discovery finishes. In conclusion, we find $\mathsf{GDD_{L}s}$ σ_1, σ_2, σ_3, σ_5, and σ_7 from this training dataset; and $sup(\sigma_1)=1$, $sup(\sigma_2)=1$, $sup(\sigma_3)=1$, $sup(\sigma_5)=2$, $sup(\sigma_7)=2$. □

Description of Algorithm. The algorithm of our $\mathsf{GDD_L}$ discovery method is shown in Algorithm 1. The input is an eid-labelled profiles graph $G=(V,E,L,F_A)$, a minimal support k for found $\mathsf{GDD_{L}s}$ to control the number of $\mathsf{GDD_{L}s}$ to be discovered, the item-names (ρ's) and possible thresholds (τ's) for each item-name in the form of $\Theta=\{\rho_1:[\tau_{11},...,\tau_{1k}],\ \rho_2:[\tau_{21},...,\tau_{2k}],\cdots\}$. Let $\mathbf{A}'$ be the set of all attributes $\mathbf{A}$ and relations in Θ.

For every entity type et (e.g., `person, location`, etc.,) in G, the algorithm, firstly, retrieves the set $H_{et}(\bar{z})$ of *all* matches of the graph pattern $Q[\{x,x'\},x.\mathsf{type}=x'.\mathsf{type}=\mathsf{et}]$ in G. Let $G_{et}=(V_{et},E_{et},L,F_A)$ be a subgraph of G, where V_{et} contains all et-typed/labelled nodes or the nodes that are directly connected to the et-nodes. E_{et} contains a subset of edges of E and the edges in the subset originate from nodes in V_{et}. L and F_A are as defined for G. The set $H_{et}(\bar{z})$ can easily be derived from G_{et} as the set of all et-node pairs in V_{et}. The algorithm then calls Function 1, to find $\mathsf{GDD_{L}s}$ in the set $H_{et}(\bar{z})$ of matches of $Q[x,x']$ for the entity type et.

The processes in Function 1 are almost self-explanatory. The generation of itemsets and detection of $\mathsf{GDD_{L}s}$ at Level-1 are performed in lines 3 and 4 respectively. In lines 6 to 8, valid itemsets are built for higher Levels, i.e., $i\geq 2$. Line 9 derives the satisfaction set of a valid itemset; and line 10 prunes itemsets with a smaller-than-k support. The detection of completed itemsets at higher levels is in line 12. The function $\mathsf{toGDD}()$ in Line 14 converts completed itemsets to distance constraints (following Property 2). Function 1 terminates if the current level has only one itemset left (line 13) or tests all itemsets of the last level in the lattice.

Function 2 composes Level-1 itemsets and their satisfaction sets. The itemsets with a lower-than-k support are ignored (Line 5).

Function 3 detects whether an itemset $\mathcal{L}$ is completed. If $\mathcal{L}$ is completed, it is removed from the search space (Line 3). Then, the GDD inference rules (in Table 2) are used to check implication of $\mathcal{L}$ *w.r.t.* the set Ω of completed itemsets: $\mathcal{L}$ is ignored if implied by Ω (Line 4), and any completed itemset $\mathcal{L}'\in\Omega$ is removed (Line 5) if it is implied by $\mathcal{L}$.

Time Complexity. Given an eid-labelled profiles graph G, let m be the number of entity types in G; and q,n,s be the averages of the number of: profiles (nodes), attributes/relations, and distance thresholds per attribute, respectively, in all m subgraphs G_{et} of G.

Algorithm 1 $\mathsf{MineGDD_{L}s}$

```
Input: eid-labelled graph G; minimum support k; thresholds Θ
Output: set Σ of GDD_Ls discovered from G
 1. $\Sigma=\emptyset$
 2. for each entity type et in graph G do
 3.    retrieve subgraph $G_{et}$ from G & get matches $H_{et}$ of Q
 4.    $\Sigma_{et}=findDep(H_{et},k,\Theta)$ // a set of GDD_Ls for type et.
 5.    $\Sigma.add(\Sigma_{et})$
 6. return Σ
```

Function 1 $findDep(H_{et},k,\Theta)$

```
 1. $sat_r=sat(\Phi_{eid})$ // text following Def. 1
 2. $\Omega=\emptyset$ // to store completed itemsets
 3. build 1st level itemsets $\mathbb{L}(1)=singleItems(H_{et},k,\Theta)$
 4. $(\Omega,\mathbb{L}(1))=detectComplete(\Omega,\mathbb{L}(1),sat_r)$
 5. for $i=2$ to $i=|\mathbf{A}'|$ do
 6.    for itemsets $(\mathcal{L}_1,\mathcal{L}_2)$ in $\mathbb{L}(i-1)\times\mathbb{L}(i-1)$ do
 7.       $\mathcal{L}=\mathcal{L}_1\cup\mathcal{L}_2$
 8.       if $|\mathcal{L}|\neq i$ or item-names in $\mathcal{L}$ repeat: continue
 9.       $sat(\mathcal{L})=sat(\mathcal{L}_1)\cap sat(\mathcal{L}_2)$
10.       if $|sat(\mathcal{L})|<k$: continue
11.       add $(\mathcal{L},sat(\mathcal{L}))$ to $\mathbb{L}(i)$.
12.    $(\Omega,\mathbb{L}(i))=detectComplete(\Omega,\mathbb{L}(i),sat_r)$
13.    if $|\mathbb{L}(i)|=1$: break;
14. return toGDD(Ω)
```

Function 2 $singleItems(H_{et},k,\Theta)$

```
 1. $\mathbb{L}(1)=\emptyset$
 2. for each $(\rho:th)$ in Θ (where $th=[\tau_0,...,\tau_k]$) do
 3.    build index $idx(\rho)=\{...,\ val:\bar{V},\ ...\}$ for et-nodes in $H_{et}$
 4.    calculate $sat(\rho[\tau_i])$ $(i=0,\cdots,k)$
 5.    $\mathbb{L}(1).add(\ (\rho[\tau_i],sat(\rho[\tau_i]))\ )$ if $|sat(\rho[\tau_i])|\geq k$.
 6. return $\mathbb{L}(1)$
```

Function 3 $detectComplete(\Omega,\mathbb{L}(i),sat_r)$

```
 1. for $(\mathcal{L},sat(\mathcal{L}))$ in $\mathbb{L}(i)$ do
 2.    if $sat(\mathcal{L})\subseteq sat_r$: then
 3.       remove $(\mathcal{L},sat(\mathcal{L}))$ from $\mathbb{L}(i)$
 4.       next loop if $\exists\ \mathcal{L}'\in\Omega$ s.t $\mathcal{L}'\succcurlyeq\mathcal{L}$
 5.       remove $\forall\ \mathcal{L}'\in\Omega$ if $\mathcal{L}\succcurlyeq\mathcal{L}'$
 6.       $\Omega.add(\mathcal{L})$
 7. return $(\Omega,\mathbb{L}(i))$
```

The retrieval of matches for the pattern $Q[\{x,x'\},x.\mathsf{type}=x'.\mathsf{type}]$, for all m node-types in the worst case is $\mathcal{O}(|G|)$. For a given subgraph G_{et}, the size of all possible LHSs (itemsets) is $(1+s)^n$, and generating all $(\mathcal{L},sat(\mathcal{L}))$ pairs in the worst case is $\mathcal{O}(q^2\cdot(1+s)^n)$. The detection of completed itemsets is in $\mathcal{O}(f)$, where f is the total number of itemsets tested. Thus, the complexity Algorithm 1 in the worst-case all together is $\mathcal{O}(|G|)+\mathcal{O}\left(m\cdot(q^2\cdot(1+s)^n+f)\right)\approx\mathcal{O}\left(q^2\cdot(1+s)^n+|G|\right)$.

6. ENTITY RESOLUTION WITH GDDS

This section presents our procedure for applying the discovered $\mathsf{GDD_{L}s}$ to find profiles that refer to the same real-world entities (*linked clusters*) in a given profiles graph G with *unknown* eids. A naïve approach would be to compute the distances of attribute/relation values of *every pair* of profiles. If the distances satisfy all the distance constraints of any discovered $\mathsf{GDD_L}$, then the two profiles are linked.

The naïve process is, however, expensive due to its extremely high number of pairwise comparisons. One million

Table 4: Blocks for Figure 1(2a)

(ρ, τ_ρ)	**Blocks**
$(\mathsf{sex}, 0)$	$B_{11} = \{1, 6, 8\}$, $B_{12} = \{2, 9\}$
$(\mathsf{dob}, 2)$	$B_{21} = \{1, 3, 8\}$, $B_{22} = \{2, 4, 9\}$
$(\mathsf{height}, 0.2)$	$B_{31} = \{1, 3, 8\}$, $B_{32} = \{6, 9\}$
$(\mathsf{name}, 4)$	$B_{41} = \{1, 2, 3, 4, 6, 8\}$, $B_{42} = \{1, 3, 6, 8\}$, $B_{43} = \{2, 4, 9\}$ $B_{44} = \{1, 2, 3, 8\}$

entities are enough to make the calculation last for days and a real-world application often has much more entities.

To speed up the computation, we design an elaborate method. Our technique improves the performance using three pruning strategies. The first, called *blocking*, groups profiles into blocks so that linking of profiles across different blocks is not meaningful. The second tactic involves the use of *blocking graph*. This method uses profile pair popularity in blocks and block sizes as a pruning principle. The third strategy, called *aggregated similarity* pruning, relies on the frequencies and similarities of values.

We highlight the details of each pruning process, show how $\mathsf{GDD_Ls}$ are used for ER, and then, present our ER algorithm.

Blocking. We group profiles in G into blocks, a block for each value of an attribute/relation ρ. The blocks are formed using an index $\mathtt{idx}(\rho)$ which is built to support searches with a 2-gram threshold. For string values, the maximum distance threshold τ_ρ translates to $\gamma \in [\tau_\rho{+}1, 2{\times}\tau_\rho]$ number of 2-gram differences. Date attributes are handled as strings and numeric attributes are handled efficiently with sorting.

For a distinct value a of ρ, the block for $\rho = a$ is a set, $B_{\rho(a)} = \{v_1, \cdots, v_k\}$, of profile/node-ids, where for any two profiles v_i and v_j in G, the difference between $v_i.A$ and $v_j.A$ is no more than $(2{\times}\tau_\rho)$ number of 2-grams. The distances of some profile pairs in $B_{\rho(a)}$ may exceed τ_ρ and these pairs will be pruned as we go. The blocks of ρ produced in this way may overlap; and blocks with a single profile are ignored.

By blocking, pairwise comparisons over the whole graph G have become pairwise comparisons within each block, and this produces a huge performance gain. As an example, consider the (attribute, max-distance-threshold) pairs of $(\mathsf{name}, 4)$, $(\mathsf{sex}, 0)$, $(\mathsf{dob}, 2)$, and $(\mathsf{height}, 0.2)$ for the graph in Figure 2(a). The resulting blocks are shown in Table 4.

We denote all blocks of every attribute/relation ρ_i and its values by $\mathcal{B} = \{B_{\rho_i(a_{ij})}, \cdots\}$.

Pruning with blocking graph. We present our second way of improving computation performance: building a blocking graph [40]. This further reduces the number of candidate profile pairs in the graph. A *blocking graph* of a set of blocks $\mathcal{B}$ is a weighted undirected graph $G_\mathcal{B} = (V_\mathcal{B}, E_\mathcal{B}, W_\mathcal{B})$, where the node set $V_\mathcal{B}$ is the set of all profiles in $\mathcal{B}$, the undirected edge set $E_\mathcal{B}$ contains edges (v, v') if v and v' co-occur in a block $B \in \mathcal{B}$, and $W_\mathcal{B}$ is a weighting function that assigns a weight to every edge in $E_\mathcal{B}$. That is:

$$V_\mathcal{B} = \cup_{B \in \mathcal{B}} B; \quad E_\mathcal{B} = \{(v, v') \mid \exists\ B \in \mathcal{B} \text{ s.t. } v \in B \wedge v' \in B\}. \quad (1)$$

The weight is the core of pruning with a blocking graph.

Weighting Edges. The weight of an edge (profile pair) is derived from two components: the popularity of the edge in the number of blocks Cbs, and the sizes of the blocks Arcs containing the edge. The weight is a combination of the two components proposed in [40], as the harmonic mean:

$$W(v, v') = 2 \cdot \frac{\mathsf{Arcs} \cdot \mathsf{Cbs}}{\mathsf{Arcs} + \mathsf{Cbs}}, \quad (2)$$

where v and v' are two nodes/profiles. Let $B_{vv'}$ be the set of all blocks containing (v, v'), and $B \in B_{vv'}$. Following [40],

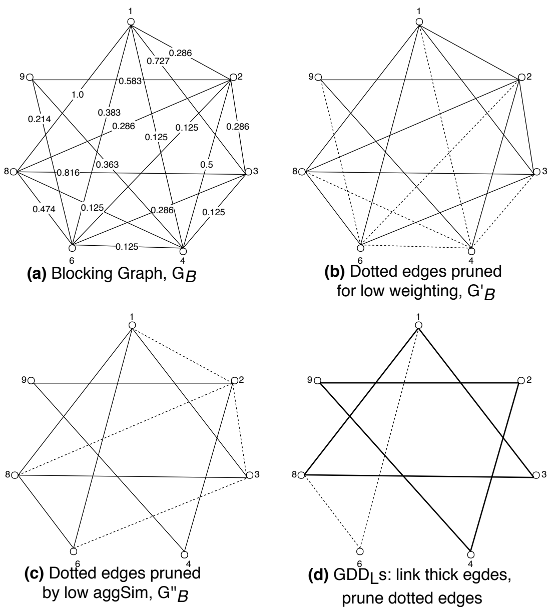

1
9
2
8
3
6
4
0.286
0.727
0.583
1.0
0.214
0.383
0.286
0.125
0.125
0.286
0.816
0.363
0.5
0.474
0.125
0.286
0.125
0.125
(a) Blocking Graph, G_B
(b) Dotted edges pruned for low weighting, G'_B
(c) Dotted edges pruned by low aggSim, G''_B
(d) GDD_Ls: link thick egdes, prune dotted edges

Figure 4: Running Example

$\mathsf{Arcs} = 1/Z_1 \sum_{B \in B_{vv'}} \frac{1}{|B|}$ represents the intuition that v and v' in small-sized blocks are more likely to be for the same entity, and $\mathsf{Cbs} = |B_{vv'}|/Z_2$ suggests profiles pairs with high co-occurrence have higher chance of being for the same entity. Z_1 and Z_2 normalizes the values $\mathsf{Arcs}, \mathsf{Cbs}$ into $[0, 1]$.

We calculate the weight of every edge in $G_\mathcal{B}$ and compute the harmonic mean of all edge weights as the average avW.

Pruning. We prune edges instead of nodes in $G_\mathcal{B}$ for better recall as the latter is more impeding of high recall [40, 41].

With the calculated average avW, for any edge $(v, v') \in E_\mathcal{B}$, if $W(v, v') < avW$, (v, v') is deleted from $E_\mathcal{B}$. This is because: "*Experimental evidence with real-world data sets suggests that the average edge weight provides an efficient (i.e., requires just one iteration over all edges) as well as reliable (i.e., low impact on effectiveness) estimation ..., regardless of the underlying weighting scheme*" [40].

The blocking graph after this step is denoted by $G'_\mathcal{B}$.

Example 8 (Blocking Graph)**.** Given the list of blocks in Table 4, the resulting blocking graph is as shown in Figure 4(a). As an example, the weight of edge $(1, 2)$ is computed as follows. $\mathsf{Arcs}(1, 2) = \frac{1}{|B_{41}|} + \frac{1}{|B_{44}|} = 1/6 + 1/4; \mathsf{Cbs}(1, 2) = |\{B_{41}, B_{44}\}| = 2$; normalized to $\mathsf{Arcs}(1, 2) = 0.417/1.667 = 0.25; \mathsf{Cbs}(1, 2) = 2/6 = 0.333$ with their respective maximum values: 1.667 and 6. $\therefore\ W(1, 2) = \frac{2 \times 0.25 \times 0.333}{0.25 + 0.333} = 0.286$. Edges with weight less than the harmonic mean of all edges ($avW = 0.245$) are pruned, shown by dotted lines in Figure 4(b) for the running example. $\square$

Pruning with aggregated similarity. The frequencies of values in a profiles graph G plays an important role in ER [34]. For example, given the first-names John and Selasi, the probability of two profiles belonging to the same person when they share John is lower than when they share Selasi. This is because, John is a much more commonly used first-name than Selasi. This pruning method computes an aggregated similarity (with consideration of value-frequency) of all attributes/relations values for edges (profile pairs) in the block graph $G'_\mathcal{B}$. The aim is to delete more edges if their aggregated similarity is less than a threshold.

Consider the edge (v, v') in $G'_{\mathcal{B}}$. The *aggregated similarity*, denoted by $\mathtt{aggSim}(v, v') \in [0, \infty]$, of (v, v') is defined as:

$$\mathtt{aggSim}(v, v') = \sum_{\rho \in \mathcal{P}} \eth_{\rho}(v, v') \cdot \wp_{\rho}(v, v'), \tag{3}$$

where $\mathcal{P}$ is the set of common attributes and relations of v and v', $\eth_{\rho}(v, v')$ is a similarity metric on ρ: $\eth_{\rho}(v, v') = \frac{\tau_{\rho} - \delta_{\rho}(v, v')}{\tau_{\rho}}$, where τ_{ρ} is the maximal distance threshold of ρ and $\delta_{\rho}(v, v')$ returns the distance of (v, v') on ρ as defined before; and $\wp_{\rho}(v, v') = \frac{1}{2}(\mathtt{pr}(v.\rho) + \mathtt{pr}(v'.\rho))$ is the average of the probabilities of the values of (v, v') on ρ.

The probability $\mathtt{pr}(v.\rho)$ of a ρ-value $v.\rho$ is modelled by a variation of the Sigmoid function using the frequency $k \in \mathbb{N}$ (calculated via the index $\mathtt{idx}(\rho)$) of the value. Specifically, $\mathtt{pr}(v.\rho) = 1/(1 + \exp(a \cdot k - b))$, where a, b control the steepness of the decay curve and its mid-point respectively. Our empirical results (omitted due to page limit) show (a, b) taking values $(0.1, 60)$ respectively gives the best estimates for our application. For example, $\mathtt{pr}(k = 10) = 0.993$, $\mathtt{pr}(k = 60) = 0.5$ and $\mathtt{pr}(k = 100) = 0.018$.

The aggregated similarity $\mathtt{aggSim}(v, v')$ is calculated for all edges in the reduced blocking graph $G'_{\mathcal{B}}$. At the same time, the calculated distance $\delta_{\rho}(v, v')$ is stored in a hash table $\mathsf{T}(\mathtt{key}, \delta_{\rho}(v, v'))$ where $\mathtt{key} = v.\rho$+'–'+$v'.\rho$. T will be used in applying $\mathsf{GDD_L}$s to find linked clusters later on.

We define the *minimum* $\mathtt{aggSim}(v, v')$ *threshold*, $\vartheta = 1$: reflecting the intuition that two profiles are for the same entity if and only if their $\mathtt{aggSim}$ is equivalent to at least one fully matching value on some attributes/relationships.

Any edge (v, v') with $\mathtt{aggSim}(v, v') < \vartheta$ is pruned from $G'_{\mathcal{B}}$. We denote the updated blocking graph after this step by $G''_{\mathcal{B}}$. Figure 4(c) shows the edges pruned by minimum aggregated similarity in dotted lines for our running example.

Linking with $\mathsf{GDD_L}$s. This is a crucial and the last step in our ER solution: classifying profile pairs as linked or not-linked. The link decision is determined by a set Σ of $\mathsf{GDD_L}$s learned from eid-labelled graph in Section 5.

Algorithm 2 Certus

```
Input: Profiles graph $G$ without eid's, set $\Sigma$ of $\mathsf{GDD_L}$s, aggSim
    threshold $\vartheta$
Output: Linked entity profiles graph $\mathcal{E}_G$.
 1. $\mathcal{E}_G := \emptyset$, hash table $\mathsf{T} = \emptyset$
 2. construct indexes $\mathtt{idx}(\rho)$ $(\forall\rho)$
 3. construct block set $\mathcal{B} = \{B_{\rho(a)}\}$ $(\forall\rho$ and $\forall a \in \mathtt{dom}(\rho))$
 4. construct blocking graph $G_{\mathcal{B}} = (V_{\mathcal{B}}, E_{\mathcal{B}}, W_{\mathcal{B}})$ (Eq 1,2)
 5. compute harmonic mean $avW$ of all edges $E_{\mathcal{B}}$ as average
 6. prune edge $e \in E_{\mathcal{B}}$ if $W(e) < avW$
 7. for each edge $e = (v, v') \in E_{\mathcal{B}}$ do
 8.     compute $d = \delta_{\rho}(v, v')$ $(\forall\rho)$
 9.     add to $\mathsf{T}(v.\rho$+'–'+$v'.\rho, d)$
10.     compute $sim = \mathtt{aggSim}(v, v')$ (Eq 3)
11.     if $sim < \vartheta$: delete $e$ from $E_{\mathcal{B}}$, goto Line 7;
12.     for each $\sigma \in \Sigma$ do
13.         for each constraint $\rho[\tau] \in \sigma.\Phi_X$ do
14.             if $(\mathsf{T}(v.\rho$+'–'+$v'.\rho) > \tau)$: goto Line 7
15.         add $(v, v')$ to $\mathcal{E}_G$
16. return $\mathcal{E}_G$
```

A profile pair with edge (v, v') in $G''_{\mathcal{B}}$ is linked if it *satisfies* a $\mathsf{GDD_L}$ $\sigma \in \Sigma$. For a $\mathsf{GDD_L}$ $\sigma = (Q[x, x'], \Phi_X \rightarrow \Phi_{\mathtt{eid}}) \in \Sigma$, the satisfaction test of the distance constraint $\delta_{\rho}(x, x') \leq \tau$ in Φ_X is instant because the distance $\delta_{\rho}(v, v')$ can be retrieved from the hash table T. If (v, v') satisfies all distance constraints in Φ_X of σ, the pair is linked and is added to the output. Figure 4(d) shows the linked (thick lines) and unlinked (dotted lines) profile pairs in our running example.

Table 5: Summary of Datasets

	DS	#A	#P	#TM	#PP	%N
A	Cora	12	1,875	17,184	296,089	58%
	Rest	4	864	112	3,693	0%
	DBAC	4	4,510	2,224	344,198	12%
	DBSc	4	66,879	5,347	680,542	11%
	PLR	112	1,028,762	4,071	21,041,359	25%
B	Cora	9	1,875	14,499	184,987	50%
	AmGo	4	4,589	1,300	97,009	23%
	DBSc	4	66,879	5,347	112,839	11%

DS: dataset; **A**: attributes; **P**: profiles; **TM**: true matching pairs; **PP**: profile pairs; **N**: avg. NULLs per attribute; Rest: Restaurant; DBSc: DBLP-Scholar; DBAC: DBLP-ACM; AmGo: Amazon-Google.

The ER Algorithm. Our ER algorithm, called **Certus**, is presented in Algorithm 2. Lines 2 & 3 construct the blocks following the 'blocking' subsection; lines 4–6 generate and prune the blocking graph; and lines 7–11 cover aggregated similarity pruning. In line 9, the calculated distance is stored in the hash table T. In line 14, the hash table is retrieved for $\mathsf{GDD_L}$ σ test. If the loop in line 13 completes, then a profile pair (v, v') satisfies σ and stored in $\mathcal{E}_G$ as linked pair.

Time Complexity. The time complexity of Algorithm 2 is $\mathcal{O}(|E_{\mathcal{B}}| \cdot c \cdot |\Sigma|)$: $|E_{\mathcal{B}}|$ is the number of edges in the blocking graph; c is the average number of distance constraints in a $\mathsf{GDD_L}$ and $|\Sigma|$ is the number of $\mathsf{GDD_L}$s. $|E_{\mathcal{B}}|$ is much larger than c and $|\Sigma|$. Let $k = \max(|B|\ \forall B \in \mathcal{B})$. Then $|E_{\mathcal{B}}|$ can be estimated as $|E_{\mathcal{B}}| \approx k \times k \times (|V_{\mathcal{B}}|/k)$.

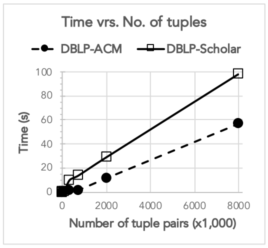
7. EMPIRICAL EVALUATION

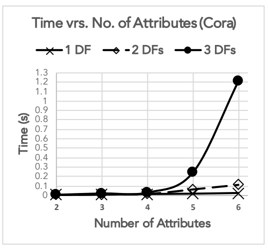
This section covers the evaluation of the proposed $\mathsf{GDD_L}$ discovery algorithm and the $\mathsf{GDD_L}$-based ER technique **Certus**, in subsections 7.1 and 7.2 respectively. All proposed algorithms in this work are implemented in Java; and the experiments were run on a 2.5GHz Intel Core i7 processor computer with 16GB of memory running macOS H. Sierra.

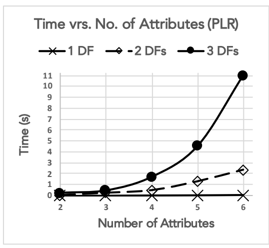
Datasets. We employ real-world ER benchmark datasets with different sizes and features captured in Table 5 for the experiments. All the datasets except PLR are open-source. The PLR dataset is a graph-modelled data consisting of entity profiles from a proprietary relational database and extracts of mentioned entities and their relations from textual documents. The Rest [3] dataset consists of records of restaurants information. Cora [1], DBAC [2] and DBSc [2] datasets are collections of citation references to scientific research papers from different on-line bibliographic portals; and AmGo [2] is a collection of products information from two online shops.

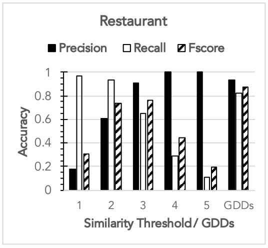
The datasets in Table 5 are in two groups: A & B. Group A datasets are used for evaluating our proposals, whereas those in B are specially generated following the descriptions in [46, 53] for a comparison to the rule-based ER solutions in [46, 53]. #TM represents the number of true matches recorded in the ground truth; and #PP represents the number of edges (profile pairs) in our initial blocking graphs (added by us, *not* part of the descriptions of the datasets).

Similarity Functions. In principle, any similarity and/or distance metric (e.g., edit distance, cosine similarity, q-grams [27], etc.) can be employed. However, we remark that the choices of functions for evaluating the closeness of values should be based on both the domain of attributes/relations and the application. For example, the PLR dataset is from law enforcement application domain. Thus, to adequately









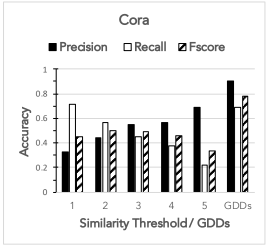


Figure 5: No. of Pairs **Figure 6:** `Cora` $\S(\mathbf{A})$ **Figure 7:** `PLR` $\S(\mathbf{A})$ **Figure 8:** `Rest` Acc. **Figure 9:** `Cora` Acc.

compute the similarity for `person` entity profiles, it is crucial to design custom metrics capable of capturing the closeness of various disparate presentation of names that may arise both unintentionally (e.g., due to errors, natural variations, etc.) and intentionally (e.g., for fraudulent purposes).

7.1 GDD Discovery Evaluation

The discovery of $\mathsf{GDD_L}$s requires the existence of labelled data, and we sampled from the ground truth of each of the open-source datasets to learn their respective rules. In the absence of labelled data, (e.g., the case of the `PLR` dataset), we generate eid-labels using mass power (crowd-sourcing) by providing experts profile pairs with various ranges of aggregated similarity for labelling.

In the following, we perform experiments to examine the performance of the discovery algorithm *w.r.t.*: (a) the size of eid-labelled data (Exp-1a); and (b) number of attributes/relations and distance functions (Exp-1b). Then, we show examples of $\mathsf{GDD_L}$s discovered in the datasets.

Exp-1a. In this experiment, we test the time performance of the $\mathsf{GDD_L}$ discovery algorithm on varying instance sizes of eid-labelled data. We sample up to $4,000$ entity profiles (i.e., ≈ 8 million possible pairs) from both `DBAC` and `DBSc` datasets for this experiment. Figure 5 captures the results of how long it takes the discovery algorithm to find a non-redundant set of $\mathsf{GDD_L}$s: the time (in seconds) is presented on the vertical axis against varying sample sizes of profiles on the horizontal axis. The results show linear performance over the range of instance sizes tested. This reflects the effectiveness of the pruning rules designed in Section 4.2.

Exp-1b. In this set of experiments, we examine the performance of the discovery algorithm *w.r.t.* the number of attributes and distance functions. The search space of possible $\mathsf{GDD_L}$s depends on both the number of attributes and distance constraints. More precisely, the relation between the search space $\S(\mathbf{A}')$ of possible $\mathsf{GDD_L}$s for a sample dataset with n attributes/relations, each with an average of s distant constraints is given by: $\S(\mathbf{A}') = (1+s)^n$.

We sample a projection of $2 \leq n \leq 6$ attributes from the `Cora` and the `PLR` datasets with a fixed sample size of 100 profiles for this test. The results are presented in Figures 6 and 7 for `Cora` and `PLR` respectively. Time (in seconds) is on the y-axes and the number of attributes are on the x-axes. The plots show characteristics for $s = [1, 2, 3]$ distance function per attribute. The time performance characteristics of the $\mathsf{GDD_L}$ discovery algorithm, although *efficient*, follows the exponential search space relation in the $\S(\mathbf{A}')$-Equation above as expected. This reveals the extra complexity of $\mathsf{GDD_L}$ discovery as opposed to other discoveries, e.g., FDs [43], CFDs [22] and GFDs [23] which involve only the equality function (i.e., $s = 1$).

Examples of discovered $\mathsf{GDD_L}$s. We present an example of $\mathsf{GDD_L}$s discovered in each of the datasets in Table 6. For brevity, we show only itemset mappings of the LHSs. The example `PLR` $\mathsf{GDD_L}$ states that for any `person` profile pair, if their name, dob differences are within $1, 2$ respectively, and the profile pair has the lives_at relation with the same `location` profile (i.e., lives_at[0]) then, they refer to the same person in the real-world. Note that this dependency can be specified with pattern $Q_4[x_4, x_5, y_3]$ in Figure 2(b) where $l_2 =$ lives_at using the LHS $\Phi_X = \{\text{name}[1], \text{dob}[2]\}$.

Table 6: Sample $\mathsf{GDD_L}$s Found

Data	Itemset of Φ_X of $\mathsf{GDD_L}$
`PLR`	{name[1], dob[2], lives_at[0]}
`Rest`	{name[2], address[2]}
`Cora`	{author[2], venue[0], pages[2]}
`DBAC`	{title[1], year[0]}
`DBSc`	{title[2]}
`AmGo`	{title[2], desc[2], manu[2], price[0]}

7.2 Evaluation of GDDs in ER

Goals & Takeaways. The objectives of these sets of experiments are to investigate: (a) the accuracy performance of our ER technique Certus *versus* merely using aggregate similarity thresholds (Exp-2a); (b) the impact of the number of $\mathsf{GDD_L}$s on the accuracy results (Exp-2b); (c) the time efficiency of Certus *w.r.t.* increasing number of $\mathsf{GDD_L}$s (Exp-2c); and lastly, (d) the performance of Certus as compared to existing state-of-the-art rule-based ER methods (Exp-2d).

A summary of the takeaways in the order of the above objectives are as follow. First, the use of $\mathsf{GDD_L}$s in Certus improve the precision of ER results without significant sacrifice of recall. Second, the accuracy of Certus' results increase with the number of $\mathsf{GDD_L}$s employed. Certus is efficient and scales well with both increasing data size and $\mathsf{GDD_L}$s. Last and not least, Certus consistently outperforms the rule-based ER method in [53], and performs generally better than all 3 methods of the current best rule-based ER system in [46].

Accuracy Metrics. We utilize the traditional metric of precision (P), recall (R), and f-measure ($\mathsf{F_1}$) to evaluate the correctness, completeness, and the overall accuracy of the ER results respectively. The definitions of the metrics are as follow: $\mathsf{P} = |\mathsf{TP}|/|\mathsf{FM}|$, $\mathsf{R} = |\mathsf{TP}|/|\mathsf{TM}|$ and $\mathsf{F_1} = 2\mathsf{PR}/(\mathsf{P}+\mathsf{R})$, where $|\mathsf{TP}|$ is the total number of correct matches; $|\mathsf{FM}|$ is the total number of matches found; and $|\mathsf{TM}|$ is the total number of true matches in the data.

Exp-2a. In this set of experiments, we investigate the benefit of including $\mathsf{GDD_L}$s in the ER match decision criteria (as in Certus) over just setting strict similarity thresholds.

For each dataset, we perform two sets of experiments: (a) using a minimum aggregated similarity threshold ($\vartheta \in [1, 5]$) as the matching decision criterion; and (b) using our approach, Certus, which employs $\mathsf{GDD_L}$s as matching rules. We randomly sample up to 100 eid-labelled profiles from

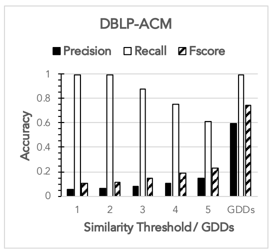


Figure 10: DBAC Acc.

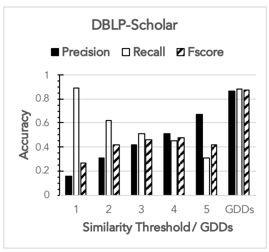


Figure 11: DBSc Acc.

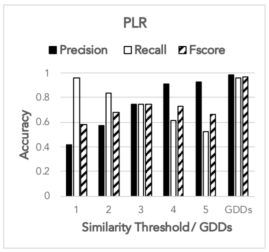


Figure 12: PLR Acc.

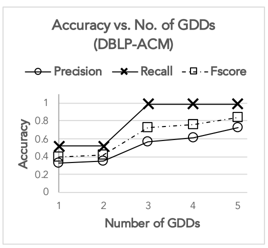


Figure 13: DBAC Imp.

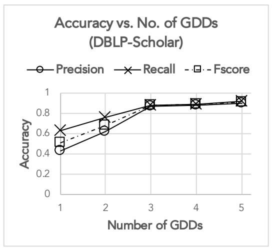


Figure 14: DBSc Imp.

each dataset for $\mathsf{GDD_L}$ discovery for Certus; and use no more than 3 of the discovered $\mathsf{GDD_L}$s as record-matching rules.

Figures 8 – 12 present bar charts showing the accuracy evaluations for the Rest, Cora, DBAC, DBSc, and PLR datasets respectively. In each plot, the x-axes represent the two scenarios above: values $1-5$ correspond to case (a) and GDDs correspond to (b). The plots show the values of P, R and F_1 for the two cases for the datasets, and case (b) which is Certus involving $\mathsf{GDD_L}$s (last group of bars in each plot) is consistently better for all the datasets on all three metrics.

For each bar chart in Figures 8 – 12, it can be seen that, increasing the minimum aggregated similarity threshold from 1 to 5 improves precision (dark-shaded bars) at a significant cost of recall (non-shaded bars). It is however noteworthy that, Certus (i.e., the GDD group of bars) achieves the highest precision without any significant compromise of recall in each plot. Indeed, the recall of Certus for each dataset is comparable to the best of case (a) when aggregated similarity is 1, yet with a considerably higher precision.

Missing attributes or null-values affect recall. This is because, attributes with missing values do not form $\mathsf{GDD_L}$s during discovery, and consequently, some true matching profile pairs may not be resolved due to absence of rules to confirm their match. For instance, the Cora dataset has the highest null-values in Table 5, hence, records the lowest recall (cf. Figure 9). The performance of Certus is, however, consistent on both semi-structured data (in Figure 12) and structured data (in Figures 8 –11).

The DBAC dataset deviates from the other datasets in two ways as Figure 10 shows. First, the effect of increasing similarity threshold has a lower pay-off than in other datasets. Second and more important, the use of $\mathsf{GDD_L}$s is not as effective as seen in the other datasets (i.e., it is the only dataset upon which Certus records precision below 85%).

We therefore probe into DBAC to unearth the reasons for the above-mentioned anomalies by examining its ground-truth, which has one-to-one mappings of duplicates. The citation reference 'journals/sigmod/SnodgrassGIMSU98', for example, is from the DBLP-portion of the dataset, and its *one and only* correct match is the ACM citation reference '290599' shown in Figure 15 (a) and (b) respectively. However, Certus finds two more matches in addition to the only documented true match: 'journals/sigmod/Snodgrass-GIMSU98a' and '390004', shown in Figure 15 (c) and (d).

A cross-check of these results in the real DBLP and ACM on-line catalogues prove that the ground truth is correct. Nonetheless, there is missing information that helps to disambiguate these four citation references in the real-world: page numbers. Since the DBAC dataset does not include the page numbers of the citations, it is impossible for even human experts to resolve these ambiguities (e.g., in Figure 15). Interestingly, this incident abounds in the DBAC dataset, hence, the relatively 'poor' precision of Certus on this dataset. In other words, we follow the ground truth to the letter, labelling all matches outside it as false-positives.

	DBLP	ACM
ID	journals/sigmod/SnodgrassGIMSU98	290599
TITLE	Chair's Message	Reminiscences on influential papers
AUTHORS	Richard T. Snodgrass	Richard Snodgrass
VENUE	SIGMOD Record	ACM SIGMOD Record
YEAR	1998	1998
	(a)	(b)
ID	journals/sigmod/SnodgrassGIMSU98a	390004
TITLE	Reminiscences on influential papers	Reminiscences in influential papers
AUTHORS	Richard T. Snodgrass	Richard Snodgrass
VENUE	SIGMOD Record	ACM SIGMOD Record
YEAR	1998	1998
	(c)	(d)

Figure 15: Some interesting 'false-positves' in DBAC dataset

Exp-2b. In this experiment, we investigate the impact of the number of $\mathsf{GDD_L}$s used by Certus on the three accuracy metrics. For this evaluation, we report the empirical results for *only* the 'worse-performing' dataset DBAC and its counterpart DBSc, due to limits on number of pages, to show how their accuracies improve with more $\mathsf{GDD_L}$s. The results of these experiments are shown in Figures 13 & 14. On the vertical axes are the accuracy values, and the increasing number of $\mathsf{GDD_L}$s are on the horizontal axes. It is distinctly clear from the plots that more $\mathsf{GDD_L}$s indeed improve the values of all three accuracy metrics for both datasets. However, for the DBAC results in Figure 13, in particular, given the discussed phenomenon, Certus finds multiple matches like those in Figure 15 which adversely impacts precision, although it improves with the number of $\mathsf{GDD_L}$s used. The recall, on the other hand, is near perfect from 3 $\mathsf{GDD_L}$s onwards, since each duplicate has only one true match which are almost always found, albeit in addition to few others.

Exp-2c. We test the efficiency of our ER solution and examine the computational overhead involved with using $\mathsf{GDD_L}$s as part of the matching decision. For this test, we use the PLR dataset, as it is the largest, using up to the full database of a million profiles; and vary the number of $\mathsf{GDD_L}$s used in the match decision of Certus. Our findings for this evaluation are in Figure 16, showing four case: the baseline with no $\mathsf{GDD_L}$s; and when 2, 4 and 8 $\mathsf{GDD_L}$s are used. The ER time (in minutes) is given on the vertical axis versus an increasing number of entity profiles from 1K to 1M. We observe a linear time performance and low extra-cost of using $\mathsf{GDD_L}$s.

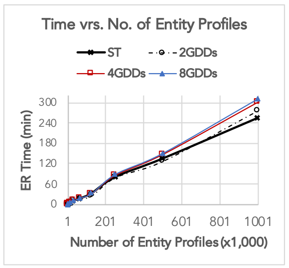


Figure 16: ER Time

Exp-2d. Here, we compare the performance of Certus to the state-of-the-art rule-based ER methods; and present the results in Table 7. The current best rule-based ER solutions

Table 7: Comparison to published results

Data	F_1 (%)				
	SIFI [53]	RS1 [46]	RS2 [46]	RS-Cons. [46]	Certus
Cora	61.50	82.89	86.40	92.23	92.31
DBSc	89.50	88.21	90.83	92.58	91.90
AmGo	61.90	56.70	61.45	63.78	65.81

in the literature are SIFI [53] and RS-Consensus [46]. Given a user/expert-provided Disjunctive Normal Forms (DNF), SIFI attempts to find the best similarity functions and their associated thresholds of similarity for ER. RS-Consensus, on the other hand, employs the program synthesis tool for automatically learning matching rules in the form of General Boolean Formulae (GBF). The advantages of the latter approach over the former are: GBFs are more expressive than DNFs, and there is no need for users to specify DNFs.

In contrast to these approaches, we define, mine and utilize a new class of data dependencies, i.e. $\mathsf{GDD_L}$s, to perform ER. $\mathsf{GDD_L}$s by definition are more expressive and able to capture the semantics of both GBFs and NDFs used in RS-Consensus and SIFI respectively. Furthermore, Certus is schema-agnostic and can be used in both structured and semi-structured data.

For a fair comparison of the accuracy of Certus to those of the above-mentioned works, we generate comparable datasets following the procedures described in [46,53], i.e., group B datasets in Table 5. Furthermore, we follow the *status quo* in [46] and perform a 5-fold cross-validation. We divided each dataset into five equal portions/folds randomly, and performed five experiments. At every instance of the five experiments, four folds were used for $\mathsf{GDD_L}$ discovery while the remaining one of the five folds is set aside as the test set. Moreover, we calibrate Certus to match the settings of the second method of [46] (i.e., RS2 in Table 7) by ensuring that the set of $\mathsf{GDD_L}$s used by Certus for each dataset has no more than a total of 15 distinct distance constraints—we refer interested readers to [46] for details.

The average of all the F_1 scores in the five experiments are reported alongside the published results in [46] in Table 7. It is noteworthy that our ER method, Certus, with settings equivalent to RS2 consistently outperforms SIFI and 2 (RS1, RS2) of the 3 techniques proposed in [46]; and performs generally better than the best method, RS-Consensus, in [46].

We remark that, a direct comparison of Certus to non-rule-based ER methods is not useful. However, it suffices to state the performance of Certus on DBSc, for example, is close to that reported by the current best non-rule-based ER solution [37] in the literature (i.e., 94.7). The ER technique in [37] uses deep learning (DL). DL methods require strenuous training of models, involving complex tuning of parameters. Moreover, it is difficult to interpret DL models and/or encode domain knowledge in them.

8. RELATED WORK

Our work is at the cutting-edge of different research fields: graph constraints R_1, learning constraints R_2, and ER R_3.

R_1. The design of formal constraints for graph data is gaining increasing research attention in recent years. Some of the pioneering works, e.g., [35,57], focus on the definition of FDs and CFDs for RDF graphs for data transformation and anomaly detection. More related graph constraints to GDDs are GKeys [18], GFDs [25], and GEDs [24]. GKeys are a class of keys for graphs based on isomorphic graph properties for identifying unique entities in graphs; whereas GFDs impose attribute-value dependencies (like FDs and CFDs) upon topological structures in graphs. GEDs unify and subsume the semantics of both GFDs and GKeys. In this work, we extend GEDs as GDDs with distance functions and introduce a new type of constraints on relation labels.

R_2. The discovery of data dependencies is a well-studied problem in the relational data. The discovery of FDs has, particularly, received significant resarch attention over last few decades with a plethora of solutions proposed in the literature (cf. [36] and [43] for a recent review and comparison of approaches, respectively). Notable works on the discovery of CFDs and other extensions of FDs can be found in a new survey in [11]. The discovery of matching rules studied in [48,51,54] are the closest to the $\mathsf{GDD_L}$ discovery problem. However, all previous works are in the relational data only; and do not include the discovery of graph patterns, distance/similarity constraints, and relationships. The existence of labelled data is the centrepiece for all matching rule learners. In the absence of good labelled data, active learning [28] and crowd-sourcing [26,52] methods can be used to generate them. This, however, is a different line of work.

R_3. Entity resolution is a well-known database problem that has attracted large volumes of contributions in the literature (cf. [38] for a comprehensive lecture on the topic). In general, works on ER can be broadly categorised as: (a) techniques that improve efficiency of ER (see [42] for a comparison of these approaches); (b) works that focus on accuracy of the ER results (cf. [32] for a thorough evaluation of methods); and (c) those that trade-off between (a) and (b), e.g., [6,13,44,55]. This work belongs to group (b) above. In this category, some techniques exploit diverse metrics to compute similarity of profile pairs based on their attributes-values, e.g., [29,46,53]; others exploit the inherent relationships amongst entity pairs, e.g., [12,16,30,31]; and some rely on probabilistics models [47,56]. Our work employs both the attribute and relational similarities to match profile pairs through the use of GDDs. In fact, the graphs patterns and distance constraints of GDDs respectively encode the relation and attribute similarity among profile pairs.

9. CONCLUSION

In this work, we presented a new effective solution to ER in graph data. This involves the proposal of a novel class of dependencies, GDDs, which are more expressive and subsume GEDs, relational MDs and CMDs. We studied the discovery of GDDs in duplicate-labelled graph and developed an algorithm for mining a non-redundant set of GDDs. Motivated by the challenge of setting the right bounds of similarity for rule-based ER methods, we showed how GDDs can improve the precision of ER results without significant compromise of recall. We performed experiments on five real-world benchmark datasets and a proprietary dataset to demonstrate the effectiveness, and efficiency of both the GDD discovery algorithm and the ER solution, Certus. The empirical results showed accuracy gains of Certus, and revealed some interesting and previously unreported phenomenon in the ground truth of one of the benchmark datasets. Furthermore, the results showed that Certus attains accuracy performances comparable to the current best ER techniques.

Acknowledgments. This work is supported by Data to Decisions CRC—ILE/ELR Project DC160031—funded by the Australian Government's CRC Programme.

10. REFERENCES

[1] Cora dataset. http://www.cs.umass.edu/~mccallum/data/cora-refs.tar.gz. Last accessed in June 2018.

[2] Dblp-acm & dblp-scholar datasets. https://dbs.uni-leipzig.de/de/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution. Last accessed in June 2018.

[3] Restaurant dataset. http://www.cs.utexas.edu/users/ml/riddle/data. Last accessed in June 2018.

[4] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[6] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progressive approach to relational entity resolution. *PVLDB*, 7(11):999–1010, 2014.

[7] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 918–929, 2006.

[8] W. W. Armstrong and C. Deobel. Decompositions and functional dependencies in relations. *ACM Trans. Database Syst.*, 5(4):404–430, Dec. 1980.

[9] C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*, pages 73–85, 1981.

[10] P. S. G. C., C. Sun, K. G. K., H. Zhang, F. Yang, N. Rampalli, S. Prasad, E. Arcaute, G. Krishnan, R. Deep, V. Raghavendra, and A. Doan. Why big data industrial systems need rules and what we can do about it. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 265–276, 2015.

[11] L. Caruccio, V. Deufemia, and G. Polese. Relaxed functional dependencies - A survey of approaches. *IEEE Trans. Knowl. Data Eng.*, 28(1):147–165, 2016.

[12] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Exploiting context analysis for combining multiple entity resolution systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 207–218, 2009.

[13] R. Cheng, E. Lo, X. S. Yang, M. Luk, X. Li, and X. Xie. Explore or exploit? effective strategies for disambiguating large databases. *PVLDB*, 3(1):815–825, 2010.

[14] L. Chiticariu, Y. Li, and F. R. Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 827–832, 2013.

[15] D. Deng, G. Li, H. Wen, and J. Feng. An efficient partition based method for exact set similarity joins. *PVLDB*, 9(4):360–371, 2015.

[16] X. Dong, A. Y. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 85–96, 2005.

[17] W. Fan. Dependencies revisited for improving data quality. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 159–170, 2008.

[18] W. Fan, Z. Fan, C. Tian, and X. L. Dong. Keys for graphs. *PVLDB*, 8(12):1590–1601, 2015.

[19] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma. Dynamic constraints for record matching. *VLDB J.*, 20(4):495–520, 2011.

[20] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[21] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2):6:1–6:48, 2008.

[22] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Trans. on Knowl. and Data Eng.*, 23(5):683–698, May 2011.

[23] W. Fan, C. Hu, X. Liu, and P. Lu. Discovering graph functional dependencies. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 427–439, 2018.

[24] W. Fan and P. Lu. Dependencies for graphs. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 403–416, 2017.

[25] W. Fan, Y. Wu, and J. Xu. Functional dependencies for graphs. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 1843–1857, 2016.

[26] D. Firmani, B. Saha, and D. Srivastava. Online entity resolution using an oracle. *PVLDB*, 9(5):384–395, 2016.

[27] C. Giannella and E. Robertson. On approximation measures for functional dependencies. *Information Systems*, 29(6):483 – 507, 2004. ADBIS 2002: Advances in Databases and Information Systems.

[28] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 601–612, 2014.

[29] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995.*, pages 127–138, 1995.

[30] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Trans. Database Syst.*, 31(2):716–767, 2006.

[31] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*, pages 262–273, 2005.

[32] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1):484–493, 2010.

[33] S. Kwashie, J. Liu, J. Li, and F. Ye. Conditional differential dependencies (cdds). In *Advances in Databases and Information Systems - 19th East European Conference, ADBIS 2015, Poitiers, France, September 8-11, 2015, Proceedings*, pages 3–17, 2015.

[34] D. Lange and F. Naumann. Frequency-aware similarity measures: why arnold schwarzenegger is always a duplicate. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 243–248, 2011.

[35] G. Lausen, M. Meier, and M. Schmidt. Sparqling constraints for RDF. In *EDBT 2008*, pages 499–509, 2008.

[36] J. Liu, J. Li, C. Liu, and Y. Chen. Discover dependencies from data - A review. *IEEE Trans. Knowl. Data Eng.*, 24(2):251–264, 2012.

[37] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 19–34, 2018.

[38] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[39] F. Panahi, W. Wu, A. Doan, and J. F. Naughton. Towards interactive debugging of rule-based entity matching. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017.*, pages 354–365, 2017.

[40] G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl. Meta-blocking: Taking entity resolutionto the next level. *IEEE Trans. Knowl. Data Eng.*, 26(8):1946–1960, 2014.

[41] G. Papadakis, G. Papastefanatos, T. Palpanas, and M. Koubarakis. Scaling entity resolution to large, heterogeneous data with enhanced meta-blocking. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016.*, pages 221–232, 2016.

[42] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas. Comparative analysis of approximate blocking techniques for entity resolution. *PVLDB*, 9(9):684–695, 2016.

[43] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J. Rudolph, M. Schönberg, J. Zwiener, and F. Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *PVLDB*, 8(10):1082–1093, 2015.

[44] T. Papenbrock, A. Heise, and F. Naumann. Progressive duplicate detection. *IEEE Trans. Knowl. Data Eng.*, 27(5):1316–1329, 2015.

[45] T. Papenbrock and F. Naumann. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 821–833, New York, NY, USA, 2016. ACM.

[46] R. Singh, V. V. Meduri, A. K. Elmagarmid, S. Madden, P. Papotti, J. Quiané-Ruiz, A. Solar-Lezama, and N. Tang. Synthesizing entity matching rules by examples. *PVLDB*, 11(2):189–202, 2017.

[47] P. Singla and P. M. Domingos. Entity resolution with markov logic. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 572–582, 2006.

[48] S. Song and L. Chen. Discovering matching dependencies. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1421–1424, New York, NY, USA, 2009. ACM.

[49] S. Song and L. Chen. Differential dependencies: Reasoning and discovery. *ACM Trans. Database Syst.*, 36(3):16:1–16:41, Aug. 2011.

[50] S. Song and L. Chen. Efficient discovery of similarity constraints for matching dependencies. *Data Knowl. Eng.*, 87:146–166, 2013.

[51] S. Song, L. Chen, and H. Cheng. On concise set of relative candidate keys. *PVLDB*, 7(12):1179–1190, 2014.

[52] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[53] J. Wang, G. Li, J. X. Yu, and J. Feng. Entity matching: How similar is similar. *PVLDB*, 4(10):622–633, 2011.

[54] Y. Wang, S. Song, L. Chen, J. X. Yu, and H. Cheng. Discovering conditional matching rules. *ACM Trans. Knowl. Discov. Data*, 11(4):46:1–46:38, June 2017.

[55] S. E. Whang, D. Marmaros, and H. Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Trans. Knowl. Data Eng.*, 25(5):1111–1124, 2013.

[56] T. Ye and H. W. Lauw. Structural constraints for multipartite entity resolution with markov logic network. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1691–1694, 2015.

[57] Y. Yu and J. Heflin. Extending functional dependency to detect abnormal data in RDF graphs. In *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, pages 794–809, 2011.