Approximation Algorithms for Schema-Mapping Discovery from Data Examples

Balder ten Cate[2,1], Phokion G. Kolaitis[1,3], Kun Qian[1], and Wang-Chiew Tan[1]

University of California Santa Cruz[1]
LogicBlox, Inc.[2]
IBM Research - Almaden[3]
{btencate,kolaitis,kunqian,tan}@soe.ucsc.edu

Abstract. In recent years, data examples have been at the core of several different approaches to schema-mapping design. In particular, Gottlob and Senellart introduced a framework for schema-mapping discovery from a single data example, in which the derivation of a schema mapping is cast as an optimization problem. Our goal is to refine and study this framework in more depth. Among other results, we design a polynomial-time $\log(n)$-approximation algorithm for computing optimal schema mappings from a given data example, for a restricted class of schema mappings; moreover, we show that this approximation ratio cannot be improved.

Keywords: Schema Mappings, Data Examples, Approximation

1 Introduction

Schema mappings between a source schema and a target schema constitute the essential building blocks for the specification of data exchange and data integration tasks [6,15,16]. However, deriving a schema mapping between two schemas can be an involved and time-consuming process. Most commercial systems (e.g., Altova Mapforce and the Stylus Studio) and research prototypes derive schema mappings automatically using correspondences between the elements of two schemas, as specified by a user through a graphical interface [7,13,18]. A shortcoming of this approach is that multiple non-equivalent schema mappings may be compatible with the same set of correspondences. More recently, data examples have been used as the basis of alternative approaches to schema-mapping design.

Data examples have been used in several different areas of data management (e.g., see [11,17,19,21]). In the context of schema-mapping design, a data example is a pair (I, J) consisting of a source instance and a target instance. In [2,22], the goal is, given a schema mapping, to *produce* meaningful data examples, that illustrate the schema mapping at hand. In [1], the goal is to investigate whether a given schema mapping can be *uniquely characterized* by a finite set of examples. In the reverse direction, there has been work on deriving a schema mapping that *fits* one or more given data examples [3,8]

Gottlob and Senellart [12] introduced and studied a cost model for deriving a schema mapping from a ground data example, i.e., a data example consisting of instances without nulls. Ideally, given a ground data example (I, J), one would like to find a GLAV schema mapping $\mathcal{M}$ that is *valid* (meaning that (I, J) satisfied the constraints of $\mathcal{M}$) and *fully explaining* for (I, J) (meaning that every fact in J belongs to every target instance K such that (I, K) satisfies the constraints of $\mathcal{M}$). However, such a schema mapping need not exist. For this reason, Gottlob and Senellart developed a framework that uses two schema-mapping languages: the standard language of GLAV constraints and an extended language GLAV$^{=,\neq}$ of *repairs* that augments, in precise way, GLAV constraints with equalities, inequalities, and ground facts. The *cost* of a GLAV schema mapping $\mathcal{M}$ w.r.t. a data example (I, J) is the minimum of the sizes of valid and fully explaining schema mappings $\mathcal{M}'$ obtained from $\mathcal{M}$ via a sequence of operations that transform $\mathcal{M}$ to a schema mapping in the extended language. Given a ground data example (I, J), the goal then is to find an *optimal* GLAV schema mapping for (I, J), that is to say, a GLAV schema mapping whose cost w.r.t. (I, J) is as small as possible. Gottlob and Senellart [12] investigated several different decision problems arising naturally in the context of the above framework, including the Existence-Cost problem: given a ground data example (I, J) and a positive integer k, is there a GLAV schema mapping $\mathcal{M}$ whose cost w.r.t. (I, J) is at most k? In [12], it is shown that the Existence-Cost problem is NP-hard, and that it belongs to the third level Σ_3^p of the polynomial hierarchy.

Here, we contribute to the study of schema-mapping discovery from data examples by refining and investigating the Gottlob-Senellart framework in more depth. We refine the framework by considering several different schema-mapping languages. At the base level, we consider sublanguages $\mathcal{L}$ of the standard language of GLAV constraints, such as the languages of GAV constraints, LAV constraints, and SH-LAV (single-head LAV) constraints. For each of these schema-mapping languages $\mathcal{L}$, we consider two corresponding repair languages, namely, $\mathcal{L}^{=,\neq}$ and $\mathcal{L}^{=}$; the former extends $\mathcal{L}$ with equalities, inequalities, and ground facts (as in [12]), while the latter extends $\mathcal{L}$ with equalities and ground facts only.

The main algorithmic problem in the Gottlob-Senellart framework is to compute an optimal schema mapping for a given ground data example. The tractability of this optimization problem was left open in [12] (the hardness of the Existence-Cost problem does not imply hardness of the optimization problem, because computing the cost of a given schema mapping for a given ground data example is itself a hard problem). We show that this optimization problem is indeed hard for GLAV mappings, w.r.t. both GLAV$^{=}$-repair and GLAV$^{=,\neq}$-repairs. More precisely, unless RP = NP, there is no polynomial-time algorithm that, given a ground data example, computes a GLAV mapping whose cost is bounded by some fixed polynomial in the cost of the optimal GLAV mapping. Moreover, an analogous result holds for GAV mappings.

After this, we design an approximation algorithm for computing near optimal schema mappings for the case of SH-LAV schema mappings. Specifically, we present a polynomial-time $O(\log n)$-approximation algorithm that, given a data example (I, J), produces a SH-LAV mapping $\mathcal{M}$ together with a SH-LAV$^{=}$-repair $\mathcal{M}'$ of $\mathcal{M}$ for (I, J), whose cost is within a logarithmic factor of the cost of the optimal SH-LAV mapping for (I, J). Finally, we establish that this is a best possible approximation result.

2 Preliminaries

Schemas and Instances An *instance* over a schema $\mathbf{R} = \{R_1, \ldots, R_k\}$ can be identified with the finite set of all *facts* $R_i(a_1, \ldots, a_m)$, such that R_i is a relation symbol of $\mathbf{R}$ and $(a_1, \ldots, a_m)$ is a tuple that belongs to the relation R_i^I of I interpreting R_i. Database instances contain values that are either *constants* or *nulls.* A *ground instance* is an instance such all of its facts are *ground.* i.e., they consist entirely of constants. In this paper, we will primarily consider ground instances, and we will, at times, drop the adjective "ground". We write $adom(I)$ to denote the *active domain* of an instance I.

Schema Mappings Let $\mathbf{S}, \mathbf{T}$ be two relational schemas, called the *source* schema and the *target* schema. A *schema mapping* is a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ consisting of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$, and a set Σ of constraints that are typically expressed in some fragment of first-order logic.

A *GLAV (Global-and-Local-As-View) constraint*, also known as a *tuple-generating dependency (tgd)*, is a FO-formula of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x},\mathbf{y}))$, where $\varphi(\mathbf{x})$ is a conjunction of atoms over $\mathbf{S}$ and $\psi(\mathbf{x},\mathbf{y})$ is a conjunction of atoms over $\mathbf{T}$. We will often drop the universal quantifiers when writing constraints.

The following are two important special cases of GLAV constraints: (1) A GAV (Global-As-View) constraint is a GLAV constraint whose right-hand side is a single atom without existential quantifiers, i.e., it is of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow T(\mathbf{x}))$. (2) A LAV (Local-As-View) constraint is GLAV constraint whose left-hand side is a single atom, i.e. it is of the form $\forall \mathbf{x}(S(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x},\mathbf{y}))$. There is another special case of LAV constraints, called *SH-LAV (Single-Head LAV) constraints.* A SH-LAV constraint is a GLAV constraint in which both the left-hand side and right-hand side are a single atom, i.e., it is of the form $\forall \mathbf{x}(S(\mathbf{x}) \rightarrow \exists \mathbf{y} T(\mathbf{x},\mathbf{y}))$. In fact every DL-lite concept subsumption axiom is equivalent to a SH-LAV constraint [5].

Example 1. $\text{Geo}(x, y) \rightarrow \text{City}(y)$ is a LAV constraint that is also a GAV constraint, and hence, in particular, is a SH-LAV contraint; $\text{Geo}(x, y) \rightarrow \exists z \text{City}(z)$ is a SH-LAV constraint that is not a GAV constraint. Finally, the constraint $\text{Geo}(x, y) \wedge \text{Geo}(x, z) \rightarrow \text{City}(y)$ is a GAV constraint that is not a LAV constraint.

A *GLAV schema mapping* is a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a finite set of GLAV constraints. The notions of GAV, LAV, and SH-LAV schema mappings are defined in an analogous way.

Data Examples Let $\mathbf{S}$ be a source schema and $\mathbf{T}$ be a target schema. A *data example* is a pair (I, J) such that I is a source instance and J is a target instance. A data example (I, J) is *ground* if both I and J are ground instances. The size $||(I, J)||$ of a data example (I, J) is the total number of facts in I and J.

Table 1. Data Example (I, J)

Source Instance		**Target Instance**
Geo(US, San Francisco)	Geo(Calif, San Jose)	City(San Francisco)
Geo(US, Los Angeles)	Geo(Calif, San Francisco)	City(San Jose)
Geo(US, Miami)	Geo(Calif, Los Angeles)	City(San Diego)
Geo(US, Boston)	Geo(Calif, San Diego)	City(Los Angeles)
Geo(US, New York)	Geo(Canada, Vancouver)	City(Boston)
Geo(NorthAm, Boston)	Geo(Germany, Berlin)	City(Toronto)
Geo(NorthAm, Toronto)	Geo(Japan, Tokyo)	City(New York)
Geo(NorthAm, New York)	Geo(China, Beijing)	City(Miami)
Geo(NorthAm, Miami)	Geo(France, Paris)	
	Geo(UK, London)	

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping, and let (I, J) be a ground data example. We say that $\mathcal{M}$ is *valid* for (I, J) if $(I, J) \models \Sigma$, that is, $(I, J) \in E$ satisfies all constraints in Σ. We say that $\mathcal{M}$ *explains* a fact f of J with respect to I if, for all target instances K such that $(I, K) \models \Sigma$, we have that $f \in K$. Finally, we say that $\mathcal{M}$ is *fully explaining* for a data example (I, J) if $\mathcal{M}$ explains each fact of J with respect to I. We will use the expression *vfe* as a shorthand for "valid and fully explaining".

In [1,3], a schema mapping $\mathcal{M}$ was said to *fit* a data example (I, J) if J is a universal solution of I w.r.t. $\mathcal{M}$, i.e., J is a solution for I w.r.t. $\mathcal{M}$ such that for every solution J' for I, there is a homomorphism from J to J' that maps constants to themselves (see [10] for more on universal solutions). It is not hard to verify that if (I, J) is a ground data example and $\mathcal{M}$ is a schema mapping, then $\mathcal{M}$ fits (I, J) if and only if $\mathcal{M}$ is vfe for (I, J).

Example 2. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{\text{Geo(area, city)}\}$, $\mathbf{T} = \{\text{City(cityName)}\}$, $\Sigma = \{\text{Geo}(x, y) \rightarrow \text{City}(y)\}$. Consider the data examples (I_i, J_i), $i = 1, 2, 3$, where

$$I_1 : \text{Geo(CA, San Francisco)}, \text{Geo(CA,San Jose)}, \text{Geo(US,Boston)}, \text{Geo(US, Los Angeles)}$$
$$J_1 : \text{City(San Francisco)}, \text{City(San Jose)}$$

$$I_2 : \text{Geo(CA, San Francisco)}$$
$$J_2 : \text{City(San Francisco)}, \text{City(New York)}$$

$$I_3 : \text{Geo(CA, San Francisco)}, \text{Geo(US,New York)}$$
$$J_3 : \text{City(San Francisco)}, \text{City(New York)}$$

In these data examples, since I_1 contains Geo(US,Boston), but City(Boston) is not in J_1, we have that $\mathcal{M}$ is not valid for (I_1, J_1); moreover, $\mathcal{M}$ is valid for (I_2, J_2), but it fails to explain the target fact City(New York); however, $\mathcal{M}$ is valid and fully explaining for (I_3, J_3). Using a simple automorphism argument, it is easy to see there is no valid and fully explaining GLAV schema mapping for (I_1, J_1). Moreover, since J_2 contains a constant that does not appear in the I_2, there is no valid and fully explaining GLAV schema mapping for (I_2, J_2).

3 Repair Framework and Cost Model

In [12], the language of GLAV constraints was used as the "base language", and an extended "repair language" was introduced; the repair language includes equalities, inequalities, and ground facts, and is used for "repairing" GLAV schema mappings, so that they are valid and fully explaining for a given ground data example. We refine this framework by considering different sublanguages of the language of GLAV constraints, as well different repair languages.

Definition 1. *Fix a schema-mapping language $\mathcal{L}$ (e.g., GLAV). An $\mathcal{L}^{=,\neq}$-*constraint *is a formula θ of the form $\forall \boldsymbol{x}(\varphi(\boldsymbol{x}) \wedge \alpha(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y}(\psi(\boldsymbol{x},\boldsymbol{y}) \wedge \beta(\boldsymbol{x},\boldsymbol{y})))$, where*

1. *$\forall \boldsymbol{x}(\varphi(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y}\psi(\boldsymbol{x},\boldsymbol{y}))$ is an $\mathcal{L}$-constraint;*
2. *$\alpha(\boldsymbol{x})$ is a possibly empty conjunction of formulas of the form $x \sim c$, where $\sim \in \{=, \neq\}$, $x \in \boldsymbol{x}$, and c is a constant;*
3. *$\beta(\boldsymbol{x},\boldsymbol{y})$ is a possibly empty conjunction of formulas of the form $(x_1 = c_1 \wedge \ldots \wedge x_n = c_n) \rightarrow y = c$, where each $x_i \in \boldsymbol{x}$, $y \in \boldsymbol{y}$, and $c_1, \ldots, c_n, c$ are constants.*

*A $\mathcal{L}^{=}$-*constraint *is a $\mathcal{L}^{=,\neq}$-constraint with no conjuncts of the form $x \neq c$ in α.*

We will write $\mathcal{L}$ to denote a schema-mapping language (e.g., GLAV), and we will write $\mathcal{L}^{=,\neq}$ and $\mathcal{L}^{=}$ to denote the corresponding repair language (with and without inequalities). We will use $\mathcal{L}^*$ to refer to either $\mathcal{L}^{=,\neq}$ or $\mathcal{L}^{=}$. We say that the formula θ as above is a $\mathcal{L}^*$-*repair* of the constraint $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x},\mathbf{y}))$.

Definition 2. *An $\mathcal{L}^*$-*repair *of an $\mathcal{L}$-schema mapping $\mathcal{M} = (\boldsymbol{S}, \boldsymbol{T}, \Sigma)$ is an $\mathcal{L}^*$-schema mapping $\mathcal{M}' = (\boldsymbol{S}, \boldsymbol{T}, \Sigma')$ such that each $\psi \in \Sigma'$ is either a ground fact or is an $\mathcal{L}^*$-repair of some $\phi \in \Sigma$, and, for each $\phi \in \Sigma$, at least one $\mathcal{L}^*$-repair of ϕ belongs to Σ'. We write $repair_{\mathcal{L}^*}(\mathcal{M})$ to denote the set of all $\mathcal{L}^*$-repairs of $\mathcal{M}$.*

The above definition differs from that of repairs in [12] in that we allow $\mathcal{M}'$ to contain multiple repairs of the same $\mathcal{L}$-constraint from $\mathcal{M}$, whereas, in [12], the $\mathcal{L}^*$-constraints of $\mathcal{M}'$ stand in a one-to-one correspondence with the $\mathcal{L}$-constraints of $\mathcal{M}$. There are cases in which a $\mathcal{L}$-constraint may need to be repaired more than once with, say, different combinations of equalities and inequalities. In such cases, the optimal schema mapping in [12] may contain multiple copies of the same GLAV constraint or multiple GLAV constraints that are identical up to a renaming of variables (see Example 4). Our definition addresses this shortcoming.

Example 3. Recall that the data example (I_1, J_1) in Example 2 had no vfe GLAV schema mapping. Consider the following $GLAV^{=,\neq}$ schema mappings:

- $\mathcal{M}_1$=$\{\text{Geo}(x,y) \wedge (x = \text{CA}) \rightarrow \text{City}(y)\}$
- $\mathcal{M}_2$=$\{\text{Geo}(x,y) \rightarrow \exists z\ \text{City}(z) \wedge (y = \text{San Jose} \rightarrow z = \text{San Jose})$ $\wedge\ (y = \text{San Francisco} \rightarrow z = \text{San Francisco})\}$
- $\mathcal{M}_3$=$\{\text{City(San Francisco)}, \text{City(San Jose)}\}$
- $\mathcal{M}_4$=$\{\text{Geo}(x,y) \wedge (x \neq \text{US}) \rightarrow \text{City}(y)\}$
- $\mathcal{M}_5$=$\{\text{Geo}(x,y) \wedge (x = \text{moon}) \rightarrow \text{City}(x)\}$

$\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$ are vfe for (I_1, J_1), but in different ways: $\mathcal{M}_1$ consists of a repair of $\text{Geo}(x,y) \rightarrow \text{City}(y)$ that uses an equality to restrict the constraint to source facts whose first attribute has value CA; $\mathcal{M}_2$ consists of a repair of $\text{Geo}(x,y) \rightarrow \exists z\text{City}(z)$ that uses two conditional equalities to explicitly specify a value of z depending on the value of y; $\mathcal{M}_3$ is a repair of the empty schema mapping, and it lists all the ground facts of J_1; finally, $\mathcal{M}_4$ consists of a repair of the same constraint as $\mathcal{M}_1$ that, instead, uses an inequality.

$\mathcal{M}_5$ is valid, but does not explain (I_1, J_1). It uses the equality $(x = \mathit{moon})$, where *moon* is outside the active domain of (I_1, J_1). We informally say that this equality *cancels* $\mathcal{M}_5$ (with respect to the data example (I_1, J_1)).

Example 4. We illustrate the need for multiple repairs of the same constraint. For the data example (I, J) in Table 1, consider the GLAV schema mappings

- $\mathcal{M}_1 = \{\text{Geo}(x, y) \rightarrow \text{City}(y)\}$,
- $\mathcal{M}_2 = \{\text{Geo}(x_1, y_1) \rightarrow \text{City}(y_1), \text{Geo}(x_2, y_2) \rightarrow \text{City}(y_2)\}$

and consider the GLAV$^{=,\neq}$-schema mapping $\mathcal{M}_r$ specified by the constraint

$$\text{Geo}(x, y) \wedge (x = \text{Calif}) \rightarrow \text{City}(y),\ \ \text{Geo}(x, y) \wedge (x = \text{NorthAm}) \rightarrow \text{City}(y).$$

Note that $\mathcal{M}_r$ is vfe for (I, J). Note also that $\mathcal{M}_2$ consists of two renamings of the GLAV constraints of $\mathcal{M}$. By our definition of repair, $\mathcal{M}_r$ is a repair of $\mathcal{M}_1$ and also of $\mathcal{M}_2$. However, according to the definition of repair in [12], $\mathcal{M}_r$ does not qualify as a repair of $\mathcal{M}_1$, and, indeed, one of the smallest repairs of $\mathcal{M}_1$ is obtained by adding an equality (x = US) to the left-hand side of the constraint in $\mathcal{M}_1$ together with adding three ground facts: City(San Jose), City(Toronto), and City(San Diego). Since the cost of a schema mapping w.r.t. a data example will be measured by the size of the smallest repair, we have that, under the cost model of [12], $\mathcal{M}_2$ has a lower cost than $\mathcal{M}_1$, which is counterintuitive.

As pointed out in [12], if (I, J) is a ground data example, then there is always a vfe GLAV$^{=,\neq}$ schema mapping for (I, J). In fact, trivially, the collection of all the ground facts in J is a vfe schema mapping for (I, J). This shows that, for all schema-mapping languages $\mathcal{L}$ considered her, every ground data example has a $\mathcal{L}^{=}$ repair. Indeed, every $\mathcal{L}$-schema mapping has a $\mathcal{L}^{=}$-repair that is vfe (obtained by cancelling all constraints and adding all ground target facts).

The cost model introduced in [12] focuses on finding a schema mapping in the base language, such that the cost of transforming it to a vfe schema mapping in the repair language is as small as possible.

Definition 3. *[12] The* size *of a first-order formula φ, denoted size(φ), is the number of occurrences of variables and constants in φ (each variable and constant is counted as many times as it occurs in φ); occurrences of variables as arguments of quantifiers do not count. A ground fact $R(a_1, \dots, a_n)$, for present purposes, is viewed as shorthand for $\exists x_1, \dots, x_n R(x_1, \dots, x_n) \wedge x_1 = a_1 \wedge \dots \wedge x_n = a_n$; therefore, we consider its* size *to be $3n$. The* size *of a repair of a schema mapping is the sum of the sizes of the constraints and the ground facts of the repair.*

Note that the motivation for the above treatment of ground facts is to discourage the use of ground facts in repairing schema mappings.

Definition 4. *The* cost *of an $\mathcal{L}$-schema mapping $\mathcal{M}$ for a data example (I, J) and a repair language $\mathcal{L}^*$, denoted by cost$(\mathcal{M}, (I, J), \mathcal{L}^*)$, is the smallest size of a vfe $\mathcal{L}^*$-repair of $\mathcal{M}$ for (I, J). An $\mathcal{L}$-schema mapping $\mathcal{M}$ is $\mathcal{L}^*$-*optimal *for (I, J) if cost$(\mathcal{M}, (I, J), \mathcal{L}^*)$ is the minimum of the quantities cost$(\mathcal{M}', (I, J), \mathcal{L}^*)$, where $\mathcal{M}'$ ranges over all $\mathcal{L}$-schema mappings.*

Example 5. We continue from Example 3 by considering the schema mappings $\mathcal{M}_a = \{\text{Geo}(x, y) \to \text{City}(y)\}$, $\mathcal{M}_b = \{\text{Geo}(x, y) \to \exists z\ \text{City}(z)\}$, and $\mathcal{M}_c = \emptyset$.

Both $\mathcal{M}_1$ and $\mathcal{M}_5$ are vfe repairs of $\mathcal{M}_a$ for (I_1, J_1) and both have size 5; it is easy to verify that no other vfe repairs of $\mathcal{M}_a$ has size less than 5, hence $cost(\mathcal{M}_a, \{(I_1, J_1)\}, GLAV^{=,\neq}) = 5$. The repair $\mathcal{M}_3$ is the only vfe repair of $\mathcal{M}_c$, and $cost(\mathcal{M}_c, \{(I_1, J_1)\}, GLAV^{=,\neq}) = 6$. Moreover, $\mathcal{M}_2$ is a vfe repair of $\mathcal{M}_b$ and $size(\mathcal{M}_2) = 11$, but the cost of $\mathcal{M}_b$ is not 11. To see this, consider the schema mapping $\mathcal{M}'_2$ specified by the constraint

$$\mathcal{M}'_2 = \{\text{Geo}(x, y) \to \exists z\ \text{City}(z) \wedge (z = \text{San Jose}), \text{City}(\text{San Francisco})\}.$$

Clearly, $\mathcal{M}'_2$ is also a vfe repair of $\mathcal{M}_b$ for (I_1, J_1) and has size of 8.

The notion of an optimal schema mapping in Definition 4 differs from the corresponding definition in [12] in that we consider a slightly different notion of repair, as explained in the remarks following Definition 2. A consequence of this is that an optimal schema mapping in our sense has cost less than or equal than the cost of an optimal schema mapping in the sense of [12]. However, the complexity-theoretic results concerning the various algorithmic problems do not depend on this, and, indeed, the proofs are not affected by this change.

4 Hardness of Computing Optimal Schema Mappings

As we have seen, the main characteristic of the framework introduced in [12] and refined here is to cast schema-mapping discovery as an optimization problem: given a finite set of ground data examples, produce a schema mapping of minimum cost. Two different decision problems naturally arise in this setting.

Definition 5. *The decision problem* $\textsc{Cost}_{\mathcal{L}^*}$ *asks: given a ground data example* (I, J)*, a* $\mathcal{L}$*-schema mapping* $\mathcal{M}$*, and an integer* $k \geq 0$*, is* $cost(\mathcal{M}, (I, J), \mathcal{L}^*) \leq k$*?*

Definition 6. *The decision problem* $\textsc{Existence-Cost}_{\mathcal{L}^*}$ *asks: given a ground data example* (I, J) *and an integer* $k \geq 0$*, does there exists a* $\mathcal{L}$*-schema mapping* $\mathcal{M}$ *such that* $cost(\mathcal{M}, (I, J), \mathcal{L}^*) \leq k$*?*

In these two problems, the source schema and the target schema are part of the input. One can also consider the variants of these problems, such as $\textsc{Existence-Cost}_{\mathcal{L}^*}(\mathbf{S}, \mathbf{T})$, obtained by fixing the source and target schemas.

The preceding problems were introduced and studied in [12] when the base language $\mathcal{L}$ is GLAV or GAV, and the repair language is $\mathcal{L}^{=,\neq}$. It was shown there that $\textsc{Cost}_{\text{GLAV}^{=,\neq}}$ belongs to Σ_3^p (the third level of the polynomial hierarchy) and is Π_2^p-hard, while $\textsc{Cost}_{\text{GAV}^{=,\neq}}$ belongs to Σ_2^p and is DP-hard. It was also shown that $\textsc{Existence-Cost}_{\text{GLAV}^{=,\neq}}$ belongs to Σ_3^p and that $\textsc{Existence-Cost}_{\text{GAV}^{=,\neq}}$ belongs to Σ_2^p; moreover, both these problems were shown to be NP-hard.[1]

We show that the problems $\textsc{Cost}$ and $\textsc{Existence-Cost}$ are NP-complete for the languages of LAV schema mappings and SH-LAV schema mappings. Neither of these schema-mapping languages was considered in [12].

[1] The NP-hardness proof given in [12] is flawed. The authors have shared with us, in private communication, a correct NP-hardness proof.

Theorem 1. *Assume that* $\mathcal{L}^* \in \{\text{LAV}^{=,\neq}, \text{LAV}^{=}, \text{SH-LAV}^{=,\neq}, \text{SH-LAV}^{=}\}$. *Then the problems* $\textsc{Cost}_{\mathcal{L}^*}$ *and* $\textsc{Existence-Cost}_{\mathcal{L}^*}$ *are NP-complete. In fact, there are fixed schemas* $\boldsymbol{S}$ *and* $\boldsymbol{T}$ *for which these problems are NP-complete.*

Theorem 1 is established via a reduction from the Set-Cover problem. Next, we show that the problem of computing optimal GLAV schema mappings is not solvable in polynomial time, unless RP=NP. Note that this does not follow directly from the Π_2^p-hardness of $\textsc{Existence-Cost}_{GLAV^{=,\neq}}$ established in [12], because $\textsc{Cost}_{GLAV^{=,\neq}}$ is DP-hard. The same holds true for GAV mappings. Actually, we show a stronger result: unless RP = NP, there is no polynomial-time algorithm that, given a ground data example (I, J), computes a GLAV schema mapping whose cost is bounded by a polynomial in the cost of the optimal GLAV schema mapping; moreover, the same holds true for GAV schema mappings.

Theorem 2. *Assume that* $\mathcal{L} \in \{GLAV, GAV\}$ *and let* $p(x)$ *be a polynomial. Unless* RP = NP*, there is no polynomial-time algorithm that, given a ground data example* (I, J)*, computes a* $\mathcal{L}$*-schema mapping* $\mathcal{M}$ *such that* $cost(\mathcal{M}, (I, J), \mathcal{L}^{=,\neq}) \leq p(cost(\mathcal{M}_{opt}, (I, J), \mathcal{L}^{=,\neq}))$*, where* $\mathcal{M}_{opt}$ *is an* $\mathcal{L}^{=,\neq}$*-optimal schema mapping for* (I, J)*. Moreover, the same holds true when* $\mathcal{L}^{=,\neq}$ *is replaced by* $\mathcal{L}^{=}$*. In fact, there are fixed source and target schemas for which these results hold.*

Theorem 2 also shows that the computational hardness is, to some extent, robust under changes to the precise cost function. The proof is based on procedure that transforms a ground data example (I, J) into another ground data example (I', J'), which, intuitively, contains many isomorphic copies of the original data example (I, J), such that every near-optimal GLAV (GAV) schema mapping for (I', J') corresponds to a fitting GLAV (GAV) schema mapping for (I, J) of near-minimal size. This is combined with a hardness result for computing fitting GAV schema mappings of near-minimal size, established in [8].

5 Approximation of Optimal Single-Head LAV Mappings

We now study the approximation properties of the following optimization problem:

Definition 7. *The* $\textsc{Optimal-Repair}_{\mathcal{L}^*}(\boldsymbol{S}, \boldsymbol{T})$ *problem asks: given a ground data example* (I, J) *with source schema* $\boldsymbol{S}$ *and target schema* $\boldsymbol{T}$*, compute a minimal-size valid and fully explaining* $\mathcal{L}^*$*-schema mapping for* (I, J) *and* $\mathcal{L}^*$*.*

This problem is equivalent to the problem that asks to compute an optimal $\mathcal{L}$-schema mapping $\mathcal{M}$ together with a minimal-size $\mathcal{L}^*$-repair of $\mathcal{M}$; in particular, it contains, as a special case, the problem of computing an optimal $\mathcal{L}$-schema mapping for a given ground data example. This is so because, from a minimal-size valid and fully explaining $\mathcal{L}^*$-schema mapping, we can immediately extract an optimal $\mathcal{L}$-schema mapping by, simply by dropping all equalities, inequalities, and ground facts. Therefore, it follows from Theorem 2 that (assuming RP $\neq$ NP) there is no polynomial-time algorithm that solves $\textsc{Optimal-Repair}_{\mathcal{L}^*}$, when $\mathcal{L}^* \in \{GLAV^{=}, GLAV^{=,\neq}, GAV^{=}, GAV^{=,\neq}\}$.

We establish a positive approximability result for SH-LAV$^{=}$ mappings.

Theorem 3. *Fix a pair of schemas $\boldsymbol{S}, \boldsymbol{T}$. There is a polynomial-time $\mathcal{H}(n)$-approximation algorithm for* OPTIMAL-REPAIR$_{\textit{SH-LAV}^{=}}(\boldsymbol{S}, \boldsymbol{T})$*, where $\mathcal{H}(n) = \sum_{i=1}^{n} \frac{1}{i}$ is the n-th harmonic number.*

Our approximation algorithm is obtained by establishing a close connection between SET-COVER and OPTIMAL-REPAIR$_{\mathcal{L}^*}$. It is known that, although the SET-COVER problem is not constant-approximable, it is $\mathcal{H}(n)$-approximable, where n is the size of the universe. Moreover, this approximation ratio is known to be asymptotically optimal: unless P = NP, SET-COVER can only be approximated up to a factor of $c \ln(n)$, where c is a constant (specified in [4,14]) and n is the size of universe (recall that $\mathcal{H}(n) = O(\ln n)$). Our approximation algorithm is largely based on the approximation algorithm of WEIGHTED SET-COVER [9]. Here, we can think of each constraint as describing a set of facts, namely, the set of target facts that it explains, and the size of the constraint is the weight of the corresponding set.

The above approximation algorithm can be extended in a straightforward manner to the case of SH-LAV$^{=,\neq}$-constraints with a bounded number of inequalities per variable, as well as to LAV$^{=,\neq}$-constraints with a bounded number of inequalities per variable and a bounded number of atoms in the right-hand side. We leave it as an open problem whether an analog of Theorem 3 holds for the general case of SH-LAV$^{=,\neq}$ and LAV$^{=,\neq}$.

We conclude with a matching lower bound for the approximability of OPTIMAL-REPAIR$_{\text{SH-LAV}^{=}}(\mathbf{S}, \mathbf{T})$. This result is established via an approximation-preserving reduction (more precisely, an L-reduction [20]) from MINIMUM SET COVER.

Theorem 4. *Let $\mathcal{L} \in \{LAV, SH\text{-}LAV\}$. There are fixed schemas $\boldsymbol{S}$ and $\boldsymbol{T}$, and a constant c such that there is no polynomial-time $c \ln(n)$-approximation algorithm for* OPTIMAL-REPAIR$_{\mathcal{L}^{=}}(\boldsymbol{S}, \boldsymbol{T})$*, where n is the total number of target facts in the input data example. The same holds true for* OPTIMAL-REPAIR$_{\mathcal{L}^{=,\neq}}(\boldsymbol{S}, \boldsymbol{T})$.

6 Concluding Remarks

The cost function in Definition 5 naturally extends to sets of data examples, where $cost(\mathcal{M}, E, \mathcal{L}^*)$ is defined as the smallest size of an $\mathcal{L}^*$-repair of $\mathcal{M}$ that is vfe for each $(I, J) \in E$ (provided that such a repair exists). Similarly, the decision problems and optimization problems we studied naturally extend to the case where the input is a finite set of data examples. All upper bounds presented in this paper hold true also in the more general setting for multiple data examples.

Two important questions that remain to be answered are the existences of a polynomial time $\mathcal{H}(n)$-approximation algorithms for computing near-optimal LAV$^{=,\neq}$ and SH-LAV$^{=,\neq}$ schema mappings.

We have focused on obtaining a complexity-theoretic understanding of the algorithmic aspects of schema-mapping discovery from data examples. Our results pave the way for leveraging further the rich area of approximation algorithms and applying it to schema-mapping discovery. In parallel, we have embarked on a prototype implementation of our approximation algorithm enhanced with heuristic rules. While much remains to be done, there is preliminary evidence that this is an approach that may lead to a reasonably efficient system for schema-mapping discovery from data examples.

7 Acknowledgments

Research partially supported by NSF Grant IIS-1217869; Tan is partially supported by NSF grant IIS-1450560.

References

1. B. Alexe, B. T. Cate, P. G. Kolaitis, and W.-C. Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23:1–23:48, 2011.
2. B. Alexe, L. Chiticariu, R. J. Miller, and W. C. Tan. Muse: Mapping Understanding and deSign by Example. In *ICDE*, pages 10–19, 2008.
3. B. Alexe, B. ten Cate, P. G. Kolaitis, and W. C. Tan. Designing and refining schema mappings via data examples. In *SIGMOD*, pages 133–144, 2011.
4. N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, Apr. 2006.
5. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The dl-lite family and relations. *JAIR*, 36:1–69, 2009.
6. P. Barceló. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.
7. A. Bonifati, E. Chang, T. Ho, V. Lakshmanan, and R. Pottinger. HePToX: Marrying XML and Heterogeneity in Your P2P Databases. In *VLDB*, pages 1267–1270, 2005.
8. B. ten Cate, V. Dalmau, and P. G. Kolaitis. Learning schema mappings. *ACM Trans. Database Syst.*, 38(4):28, 2013.
9. V. Chvtal. A greedy heuristic for the set covering problem. *Math. Oper. Res.*, 4:233–235, 1979.
10. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *TCS*, 336(1):89–124, 2005.
11. G. H. Fletcher and C. M. Wyss. Towards a general framework for effective solutions to the data mapping problem. *Journal on Data Semantics*, XIV, 2009.
12. G. Gottlob and P. Senellart. Schema mapping discovery from data instances. *JACM*, 57(2), 2010.
13. L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio Grows Up: From Research Prototype to Industrial Tool. In *ACM SIGMOD*, pages 805–810, 2005.
14. D. S. Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of STOC*, pages 38–49, New York, NY, USA, 1973. ACM.
15. P. G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *ACM PODS*, pages 61–75, 2005.
16. M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM PODS*, pages 233–246, 2002.
17. H. Mannila and K.-J. Räihä. Automatic generation of test data for relational queries. *JCSS*, 38(2):240–258, 1989.
18. R. J. Miller, L. M. Haas, and M. A. Hernández. Schema Mapping as Query Discovery. In *VLDB*, pages 77–88, 2000.

19. C. Olston, S. Chopra, and U. Srivastava. Generating example data for dataflow programs. In *ACM SIGMOD*, pages 245–256, 2009.
20. C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of STOC*, pages 229–234, New York, NY, USA, 1988. ACM.
21. A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and J. Widom. Synthesizing view definitions from data. In *ICDT*, pages 89–103, 2010.
22. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-Driven Understanding and Refinement of Schema Mappings. In *ACM SIGMOD*, pages 485–496, 2001.