Approximation Algorithms for Schema-Mapping Discovery from Data Examples

BALDER TEN CATE, UC Santa Cruz*
PHOKION G. KOLAITIS, UC Santa Cruz and IBM Research - Almaden
KUN QIAN, UC Santa Cruz
WANG-CHIEW TAN, Recruit Institute of Technology and UC Santa Cruz

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 set of data examples (where n is the combined size of the given data examples), for a restricted class of schema mappings; moreover, we show that this approximation ratio cannot be improved. In addition to the complexity-theoretic results, we implemented the aforementioned $\log(n)$-approximation algorithm and carried out an experimental evaluation in a real-world mapping scenario.

CCS Concepts: •**Theory of computation** → *Approximation algorithms analysis; Data exchange;*

Additional Key Words and Phrases: Approximation Algorithms, Schema Mappings, Data Examples

ACM Reference format:
Balder ten Cate, Phokion G. Kolaitis, Kun Qian, and Wang-Chiew Tan. 9999. Approximation Algorithms for Schema-Mapping Discovery from Data Examples. *ACM Trans. Datab. Syst.* 0, 0, Article 0 (9999), 43 pages.
DOI: 0000001.0000001

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 [8, 24, 28]. 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

*Now working at Google, Inc.

This work is partially supported by NSF grant IIS-1217869; Qian is partially supported by NSF grants IIS-1217869 and IIS-1450560; Tan is partially supported by NSF grants IIS-1450560 and IIS-1524382 at UCSC. Author's addresses: B. ten Cate, Google, Inc.; P. Kolaitis, Computer Science Department, UC Santa Cruz; K. Qian, Computer Science Department, UC Santa Cruz; W.C. Tan, Computer Science Department, UC Santa Cruz.

 0362-5915/9999/00-ART0 $15.00
DOI: 0000001.0000001

between the elements of two schemas, as specified by a user through a graphical interface [9, 21, 30]. 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 [15, 18, 29, 31]). 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 [1, 34], the goal is, given a schema mapping, to *produce* meaningful data examples, that illustrate the schema mapping at hand. In [3], 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 [4, 11].

Gottlob and Senellart [19] 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) satisfies 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 $\text{GLAV}^{=,\neq}$ 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 size of a valid and fully explaining repair of $\mathcal{M}$, where a *repair* of $\mathcal{M}$ is a schema mapping $\mathcal{M}'$ in the extended language that can be obtained from $\mathcal{M}$ via a sequence of specified repair operations (see Definitions 3.2, 3.7, and 3.8). 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 [19] 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 [19], 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, extending, and investigating the Gottlob-Senellart framework in more depth. We first extend the framework by allowing any finite number of ground data examples, instead of a single ground data example. While, given a finite set E of data examples, there may not be a valid and fully explaining schema mapping for E in the extended language, we give a simple necessary and sufficient condition for the existence of such a schema mapping, a condition that is also easy to verify algorithmically. We also 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 [19]), 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 [19] (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, for both GLAV$^{=}$-repairs 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.

On the positive side, we design a $\log(n)$-approximation algorithm for computing near-optimal schema mappings in the more restricted case of SH-LAV schema mappings. Specifically, we present a polynomial-time $O(\log n)$-approximation algorithm that, given a set E of data examples, produces a SH-LAV mapping $\mathcal{M}$ whose cost is within a logarithmic factor of the cost of the optimal SH-LAV mapping for E, together with a witnessing SH-LAV$^{=}$-repair $\mathcal{M}'$ of $\mathcal{M}$ for E. Furthermore, we show that this is a best possible approximation result. SH-LAV schema mappings and the corresponding repair languages form important classes of schema mappings. For instance, many of the primitives of two well-known schema mapping benchmarks can be expressed in these languages (see Section 4.3 for a more detailed discussion).

In addition to the complexity-theoretic results, we implemented the aforementioned $\log(n)$-approximation algorithm and then tested the implementation on a real-world mapping scenario, where the task is to find a schema mapping from a relational database to an ontology.

A preliminary version of the present paper has appeared in [13], without proofs of the main results. Here, we provide detailed proofs. In addition, we extend the work in [13] by considering a more general schema mapping discovery setting, where the input is a set of ground data examples. We also report on an experimental evaluation of the log(n)-approximation algorithm based on a real-world schema mapping scenario.

The rest of this paper is organized as follows: In Section 2, we introduce the basic concepts needed. In Section 3, we review the repair framework and cost model introduced by Gottlob and Senellart, and also extend it to a finite set of data examples. In Section 4, we study the complexity of computing optimal and near-optimal schema mappings. In particular, we first present several negative results on approximating GAV schema mappings (Section 4.1) and also show how we can extend the results to the GLAV case (Section 4.2), and then we present the complexity results regarding LAV and SH-LAV schema mappings (Section 4.3). In Section 5, we present a positive result, that is, a $\log(n)$-approximation algorithm for SH-LAV schema mappings with respect to the repair language SH-LAV$^{=}$; moreover, we show that the approximation factor (i.e., logarithmic factor) cannot be improved (assuming P$\neq$NP). In Section 6, we present an experimental evaluation of that approximation algorithm. Finally, in Section 7, we summarize our results and discuss open questions and directions for future work.

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 relaton 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. A *homomorphism* $h : I_1 \to I_2$

is a function from $\text{adom}(I_1)$ to $\text{adom}(I_2)$ such that for every fact $P(a_1, \ldots, a_m)$ of I_1, we have that $P(h(a_1), \ldots, h(a_m))$ is a fact of I_2.

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 $\forall \mathbf{x}(S(\mathbf{x}) \rightarrow \exists \mathbf{y} T(\mathbf{x},\mathbf{y}))$.

Example 2.1. $\text{Geo}(x, y) \rightarrow \text{City}(y)$ 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. The size $||E||$ of a set of data examples is the sum of the sizes of the data examples in E.

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) satisfies all constraints in Σ. In this case, we will also say that J is a *solution* for I with respect to $\mathcal{M}$. 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. In the scenario of multiple data examples, we say that $\mathcal{M}$ is valid for a set E of data examples if for every $(I_k, J_k) \in E$, $(I_k, J_k) \models \Sigma$. We say that $\mathcal{M}$ is fully explaining for E if $\mathcal{M}$ is fully explaining for every data example (I_k, J_k) in E. We will use the expression *vfe* as a shorthand for "valid and fully explaining".

In [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. $\mathcal{M}$. Formally, J is a universal solution for I with respect to $\mathcal{M}$ if J is a solution for I w.r.t. $\mathcal{M}$ and, moreover, for every solution J' for I w.r.t. $\mathcal{M}$, there is a homomorphism from J to J' that maps all constants to themselves (see [16] for more details). It is not hard to verify that if (I, J) is a ground data example, then a schema mapping $\mathcal{M}$ fits (I, J) if and only if $\mathcal{M}$ is vfe for (I, J). Same claim holds for the case where we have multiple ground data examples.

Example 2.2. 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 : {Geo(CA, San Francisco), Geo(CA,San Jose), Geo(US,Boston), Geo(US, Los Angeles)}
J_1 : {City(San Francisco), City(San Jose)}

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

I_3 : {Geo(CA, San Francisco), Geo(US,New York)}
J_3 : {City(San Francisco), 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 not fully explaining, as 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 can be shown that there is no valid and fully explaining GLAV schema mapping for (I_1, J_1). Similarly, 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 [19], 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 become 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 3.1. Fix a schema-mapping language $\mathcal{L}$ (e.g., GLAV). An $\mathcal{L}^{=,\neq}$*-constraint* is a formula θ of the form $\forall\mathbf{x}(\varphi(\mathbf{x}) \wedge \alpha(\mathbf{x}) \rightarrow \exists\mathbf{y}(\psi(\mathbf{x},\mathbf{y}) \wedge \beta(\mathbf{x},\mathbf{y})))$, where

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

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

In what follows, we will continue using $\mathcal{L}$ 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 $\forall\mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists\mathbf{y}\psi(\mathbf{x},\mathbf{y}))$.

The definition of $\mathcal{L}^{=,\neq}$ above was given in [19], and extensive justification for this repair language was given in that work. The significance of $\mathcal{L}^{=}$ (introduced here) will become clear later on.

Definition 3.2. An $\mathcal{L}^{*}$*-repair* of an $\mathcal{L}$-schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is an $\mathcal{L}^{*}$-schema mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ such that each $\psi \in \Sigma'$ is either a ground target 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 $\text{repair}_{\mathcal{L}^{*}}(\mathcal{M})$ to denote the set of all $\mathcal{L}^{*}$-repairs of $\mathcal{M}$.

The above definition differs from that of repairs in [19] in that we allow $\mathcal{M}'$ to contain multiple repairs of the same $\mathcal{L}$-constraint from $\mathcal{M}$, whereas, in [19], the $\mathcal{L}^{*}$-constraints of $\mathcal{M}'$

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)	

stand in a one-to-one correspondence with the $\mathcal{L}$-constraints of $\mathcal{M}$. There are cases in which an $\mathcal{L}$-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 [19] may contain multiple copies of the same GLAV constraint or multiple GLAV constraints that are identical up to a renaming of variables (see Example 3.4). Our definition addresses this shortcoming. It does not impact our complexity results.

Example 3.3. Recall that the data example (I_1, J_1) in Example 2.2 had no vfe GLAV schema mapping. Let $\mathbf{S} = \{\text{Geo(are, city)}\}$ and $\mathbf{T} = \{\text{City(cityName)}\}$. Consider the GLAV$^{=,\neq}$ schema mappings $\mathcal{M}_k = (\mathbf{S}, \mathbf{T}, \Sigma_k)$, where $k = 1, \ldots, 5$, specified by the following sets of GLAV$^{=,\neq}$ constraints:

- Σ_1=$\{\text{Geo}(x, y) \wedge (x = \text{CA}) \rightarrow \text{City}(y)\}$
- Σ_2=$\{\text{Geo}(x, y) \rightarrow \exists z\ \text{City}(z) \wedge (y = \text{San Jose} \rightarrow z = \text{San Jose})$ $\wedge\ (y = \text{San Francisco} \rightarrow z = \text{San Francisco})\}$
- Σ_3=$\{\text{City(San Francisco)}, \text{City(San Jose)}\}$
- Σ_4=$\{\text{Geo}(x, y) \wedge (x \neq \text{US}) \rightarrow \text{City}(y)\}$
- Σ_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 are obtained in different ways: $\mathcal{M}_1$ includes 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; the mapping $\mathcal{M}_2$ includes 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, $\mathcal{M}_4$ includes a repair of the same constraint as $\mathcal{M}_1$ that, instead, uses an inequality.

$\mathcal{M}_5$ is valid, but not fully explaining for (I_1, J_1). It uses the equality $(x = \textit{moon})$, 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 3.4. The following example illustrates why, in Definition 3.2, we allow for multiple repairs of the same constraint. Consider the data example (I, J) in Table 1, and consider the GLAV schema mappings $\mathcal{M}_k = (\{\text{Geo(area,city)}\}, \{\text{City(cityName)}\}, \Sigma_k)$, where $k = 1, 2$, specified by the following sets of GLAV$^{=,\neq}$-constraints

- $\Sigma_1 = \{\text{Geo}(x, y) \rightarrow \text{City}(y)\}$,
- $\Sigma_2 = \{\text{Geo}(x_1, y_1) \rightarrow \text{City}(y_1), \text{Geo}(x_2, y_2) \rightarrow \text{City}(y_2)\}$.

and consider the GLAV$^{=,\neq}$-schema mapping $\mathcal{M}_r = (\{\text{Geo}\}, \{\text{City}\}, \Sigma_r)$, where Σ_r contains the following constraints:

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

Note that $\mathcal{M}_r$ is vfe for (I, J). Note also that Σ_2 consists of two renamings of the GLAV constraint of $\mathcal{M}_1$. 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 [19], $\mathcal{M}_r$ does not qualify as a repair of $\mathcal{M}_1$, and, indeed, an example of a minimal-size repair 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 (see Definitions 3.7 and 3.8 below), we have that, under the cost model of [19], $\mathcal{M}_2$ has a lower cost than $\mathcal{M}_1$, which is counterintuitive. The definition of cost we use here solves this problem. □

As pointed out in [19], 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 here, every ground data example has a vfe $\mathcal{L}^{=}$ repair. Indeed, for every data example (I, J) it holds that every $\mathcal{L}$-schema mapping has a $\mathcal{L}^{=}$-repair that is vfe (which can be obtained, for instance, by cancelling all constraints and adding all ground target facts). However, the same result does not hold for a set of ground data examples (note that the trivial schema mapping consisting of all the ground target facts occurring in the input set of data examples is not always vfe). Theorem 3.5, below, establishes a necessary and sufficient condition for the existence of a vfe GLAV$^{=,\neq}$ (resp. GLAV$^{=}$) schema mapping for a set E of data examples.

THEOREM 3.5. *Let E be a finite set of ground data examples. There exists a vfe GLAV$^{=,\neq}$ schema mapping for E if and only if the following condition holds: for every pair of data examples $(I, J), (I', J') \in E$, if $I \subseteq I'$, then $J \subseteq J'$. The same holds true when GLAV$^{=,\neq}$ is replaced by GLAV$^{=}$. Moreover, this condition is checkable in polynomial time.*

Note that, in particular, Theorem 3.5 implies that GLAV$^{=}$ and GLAV$^{=,\neq}$ are, in a sense, equally powerful repair languages: a finite set of ground data examples has a vfe GLAV$^{=}$ schema mapping if and only if it has a vfe GLAV$^{=,\neq}$ schema mapping. Before we give the proof of Theorem 3.5, we need a definition:

Definition 3.6. The complete-description constraint of a ground data example (I, J) is the GLAV$^{=}$ constraint t that is of the form $\forall \mathbf{x}(q_I(\mathbf{x}) \wedge \alpha(\mathbf{x}) \rightarrow \exists \mathbf{y} q_J(\mathbf{y}) \wedge \beta(\mathbf{y}))$, where $q_I(\mathbf{x})$ is the conjunction of all facts of I (with each value from the active domain of I replaced by a universally quantified variable from $\mathbf{x}$) and $q_J(\mathbf{y})$ is the conjunction of all facts of J (with each value replaced by an existentially quantified variable from $\mathbf{y}$); $\alpha(\mathbf{x})$ is the conjunction of all equalities $(x_i = c_i)$, where x_i is the variable corresponding to c_i, and likewise for $\beta(\mathbf{y})$. For instance, if $I = \{S(a), S(b)\}$ and $J = \{T(a, b)\}$, then the complete-description constraint of (I, J) is $\forall xy(S(x) \wedge S(y) \wedge (x = a) \wedge (y = b) \rightarrow \exists wu(T(w, u) \wedge (w = a) \wedge (u = b))$.

PROOF OF THEOREM 3.5. ($\Leftarrow$) Suppose the condition holds, and consider the set Σ consisting of the complete-description constraint of each data example $(I_k, J_k) \in E$. We show that the schema mapping $\mathcal{M}$ specified by Σ is vfe for E.

Validity. We need to show that for each data example $(I_k, J_k) \in E$ and for each $\sigma \in \Sigma$, we have $(I_k, J_k) \models \sigma$. By construction, σ is the complete-description constraint of some data example $(I_\sigma, J_\sigma) \in E$. If $I_k \not\subseteq I_\sigma$, then σ is trivially valid in (I_k, J_k), because facts of I_k

would never fire the left-hand side of σ. If, on the other hand, $I_\sigma \subseteq I_k$, then, by assumption $J_\sigma \subseteq J_k$, and hence σ is valid in (I_k, J_k)., We conclude that $\mathcal{M}$ is valid for E.

Explanation. We need to show that $\mathcal{M}$ fully explains every data example of E. Consider any $(I, J) \in E$, and let σ be the complete-description constraint of of (I, J). In order to show that σ fully explains (I, J), we need to show that for every instance K such that $(I, K) \models \sigma$, we have $J \subseteq K$. Since $(I, K) \models \sigma$,we have that the homomorphism from the left-hand side of σ to I can be extended to a homomorphism from the right-hand side of σ to K. By construction of σ, it follows that $J \subseteq K$, and thus σ fully explains (I, J). Since the same argument applies to each data example in E, we have that the schema mapping specified by Σ fully explains E.

($\Rightarrow$). Suppose that there is a vfe schema mapping $\mathcal{M}$ for E, and consider any pair of data examples (I, J), (I', J') in E, such that $I \subseteq I'$. We have that, in this case, J' must be also a solution for I w.r.t. $\mathcal{M}$ (because every homomorphism from the left-hand side of a constraint to I is also a homomorphism from the left-hand side of the same constraint to I'). Since $\mathcal{M}$ is fully explaining for E, we conclude that $J \subseteq J'$.

Since no inequality is used, the same holds for when $\text{GLAV}^{=,\neq}$ is replaced by $\text{GLAV}^{=}$.

The condition is checkable in polynomial time, because if there are n data examples in E where each data example has at most n facts, the checking function needs to compare n^2 pairs of examples. In each comparison, we need to examine at most $4n$ facts, thus this can be done in polynomial time. □

The cost model introduced in [19] 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 3.7. [19] 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 $R(a_1, \ldots, a_n)$, for present purposes, is viewed as shorthand for $\exists x_1, \ldots, x_n R(x_1, \ldots, x_n) \wedge x_1 = a_1 \wedge \cdots \wedge x_n = a_n$; 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 3.8. The *cost* of an $\mathcal{L}$-schema mapping $\mathcal{M}$ for a set E of data examples and a repair language $\mathcal{L}^*$, denoted by cost($\mathcal{M}, E, \mathcal{L}^*$), is the smallest size of a vfe $\mathcal{L}^*$-repair of $\mathcal{M}$ for E. An $\mathcal{L}$-schema mapping $\mathcal{M}$ is *$\mathcal{L}^*$-optimal for E* if cost($\mathcal{M}, E, \mathcal{L}^*$) is the minimum of the quantities cost($\mathcal{M}', E, \mathcal{L}^*$), where $\mathcal{M}'$ ranges over all $\mathcal{L}$-schema mappings.

Example 3.9. We continue from Example 3.3 by considering the following schema mappings:

$\mathcal{M}_a = (\{\text{Geo}\}, \{\text{City}\}, \{\text{Geo}(x, y) \rightarrow \text{City}(y)\})$;

$\mathcal{M}_b = (\{\text{Geo}\}, \{\text{City}\}, \{\text{Geo}(x, y) \rightarrow \exists z\ \text{City}(z)\})$;

$\mathcal{M}_c = (\{\text{Geo}\}, \{\text{City}\}, \emptyset)$.

Both $\mathcal{M}_1$ and $\mathcal{M}_4$ are vfe repairs of $\mathcal{M}_a$ for (I_1, J_1) and both have size 5; in fact no other vfe repairs of $\mathcal{M}_a$ has size less than 5, hence cost($\mathcal{M}_a, \{(I_1, J_1)\}, \text{GLAV}^{=,\neq}$) = 5. The repair $\mathcal{M}_3$ is the only vfe repair of $\mathcal{M}_c$, and cost($\mathcal{M}_c, \{(I_1, J_1)\}, \text{GLAV}^{=,\neq}$) = 6 (recall that

the size of a ground fact is three times its arity). 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

$$\text{Geo}(x, y) \rightarrow \exists z\ \text{City}(z) \wedge (z = \text{San Jose}), \text{City}(\text{San Francisco}).$$

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

Recall that we consider a slightly different notion of repair than the one in [19], as explained in the remarks following Definition 3.2. As a consequence, the cost of a schema mapping, under our cost model, may be less than the cost under the cost model in [19]. Nevertheless, every optimal schema mapping in our cost model has the same cost as the cost of an optimal schema mapping in the cost model of [19]. 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 AND NEAR-OPTIMAL SCHEMA MAPPINGS

As we have seen, the main idea of the framework introduced in [19] 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 4.1. The decision problem $\text{Cost}_{\mathcal{L}^*}$ asks: given a set E of ground data examples, a $\mathcal{L}$-schema mapping $\mathcal{M}$, and an integer $k \geq 0$, is $\text{cost}(\mathcal{M}, E, \mathcal{L}^*) \leq k$?

Definition 4.2. The decision problem Existence-$\text{Cost}_{\mathcal{L}^*}$ asks: given a set E of ground data examples and an integer $k \geq 0$, does there exist a $\mathcal{L}$-schema mapping $\mathcal{M}$ such that $\text{cost}(\mathcal{M}, (I, J), \mathcal{L}^*) \leq k\text{cost}(\mathcal{M}, E, \mathcal{L}^*) \leq k$?

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 Existence-$\text{Cost}_{\mathcal{L}^*}(\mathbf{S}, \mathbf{T})$, obtained by fixing the source and target schemas.

The above problems (where E consists of a single ground data example) were introduced and studied in [19] for the case where the base language $\mathcal{L}$ is GLAV or GAV, and the repair language is $\mathcal{L}^{=,\neq}$. It was shown there that $\text{Cost}_{\text{GLAV}^{=,\neq}}$ belongs to Σ_3^p (the third level of the polynomial hierarchy) and is Π_2^p-hard, while $\text{Cost}_{\text{GAV}^{=,\neq}}$ belongs to Σ_2^p and is DP-hard. It was also shown that Existence-$\text{Cost}_{\text{GLAV}^{=,\neq}}$ belongs to Σ_3^p and that Existence-$\text{Cost}_{\text{GAV}^{=,\neq}}$ belongs to Σ_2^p; moreover, both these problems were shown to be NP-hard[1] These results are also summarized in Table 2.

Table 2. Relevant complexity results in [19]

Decision problem	complexity results
$\text{Cost}_{\text{GLAV}^{=,\neq}}$	in Σ_3^p, Π_2^p-hard
$\text{Cost}_{\text{GAV}^{=,\neq}}$	in Σ_2^p, DP-hard
Existence-$\text{Cost}_{\text{GLAV}^{=,\neq}}$	in Σ_3^p, NP-hard
Existence-$\text{Cost}_{\text{GAV}^{=,\neq}}$	in Σ_2^p, NP-hard

[1]The NP-hardness proof given in [19] is flawed. The authors have shared with us, in private communication, a correct NP-hardness proof.

The Cost and Existence-Cost problems for LAV schema mappings and its sub-language SH-LAV schema mappings were not considered in [19]. In addition, exploring the approximation properties of different schema mappings was left as a future work in [19]. In the rest of this paper, we study the aforementioned open questions in more depth. The main results, in this section, are:

(1) We show that (assuming RP $\neq$ NP) there is no polynomial time algorithm that, given a ground data example, computes an optimal GLAV schema mapping, or even a GLAV schema mapping whose cost is bounded by a fixed polynomial in the cost of the optimal GLAV schema mapping. The same result holds for GAV schema mappings, regardless of the repair language considered (see Theorem 4.3).
(2) We show that the Cost and Existence-Cost problems are NP-complete for LAV schema mappings and SH-LAV schema mappings, regardless of the repair language considered (see Theorem 4.12).

4.1 Hardness of Computing Optimal and Near-Optimal GAV Schema Mappings

We first 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 NP-hardness of Existence-Cost$_{\text{GLAV}^{=,\neq}}$ established in [19], because Cost$_{\text{GLAV}^{=,\neq}}$ is Π_2^p-hard.

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 4.3. *Let $\mathcal{L} \in \{GLAV, GAV\}$, let $\mathcal{L}^* \in \{\mathcal{L}^{=}, \mathcal{L}^{=,\neq}\}$, and let $p(x)$ be any 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 $cost(\mathcal{M}, (I, J), \mathcal{L}^*) \leq p(cost(\mathcal{M}_{opt}, (I, J), \mathcal{L}^*))$, where $\mathcal{M}_{opt}$ is an $\mathcal{L}^*$-optimal schema mapping for (I, J). In fact, there are fixed source and target schemas for which this result holds.*

Theorem 4.3 also shows that the computational hardness is, to some extent, robust under changes to the precise cost function.

The proof of Theorem 4.3 will make use of a result from [11] (the preliminary version appeared in [10]); we spell that result first and then give a roadmap for the proof of the desired Theorem 4.3

We say that a class of data examples is *GLAV-tractable* if there is a polynomial time algorithm that, given as input a data example from the class and a GLAV-constraint, checks if the data example satisfies the GLAV-constraint.

Theorem 4.4. *(from [11]) There exist schemas* $\mathbf{S}$ *and* $\mathbf{T}$ *and a GLAV-tractable class K of data examples over $\boldsymbol{S}$ and $\boldsymbol{T}$, where $\boldsymbol{T}$ consists of unary relations, such that the following holds: let $p(x)$ be any polynomial function. Unless* RP = NP*, there is no polynomial-time algorithm that, given a ground data example $(I, J) \in K$ for which there exists vfe GAV-schema mappings, computes a vfe GAV-schema mapping $\mathcal{M}$ such that $size(\mathcal{M}) \leq p(size(\mathcal{M}_{min}))$, where $\mathcal{M}_{min}$ is the minimal-size vfe GAV-schema mapping for (I, J).*

Proof. The result was proved in [11] without reference to GLAV-tractability. Careful inspection of the proof in [11] shows that the class of source instances used is CQ-tractable (defined below), and that the target schema is unary. As we will show, this implies the

data examples are GLAV-tractable. We say that a class C of instances (for a schema $\mathbf{S}$) is *CQ-tractable* if there is a polynomial-time algorithm that given an instance $I \in C$, an n-ary conjunctive query q ($n \geq 0$), and a tuple $\mathbf{a} \in adom(I)^n$, decides if $\mathbf{a} \in q(I)$. Note that CQ-tractable classes of instances are rare. Indeed, the classic reduction from 3SAT to conjunctive query evaluation shows that there is singleton class C consisting of a two-element instance, such that C is CQ-intractable unless P = NP. A non-trivial example of a CQ-tractable class of instances is the class of disjoint unions of oriented paths (viewed as instances over a schema consisting of a single binary relation) [20].

Let K be a class of data examples over source and target schema $\mathbf{S}$ and $\mathbf{T}$, such that the source instances of the data examples in K belong to a CQ-tractable class, and $\mathbf{T}$ contains only unary relations. We will show that K is a GLAV-tractable class of data examples.

Let (I, J) be a data example from K and consider any GLAV-constraint t of the form $\forall \mathbf{xy}\ \varphi(\mathbf{x}, \mathbf{y}) \rightarrow\ \exists \mathbf{z}\ \psi(\mathbf{x}, \mathbf{z})$. We can split t into a finite set Σ of GLAV-constraints such that each GLAV-constraint in Σ has only one variable occurring in the right-hand side. Concretely, for each GLAV-constraint t with m variables occurring in the right-hand side, where $m > 1$, we split t into m GLAV-constraints in the following way: for each variable z occurring in the right-hand side of t, we create a new GLAV-constraint t' with identical left-hand side as t, and the right-hand side of t' contains only those atoms that use the variable z. After this splitting, each GLAV-constraint is of the form (1) $\forall \mathbf{x}y\ \varphi(\mathbf{x}, y) \rightarrow\ \psi(y)$, or (2) $\forall \mathbf{x}\ \varphi(\mathbf{x}) \rightarrow\ \exists y\ \psi(y)$, where ψ is a conjunction of relational atoms. Note that t and Σ are logically equivalent, and the combined size of the constraints in Σ is $O(n^2)$ where n is the size of t. It therefore suffices to show that we can check the validity of the constraints from Σ in (I, J) in polynomial time. This follows immediately from the fact that I belongs to a CQ-tractable class: we can evaluate the left-hand side $\exists \mathbf{x}\varphi$ in I in polynomial time, and check that the right-hand side is satisfied. We remark that the above argument works because the target schema has only unary relations. □

We now provide a roadmap of the proof of Theorem 4.3. The proof proceeds by contradiction: we show that if there is a polynomial-time algorithm A that solves the problem described in Theorem 4.3, then, using A as an oracle, we can also solve the problem described in Theorem 4.4 in polynomial time. More precisely, our construction is based on a procedure that transforms a ground data example (I, J) (for which there is a vfe GAV schema mapping) into another ground data example (I', J') that contains many isomorphic copies of the original data example (I, J). Intuitively, after creating "sufficiently" many isomorphic copies of (I, J) in (I', J'), every GAV schema mapping $\mathcal{M}$ that needs to be repaired so as to be vfe for (I, J) will have to contain a large number of repairs so as to be vfe for (I', J') (see Lemma 4.9). As a result, if the hypothetical Algorithm A returns a near-optimal GAV schema mapping $\mathcal{M}'$ for (I', J'), then a "small" vfe GAV schema mapping $\mathcal{M}_{\text{small}}$ for (I, J) can be extracted from $\mathcal{M}'$ in polynomial time (see Lemma 4.9 and Proposition 4.10). We show that $\mathcal{M}_{\text{small}}$ is a solution for the problem in Theorem 4.4, and the contradiction is established.

The rest of the section is dedicated to a detailed proof of Theorem 4.3. We will give a detailed proof for the GAV$^{=,\neq}$ case, which applies also to the GAV$^{=}$ case. In Section 4.2, we will explain how the proof can be modified to be a proof for the GLAV$^{=,\neq}$ and GLAV$^{=}$ case.

LEMMA 4.5 ([4]). *Let E be any finite set of data examples. If there is a vfe GAV schema mapping for E, then there is one of size at most $||E||^2$.*

We say that a GAV constraint is *connected* if the graph, where the nodes are the atoms (both on the left-hand side and on the right-hand side) in the constraint and two atoms are connected by an edge if they have a variable in common, is connected. Thus, for example, the GAV constraints $\forall xy(R(x,y) \rightarrow S(x,y))$ and $\forall xy P(x) \wedge Q(y) \rightarrow R(x,y)$ are connected, while the GAV constraint $\forall xyzv(R(x,y) \wedge U(z,v) \rightarrow S(x,y))$ is not connected. We say that a GAV schema mapping is connected if it consists of connected GAV constraints.

LEMMA 4.6. *Every GAV schema mapping $\mathcal{M}$ that is a minimal-size vfe GAV schema mapping for a data example (I,J) is connected.*

PROOF. Let $\mathcal{M}$ be a minimal vfe GAV schema mapping for (I,J), and suppose, for the sake of a contradiction, that $\mathcal{M}$ is not connected. If a GAV constraint is not connected, (the graph of) its atoms can be divided into two or more connected components. One of these connected components must contain the right-hand side of the constraint. Let $\widehat{\mathcal{M}}$ be obtained from $\mathcal{M}$ simply by removing, from left-hand side of each GAV constraint, all atoms that do not belong to the connected component containing the right-hand side. Note that $\widehat{M}$ is strictly smaller than $\mathcal{M}$. We will show that $\widehat{\mathcal{M}}$ is vfe for (I,J), which contradicts the assumption that $\mathcal{M}$ was a minimal vfe GAV schema mapping for (I,J).

Since $\mathcal{M}$ is assumed to be a minimal vfe schema mapping for (I,J), every constraint σ of $\mathcal{M}$ explains at least one target fact. This implies that there is at least one homomorphism h from the left-hand side of σ to I. Let $\widehat{\sigma}$ be the connected constraint of $\widehat{\mathcal{M}}$ that was obtained from σ. Clearly, every target fact that is explained by σ is also explained by $\widehat{\sigma}$. Conversely, every target fact explained by $\widehat{\sigma}$ is explained by σ, because every homomorphism g from the left-hand side of $\widehat{\sigma}$ to I can be extended to a homomorphism g' from the left-hand side of σ to I (where g' simply extends g by mapping every variable $x \in dom(h) \setminus dom(g)$ to $h(x)$). It follows that $\mathcal{M}$ and $\widehat{\mathcal{M}}$ explain the same target facts, and hence $\widehat{\mathcal{M}}$ is vfe for (I,J). □

LEMMA 4.7. *For each GAV schema mapping $\mathcal{M} = (\boldsymbol{S}, \boldsymbol{T}, \Sigma)$, there is a connected GAV schema mapping $\mathcal{M}'$, such that, for all data examples (I,J),*

$$cost(\mathcal{M}', \{(I,J)\}, GAV^{=,\neq}) \leq 2 \cdot cost(\mathcal{M}, \{(I,J)\}, GAV^{=,\neq})$$

and the same holds for when $GAV^{=,\neq}$ is replaced by $GAV^{=}$. Moreover, in both cases, $\mathcal{M}'$ can be computed from $\mathcal{M}$ in polynomial time.

PROOF. If a GAV constraint is not connected, (the graph of) its atoms can be divided into two or more connected components. One of these connected components must contain the right-hand side of the constraint. Let $\widehat{\mathcal{M}} = (\mathbf{S}, \mathbf{T}, \widehat{\Sigma})$ be obtained from $\mathcal{M}$ simply by removing, from left-hand side of each GAV constraint, all atoms that do not belong to the connected component containing the right-hand side. This can clearly be done in polynomial time. We claim that $\widehat{\mathcal{M}}$ satisfies the required properties.

Let (I,J) be an arbitrary data example and $\mathcal{M}_r$ be a minimal vfe repair of $\mathcal{M}$ with respect to (I,J). We have to show that there is a repair $\widehat{\mathcal{M}_r} = (\mathbf{S}, \mathbf{T}, \widehat{\Sigma_r})$ of $\widehat{\mathcal{M}}$ such that $\widehat{\mathcal{M}_r}$ is vfe with respect to (I,J), and such that the size of $\widehat{\mathcal{M}_r}$ is at most twice the size of $\mathcal{M}_r$. We take

$$\widehat{\Sigma_r} = \{\widehat{\psi} \mid \psi \in \Sigma_r\}$$

where, for each repaired GAV constraint $\psi \in \mathcal{M}_r$, $\widehat{\psi}$ is obtained as follows: let $\phi \in \Sigma$ be the GAV constraint of which ψ is a repair, and let $\widehat{\phi}$ be the corresponding GAV constraint of

$\widehat{\mathcal{M}}$ (that is, $\widehat{\phi}$ is obtained from ϕ by removing disconnected atoms from the left-hand side). We distinguish two cases:

First, we consider the case where ψ explains at least one fact of J. In particular, this means that there is a homomorphism h from the left-hand side of ψ into I. In this case, we simply let $\widehat{\psi}$ be the repair of $\widehat{\phi}$ obtained by applying the same repair operations as in ψ, insofar as they concern variables that still belong to $\widehat{\phi}$. Clearly, all atoms in the left-hand side of $\widehat{\psi}$ occur also in the left-hand side of ψ. Therefore, $\widehat{\psi}$ explains all facts of J explained by ψ. Moreover, $\widehat{\psi}$ is valid in (I, J) because every homomorphism g from the left-hand side of $\widehat{\psi}$ can be extended to a homomorphism g' from the left-hand side of ψ to I (where g' simply extends g by mapping every variable $x \in dom(h) \setminus dom(g)$ to $h(x)$).

Next, consider the case where ψ does not explain any fact of J. Since $\mathcal{M}_r$ is valid with respect to (I, J), this implies that there is no homomorphism from the left-hand side of ψ to I (because, if there were, then the right-hand side of ψ would be satisfied in J, and hence ψ would explain at least one fact of J). If ϕ is connected (and hence $\widehat{\phi} = \phi$), we simply take $\widehat{\psi} = \psi$. Otherwise, we construct $\widehat{\psi}$ from $\widehat{\phi}$ by adding a single equality with a constant outside the active domain of I. We then have that the size of $\widehat{\psi}$ is at most twice the size of ψ, and ψ is trivially valid in (I, J).

In each of the above cases, $\widehat{\psi}$ was constructed so that it is valid with respect to (I, J) and it explains all facts of J explained by ψ, while at the same time having a size at most twice the size of ψ. It follows that the cost of $\widehat{\mathcal{M}}$ with respect to (I, J) is at most twice as large as the cost of $\mathcal{M}$ with respect to (I, J). □

We will prove Theorem 4.3 by contradiction: if a poly-time algorithm A for the problem in Theorem 4.3 exists, then, using A as an oracle, we can solve the problem in Theorem 4.4 in polynomial-time as well.

Let p be a fixed polynomial. Take an arbitrary ground data example $(I, J) \in K$ for which there exists a vfe GAV schema mapping, where K is the GLAV-tractable class of data examples given by Theorem 4.4. Let $\mathbf{S}$ and $\mathbf{T}$ be the source schema and target schema involved. We transform (I, J) to another data example (I', J') over the same source and target schemas: let $E = \{ (I_1, J_1), \cdots, (I_n, J_n) \}$, where each (I_k, J_k) is an isomophic copy of (I, J) using fresh constants, and where $n = 3 \cdot p(||(I, J)||^2)$. We define (I', J') to be the data example where $I' = \bigcup_{k=1\ldots n} I_k$ and $J' = \bigcup_{k=1\ldots n} J_k$.

Clearly, the transformation from (I, J) to (I', J') described above is computable in polynomial time. In the remainder of this section, we show that, from a near-optimal GAV schema mapping for (I', J') (in the sense of Theorem 4.3) we can extract, in polynomial time, a near-minimal vfe GAV schema mapping for (I, J) (in the sense of Theorem 4.4).

LEMMA 4.8. *Let $\mathcal{M} = (\boldsymbol{S}, \boldsymbol{T}, \Sigma)$ be any connected GAV schema mapping (where $\boldsymbol{S}$ and $\boldsymbol{T}$ are the source and target schema of the data example (I, J)). If $\mathcal{M}$ is vfe for (I, J), then $\mathcal{M}$ is vfe for (I', J').*

PROOF. By invariance under isomorphism, $\mathcal{M}$ is vfe for (I_k, J_k), for all $k \leq n$. Since $\mathbf{T}$ consists of unary relations, it follows that, for every connected GAV constraint $\sigma \in \Sigma$, we in fact have that the left-hand side of σ is connected, and hence, for every homomorphism h from the left-hand side of σ into I', the range of h is, in fact, contained in I_k for some $k \leq n$, and therefore, since $(I_k, J_k) \models \Sigma$, h is a homomorphism from the right-hand side of σ to J_k, which is contained in J. It follows that $\mathcal{M}$ is valid for (I', J'). Moreover, since every fact in J' belongs to J_k for some k and $\mathcal{M}$ is vfe for (I_k, J_k), we have that $\mathcal{M}$ explaines every fact in J'. □

LEMMA 4.9. *Let $\mathcal{M} = (\boldsymbol{S}, \boldsymbol{T}, \Sigma)$ be any GAV schema mapping, where $\boldsymbol{S}$ and $\boldsymbol{T}$ are the source and target schema of the data example (I, J).*

(1) *If every vfe repair of $\mathcal{M}$ for (I, J) contains at least one constant in (I, J), then $cost(\mathcal{M}, E, GAV^{=,\neq}) \geq 2n$, where n is the number of data examples in E.*
(2) *If there is a vfe repair of $\mathcal{M}$ for (I, J) that does not contain any constant in (I, J), then there is a subset $\Sigma' \subseteq \Sigma$ such that $\mathcal{M}' = (\boldsymbol{S}, \boldsymbol{T}, \Sigma')$ is a vfe GAV schema mapping for (I, J).*

PROOF. (1) Since every vfe repair of $\mathcal{M}$ for (I, J) contains at least one constant in (I, J), every vfe repair of $\mathcal{M}$ for (I_k, J_k) also contains at least one constant in (I_k, J_k) , where $k = 1, \cdots, n$ due to the fact that (I_k, J_k) is a copy of (I, J). Let $\mathcal{M}_r$ be a minimal vfe repair of $\mathcal{M}$ for E, by definition of vfe for a set of ground data examples, $\mathcal{M}_r$ is also a valid and fully explaining GAV$^{=,\neq}$ schema mapping for each $(I_k, J_k) \in E$. Therefore, $\mathcal{M}_r$ contains at least one constant from each data example $(I_k, J_k) \in E$. This indicates that $\mathcal{M}_r$ contains at least n distinct constants. By the definition of repair language, every repaired schema mapping containing a constant has size at least 2, hence $\text{cost}(\mathcal{M}, E, GAV^{=,\neq}) \geq 2n$.

(2) Let $\mathcal{M}_r = (\mathbf{S}', \mathbf{T}', \Sigma'')$ be a valid and fully explaining repair of $\mathcal{M} = (\mathbf{S}', \mathbf{T}', \Sigma)$ for (I, J) and that does not contain any constant in (I, J). Consider the following repairs that can appear in $\mathcal{M}_r$.

(i) An equality $(x = c)$, where c is not in (I, J), in the left-hand side of a GAV-constraint t in $\mathcal{M}$. This equality is cancelling t, thus t can be removed from $\mathcal{M}_r$ without changing the validity and explanation of $\mathcal{M}_r$ for (I, J).
(ii) An inequality $(x \neq c)$, where c is not in (I, J), in the left-hand side of a GAV constraint t in $\mathcal{M}$. Clearly, it is a redundant repair and can be removed from t. This implies a contradiction to the fact that $\mathcal{M}_r$ is a minimal, hence cannot be the case.
(iii) A ground fact f such that $f \notin J$. This cannot be the case, because it makes $\mathcal{M}_r$ invalid in (I, J).

Since Case (ii) and Case (iii) cannot arise, the only possible repair is Case (i). If $\mathcal{M}_r$ contains such an equality, then we can remove the constraint that contains such kind of equality, because the repair is cancelling the constraint. Therefore, we can find a subset $\Sigma' \subseteq \Sigma''$ and that is a vfe for (I, J), and, clearly, Σ' is also a subset of Σ. □

PROPOSITION 4.10. *Let (I, J) be any data example from the GLAV-tractable class K given by Theorem 4.4, and let (I', J') be as constructed above (that is, by making isomorphic copies of (I, J)). Let $\mathcal{M}$ be any connected GAV schema mapping such that $cost(\mathcal{M}, (I', J'), GAV^{=,\neq}) \leq p(cost(\mathcal{M}^{opt}, (I', J'), GAV^{=,\neq}))$, where $\mathcal{M}^{opt}$ is the optimal GAV schema mapping for (I', J'). From $\mathcal{M}$ and (I, J), we can compute in polynomial time a GAV schema mapping $\mathcal{M}'$ such that $size(\mathcal{M}') \leq 2p(size(\mathcal{M}^{min}))$, where $\mathcal{M}^{min}$ is the smallest vfe GAV schema mapping for (I, J).*

PROOF. By Lemma 4.5 and Lemma 4.6, there is a vfe connected GAV schema mapping for (I, J) of size at most $||(I, J)||^2$. By Lemma 4.8, the same schema mapping is also vfe for (I', J'), and, therefore, $\text{cost}(\mathcal{M}^{opt}, (I', J'), GAV^{=,\neq}) \leq ||(I, J)||^2$. Let $\mathcal{M}_{conn}$ be the connected GAV schema mapping obtained from $\mathcal{M}$ through Lemma 4.7. Then

$$\text{cost}(\mathcal{M}_{conn}, (I', J'), GAV^{=,\neq}) \leq 2\text{cost}(\mathcal{M}, (I', J'), GAV^{=,\neq}) \leq 2p(||(I, J)||^2) \ .$$

Recall that $n = 3p(||(I, J)||^2)$. By Lemma 4.9, we obtain that $\mathcal{M}_{conn} = (\mathbf{S}, \mathbf{T}, \Sigma)$ "contains" a GAV mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ (with $\Sigma' \subseteq \Sigma$) that is vfe for (I, J). Moreover, $\mathcal{M}'$ is

computable in polynomial time due to the GLAV-tractability of K, and

$$size(\mathcal{M}') \leq size(\mathcal{M}_{conn}) \leq 2\text{cost}(\mathcal{M}, (I', J'), GAV^{=,\neq}) \leq 2p(\text{cost}(\mathcal{M}^{opt}, (I', J'), GAV^{=,\neq}))\,.$$

By Lemma 4.6, we may assume without loss of generality that $\mathcal{M}^{min}$ is connected. By Lemma 4.8, then, $\mathcal{M}^{min}$ is vfe for (I', J') and we have that $\text{cost}(\mathcal{M}^{opt}, (I', J'), GAV^{=,\neq}) \leq size(\mathcal{M}^{min})$.

Putting everything together, we have that

$$size(\mathcal{M}') \leq 2p(\text{cost}(\mathcal{M}^{opt}, (I', J'), GAV^{=,\neq})) \leq 2p(size(\mathcal{M}^{min})).$$

□

Proposition 4.10 together with Theorem 4.4 immediately implies Theorem 4.3.

4.2 Extending the proof of Theorem 4.3 for GLAV schema mappings

Here, we extend the proof of Theorem 4.3 to hold both for $\text{GLAV}^{=,\neq}$ schema mappings and for $\text{GLAV}^{=}$ schema mappings.

PROPOSITION 4.11. *Let (I, J) be a data example over source and target schema $\boldsymbol{S}$ and $\boldsymbol{T}$, where $\boldsymbol{T}$ contains only unary relations. For every GLAV schema mapping $\mathcal{M}$ there is a GAV schema mapping $\mathcal{M}'$, which can be obtained from $\mathcal{M}$ in polynomial time, such that*

i *$cost(\mathcal{M}', (I, J), GAV^{=,\neq}) = O(cost(\mathcal{M}, (I, J), GLAV^{=,\neq})^2)$, and*
ii *$cost(\mathcal{M}', (I, J), GAV^{=}) = O(cost(\mathcal{M}, (I, J), GLAV^{=})^2)$*

PROOF. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$. Consider the GAV mapping $\mathcal{M}^{\text{gav}} = (\mathbf{S}, \mathbf{T}, \bigcup_{\sigma\in\Sigma} \Sigma^{\text{gav}}(\sigma))$, where $\Sigma^{\text{gav}}(\sigma)$ is the set of GAV constraints obtained by removing all atoms containing existentially quantified variables, as well as all existential quantifiers, from the right-hand side of σ (where $\sigma \in \Sigma$), and, in case of multiple atoms in the right-hand side, splitting the constraint into several constraints, each having a single atom in the right-hand side. For example, if σ is the GLAV constraint $\forall xy(P(x) \wedge Q(y) \rightarrow \exists z(R(x) \wedge S(y) \wedge T(z) \wedge U(z))$, then $\Sigma^{\text{gav}}(\sigma)$ consists of the GAV constraints $\forall xy(P(x) \wedge Q(y) \rightarrow R(x))$ and $\forall xy(P(x) \wedge Q(y) \rightarrow S(y))$.

We claim that $\Sigma^{\text{gav}}(\sigma)$ satisfies the properties. We describe the argument for (i), but the argument for (ii) is similar. Consider any $\text{GLAV}^{=,\neq}$ repair $\widehat{\mathcal{M}}$ of $\mathcal{M}$ that is vfe for (I, J). We construct a corresponding $\text{GAV}^{=,\neq}$ repair $\widehat{\mathcal{M}^{\text{gav}}}$ for $\mathcal{M}^{\text{gav}}$ as follows:

(a) whenever $\widehat{\mathcal{M}}$ includes a repaired constraint $\widehat{\sigma}$ of a constraint $\sigma \in \Sigma$, then we add to $\widehat{\mathcal{M}^{\text{gav}}}$ a set $\widehat{\Sigma^{\text{gav}}(\sigma)}$ of repaired GAV constraints obtained as follows: for every GAV constraint $t \in \Sigma^{\text{gav}}(\sigma)$, we add to $\widehat{\Sigma^{\text{gav}}(\sigma)}$ a $\text{GAV}^{=,\neq}$-constraint t' obtained from t by applying the same equalities and inequalities (used in $\widehat{\sigma}$) that involve universally quantified variables occurring in t.
(b) whenever $\widehat{\mathcal{M}}$ includes a repaired constraint $\widehat{\sigma}$ containing a conditional equality of the form $(\ldots \rightarrow y = c)$, where y is an existentially quantified variable, then for each atom $T(y)$ in the right-hand side of $\widehat{\sigma}$, we add the ground fact $T(c)$ to $\widehat{\mathcal{M}^{\text{gav}}}$.
(c) whenever there is a ground fact f in $\widehat{\mathcal{M}}$, we add f to $\widehat{\mathcal{M}^{\text{gav}}}$.

We next show that $\widehat{\mathcal{M}^{\text{gav}}}$ is vfe for (I, J) and it is only quadratically larger than $\widehat{\mathcal{M}}$.

Validity. By construction, the ground facts included in $\widehat{\mathcal{M}^{\text{gav}}}$ are valid for (I, J), and it suffices to show that the GAV constraints of $\widehat{\mathcal{M}^{\text{gav}}}$ are also valid for (I, J). To show that, we take an arbitrary homomorphism h from the left-hand side of a constraint $t' \in \widehat{\mathcal{M}^{\text{gav}}}$ to I. By procedure (a), we know that t' was obtained from the repaired version $\widehat{\sigma}$ of some constraint

$\sigma \in \Sigma$. By the operations described in procedure (a), t' must have identical left-hand side as $\widehat{\sigma}$. Therefore, h is also a homomorphism from the left-hand side of $\widehat{\sigma}$ to I. Since every constraint in $\widehat{\mathcal{M}}$ is valid for (I, J), it means h can be extended to a homomorphism h' from the right-hand side of $\widehat{\sigma}$ to J. We then can easily extract a homomorphism h'' from h' such that h'' is a homomorphism from the right-hand side of t' to J. In particular, h'' is the restriction of h' to the universally quantified variables. Therefore, $\widehat{\mathcal{M}^{\text{gav}}}$ is valid for (I, J).

Fully Explaining. Since $\widehat{\mathcal{M}}$ is fully explaining for (I, J), we have that each fact in J is either one of the ground target facts of $\widehat{\mathcal{M}}$ or is explained by some constraint of $\widehat{\mathcal{M}}$. For those facts in J that are among the ground target facts in $\widehat{\mathcal{M}}$, they will also belong to $\widehat{\mathcal{M}^{\text{gav}}}$ (because of Procedure (c) described above). Now, consider a (unary) target fact $P(a)$ that is explained by a constraint $\widehat{\sigma} \in \widehat{\mathcal{M}}$. Let σ' be the constraint obtained from σ by dropping all atoms containing existentially quantified variables, and let $\widehat{\sigma'}$ be the repair of σ' obtained from $\widehat{\sigma}$ in the same way. Note that, by previous argument for validity, $\widehat{\sigma}$ is valid for (I, J), thus there is a homomorphism h from the left-hand side of $\widehat{\sigma}$ to I and it can be extended to a homomorphism h' from the right-hand side of $\widehat{\sigma}$ to J. Since $\widehat{\sigma}$ explains $P(a)$, the right-hand side of $\widehat{\sigma}$ contains an atom $P(z)$ such that $h'(z) = a$. We distinguish two cases:

Case I: the fact $P(a)$ is explained by $\widehat{\sigma}$ without using the extension h'. In this case, $P(a)$ is explained by $\widehat{\sigma'}$ using the homomorphism h.

Case II: the fact $P(a)$ is explained by $\widehat{\sigma}$ using the extension h'. In this case, the right-hand side of $\widehat{\sigma}$ must contain an atom of the form $P(y)$ and a conditional equality of the form $(\cdots \rightarrow y = a)$. It follows that, from our construction, the fact $P(a)$ is among the ground facts of schema mapping $\mathcal{M}^{\text{gav}}$ (because of procedure (b)).

This concludes the proof for fully explaining.

We now show that the size of $\widehat{\mathcal{M}^{\text{gav}}}$ is bounded quadratically in the size of $\widehat{\mathcal{M}}$. By construction, the operations described in procedure (c) will never result in a schema mapping that has greater size than the original one, and the procedure described in (b) will blows up the size by a constant factor. This is because each conditional equality explains at most one ground target fact (note that the target schema contains unary relations only), and it has size at least two). However, the operations described in procedure (a) may blow up the size to a quadratic factor. Therefore, the size of $\widehat{\mathcal{M}^{\text{gav}}}$ is bounded quadratically in the size of $\widehat{\mathcal{M}}$, and the proposition is proved. □

Proposition 4.11, intuitively, shows that the problem of computing near-optimal GAV schema mappings (for unary target schemas) reduces to the problem of computing near optimal GLAV schema mappings. Since the proof of Theorem 4.3 uses a unary target schema, it immediately implies that Theorem 4.3 holds true where GAV is replaced by GLAV.

4.3 Complexity of the problems Cost and Existence-Cost for LAV and SH-LAV schema mappings

Next, we show that the problems Cost and 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 [19].

Although restrictive, SH-LAV schema mappings together with the SH-LAV$^{=}$ repair language, can capture projection, column addition, column reordering, and selection conditions. Many of the primitives of two well-known primitive-based schema mapping benchmarks (STBenchmark [2] and iBench [6]) can be expressed as SH-LAV schema mappings and

SH-LAV$^{=}$ repairs. In particular, three out of eleven primitives of STBenchmark correspond to SH-LAV schema mappings and SH-LAV$^{=}$-repair: Copy, Constant Value Generation, Horizontal Partition. In iBench, there are five out of fifteen: Copy, Horizontal Partition, Add Attribute, Delete Attribute, and Add & Delete Attribute.

In addition, SH-LAV constraints capture a large fragment of the DL-Lite language used in ontology based data access (OBDA) [7]. More precisely, as pointed out in [12], a knowledge base (as defined in the OBDA literature) can be captured, in a precise formal sense, by a schema mapping that is specified by source-to-target copy-constraints as well as target constraints. If the knowledge base is specified in the core DL-Lite language, it turns out that each target constraint is either a SH-LAV constraint or a denial constraint (i.e., a first-order sentence of the form $\forall \mathbf{x}(\phi \rightarrow \bot)$ where ϕ is a conjunction of atoms).

THEOREM 4.12. *Let $\mathcal{L}^* \in \{\text{LAV}^{=,\neq}, \text{LAV}^{=}, \text{SH-LAV}^{=,\neq}, \text{SH-LAV}^{=}\}$. Then the problems* COST$_{\mathcal{L}^*}$ *and* 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 4.12 is established via a reduction from the SET-COVER problem. The following lemmas and propositions will be helpful for establishing the NP-upper bounds.

LEMMA 4.13. *Let E be a finite set of ground data examples and $\mathcal{M}$ a LAV schema mapping with cost$(\mathcal{M}, E, \text{LAV}^{=,\neq}) = n$. Then there is a SH-LAV schema mapping $\mathcal{M}'$ such that cost$(\mathcal{M}', E, \text{SH-LAV}^{=,\neq}) \leq n^2$. The same holds for when $=, \neq$ is replaced by $=$.*

PROOF. Let E be a finite set of ground data examples, let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a LAV-schema mapping, and let $\mathcal{M}_r = (\mathbf{S}, \mathbf{T}, \Sigma_r)$ be a minimal-size vfe LAV*-repair of $\mathcal{M}$ for E, where $*$ is either $=$ or $=, \neq$.

We will denote by $\mathcal{M}_r^{\text{shlav}}$ the SH-LAV*-schema mapping $(\mathbf{S}, \mathbf{T}, \Sigma_r^{\text{shlav}})$ where Σ_r^{shlav} is obtained as follows: for each LAV*-constraint $\sigma \in \Sigma_r$ of the form

$$R(\mathbf{x}) \wedge \alpha(\mathbf{x}) \rightarrow \exists \mathbf{y}(S_1(\mathbf{x}, \mathbf{y}_1) \wedge \cdots \wedge S_n(\mathbf{x}, \mathbf{y}_n) \wedge \beta(\mathbf{x}, \mathbf{y})),$$

(with $\mathbf{y}_1, \ldots, \mathbf{y}_n \subseteq \mathbf{y}$) we take all n SH-LAV* constraints σ' of the form

$$R(\mathbf{x}) \wedge \alpha(\mathbf{x}) \rightarrow \exists \mathbf{y}_i(S_i(\mathbf{x}, \mathbf{y}_i) \wedge \beta'(\mathbf{x}, \mathbf{y}_i)),$$

where $1 \leq i \leq n$, and where β' is obtained from β by keeping only those conjuncts that contain an existentially quantified variable belonging to $\mathbf{y}_i$. Thus, for example, if $\mathcal{M}_r$ contains the LAV$^{=}$-constraint $R(x, y) \rightarrow \exists uv\ S(x, u) \wedge T(z, u, v) \wedge (x = c_1 \wedge y = c_2 \rightarrow u = c_3) \wedge (x = c_1 \rightarrow v = c_4)$, then $\mathcal{M}_r^{\text{shlav}}$ contains $R(x, y) \rightarrow \exists u\ S(x, u) \wedge (x = c_1 \wedge y = c_2 \rightarrow u = c_3)$ and $R(x, y) \rightarrow \exists uv\ T(z, u, v) \wedge (x = c_1 \wedge y = c_2 \rightarrow z = c_3) \wedge (x = c_1 \rightarrow v = c_4)$.

We will denote by $\mathcal{M}_r^{\text{shlav}}$ the SH-LAV schema mapping obtained in the same way from $\mathcal{M}$. Observe that $\mathcal{M}_r^{\text{shlav}}$ is a SH-LAV*-repair of $\mathcal{M}^{\text{shlav}}$, and that $size(\mathcal{M}_r^{\text{shlav}}) \leq size(\mathcal{M}_r^2)$. It only remains to show that $\mathcal{M}_r^{\text{shlav}}$ is vfe for E. Validity is immediate from the construction (note that $\mathcal{M}_r$ logically implies $\mathcal{M}_r^{\text{shlav}}$). It is also clear from the construction that every fact in the target instance of a data example from E explained by a LAV*-constraint σ of $\mathcal{M}_r$ is also explained by one of the corresponding SH-LAV*-constraints of $\mathcal{M}_r^{\text{shlav}}$. Note that the above argument holds true because, according to the definition of the repair language, every conjunct in β can contain only one existential variable. □

Definition 4.14 (csize(E)). The constant-based size of a set E of ground data examples, denote by csize(E), is the total number of occurrences of constants in all data examples in E.

Note that the quantity csize(E) is different from the quantity $||E||$ introduced in Section 2: the former gives the total number of occurrences of constants in E, while the latter gives the total number of facts in E. We use $||E||$ as an amplification parameter in Lemma 4.5, which is part of the proof of Theorem 4.3. In the next lemma, we will use csize(E) to show that when a vfe SH-LAV$^=$-schema mapping for E exists, then there is a "small" one.

LEMMA 4.15. *Let E be a finite set of ground data examples for which there exists a vfe SH-LAV$^=$-schema mapping. There exists a SH-LAV schema mapping $\mathcal{M}$ such that cost($\mathcal{M}, E$, SH-LAV$^=$) is bounded by a polynomial in csize(E).*

PROOF. We know that there is a vfe SH-LAV$^=$-schema mapping for E. Each constraint t of a minimal vfe SH-LAV$^=$-schema mapping can explain at least one target fact (because otherwise, we can remove all non-explaining constraints and that will still be vfe for E but has smaller size). In the worst case, suppose that each SH-LAV$^=$ constraint of a minimal vfe SH-LAV$^=$-schema mapping only explains one target fact in E. Let $\mathcal{M}$ be a minimal vfe SH-LAV$^=$-schema mapping, let n be csize(E). The arity of atoms occurring in a SH-LAV$^=$-constraint t of $\mathcal{M}$ are both bounded by n, because $\mathcal{M}$ is minimal vfe for E.

Each variable occurring in the left-hand side of t can only be involved in at most one equality, because, if the left-hand side of t has two equalities involving same universally quantified variable but using different constants, then this constraint is trivial and can be removed. For example, the following constraint cannot be included in any minimal vfe schema mapping $\mathcal{M}$ for a set E of ground data examples, because we can always remove it from $\mathcal{M}$ to obtain a schema mapping of smaller size, and the resulting schema mapping is vfe for E.

$$\forall \mathbf{x}\ \varphi(\mathbf{x}) \wedge (x_i = a) \wedge (x_i = b) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}).$$

Also, in the worst case that all variables occurring in the right-hand side are existentially quantified, we have to add repairs that involve every variable in the right-hand side. The most costly way to repair an existentially quantified variable is to add a conditional equality with a conjunction of n equalities in the premise, and thus the maximum size of a repair on the right-hand side is $n \times (2n + 1)$. (Note that in fact the same existentially quantified variable can be repaired by multiple conditional equalities. However, it is enough to explain one fact with the repaired constraints with one conditional equality per existentially quantified variable). Therefore, the maximum size of an explaining SH-LAV$^=$ constraint for E is $2n^2 + 5n$ (n for unrepaired left-hand side, n for unrepaired right-hand side, $2n$ for the equalities on the left-hand side, and $2n^2 + n$ for the conditional equalities on the right-hand side). Since the number of facts in E is at most n, the maximum size of a minimal vfe SH-LAV$^=$ schema mapping for E is at most $2n^3 + 5n^2$, therefore polynomial in csize(E). □

The preceding lemmas imply the following result.

PROPOSITION 4.16. *Fix a source $\boldsymbol{S}$ and target schema $\boldsymbol{T}$. Let E be a finite set of ground data examples over $\boldsymbol{S}$ and $\boldsymbol{T}$. If there exists a vfe LAV$^{=,\neq}$-schema mapping for E, then there is a vfe LAV$^{=,\neq}$-schema mapping (in fact, a SH-LAV$^=$-schema mapping) $\mathcal{M}$ for E whose size is bounded by a polynomial in csize(E).*

PROOF. Lemma 4.13 shows that if a vfe LAV$^{=,\neq}$-schema mapping for E exists with cost n, then a vfe SH-LAV$^{=,\neq}$-schema mapping, whose cost is bounded by n^2, for E exists. To prove the Proposition, we first show that for every vfe SH-LAV$^{=,\neq}$-schema mapping $\mathcal{M}$ for E, we can efficiently (in polynomial time) rewrite $\mathcal{M}$ into a SH-LAV$^=$-schema mapping $\mathcal{M}'$ such that $\mathcal{M}$ and $\mathcal{M}'$ are logically equivalent. This will show the existence of a vfe

SH-LAV$^{=}$-schema mapping for E. Then, by Lemma 4.15, we know that there must be a SH-LAV$^{=}$-schema mapping for E whose size is polynomial in $csize(E)$.

Let $\mathcal{M}$ be the SH-LAV$^{=,\neq}$-schema mapping that is vfe for E, and assume that every constraint $t \in \mathcal{M}$ is of the form:

$$S(\mathbf{x}) \wedge \alpha(\mathbf{x}) \wedge \beta(\mathbf{x}) \rightarrow \exists \mathbf{y} T(\mathbf{x}, \mathbf{y}) \wedge \theta(\mathbf{x}, \mathbf{y}), \text{ where}$$

$\alpha(\mathbf{x})$ is a collection of equalities with each of the form $(x = c)$,
$\beta(\mathbf{x})$ is a collection of inequalities with each of the form $(x \neq c)$,
and $\theta(\mathbf{x}, \mathbf{y})$ is a collection of conditional equalities with each of the form $\bigwedge_i x_i = c_i \rightarrow y_k = c_k)$ (note that $x_i \in \mathbf{x}, y_i \in \mathbf{y}$, and c, c_i, c_k are constants).

Let $adom(E)$ be the union of the active domain of all source and target instances in E. We construct a SH-LAV$^{=}$-schema mapping $\mathcal{M}'$ from $\mathcal{M}$ such that $\mathcal{M}'$ is the schema mapping that contains, for each constraint $t \in \mathcal{M}$, all constraints t' that can be obtained from t by replacing each inequality $x \neq c$ by an equality $x = d$ for some constant $d \in adom(E)$ such that $d \neq c$, provided that t' does not contain multiple equalities $x = d_1$ and $x = d_2$ for the same universally quantified variable, with $d_1 \neq d_2$. As an illustrative example, assume that we have a ground data example (I, J), where

$I = \{S(a),\ S(b),\ S(c)\}$ and

$J = \{R(a),\ R(b)\}$,

and we are given the following SH-LAV$^{=,\neq}$-schema mapping that is vfe for (I, J):

$$\mathcal{M} = \{\{S\}, \{R\}, \Sigma = \{S(x) \wedge (x \neq c) \rightarrow R(x)\}\}.$$

Then we can obtain a new schema mapping $\mathcal{M}'$ by rewriting Σ into the set Σ' of constraints (see below):

$$\mathcal{M}' = \{\{S\}, \{R\}, \Sigma' = \{S(x) \wedge (x = a) \rightarrow R(x),\ S(x) \wedge (x = b) \rightarrow R(x)\}$$

In this way, the newly constructed schema mapping $\mathcal{M}'$ has the same validity and explanation properties with respect to E as the original schema mapping $\mathcal{M}$. Moreover, since the number of universally quantified variables in a constraint is bounded by a constant (i.e., the arity of the source schema), the number of constraints in $\mathcal{M}'$ is polynomial in the number of constraints of $\mathcal{M}$.

Note that a SH-LAV$^{=}$-schema mapping is also a SH-LAV$^{=,\neq}$-schema mapping and a LAV$^{=,\neq}$-schema mapping. Moreover, for any schema mapping language $\mathcal{L}$, given a finite set E of ground data examples, the size of smallest vfe $\mathcal{L}^{=,\neq}$-schema mapping for E is no greater than the size of smallest vfe $\mathcal{L}^{=}$-schema mapping. Therefore, the above two lemmas (Lemmas 4.13 and 4.15) imply Proposition 4.16. □

Before presenting the reduction from the Set-Cover problem to our problem, we need to show that our problem is in NP and the following lemma serves the purpose.

Lemma 4.17. *Given a finite set E of ground data examples and a LAV$^{=,\neq}$ schema mapping $\mathcal{M} = (\boldsymbol{S}, \boldsymbol{T}, \Sigma)$, determining whether $\mathcal{M}$ is vfe for E is in NP. If $\mathcal{M}$ is a SH-LAV$^{=,\neq}$ schema mapping, the same problem is in P.*

Proof. Since $\mathcal{M}$ is LAV$^{=,\neq}$, each constraint in Σ is of the form

$$\forall \mathbf{x}(S(\mathbf{x}) \wedge \alpha(\mathbf{x}) \rightarrow \exists \mathbf{y}(T_1(\mathbf{x}_1, \mathbf{y}_1) \wedge \cdots \wedge T_m(\mathbf{x}_m, \mathbf{y}_m) \wedge \beta(\mathbf{x}, \mathbf{y}))),$$

where $\mathrm{S}(\mathbf{x})$ is a relational atom over source schema, $\alpha(\mathbf{x})$ is a collection of equalities with each of the form $(x = c)$, $\beta(\mathbf{x}, \mathbf{y})$ is a collection of conditional equalities with each of the form $\bigwedge_i x_i = c_i \rightarrow y_k = c_k)$ (note that $x_i \in \mathbf{x}, y_i \in \mathbf{y}$, and c, c_i, c_k are constants), and $T_i(\mathbf{x}_i,\mathbf{y}_i)$ is a relational atom over target schema. Let n be the combined size of E and $\mathcal{M}$, that is, $n = \mathrm{csize}(E) + size(\mathcal{M})$. For simplicity, we consider the case where Σ contains only one constraint σ. The same arguments apply in the case of a set of constraints.

Validity checking. There are at most n homomorphisms from $S(\mathbf{x})$ (one for each source fact in E) to a source instance I, with $(I, J) \in E$. For every such homomorphism, we can efficiently check if $\alpha(\mathbf{x})$ is satisfied. For each homomorphism h that makes $S(h(\mathbf{x})) \wedge \alpha(h(\mathbf{x}))$ be true, we non-deterministically guess an extension h' of h w.r.t. the existential variables in the right-hand side of σ, such that $\beta(h'(\mathbf{x}), h'(\mathbf{y}))$ holds. We can then verify in polynomial time that h' is a homomorphism from the right-hand side of σ to the target instance J.

Explanation checking. The total number of target facts in the data examples is at most n, and it suffices to check that each target fact is explained by $\mathcal{M}$. Let $(I, J) \in E$ and let $T(\mathbf{a})$ be a fact belonging to J. To test if $\mathcal{M}$ explains $T(\mathbf{a})$, we guess a homorphisms h from the left-hand side of σ to I such that the right-hand side of σ under h (i.e., $\exists \mathbf{y}(T_1(h(\mathbf{x}_1), \mathbf{y}_1) \wedge \cdots \wedge T_m(h(\mathbf{x}_m), \mathbf{y}_m) \wedge \beta(h(\mathbf{x}), \mathbf{y}))$) logically implies $T(\mathbf{a})$. This can be done in polynomial time (due to the fact that $T(\mathbf{a})$ is a ground atom). If the answer is positive, then, clearly, every solution of I w.r.t. $\mathcal{M}$ (that is, every target instance J such that $\mathcal{M}$ is valid w.r.t. (I, J)) must contain $T(\mathbf{a})$, and hence, $\mathcal{M}$ explains $T(\mathbf{a})$ w.r.t. (I, J). If not, then construct a solution of I with respect to $\mathcal{M}$ that does not contain $T(\mathbf{a})$, and hence, $\mathcal{M}$ does not explain $T(\mathbf{a})$ w.r.t. (I, J).

On the other hand, checking the validity and explanation of SH-LAV$^{=,\neq}$-schema mapping is in P. Because there are at most n homomorphisms from the left-hand side of a SH-LAV$^{=,\neq}$-constraint t to E and for each homomorphism h that makes the left-hand side of t be true, there are also at most n homomorphisms from the right-hand side of a SH-LAV$^{=,\neq}$-constraint to E. We can try then all, and the total computation is polynomial in the size of input. □

We are ready to prove Theorem 4.12.

Proof of Theorem 4.12. We will prove NP-completeness of Existence-Cost$_{\mathrm{LAV}^{=,\neq}}$. The proof of the other items in Theorem 4.12 is identical.

First, we need to show the problem is in NP. Let $\mathbf{S}$ be a source schema, $\mathbf{T}$ be a target schema. Given a finite set E of ground data examples such that E conforms to $\mathbf{S}$ and $\mathbf{T}$. By Proposition 4.16 we know that if there is a vfe LAV$^{=,\neq}$ schema mapping for E, then there is one whose size is bounded by $p(||E||)$, for some polynomial p. This places the problem in NP: we can guess a schema mapping of size at most $p(||E||)$, and verify the guess in polynomial-time with an additional non-deterministic guess using Lemma 4.17. Note that both non-deterministic guesses can be combined into one guess. The NP-hardness is shown via a reduction from Set-Cover [23].

Set-Cover: an instance of the Set-Cover problem is a pair (X, S) where X is a set and S is a collection of subsets of X, such that $\bigcup_{S_i \in S} S_i = X$. A solution for (X, S) is a subset S' of S such that $\bigcup_{S_k \in S'} S_k = X$. An optimal solution is a solution such that the size of S' is minimum.

The following problem is known to be NP-complete: given an instance of the Set-Cover problem (X, S) and a natural number K, does there exist a solution S' with $|S'| \leq K$?

We encode an instance (X, S) of the Set-Cover problem by a data example (I, J) over the schemas $\mathbf{S}, \mathbf{T}$, where $\mathbf{S}$ consists of a single 5-ary relation P, and $\mathbf{T}$ consists of a single 4-ary relation Q. For each element $a \in X$, we create four distinct constants a_1, a_2, a_3, a_4. The source instance I contains all facts $P(i, a_1, a_2, a_3, a_4)$ for $S_i \in S$ and $a \in S_i$ (we assume, without loss of generality, that the index i is not itself an element in X). In addition, I contains $7|X|$ facts of the form $P(b, c, d, e, f)$, where b, c, d, e, f are fresh distinct constants not contained in X. We will refer to these additional facts as *garbage facts* below. The target instance J contains all facts $Q(a_1, a_2, a_3, a_4)$ for $a \in X$. Obviously, the construction of (I, J) from (X, S) is a polynomial-time transformation.

Claim: the optimal solution for (X, S) has size of k iff $\text{cost}(M_{opt}, \{(I, J)\}, LAV^{=,\neq}) = 11k$, where M_{opt} is LAV$^{=,\neq}$-optimal for (I, J).

($\Rightarrow$) Let $S' \subseteq S$ be an optimal solution for the Set-Cover problem (X, S), and let $|S'| = k$. We will construct a vfe LAV$^{=,\neq}$ schema mapping for (I, J) and that has size $11k$. Consider the LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ where $\Sigma = \{t\}$ for

$$t : P(x_1, y_1, y_2, y_3, y_4) \rightarrow Q(y_1, y_2, y_3, y_4).$$

Let Σ_r be the set containing, for each $S_i \in S'$, the LAV$^{=,\neq}$ constraint

$$t'_i : \ P(x, y_1, y_2, y_3, y_4) \wedge (x = i) \rightarrow Q(y_1, y_2, y_3, y_4).$$

Clearly, $\mathcal{M}_r = (\mathbf{S}, \mathbf{T}, \Sigma_r)$ is a repair of $\mathcal{M}$, and $\mathcal{M}_r$ is vfe for (I, J), and the size of $\mathcal{M}_r$ is $11k$. Now we will show that $\mathcal{M}$ is LAV$^{=,\neq}$-optimal for (I, J), because no other vfe LAV$^{=,\neq}$ schema mapping for (I, J) has smaller size than Σ_r.

Let $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ be any vfe LAV$^{=,\neq}$-schema mapping for (I, J) of size ℓ. We may assume that Σ' does not contain ground facts: every ground fact costs 12 (recall that a ground fact costs $3n$, where n is the arity of the fact), while the same fact can be explained by a valid constraint of the form t'_i above, for suitable choice of i, which costs only 11. Therefore, Σ' consists of LAV$^{=,\neq}$-constraints of the form

$$P(\mathbf{x}) \wedge \alpha(\mathbf{x}) \rightarrow \exists \mathbf{w} Q(\mathbf{x}, \mathbf{w}) \wedge \cdots \wedge Q(\mathbf{x}, \mathbf{w}) \wedge \beta(\mathbf{x}, \mathbf{w}).$$

We may furthermore assume that each constraint in Σ' explains at least one target fact (all constraints that do not explain any target fact can be removed without increasing the size of $\mathcal{M}'$ and the resulting schema mapping is still vfe for (I, J)).

Consider the left-hand side of the above constraint. We claim that all variables occurring in the P-atom are distinct, that is, $P(\vec{x})$ is of the form $P(x, y_1, y_2, y_3, y_4)$: otherwise, the constraint cannot explain any fact in J. Furthermore, the variable x that is in the *first* position of P-atom does not occur in the right-hand side of the constraint (otherwise either the constraint is invalid for (I, J), or it does not explain any fact), and that the variable y_j (for $1 \leq j \leq 4$) occurs only in the j-th position of atom in the right-hand side of the constraints (again, otherwise the constraint is either invalid or unable to explain a fact).

Now consider the right-hand side of the constraint. We may assume without loss of generality that every Q-atom contains at least one variable that is not existentially quantified. Indeed, if some atom $Q(w_1, \ldots, w_4)$ consisted entirely of existentially quantified variables, then, in order for the atom to explain a fact, β must include 4 (possibly conditional) equalities of the form $(w_j = c_j)$ $(1 \leq j \leq 4)$. These equality conditions can be removed at the expense of adding the constraint t'_i (as defined above) for some $S_i \in S$ with $c \in S_i$ (which is valid and explains the same fact, while having a lower cost), after which the atom $Q(w_1, \ldots, w_4)$ can be removed from the right-hand side of the constraint entirely.

Thus, each Q-atom contains one of the variables y_j in its j-th position. It follows, through the construction of (I, J) that, under any given assignment of values to the variables $x, y_1, \ldots, y_n$ that makes the left-hand side true, each Q-atom in the right-hand side can explain at most one fact of J, namely the fact that has the value of y_j in its j-th position. But this means that any Q-atom of the form $Q(\ldots, y_j, \ldots)$ can be replaced by $Q(y_1, \ldots, y_n)$ without changing the collection of facts explained by the constraint, or the validity of the constraint, and without increasing the size of the constraint. This also shows that there is no benefit in having more than one Q-atom in the right-hand side of the constraint. To summarize, from the assumption that $\mathcal{M}'$ is a minimal vfe LAV$^{=,\neq}$-schema mapping for (I, J), we have derived that every constraint of $\mathcal{M}'$ is of the form

$$P(x, y_1, y_2, y_3, y_4) \wedge \alpha(x, \mathbf{y}) \rightarrow Q(y_1, y_2, y_3, y_4)$$

Finally, we argue that, without loss of generality, $\alpha(x, \mathbf{y})$ is of the form $(x = i)$. If $\alpha(\mathbf{x})$ contains an equality of the from $y_j = c$ we can replace it by an equality of the form $x = i$ for any i such that $c \in S_i$, without affecting validity and the resulting constraint explains all the same facts and possibly more. If α contains an inequality $z \neq c$, then, it follows from the construction of the data example (I, J) that either the inequality can be removed without affecting the validity of the constraint, or the left-hand-side must contain as many inequalities as the number of garbage facts, in which case the size of the constraint is at least $14|X|$ (each inequality has size 2). In the latter case, we can replace the schema mapping by one that consists of $|X|$ many constraints of the form t'_i — at most one for each element of X in the worst case, with a total cost of at most $11|X|$.

In summary, we may assume without loss of generality that the constraint is of the form

$$P(x, y_1, y_2, y_3, y_4) \wedge (x = i) \rightarrow Q(y_1, y_2, y_3, y_4),$$

and indeed, by the above arguments, every minimal-size vfe LAV$^{=,\neq}$-schema mapping for (I, J) must consist entirely of constraints of the above form. The collection of target facts explained by this constraint corresponds precisely to the the set S_i from the Set-Cover problem. In this way, we obtain that every minimal-size vfe LAV$^{=,\neq}$-schema mapping for (I, J) has cost $11k$ for some k, and induces a solution of the Set-Cover problem (X, S) that has size k.

($\Leftarrow$) Suppose the cost of a LAV$^{=,\neq}$-optimal schema mapping $\mathcal{M}$ for (I, J) is $11k$. Let $\mathcal{M}_r$ be the minimal vfe $LAV^{=,\neq}$-repair of $\mathcal{M}$. By the above arguments, we know that each constraint of $\mathcal{M}_r$ is of the form $P(x, y_1, \ldots, y_n) \wedge (x = c) \rightarrow Q(y_1, \ldots, y_n)$. But then, by the above arguments, we have that the Set-Cover problem (X, S) admits a solution of size k, and, moreover, we have that the Set-Cover problem admits no smaller solution (as such as smaller solution would entail a smaller vfe LAV$^{=,\neq}$ for (I, J).

The NP-completeness of the Cost problems now follows, since Set-Cover is NP-hard. □

5 APPROXIMATION OF OPTIMAL SINGLE-HEAD LAV MAPPINGS

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

Definition 5.1. The Optimal-Repair$_{\mathcal{L}^*}(\mathbf{S}, \mathbf{T})$ problem asks: given a set E of ground data examples with source schema $\mathbf{S}$ and target schema $\mathbf{T}$, compute a minimal-size valid and fully explaining $\mathcal{L}^*$-schema mapping for E.

Note that, in the above formulation of the problem, the source and target schemas are fixed. One may also consider a variant where the schemas are part of the input. We will not consider the latter here, and we leave its complexity as an open problem to be addressed

in future work. We note, though, that the difference between these two variants (i.e., the variant in which the source and target schemas are fixed and the variant in which they are part of the input) is, in some sense, the difference between *data complexity* and *expression complexity*; for this reason, we expect that the second variant will be of higher complexity than the first.

The above 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 dropping all equalities, inequalities, and ground facts. Therefore, it follows from Theorem 4.3 that (assuming RP $\neq$ NP) there is no polynomial-time algorithm that solves Optimal-Repair$_{\mathcal{L}^*}$, when $\mathcal{L}^* \in \{\text{GLAV}^{=}, \text{GLAV}^{=,\neq}, \text{GAV}^{=}, \text{GAV}^{=,\neq}\}$.

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

Theorem 5.2. *Fix a pair of schemas $\boldsymbol{S}$, $\boldsymbol{T}$. There is a polynomial-time $\mathcal{H}(n)$-approximation algorithm for* Optimal-Repair$_{SH\text{-}LAV^{=}}(\boldsymbol{S}, \boldsymbol{T})$, *where $\mathcal{H}(n) = \sum_{i=1}^{n} \frac{1}{i}$ is the n-th harmonic number.*

Our approximation algorithm, see Algorithm GreedySHLAV$^{=}(\mathbf{S}, \mathbf{T})$ shown in Figure 1, is obtained by establishing a close connection between Optimal-Repair$_{\mathcal{L}^*}$ and Set-Cover. 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 [5, 22]) 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 for Weighted Set-Cover [14]. 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 5.2 holds for the general case of SH-LAV$^{=,\neq}$ and LAV$^{=,\neq}$.

For a data example (I, J) and a SH-LAV$^{=}$ constraint t, we denote by $Facts_{(I,J)}(t)$ the set of ground target facts that are explained by t in (I, J). For a ground target fact t, $Facts_{(I,J)}(t) = \{t\}$ if $t \in J$ and $Facts_{(I,J)}(t) = \emptyset$ otherwise. Our algorithm, called GreedySHLAV$^{=}$ is given in Figure 1.

The Algorithm GreedySHLAV$^{=}(\mathbf{S}, \mathbf{T})$ is established based on a known greedy algorithm for the Weighted Set-Cover problem. The set of all target facts occurring in a given finite set E of data examples can be seen as the universe of elements that needs to be covered (explained), and each valid SH-LAV$^{=}$-constraint can be seen as inducing a subset that covers a number of elements (explained target facts) of the universe, and the size of such a SH-LAV$^{=}$-constraint can be seen as the cost associated to the corresponding subset. Specifically, Line 1 initializes the variable "Candidates" that contains all possible SH-LAV$^{=}$ constraints that are valid for E over $\mathbf{S}$ and $\mathbf{T}$, and this corresponds to creating the collection of "subsets" of the Weighted Set-Cover instance. Line 2 initializes the variable "Unexplained" that corresponds to the "universe" of the Weighted Set-Cover instance for which we are trying to find a cover. The while loop in the algorithm is precisely the greedy algorithm

Input: finite set E of data examples
Output: a SH-LAV$^{=}$-schema mapping that is vfe for E or "None exists"
Candidates $\leftarrow$ all SH-LAV$^{=}$ constraints over $\mathbf{S}, \mathbf{T}$ (up to variable renaming) that are valid for E, plus all ground target facts that belong to *every* data example in E.
Unexplained $\leftarrow$ all triples (I, J, F) with $(I, J) \in E$ and F a ground fact belonging to J.
$M \leftarrow \emptyset$
while *Unexplained* $\neq \emptyset$ **do**
  if *Candidates*$=\emptyset$ **then**
    return "No valid and fully explaining SH-LAV$^{=}$-schema mapping exists"
  end
  Find a $t \in$ Candidates that minimizes

$$\alpha = \frac{size(t)}{|\{(I, J, F) \mid (I, J) \in E \text{ and } F \in Facts_{(I,J)}(t)\} \cap \text{Unexplained}|}$$

  $M \leftarrow M \cup \{t\}$
  Unexplained $\leftarrow$ Unexplained $\setminus \{(I, J, F) \mid (I, J) \in E$ and $F \in Facts_{(I,J)}(t)\}$
  Candidates $\leftarrow$ Candidates $\setminus \{t\}$
end

Fig. 1. Algorithm GreedySHLAV$^{=}(\mathbf{S}, \mathbf{T})$

for Weighted Set-Cover. In particular, at each iteration, the algorithm will pick a constraint (corresponds to a subset) that has the highest cost effectiveness (which is defined on Line 8). The following theorem formally shows that this algorithm is a polynomial-time $\mathcal{H}(n)$-approximation algorithm for our problem.

Theorem 5.3. *For each fixed pair of schemas $\boldsymbol{S}$, $\boldsymbol{T}$, GreedySHLAV$^{=}(\boldsymbol{S}, \boldsymbol{T})$ is a polynomial time $\mathcal{H}(n)$-approximation algorithm for* Optimal-Repair$_{SH\text{-}LAV^{=}}(\boldsymbol{S}, \boldsymbol{T})$.

We first introduce the following helper lemma for the proof of Theorem 5.3

Lemma 5.4. *Let E be a finite set of ground data examples with fixed source schema $\boldsymbol{S} = \{S_1, \cdots, S_p\}$ and target schema $\boldsymbol{T} = \{T_1, \cdots, T_q\}$. Let m be the maximum arity of source relations and n the maximum arity of target relations, and let $|adom(E)|$ be the size of active domain of E. There are at most $pqm^m(m+n)^n(|adom(E)|^m n)^n$ distinct SH-LAV$^{=}$-constraints for E, up to renaming of variables.*

Proof. Note that p, q, m, n are constants since schemas are fixed. Observe that there are at most $pqm^m(m+n)^n$ distinct SH-LAV constraints over $\mathbf{S}, \mathbf{T}$, up to renaming of variables. In particular, we can decompose the number as follows.

(1) $p \times q$ - maximum of combinations of source relation and target relation.
(2) m^m - given a source relation S_k in $\mathbf{S}$, the maximum number of distinct single relational atoms over S_k.
(3) $(m+n)^m$ - given a source relation $S_k \in \mathbf{S}$ and a target relation $T_k \in \mathbf{T}$, the maximum number of distinct single relational atoms over T_k. In particular $(m+n)$ is the maximum number of variables, both universally quantified and existentially quantified, that can be used for constructing an atom over T_k.

Since only equalities are allowed in the repairs, for each SH-LAV-constraint there are at most $|adom(E)|^m$ distinct repairs on the left-hand side, and at most n existentially quantified variables on the right-hand side. For each existentially quantified variable y_k on the right-hand side, there is possibly a conditional equality. It follows that we have at most $|adom(E)|^m$ distinct ways to construct the premise of a conditional equality, and at most n

ways to construct the conclusion of the corresponding equality. It follows that there are at most $(|adom(E)|^m n)^n$ ways to construct the right-hand side. Therefore, there are at most $pqm^m(m+n)^n(|adom(E)|^m n)^n$ distinct SH-LAV$^=$-constraints over $\mathbf{S}$, $\mathbf{T}$, and E. □

We are now ready to prove Theorem 5.3.

Proof of Theorem 5.3. The GreedySHLAV$^=(\mathbf{S},\mathbf{T})$ algorithm is derived from the $\mathcal{H}(n)$-approximation algorithm for Weighted Set-Cover problem. Indeed, the problem of finding a minimal-size vfe SH-LAV$^=$-schema mapping can be cast as a weighted set-cover problem: the set of all target facts of a given finite set E of data examples can be seen as the universe of elements that needs to be covered (explained), and each valid and explaining SH-LAV$^=$-constraint can be seen as a subset that covers (explain) a number of elements (target facts) of the universe, and the size of such a SH-LAV$^=$-constraint can be seen as the cost associated to the corresponding subset. Note that the optimal SH-LAV$^=$-schema mapping for a finite set E of data examples, if exists, must be a subset of the set of all valid and explaining SH-LAV$^=$-constraints over $\mathbf{S}$, $\mathbf{T}$, E, plus all ground facts of E. We can view the GreedySHLAV$^=(\mathbf{S},\mathbf{T})$ algorithm as a special case of the well-known greedy $\mathcal{H}(n)$-approximation algorithm for Weighted Set-Cover problem. In particular, the correctness of our algorithm follows from the correctness of the aforementioned approximation algorithm for Weighted Set-Cover [14].

It remains to show that the algorithm runs in polynomial time. Since the schema mappings in consideration are SH-LAV$^=$, by Lemma 4.17, checking validity and explanation of a SH-LAV$^=$-schema mapping for a finite set E of ground data examples can be done in polynomial time. Moreover, since the source schema and target schema in question are fixed, it suffices to show that the number of SH-LAV$^=$-constraints is polynomial in the size of E. By Lemma 5.4, this is indeed the case. Therefore, GreedySHLAV$^=(\mathbf{S},\mathbf{T})$ runs in polynomial time and it is a poly-time $\mathcal{H}(n)$-approximation algorithm for Optimal-Repair$_{\text{SH-LAV}^=}(\mathbf{S},\mathbf{T})$, and it is also easy to see that the algorithm can be extended to a poly-time $\mathcal{H}(n)$-approximation algorithm for Optimal-Mapping$_{\text{SH-LAV}^=}$. □

In Section 6.1, we will discuss practical implementation details for the approximation algorithm.

5.1 A Matching Lower Bound

We conclude with a matching lower bound for the approximation properties of Optimal-Repair$_{\text{SH-LAV}^=}(\mathbf{S},\mathbf{T})$. This result is established via an approximation-preserving reduction (more precisely, an L-reduction [32]) from Minimum Set Cover.

Theorem 5.5. *Let $\mathcal{L} \in \{$LAV,SH-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$_{\mathcal{L}^{=,\neq}}(\boldsymbol{S},\boldsymbol{T})$.

We first recall the definition of L-reduction. Let Π, Π' be two optimization problems, we say Π L-reduces to Π', denoted by $\Pi \preceq_L \Pi'$, if there is a pair of polynomial-time algorithms f, g, and constants $\alpha, \beta > 0$ such that for each instance Ins of Π, the following hold:

(a) Algorithm f produces an instance $Ins' = f(Ins)$ of Π' such that $OPT(Ins') \leq \alpha \cdot OPT(Ins)$, where $OPT(Ins')$ is the cost of an optimal solution for Ins' and $OPT(Ins)$ is the cost of an optimal solution for Ins.

(b) Given any solution of Ins' with cost c', algorithm g produces a solution of Ins with cost c such that $|c - OPT(Ins)| \leq \beta|c' - OPT(Ins')|$.

To prove Theorem 5.5, it suffices to prove the following theorem.

THEOREM 5.6. *Let $\mathcal{L} \in \{LAV, SH\text{-}LAV\}$. There are L-reductions from* MINIMUM SET-COVER *to* OPTIMAL-REPAIR$_{\mathcal{L}^=}(\boldsymbol{S}, \boldsymbol{T})$ *and to* OPTIMAL-REPAIR$_{\mathcal{L}^{=,\neq}}(\boldsymbol{S}, \boldsymbol{T})$.

PROOF. We spell out the proof for the LAV$^{=,\neq}$ case. The proof can be adapted to obtain also the L-reductions to $\mathcal{L}^=$ and $\mathcal{L}^{=,\neq}$, where $\mathcal{L} \in \{\text{LAV}, \text{SH-LAV}\}$.

Let f be the transformation function used in the reduction of the proof of Theorem 4.12. f is a poly-time computable function that transforms an instance Ins of MIN SET-COVER to an instance (I, J) (a LAV$^{=,\neq}$-schema mapping) of OPTIMAL-REPAIR$_{\text{LAV}^{=,\neq}}(\mathbf{S}, \mathbf{T})$.

We must show that the following hold for suitable $\alpha, \beta > 0$.

(a) For all instance Ins of MIN SET-COVER, algorithm f produces an instance $(I, J) = f(Ins)$ such that $OPT((I, J)) \leq \alpha \cdot OPT(Ins)$, where $OPT((I, J))$ is the cost of the optimal LAV schema mapping for (I, J) in the LAV$^{=,\neq}$ repair language, and $OPT(Ins)$ is the cost of an optimal solution for the MIN SET-COVER instance Ins.

(b) For all vfe LAV$^{=,\neq}$-schema mappings for (I, J) with size c', there is a polynomial-time algorithm g that produces a set cover for Ins with cost c such that $|c - OPT(Ins)| \leq \beta|c' - OPT(Ins')|$.

Since we know that $OPT(Ins) = k$ iff $OPT((I, J)) = 11k$, property (a) holds for $\alpha = 11$.

For property (b), suppose that we are given an arbitrary vfe LAV$^{=,\neq}$-schema mapping $\mathcal{M} = \{\mathbf{S}, \mathbf{T}, \Sigma\}$. By the arguments given in the proof of Theorem 4.12, we may assume without loss of generality that Σ consists entirely of LAV$^{=,\neq}$-constraints of the form

$$t_i = P(x, y_1, y_2, y_3, y_4) \wedge (x = i) \rightarrow Q(y_1, y_2, y_3, y_4).$$

Note that the relevant arguments in the proof of Theorem 4.12 involve polynomial-time computable transformations. In particular, we are using the fact that the problem whether a LAV$^{=,\neq}$-constraint explains a target fact is polynomial-time computable. The fact is true because we can chase the source instance with the given LAV$^{=,\neq}$-constraint and see if the chase result contains the target fact in question. Note that the chase procedure, introduced in [17], runs in polynomial time.

We define the function $g(\mathcal{M})$ to be $\{\ i \mid t_i \in \Sigma\ \}$, and $g(\mathcal{M})$ is a set cover for Ins. Let c be the cardinality of $g(\mathcal{M})$. We show that $g(\mathcal{M})$ satisfies the property (b). Let c' be the size of $\mathcal{M}$. Obviously, c' is a multiple of 11 because each constraint of the form t_i in $\mathcal{M}$ has cost 11. Moreover, we have that $c' \geq 11c$ because the number of constraints in $\mathcal{M}$ is at least c. It follows that we have following inequality:

$$|c - k| = \frac{1}{11}|11c - 11k| \leq \frac{1}{11}|c' - 11k|.$$

This shows that property (b) holds with $\beta = \frac{1}{11}$. The theorem is proved. Note that the proof for the $\mathcal{L}^=$ case follows immediately from the proof of $\mathcal{L}^{=,\neq}$ case. □

It was shown in [33] that there is a constant c such that there is no polynomial time $c\ln(n)$-approximation algorithm for MINIMUM SET COVER, where n is the size of the universe of a given set-cover problem instance. The L-reduction established in Theorem 5.6 translates every MINIMUM SET COVER problem instance with universe of size n into an OPTIMAL-REPAIR problem instance consisting of a single data example with at most n target facts. Therefore, Theorem 5.5 is proved.

6 IMPLEMENTATION AND EXPERIMENTAL EVALUATION

We implemented the approximation algorithm presented in the previous section. In this section, we discuss the optimizations we used to improve the efficiency of the algorithm (detailed discussion can be found in the Appendix). We also present an experimental evaluation of the optimized implementation on a real-world schema mapping scenario.

6.1 Optimized Implementation

The first step of the approximation algorithm, described in Figure 1, consists of two phases:

(1) *Constraints-generation phase*: compute all SH-LAV$^=$-constraints for the input set E of data examples.
(2) *Validity-checking phase*: check the validity of each constraint obtained in Step 1 w.r.t. E, and omit the constraints that are invalid.

The Constraints-generation phase (Phase 1) is computationally expensive, and must be optimized so that the approximation algorithm is practically feasible. We have applied several optimizations to improve the efficiency of the implementation. The optimizations help reduce the space of SH-LAV$^=$-constraints that is considered by the algorithm. In this way, the optimized algorithm can produce vfe SH-LAV$^=$-schema mappings that are still within the $\log(n)$-factor approximation ratio. In our implementation, we have applied two kinds of optimizations: (1) *zero-amplification* optimizations, and (2) *constant-amplification* optimizations.

Intuitively, zero-amplification optimizations prune away constraints that would not be picked by the greedy algorithm. Hence, the approximation bound remains unchanged by this kind of optimizations. For instance, one of the zero-amplification optimizations we applied is the *Global Threshold Rule*. This rule is established based on the observation that, for a given ground data example (I, J), the schema mapping consisting of all facts in J is a trivial vfe schema mapping for (I, J). We can therefore use the cost of the trivial vfe mapping as a global threshold so that we can ignore every candidate constraint whose size is greater than or equal to this threshold. Clearly, such an optimization never increases the $\log(n)$-approximation bound. In contrast, we also consider constant-amplification optimizations which may improve the running time of the algorithm but at the expense of increasing the approximation bound by a fixed constant-factor. The detailed discussion of the two optimizations is omitted here, and we refer the interested reader to Appendix A for more details on these two kinds of optimizations. Example 6.1 below illustrates both the algorithm and some of the optimizations we have implemented.

Example 6.1. In this example, we will first illustrate the workflow of the approxiamtion algorithm (PART I) and then illustrate an example of constant-amplification optimizations (PART II).

PART I- workflow of the approximation algorithm.

Consider the data example (I, J) shown in Table 1. There is no vfe SH-LAV schema mapping for (I, J) and that there are only five pairwise logically inequivalent SH-LAV constraints (up to variable renaming) for these schemas:

σ_1:Geo$(x, x) \rightarrow$City(x) σ_2:Geo$(x, y) \rightarrow$City(y) σ_3:Geo$(x, y) \rightarrow$City(x)
σ_4:Geo$(x, x) \rightarrow \exists m$City$(m)$ σ_5:Geo$(x, y) \rightarrow \exists m$City$(m)$

Each of the five SH-LAV constraints has many possible SH-LAV$^=$ repairs. The first step of the algorithm computes the set of all SH-LAV$^=$ constraints (up to renaming of variables) that are valid in (I, J), and all ground target facts in J. Our implementation

will perform the following optimizations to avoid searching through the large number of SH-LAV$^=$-constraints in general.

For our example, observe that all SH-LAV$^=$-repairs of σ_1 can be immediately disregarded because their left-hand side cannot be satisfied in I. The same argument applies to σ_4. Similarly, all SH-LAV$^=$-repairs of the SH-LAV constraint σ_5 can be omitted, because the only way they could explain a target fact is if they have a conditional equality in their right-hand side of the form $(x = c \rightarrow m = d)$ or $(m = d)$. Each implication of the form $(x = c \rightarrow m = d)$ costs 4 and only helps explain one target fact. The equality of the form $(m = d)$ costs 2 but the entire constraint can only explain one target fact. Hence, the cost would be lower if we had simply included the corresponding target facts as ground facts in the SH-LAV$^=$ schema mapping. These optimizations are examples of zero-amplification optimizations, because they do not increase the approximation bound. Next, consider σ_3. We know that there is no SH-LAV$^=$-repair of σ_3 that is both valid for (I, J) and that explains even a single fact (in general, we can restrict attention to $\mathcal{L}$-constraints where a universally quantified variable occurs in a specific position of the right-hand side, only when one of the data examples actually contains facts matching it). This is another example of a zero-amplification optimization.

To summarize, for this data example, it is sufficient to consider SH-LAV$^=$ repairs of σ_2, and ground target facts, without affecting the quality of the schema mapping produced by the algorithm. Finally, observe that it is sufficient to consider only equalities that "are realized in the data example" when considering repairs of σ_2. In other words, it is unnecessary to consider a repair of σ_2 that includes equalities such as $(x = \text{US}) \wedge (y = \text{London})$ because no fact of I will satisfy these equalities.

After applying these optimizations, the number of candidate SH-LAV$^=$-constraints is 57 (plus ground target facts).

Next, the algorithm will repeatedly pick a constraint or ground fact that has highest cost effectiveness at each iteration until all facts from J are explained. The algorithm will perform the following in order

- 1: select "Geo(x,y) $\wedge$ (x = US)$\rightarrow$ City(y)", which explains five previously-unexplained facts
- 2: select "Geo(x,y) $\wedge$ (x = California)$\rightarrow$City(y)", which explains two previously-unexplained facts
- 3: add a ground fact "City(Toronto)".

After these three iterations, the algorithm terminates and output the following set of SH-LAV$^=$-constraints:

$$\{\text{Geo(x,y)} \wedge (\text{x} = \text{US}) \rightarrow \text{City(y)}, \text{City(Toronto)}, \text{Geo(x,y)} \wedge (\text{x} = \text{California}) \rightarrow \text{City(y)}\}.$$

Hence, our algorithm obtains a SH-LAV schema mapping of $size(\mathcal{M}_{greedy}) = 13$. The following SH-LAV$^=$-schema mapping (which can be shown to be minimal) has size 10.

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

PART II - constant-amplification optimizations.

The first step of our approximation algorithm is to compute a set of candidate SH-LAV-constraints (in the base language) together with their SH-LAV$^=$-repairs which will be later used by the Set-Cover greedy algorithm. To illustrate the constant-amplification optimizations, consider the following simple data example (I, J) and a SH-LAV-constraint t

for which we need to compute repairs.

$$I = \{S(a,1), S(b,2), S(c,3)\}, \quad J = \{T(a,a,a), T(b,b,b)\}$$

$$t : S(x,y) \rightarrow \exists m\ T(m,m,m).$$

The optimal SH-LAV$^{=}$-repair of t that is vfe for (I,J) is

$$t' : S(x,y) \rightarrow \exists m\ T(m,m,m) \wedge (y = 1 \rightarrow m = a) \wedge (y = 2 \rightarrow m = b)$$

The constraint t' uses multiple conditional equalities to explain more than one target fact. In practice, computing all possible combinations of conditional equalities that are valid and explaining for multiple target facts is computationally expensive. In order to make the runtime be within reasonable range, our implementation adds only one conditional equality to each existentially quantified variable in a constraint in consideration. If we want to add an additional conditional equality over the same existentially quantified variable, we make a copy of the original SH-LAV-constraint and then add the new conditional equality. For instance, for the data example (I,J) and t shown above, our implementation will generate two SH-LAV$^{=}$-constraints as follows:

$$t_1 : S(x,y) \rightarrow \exists m\ T(m,m,m) \wedge (y = 1 \rightarrow m = a)$$

$$t_2 : S(x,y) \rightarrow \exists m\ T(m,m,m) \wedge (y = 2 \rightarrow m = b)$$

We can see that t' and $\{t_1, t_2\}$ are both vfe for (I,J), but $\{t_1, t_2\}$ has larger size. In Appendix A, we have results that show that this type of optimizations may increase the size of a schema mapping but the increase in the size of the schema mappings is bounded by a constant-factor that depends only on the source and target schemas (assuming schemas are fixed). Altogether, this means that we are able to improve the runtime efficiency at the expense of increasing the cost of the solution by a fixed constant factor.

6.2 Experimental Evaluation

We now demonstrate that the optimized approximation algorithm is practically feasible in a real-world mapping scenario, and show the quality of the mappings returned by the algorithm is good.

6.2.1 Mapping Scenario. We use the mapping scenario derived from the OBDA (Ontology-Based Data Access) mappings in [26]. As part of the Ontop project, the authors of [26] developed an ontology for movies[2] and map the schema of the IMDB database to the ontology so as to populate the movie ontology with data from the IMDB database. The Ontop project contains a mapping consisting of a collection of OBDA constraints, and that were developed by UNIBZ students as part as a lab assignment and later improved by the research team. We use their OBDA constraints as the "ground truth" and compare the constraints returned by our approximation algorithm against these constraints to understand how well our algorithm performs.

6.2.2 Statistics of source and target schemas. The movie ontology uses a triple-based model: $\langle subject, predicate, object\rangle$. We turn every $\langle subject, predicate, object\rangle$ ontological concept into a binary fact $predicate(subjuect, object)$. In our experiment, we draw data examples from IMDb dataset (source) and Movie ontology (target). IMDB dataset contains twenty-one relations and eight of them are used in the mapping from IMDb to movie ontology (A full description of the IMDB schema can be found at [27], and the OBDA mappings

[2]see https://github.com/ontop/ontop/wiki/Example_MovieOntology for more details

that are constructed by hand can be found at [25]). There are sixty-eight authored OBDA constraints and therefore there are sixty-eight movie ontology concepts (i.e., sixty-eight target binary relations). A close examination reveals that these sixty-eight OBDA constraints can be categorized into following four types:

- **Type I**: Copying SH-LAV-constraints.
  E.g., name(id, name) $\rightarrow$ HasBirthName(id, name)
- **Type II**: Projection SH-LAV-constraints.
  E.g., company_name(id, name, country_code, IMDB_id, name_nf, name_sf) $\rightarrow$ compHasName(id, name)
- **Type III**: SH-LAV$^{=}$-constraints with one equality on the left-hand side.
  E.g., cast_info (id, person_id, movie_id, p_role_id, note, nr_order, role_id) $\wedge$ (role_id=8) $\rightarrow$ hasDirector(movie_id, person_id)
- **Type IV**: SH-LAV$^{=}$-constraints with two equalities on the left-hand side.
  E.g., movie_info (id, movie_id, info_type_id, info, note) $\wedge$ (info_type_id=3) $\wedge$ (info=Thriller) $\rightarrow$ SensibleThrilling(movie_id, info)

In our experiments, we have chosen some constraints in each category to test our algorithm. Concretely, we have chosen one copying constraint, one project constraint, and three constraints from Type III and Type IV (for Type III and IV, we choose those constraints whose source relations have different arities). We chose only one constraint from Type I and Type II because copying and projection are relatively simple transformations comparing with the transformations in Type III and Type IV. The statistics of selected IMDB relations and Movie ontology concepts are shown in Table 3.

Table 3. Statistics of source and target schemas

	IMDB	Movie ontology
number of relations	8	8
average arity	7.5	2
max. arity	11	2
min. arity	5	2

6.2.3 Adjusted mapping scenario. As mentioned, we use the OBDA constraints that were developed by human for the IMDB-to-Ontology mapping as ground truth. Hence, we can directly compare them with the constraints that are returned by our approximation algorithm. The IMDB database is populated with real data from IMDB.com and the size of the smallest table has about 219K records. Since our schema-mapping approximation algorithm is designed to run over small data examples and not over large datasets in general, we cannot simply execute our algorithm directly on the datasets obtained. Hence, in our experiments, we first generate small data examples from the real datasets before we apply our approximation algorithm and test the scalability of our algorithm by executing it on data examples of increasing sizes.

6.2.4 Data example generation. The purpose is to obtain a series of data examples of different sizes. We generate data examples with the following procedures. All OBDA constraints in the mapping specified in the Ontop project is of the form

$$S(\mathbf{x}) \wedge \alpha(\mathbf{x}) \rightarrow T(\mathbf{x})$$
$$\text{where } \alpha(\mathbf{x}) \text{ is of a collection of equalties } (x_1 = c_1) \wedge \cdots \wedge (x_n = c_n)$$

We first take an OBDA constraint t, and then randomly sample k (e.g., $k = 10$) facts, denoted by $I = \{f_1, f_2, \ldots, f_k\}$, from the S-relation, and we chase the k sampled facts with t (see [16] for more details about the chase procedure for data exchange) to generate a set J of T-facts which will be served as target facts. It follows that, (I, J) will be the data example generated. To obtain larger data examples, we simply increase the sampling size and repeat the above procedure. For generating a set of m data examples (used in Experiment 4), we simply run the same procedures m times to obtain m data examples, and these examples form a set E of data examples.

6.2.5 Experiments. Our goal is to understand the actual running time of the implementation in various settings, and the quality of constraints returned by the approximation algorithm on the IMDB-to-Ontology scenario. The approximation algorithm was implemented in Java, with PostgreSQL v9.3 as the underlying database engine. All the experiments were running on an Intel Core i7-4770 3.4GHz CPU Linux machine with 16GB memory. Experiments presented here take a single ground data example as input unless otherwise stated. We are interested in the following aspects:

(a) Compare the running time of the approximation approach with the exhaustive algorithm (i.e., the exact algorithm).
(b) Scalability of the optimized implementation with respect to:
  – data examples of different sizes.
  – different number of source relations and target relations.
(c) Running time of each of the three phases of the approximation algorithm (i.e., constraints-generation phase, validity-checking phase, and greedy phase).
(d) Compare the constraints produced by our approximation algorithm with the ground truth OBDA constraints.
(e) Understand the runtime efficiency of our algorithm on multiple data examples.

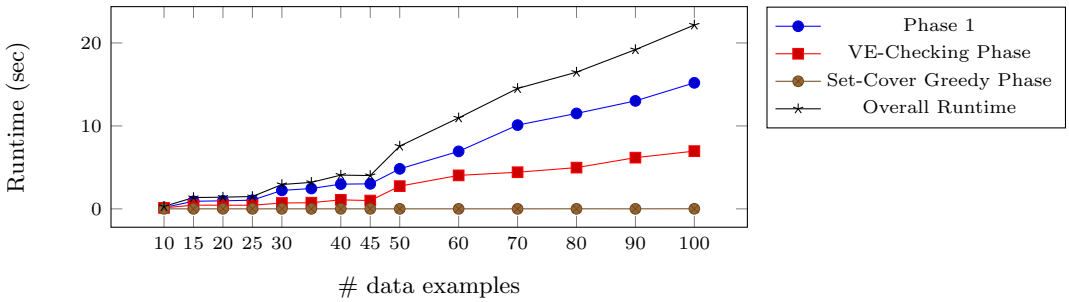
Experiment 1 - Approximation algorithm vs. exhaustive algorithm. We show that the exhaustive algorithm is not feasible because it runs out of memory even for a small input data example. The exhaustive algorithm is implemented in a straightforward way: after computing the set C of candidate valid and explaining constraints for a given data example, the exhaustive algorithm will then compute all subsets of C and then try to find a a minimum subset of C that is vfe for the given data example. Obviously, this is a $O(2^n)$ algorithm where n is the size of C. We run our approximation algorithm and the exhaustive algorithm on three small data examples of sizes 10, 15, and 20 (total number of source and target facts). The results are list in Table 4.

Clearly, as shown in Table 4, the exhaustive search performs much worse than the approximation algorithm in terms of running time. Table 4 shows that even for the case where the number of valid and explaining constraints is small, the exhaustive algorithm ran out of memory and therefore it is not feasible. On the other hand, the approximation algorithm is very efficient.

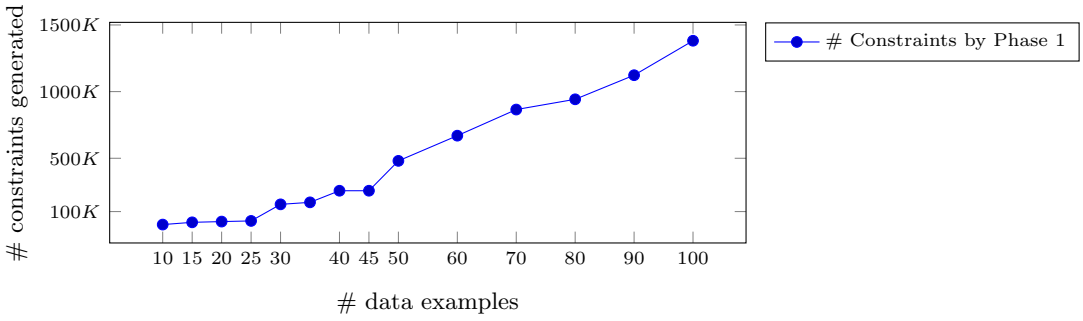
Table 4. Running time - approximation algorithm vs. exhaustive algorithm

Size of data examples	# candidate valid & explaining constraints	running time of approx. algorithm (seconds)	running time of exhaustive algorithm (seconds)
10	22	0.001	7.14
15	24	0.001	31.25
20	26	0.001	Out of memory

Experiment 2 - Scalability in terms of data examples of different sizes. In this experiment, the data examples used are composed of facts from two source relations and two target relations. The two source relations have arity five and seven respectively, and the two target relations are binary relations. These four relations are selected from two OBDA constraints in the ground truth solution (one from Type III and one from Type IV). We start our test with a small data example, and increase the sizes of data examples in the subsequent tests. Initially, the data example has six source facts from two source relations and four target facts from two target relations. In the last test, there are a total of 100 facts in the example tested (note that in each test, there are roughly 60% of the facts contained in the data examples are source facts, and roughly 40% are target facts). The overall running time is shown in Figure 2a. Moreover, the running time of each phase of our algorithm is also shown in the same figure.



(a) Overall running time with respect to data examples of different sizes



(b) Number of constraints generated in Phase 1

Fig. 2. Runtime efficiency and size of generated constraints

Figure 2a shows that our algorithm executes fast for data examples with smaller number of facts (e.g., less than six seconds for a data example of size forty), and is still efficient for large data examples (e.g., finished within twenty-four seconds for a data example of size 100). The approximation algorithm is a high polynomial degree algorithm. The running time shows that the optimizations applied in the implementation is capable of improving the efficiency of the algorithm. Another observation is that the greedy phase is extremely fast (in all tests, the greedy phase finished within a second). We also see that the constraints generation phase (i.e., Phase 1 of the algorithm) is the most time-consuming part of the algorithm. We did another experiment to record the number of constraints generated in Phase 1. The numbers are shown in Figure 2b. We can see that the runtime of Phase 1 and the number of constraints generated in Phase 1 are highly related. Moreover, the validity and explanation checking phase (i.e., VE-Checking phase) has similar runtime as Phase 1, this is because the runtime of VE-Checking phase is determined by the number of constraints generated in Phase 1. Therefore, we can conclude that, Phase 1 contributes the most to the overall runtime of our approximation algorithm.

Quality of results in Experiment 2. We did fourteen tests in Experiment 2, and we compared the constraints returned by the approximation algorithm with the ground truth OBDA constraints. It turned out that our approximation algorithm produced optimal schema mappings in all cases.

Experiment 3 - Scalability in terms of different numbers of relations. In each test of this experiment, the data examples are constructed by drawing five facts from each source instance and five facts from each target instance, and we keep increasing the number of source relations and target relations. We increase the number of source relations and target relations in turn in each subsequent test. In this experiment, only constraints of Type III and Type IV are involved. Initially, there are one source relation and one target relation involved (total 10 facts), and eventually there are five source relations and five target relations involved (total 100 facts).

Table 5. Running time with respect to different number of relations

Total # of relations	# of source relations	# of target relations	max. source schema arity	# of constraints after validity checking	Running time of the approximation algorithm (seconds)
2	1	1	5	16	0.069
3	2	1	7	18	0.11
4	2	2	7	1908	0.275
5	3	2	7	3664	0.166
6	3	3	7	5414	0.382
7	4	3	7	6284	0.346
8	4	4	7	9667	0.356
9	5	4	11	14301	10.175
10	5	5	11	38465	48.284

The results are listed in Table 5. In the first seven tests, our algorithm executes to completion under one second in all cases. However, the runtime became significantly slow in the cases when larger (arity of eleven) source relations are involved. Concretely, in the last two tests, we included facts from a source relation of arity eleven. In contrast, in the first seven tests, all relations involved have arity either five or seven. The results in Table 5 are

not surprising since our approximation algorithm has exponential complexity in the worst case when the schema is part of the input.

Quality of results in Experiment 3. We did nine tests in this experiment, and the approximation algorithm produced the same OBDA constraints that are specified in the ground truth in eight of the tests. There is one test in which our approximation algorithm produced a constraint that is different from the corresponding ground truth OBDA constraint. Concretely, the constraint t shown below is the one included in the ground truth, and the constraint t' is the one returned by our approximation algorithm. The constraint t' does not have the equality (info_type_id=3) showing on its left-hand side.

$$t: \text{movie\_info (id, movie\_id, info\_type\_id, info, note)} \wedge \text{(info\_type\_id=3)} \wedge \text{(info=Thriller)} \rightarrow \text{SensibleThrilling(movie\_id, info)}$$

$$t': \text{movie\_info (id, movie\_id, info\_type\_id, info, note)} \wedge \text{(info=Thriller)} \rightarrow \text{SensibleThrilling(movie\_id, info)}$$

We looked into the full IMDB dataset and verified that our approximation algorithm made the right decision. The equality (`info_type_id` = 3) is redundant, since every tuple p in the `movie_info` table that has value "3" in attribute `info_type_id`, the tuple p also has value "`Thriller`" in attribute `info`. In other words, t and t' agree on validity and explanation, but t' has smaller size. In other words, our algorithm derived a constraint that is in fact better than the constraint that corresponds to our ground truth.

Experiment 4 - runtime efficiency on multiple data examples. In this experiment, we try to understand the runtime efficiency of our approximation algorithm on multiple data examples. In this experiment, the input to our approximation algorithm is a set E of data examples, and we test our algorithm with respect to different sizes of E (i.e., test with respect to different numbers of data examples in E). First, we selected four OBDA constraints (which are exactly those constraints shown in Section 6.2.2 as example constraints) to generate our data examples. We then randomly choose five facts from each source relation specified in the four OBDA constraints, and obtain target instance by chasing the randomly selected facts with the four OBDA constraints. To get $m \geq 2$ examples, we simply repeat the same procedure m times. We have tested our approximation algorithm on ten different sets of data examples, and the results are listed in Figure 3. We can see that the overall runtime grows linearly, and the runtime of Phase 1 has similar behaviour. However, the runtime of the validity and explanation checking phase grows a bit faster. Note that the data examples used in the experiments have comparable sizes due to the way we generate them. We run our approximation algorithm on each individual data example of the input set E of data examples and obtain a collection C of constraints. We then check the validity and explanation of C on every data example in E, and keep those that are valid and explaining for every data example in E. Hence, the runtime of VE-checking phase grows quadratically in the number of examples in the input set of data examples. On the other hand, the runtime of Phase 1 grows linearly in the size of input set of data examples because the algorithm generates constraints for each data example independently. However, as the number of examples grows, the runtime of our approximation algorithm will be dominated by the runtime of VE-checking phase, since it grows quadratically. Notice that, in this set of experiments, we no longer have the ground truth constraints since we now have multiple data examples as input (The ground truth OBDA constraints do not need to be vfe when

there are multiple data examples). Therefore, we do not measure the quality of results in this experiment.

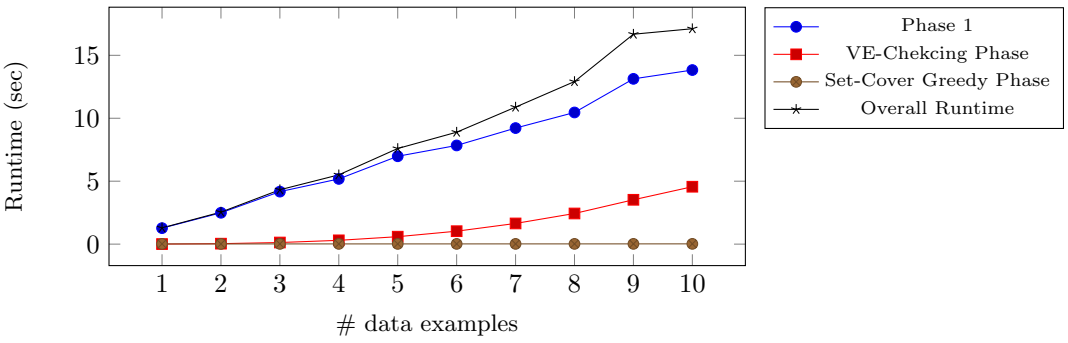


Fig. 3. Runtime with respect to multiple data examples

Discussion. We have conducted experiments to evaluate our optimized implementation on a real mapping scenario. Our experimental results reveal that the optimized approximation algorithm is efficient for this mapping scenario. Moreover, the SH-LAV$^{=}$-schema constraints returned by the approximation algorithm for the case where the input consists of a single data example are as good as authored OBDA constraints which we take as ground truths. In fact, our approximation algorithm produced a constraint that is better than the authored OBDA constraint in one of the data examples. We also validated that our approximation algorithm is efficient when the input consists of multiple data examples. Our experimental results show that the performance of the algorithm is dominated by the arity of the schemas; as the schemas are larger, our algorithm runs slower. In fact, if schemas are part of the input, our approximation algorithm runs in exponential time, and that is inevitable.

7 CONCLUDING REMARKS

We investigated the derivation of an optimal schema mapping from a set of data examples and focused on the approximation properties of this optimization problem. We considered several different sublanguages of the language of GLAV schema mappings and delineated the boundary as regards good approximation properties. To this effect, we established negative results about the existence of approximation algorithms for GAV and GLAV schema mappings, but also showed that the collection of SH-LAV schema mappings is the largest subclass of GLAV mappings for which we can obtain a positive approximation result. These results pave the way for leveraging further the rich area of approximation algorithms and applying it to schema-mapping discovery.

We enchanced our approximation algorithm with heuristic rules, implemented it, and evaluated it on a real-world schema mapping scenario. Our experimental evaluation suggests that this approach may indeed be applied in practice to construct a near-optimal schema mapping from a set of data examples.

An important question that remains to be answered is the existence of a polynomial time $\mathcal{H}(n)$-approximation algorithms for computing near-optimal LAV$^{=,\neq}$ schema mappings, as well as near optimal SH-LAV$^{=,\neq}$ schema mappings.

REFERENCES

[1] Bogdan Alexe, Laura Chiticariu, Renée J. Miller, and Wang Chiew Tan. 2008. Muse: Mapping Understanding and deSign by Example. In *ICDE*. 10–19.

[2] Bogdan Alexe, Wang-Chiew Tan, and Yannis Velegrakis. 2008. STBenchmark: Towards a Benchmark for Mapping Systems. *Proc. VLDB Endow.* 1, 1 (Aug. 2008), 230–244. DOI:http://dx.doi.org/10.14778/1453856.1453886

[3] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang-Chiew Tan. 2011. Characterizing Schema Mappings via Data Examples. *ACM Trans. Database Syst.* 36, 4, Article 23 (Dec. 2011), 48 pages. DOI:http://dx.doi.org/10.1145/2043652.2043656

[4] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. 2011. Designing and refining schema mappings via data examples. In *SIGMOD*. 133–144.

[5] Noga Alon, Dana Moshkovitz, and Shmuel Safra. 2006. Algorithmic Construction of Sets for K-restrictions. *ACM Trans. Algorithms* 2, 2 (April 2006), 153–177. DOI:http://dx.doi.org/10.1145/1150334.1150336

[6] Patricia C. Arocena, Boris Glavic, Radu Ciucanu, and Renée J. Miller. 2015. The iBench Integration Metadata Generator. *Proc. VLDB Endow.* 9, 3 (Nov. 2015), 108–119. DOI:http://dx.doi.org/10.14778/2850583.2850586

[7] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. 2009. The DL-Lite family and relations. *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH (JAIR)* 36 (2009), 1–69.

[8] Pablo Barceló. 2009. Logical foundations of relational data exchange. *SIGMOD Record* 38, 1 (2009), 49–58.

[9] A. Bonifati, E. Q. Chang, T. Ho, V. S. Lakshmanan, and R. Pottinger. 2005. HePToX: Marrying XML and Heterogeneity in Your P2P Databases. In *VLDB*. 1267–1270.

[10] Balder ten Cate, Víctor Dalmau, and Phokion G. Kolaitis. 2012. Learning Schema Mappings. In *Proceedings of ICDT*. ACM, New York, NY, USA, 182–195. DOI:http://dx.doi.org/10.1145/2274576.2274596

[11] Balder ten Cate, Víctor Dalmau, and Phokion G. Kolaitis. 2013. Learning Schema Mappings. *ACM Trans. Database Syst.* 38, 4, Article 28 (Dec. 2013), 31 pages. DOI:http://dx.doi.org/10.1145/2539032.2539035

[12] Balder ten Cate, Richard Halpert, and Phokion Kolaitis. 2016. Exchange-Repairs: Managing Inconsistency in Data Exchange. *Journal of Data Semantics* 5, 2 (2016), 77–97.

[13] Balder ten Cate, Phokion Kolaitis, Kun Qian, and Wang-Chiew Tan. 2015. Approximation Algorithms for Schema-Mapping Discovery from Data Examples. In *AMW 2015*.

[14] V. Chvtal. 1979. A greedy heuristic for the set covering problem. *Math. Oper. Res.* 4 (1979), 233–235.

[15] Anish Das Sarma, Aditya G. Parameswaran, Hector Garcia-Molina, and Jennifer Widom. 2010. Synthesizing view definitions from data. In *ICDT*. 89–103.

[16] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. 2005. Data Exchange: Semantics and Query Answering. *TCS* 336, 1 (2005), 89–124.

[17] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. 2005. Composing Schema Mappings: Second-order Dependencies to the Rescue. *ACM Transactions on Database Systems (TODS)* 30, 4 (2005), 994–1055.

[18] George H.L. Fletcher and Catharine M. Wyss. 2009. Towards a general framework for effective solutions to the data mapping problem. *Journal on Data Semantics* XIV (2009).

[19] Georg Gottlob and Pierre Senellart. 2010. Schema mapping discovery from data instances. *JACM* 57, 2 (2010).

[20] Wolfang Gutjahr, Emo Welzl, and Gerhart Woeginger. 1992. Polynomial graph-colorings. *Discrete Appl. Math.* 35 (1992), 29–46.

[21] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. 2005. Clio Grows Up: From Research Prototype to Industrial Tool. In *ACM SIGMOD*. 805–810.

[22] David S. Johnson. 1973. Approximation Algorithms for Combinatorial Problems. In *Proceedings of STOC*. ACM, New York, NY, USA, 38–49. DOI:http://dx.doi.org/10.1145/800125.804034

[23] Richard M. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (Eds.). Plenum, 85–103.

[24] P. G. Kolaitis. 2005. Schema Mappings, Data Exchange, and Metadata Management.. In *ACM PODS*. 61–75.

[25] KRDB. 2013. OBDA mappings from IMDB to Ontology. (2013). https://raw.githubusercontent.com/wiki/ontop/ontop/attachments/Example_MovieOntology/movieontology.obda

[26] KRDB. 2013. ontop project. (2013). https://github.com/ontop/ontop/wiki/Example_MovieOntology
[27] KRDB. 2013. SQL script to generate the schema only IMDB database. (2013). https://raw.githubusercontent.com/wiki/ontop/ontop/attachments/Example_MovieOntology/imdbpg.sql
[28] M. Lenzerini. 2002. Data Integration: A Theoretical Perspective. In *ACM PODS*. 233–246.
[29] Heikki Mannila and Kari-Jouko Räihä. 1989. Automatic Generation of Test Data for Relational Queries. *JCSS* 38, 2 (1989), 240–258.
[30] R. J. Miller, L. M. Haas, and M. A. Hernández. 2000. Schema Mapping as Query Discovery. In *International Conference on Very Large Data Bases (VLDB)*. 77–88.
[31] Christopher Olston, Shubham Chopra, and Utkarsh Srivastava. 2009. Generating example data for dataflow programs. In *ACM SIGMOD*. 245–256.
[32] Christos Papadimitriou and Mihalis Yannakakis. 1988. Optimization, Approximation, and Complexity Classes. In *Proceedings of STOC*. ACM, New York, NY, USA, 229–234. DOI:http://dx.doi.org/10.1145/62212.62233
[33] Ran Raz and Shmuel Safra. 1997. A Sub-constant Error-probability Low-degree Test, and a Sub-constant Error-probability PCP Characterization of NP. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing (STOC '97)*. ACM, New York, NY, USA, 475–484. DOI: http://dx.doi.org/10.1145/258533.258641
[34] L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. 2001. Data-Driven Understanding and Refinement of Schema Mappings. In *ACM SIGMOD*. 485–496.

APPENDIX

A DETAILED DISCUSSION OF OPTIMIZED IMPLEMENTATION

This section is dedicated to the presentation of details concerning the optimized implementation of the approximation algorithm. The approximation algorithm described in Figure 1 is composed of three major phases:

(1) Constraints-generation phase (Line 1 of Algorithm GreedySHLAV$^{=}$($\mathbf{S}, \mathbf{T}$))
(2) Validity-checking phase (Line 1 of Algorithm GreedySHLAV$^{=}$($\mathbf{S}, \mathbf{T}$))
(3) Greedy phase (The while loop in Algorithm GreedySHLAV$^{=}$($\mathbf{S}, \mathbf{T}$))

The first two steps of the algorithm are actually consolidated into one dense step in the algorithm depicted in Algorithm GreedySHLAV$^{=}$($\mathbf{S}, \mathbf{T}$). The greedy phase of the approximation algorithm was established based on the connection to the greedy algorithm of Weighted Set-Cover problem. The greedy phase is the least expensive part of the algorithm based on our experimental evaluation, and we do not need extra efforts to optimize this phase. The main focus of this section is to discuss the optimizations used in the first step of the approximation algorithm, i.e., the constraints-generation phase. In the main part of the paper, we have mentioned that there are two kinds of optimizations applied in the implementation, which are (1) zero-amplification optimizations and (2) constant-amplification optimizations. Recall that, we say an optimization is zero-amplification if its application never blows up the approximation bound; we say an optimization is constant-amplification if its application would blow up the approximation bound within some fixed constant factor. We have applied several zero-amplification optimizations and one constant-amplification optimization in the prototype implementation. To better understand the applied optimizations, we need the introduction of following definitions and lemmas:

Definition A.1 (VE-Equivalence). Given a ground data example (I, J) over source schema $\mathbf{S}$ and target schema $\mathbf{T}$. Let t and t' be two SH-LAV$^{=}$-constraints over $\mathbf{S}$ and $\mathbf{T}$, and (I, J). We say that t and t' are *VE-Equivalent* if the following conditions hold:

(a) The two constraints t and t' agree on their validity with respect to (I, J).
(b) The two constraints t and t' explain the same set of target facts in J. □

Definition A.2 (Redundant Equality). Given a data example (I, J) over source schema $\mathbf{S}$ and target schema $\mathbf{T}$, and a SHLAV$^=$-constraint c of the form (Assuming that c contains relational atoms over $\mathbf{S}$ and $\mathbf{T}$ only, and all constants in c belongs to the active domain of I only).

$$\forall x \forall y \left(S(x, y) \wedge \alpha \rightarrow \exists m T(y, m) \wedge \beta \right),$$

where α is a collection of equalities with each of the form $(x = p)$ and β is a collection of conditional equalities with each of the form $(\wedge_i (x_i = c_i) \rightarrow (m_j = c_j))$. We say an equality E, which occurs in α or in the premise of a conditional equality in β, is redundant if the SH-LAV$^=$-constraint c' that is obtained from c by removing E is VE-Equivalent to c. □

Lemma A.3. *The GreedySHLAV$^=$ ($\boldsymbol{S}$, $\boldsymbol{T}$) approximation algorithm never produces a schema mapping that has a constraint that contains a redundant equality.*

The proof of the Lemma A.3 is immediately followed by the definition of the GreedySHLAV$^=$ ($\mathbf{S}$, $\mathbf{T}$) approximation algorithm. Before presenting the optimized algorithm that implements various optimizations, we need one more definition, which we call *Key Constant*.

Definition A.4 (Key Constant). Given a ground data example (I, J) where I contains a set $F = \{F_1, \ldots, F_m\}$ of facts. We say a constant $c \in adom(I)$ is a key constant for I if c appears in only one fact in I. □

The pseudo-code of generating candidate constraints for an input data example (I, J) is shown in Function collectCandidateConstraints() described in Figure 4 .

The Function collectCandidateConstraints() takes as input a single source fact s from I, a single target fact from t J, a *GlobalThreshold* that is an integer, and a set of key constants for I. It outputs a set of candidate SH-LAV$^=$-constraints that will be consumed by the validity-checking phase of the approximation algorithm. The main function, i.e., collectCandidateConstraints(), is called on every pair of source fact and target fact. The main function also uses a subroutine function, i.e., computeRightRepairs(), which is shown in Figure 5. The Function computeRightRepairs is a function that computes the conditional equalities to be added to the right-hand side of a partially constructed SH-LAV$^=$-constraint. We have included a running example in the presentation of the main function so as to make it self-explanatory. The main idea of the function is that we construct candidate SH-LAV$^=$-constraints based on every pair of source and target facts. Intuitively, every target fact can be explained by a source fact through a SH-LAV$^=$-constraint. When we construct the candidate constraints over a pair of source and target fact, we look into the constants occurring both in the source fact and in the target fact. We first construct a collection of constraints in the base language (i.e., SH-LAV) by analyzing the positions of constants occurring in the source fact and target fact. Then we add equalities and conditional equalities to those SH-LAV constraints based on the input pair of source fact and target fact.

Theorem A.5. *Given a ground data example (I, J). Let A be the optimized approximation algorithm that uses Function collectCandidateConstraints() to produce candidate SH-LAV$^=$-constraints, and let B be the approximation algorithm without optimizations. Let $\mathcal{M}_A$ and $\mathcal{M}_B$ be the SH-LAV$^=$-mappings returned by running A and B over (I, J) respectively. Then the sizes of $\mathcal{M}_A$ and $\mathcal{M}_B$ are same up to some fixed constant number.*

To prove Theorem A.5, it suffices to show that the optimizations applied in the implementation are either zero-amplification or constant-amplification. The rest of the section is dedicated to prove that the optimizations used in this implementation are indeed the case. We first present three optimizations and show that they are zero-amplification.

input : (I, J): a ground data example
t: a single target fact
GlobalThreshold: the cost of adding all ground facts
keyConstants: the set of key constants
output: A set of SH-LAV$^{=}$-constraints that are valid for (I, J)

```
candidate ← ∅
for every source fact s ∈ I and t ∈ J do
    // We will use s=S(a,b), and t=T(a,a) as a running example
    rightAtom ← Scan each constant in s and construct an atom such that, each constant is
    replaced by a distinct variable. (e.g., if s is S(a,b), then leftAtom is S(x,y); if s is S(a,a), then
    leftAtom is S(x,x) )
    rightAtomList ← compute all possible r combinations, where r is the arity of t, of variables
    occurring in leftAtom as well as existentially quantified variables. For each obtained
    combination, create an atom and add it to rightAtomList. (as for the running example the
    rightAtomList will be {∃nT(x,y,n), ∃m∃pT(m,y,p), ∃n∃pT(x,n,p), ∃m∃n∃pT(m,n,p)}
    equalitySet ← compute a set of equalities according to s and leftAtom in the following way:
    supposing s is of the form S(c_0,...,c_n), and leftAtom is of the form S(x_1,...,x_n), then we
    have equalitySet = {(x_k = c_k | k = 0,1,...,n}. (As for the running example, equalitySet =
    {(x = a),(y = b)}).
    leftRepairs ← compute all subsets of equalitySet and each subset become a member of
    leftRepairs. (For the running example,
    leftRepairs={{},{(x = a)},{(y = b)},{(x = a),(y = b)}}
    for repair in leftRepair do
        for rightAtom in rightAtomList do
            shlav ← construct a partial constraint of the form: (leftAtom + repair )→ rightAtom.
            if shlav is a LAV∩GAV-constraint then
                if size(shlav)< GlobalThreshold then
                    candidate ← candidate ∪ {shlav }
                end
            end
            else
                RightRepairList ← computeRightRepairs(shlav, equalitySet, s, t)
                for rightRepair in RightRepairList do
                    shlav ← update shlav by adding rightRepair to its right-hand side
                    if size(shlav) < GlobalThreshold then
                        candidate ← candidate ∪ {shlav }
                    end
                end
            end
        end
    end
end
for constraint t ∈ candidate do
    if t is not valid for (I, J) then
        candidate ← candidate \{t}. // check validity by chase
    end
end
Return candidate
```

Fig. 4. Function collectCandidateConstraints

(1) *Global threshold rule (zero-amplification).*
There is a simple fact that for a single ground data example (I, J), there is a trivial vfe schema mapping for (I, J) that is composed of all facts in J. We use the size of that trivial schema mapping as a threshold such that, whenever we are constructing a candidate SH-LAV$^{=}$-constraint and its size is greater than the threshold, we stop the construction process. This optimization is applied in various places in Function collectCandidateConstraints() and Function computeRightRepairs() . It is easy to see that this is a zero-amplification optimization

(2) *Key constants rule (zero-amplification).*
The definition of key constants suggest an nice property. Concretely, suppose that there are multiple equalities in the left-hand side of a SH-LAV$^{=}$-constraint (or in the premise part of a conditional equality in the right-hand side). If there is one equality E that involves a key constant, then we can ignore the rest of equalities and just keep E alone. In other words, E makes all other co-occurring equalities be redundant. This is very useful in practice and can save a lot of time-consuming process that would produce a constraint with redundant equalities. This optimization is applied in Step 5 of Function collectCandidateConstraints() , and the inner for loop in Step 9 of Function computeRightRepairs() . (Remark: this optimization can be extended to a concept named *Key Equality* that takes into account the occurrences of a constant both in rows and columns. For instance, suppose that the input source instance contains two facts $S(a, a, b), S(c, c, b)$, and the leftAtom is $S(x, y, z)$. Then $(x = a)$ and $(y = a)$ are both key equalities). The optimization is zero-amplification because Lemma A.3.

(3) *Realized equality rule (zero-amplification).*
Both in Functions collectCandidateConstraints() and computeRightRepairs() , we have to compute a lot of candidate equalities that are going to be added to candidate SH-LAV$^{=}$-constraints. It is easy to see that, a simple brute-force algorithm will construct a lot of unrealized equalities. For example, suppose that we are at the Step 4 of Function collectCandidateConstraints() , where the source fact and leftAtom in consideration are $S(a, b)$ and $S(x, y)$. A brute-force algorithm would construct equalities like $(x = b)$ and $(y = a)$, and those equalities will never be realized and thus are actually cancelling the constraints that containing them. In our implementation, we address this problem by detecting and removing unrealized equalities in the constraints-generation phase. This optimization is applied in Step 5 of Function collectCandidateConstraints() , and Step 4 and 5 of Function computeRightRepairs() . The optimization is zero-amplification because Lemma A.3.

The optimizations discussed so far are all zero-amplification optimizations and obviously never blow up the approximation bound. Next, we discuss the constant-amplification optimization that is applied in the implementation. The only one constant-amplification optimization in our implementation is established based on the fact that the main function, i.e., collectCandidateConstraints(), is called on every pair of source and target facts. To better understand why this is an implicit optimization that blows up the approximation bound, the following example would be helpful. Suppose that we have following input data

input : *shlav*: a partially constructed SH-LAV constraint:
equalitySet: from collectCandidateConstraints()
s: source fact from collectCandidateConstraints()
t: target fact from collectCanddiateConstraints()

output : A RightRepairList in which each element is a group of conditional equalities

1. *premiseList* $\leftarrow \emptyset$
2. *consequenceList* $\leftarrow \emptyset$
3. *numExistentialVars* $\leftarrow$ number of existential variables occuring in *shlav*
4. **for** *existVar* $_k$ *in* *shlav* **do**
   // *existVar* $_k$ is the $\exists$-quantified variable at the k-position in *shlav*
   // c_k is the constant occurring at the k-th position of t
   consequenceList $\leftarrow$ *consequenceList* $\cup \{(existVar_k = c_k)\}$
   end
5. *leftOverEqualities* $\leftarrow$ *equalitySet* $\setminus$ { leftReparis in *shlav* }
6. *premiseList* $\leftarrow$ compute all possible subsets of *leftOverEqualities*, and each obtained subset become a member of *premiseList*
7. *premiseCombinations* $\leftarrow$ compute all possible *numExistentialVars*-elements combinations of items in *premiseList*
8. *RightRepairList* $\leftarrow \emptyset$
9. **for** *premiseGroup in* *premiseList* **do**
   rightRepair $\leftarrow \emptyset$
   for $i \leftarrow 1$ **to** *numExistentialVars* **do**
   premise $\leftarrow$ the i-th element in *premiseGroup*
   if *premise has an equality e that involves* *keyConstants* **then**
   premise $\leftarrow e$
   end
   conditionalEquality $\leftarrow$ create a new conditional equality of the form:
   $$premise \rightarrow consequenceList.At(i)$$
   rightRepair $\leftarrow$ *conditionalEquality*
   end
   end
10. **Return** *RightRepairList*

Fig. 5. Function computeRightRepairs

example (I, J)

$$I = \{S(a), S(b)\},\ J = \{T(a), T(b)\}$$

It follows that, we will call the main function four times, one for each of the following pairs: $\langle S(a), T(a)\rangle$, $\langle S(a), T(b)\rangle$, $\langle S(b), T(a)\rangle$, and $\langle S(b), T(b)\rangle$. Then the algorithm will compute repairs for the four pairs independently. The issue with this is that, the algorithm would never produce a repair such as

$$S(x) \rightarrow \exists m T(m) \wedge (x = a \rightarrow m = a) \wedge (x = b \rightarrow m = b),$$

because the approximation algorithm cannot have both $T(a)$ and $T(b)$ as input at the same time. Instead, the approximation algorithm will produce the following two constraints,

one for the pair $\langle S(a), T(a)\rangle$, and one for the pair $\langle S(a), T(b)\rangle$.

$$S(x) \to \exists m T(m) \wedge (x = a \to m = a),\ \ S(x) \to \exists m T(m) \wedge (x = b \to m = b).$$

Although the above blow-up may happen, we next show that the blow-up is bounded by a fixed constant factor. We first need following definition and lemma.

Definition A.6 (SPLIT(t)). Given a ground data example (I, J) that conforms to some fixed source and target schemas, and a SHLAV$^{=}$-constraint t of the form: $\forall \mathbf{x}, \mathbf{y}\ S(\mathbf{x}, \mathbf{y}) \wedge \alpha \to \exists \mathbf{m} T(\mathbf{y}, \mathbf{m}) \wedge \beta$, where α is a (possibly empty) collection of equalities and β is a (possibly empty) collection of conditional equalities with each of the form $(\wedge_i (x_i = c_i) \to (m_j = c_j))$. We define SPLIT($t$) be a set of SHLAV$^{=}$-constraints obtained from t by following steps:

(1) Partition β into $|\mathbf{m}|$ groups, where $|\mathbf{m}|$ is the number of existential variables in t. Concretely, two conditional equalities are in the same group if their consequence parts involve the same existential variable.
(2) Suppose that the $|\mathbf{m}|$ groups are $G_1, G_2, \ldots, G_m$, and each group has cardinality $C_1, C_2, \ldots, C_m$. Then we compute all $|\mathbf{m}|$-element combinations of the conditional equalities from the $|\mathbf{m}|$ groups such that, each combination contains exactly one conditional equality from each group. This process will end up with $C_1 \times C_2 \times \cdots \times C_m$ combinations.
(3) For each combination c obtained in Step 2, add to SPLIT(t) a new constraint as follows

$$\forall \mathbf{x} \forall \mathbf{y}\ \ (S(\mathbf{x}, \mathbf{y}) \wedge \alpha \to \exists \mathbf{m} T(\mathbf{y}, \mathbf{m}) \wedge c)\,.$$

□

Lemma A.7. *Fix a source schema* $\boldsymbol{S}$ *and a target schema* $\boldsymbol{T}$*. Given a ground data example* (I, J) *that conforms to some* $\boldsymbol{S}$ *and* $\boldsymbol{T}$*. Let* t *be an arbitrary SHLAV*$^{=}$*-constraint that is constructed over* (I, J)*. Then SPLIT(t) and* t *are VE-Equivalent with respect to* (I, J)*. Moreover, the size of SPLIT(t) is bounded by a constant factor in the size of* t*.*

Proof. First if t is a LAV and GAV constraint, the lemma holds trivially. If t contains existential variables, then the proof of the lemma is the following.

VE-Equivalence. Suppose that t is valid in (I, J), which means for every homomorphism h from the left-hand side of t to I, h can be extended to a homomorphism h' from the right-hand side of t to J. By construction of SPLIT(t), we know that h is also a homomorphism from the left-hand side of every constraint in SPLIT(t) to I, because all constraints in SPLIT(t) have the same atom in the left-hand sides, which coincides with the left-hand side of t. It follows that h' is also a homomorphism from the right-hand side of every constraint in SPLIT(t) to J. Suppose that t is invalid in (I, J), it means there is a homomorphism h from the left-hand side of t to I, and that cannot be extended to another homomorphism from the right-hand side of t to J. It follows that, h is also a homomorphism from the left-hand side of every constraint in SPLIT(t) to I by the same argument above. We need to show that h cannot be extended to a homomorphism from the right-hand side of some constraint in SPLIT(t) to J. We prove it by contradiction, suppose that h can be extended to a homomorphism h' from the right-hand side of every constraint in SPLIT(t) to J. It is easy to see that, by construction of SPLIT(t), h' is also a homomorphism from the right-hand side of t to J, which is a contradiction to the assumption. Therefore, the VE-equivalence is proved.

constant-factor bound. Since the source and target schemas are fixed, the number of existential variables in t is fixed. Let *sArity* be the maximum arity of source schema $\mathbf{S}$, and

let $tArity$ be the maximum arity of the target schema $\mathbf{T}$. Then the size of SPLIT(t) is at most $|C| * size(t)$, where $|C|$ is bounded by $O(sArity^{tArity})$, i.e., a constant-factor bound. The lemma is proved. □

LEMMA A.8. *Fix a source schema $\boldsymbol{S}$ and a target schema $\boldsymbol{T}$. Given a single ground data example (I, J) that conforms to $\boldsymbol{S}$ and $\boldsymbol{T}$. Let A be the original approximation algorithm specified in Figure 1, and let B be the optimized algorithm that implements Line 1 of Algorithm A by making use of Function collectCandidateConstraints() . Let $\mathcal{M}$ (resp. $\mathcal{M}'$) be the SH-LAV$^{=}$-schema mapping returned by running A (resp. B) on (I, J). Then we have that $size(\mathcal{M}') \leq c \cdot size(\mathcal{M})$ where c is some fixed constant.*

PROOF. (sketch) Given a ground data example (I, J) where $I = \{S_1, \ldots, S_n\}$ and $J = \{R_1, \ldots, R_m\}$. Let t be a SH-LAV$^{=}$-constraint that is valid and fully explaining for (I, J). If we run collectCandidateConstraints() on over every pair (S_k, R_p) of source and target facts, where $S_k \in S$ and $R_p \in J'$, and obtained a collection C of SH-LAV$^{=}$-constraints, then every constraint in SPLIT(t) is also contained in C. By Lemma A.8 we know that SPLIT(t) and t is VE-Equivalent with respect to (I, J). Note that SPLIT(t) is contained in C which will be the input of the second step of proposed approximation algorithm (i.e., SET-COVER-like greedy algorithm), therefore in the worst case the proposed approximation algorithm will output vfe SH-LAV$^{=}$-schema mapping $\mathcal{M}$ for (I, J). It follows that the size of $\mathcal{M}$ is at most the size of SPLIT(t), therefore, it would blow up the approximation bound within some fixed constant factor. □

PROOF OF THEOREM A.5. (sketch) As discussed earlier, the zero-amplification optimizations never blow up the approximation bound of the original approximation algorithm. Lemma A.8 shows that, in the worst case, the constant-amplification optimization blows up the approximation bound within some fixed constant factor. □

Received December 2015; revised XXX 2016; accepted XXX