On Concise Set of Relative Candidate Keys

Shaoxu Song[†] Lei Chen[‡] Hong Cheng[§]

[†]KLiss, MoE; TNList; School of Software, Tsinghua University, China sxsong@tsinghua.edu.cn

[‡]The Hong Kong University of Science and Technology, China leichen@cse.ust.hk

[§]The Chinese University of Hong Kong, China hcheng@se.cuhk.edu.hk

ABSTRACT

Matching keys, specifying *what attributes to compare* and *how to compare them* for identifying the same real-world entities, are found to be useful in applications like record matching, blocking and windowing [7]. Owing to the complex redundant semantics among matching keys, capturing a proper set of matching keys is highly non-trivial. Analogous to minimal/candidate keys w.r.t. functional dependencies, relative candidate keys (RCKs [7], with a minimal number of compared attributes, see a more formal definition in Section 2) can clear up redundant semantics w.r.t. "what attributes to compare". However, we note that redundancy issues may still exist among RCKs on the same attributes about "how to compare them". In this paper, we propose to find a concise set of matching keys, which has *less redundancy* and can still meet the requirements on coverage and validity. Specifically, we study approximation algorithms to efficiently discover a near optimal set. To ensure the quality of matching keys, the returned results are guaranteed to be RCKs (minimal on compared attributes), and most importantly, minimal w.r.t. distance restrictions (i.e., redundancy free w.r.t. "how to compare the attributes"). The experimental evaluation demonstrates that our concise RCK set is more effective than the existing RCK choosing method. Moreover, the proposed pruning methods show up to 2 orders of magnitude improvement w.r.t. time costs on concise RCK set discovery.

1. INTRODUCTION

For matching records that denote the same real-world entities, a variety of approaches have been proposed, such as probabilistic matching [18], learning-based [2], distance-based [4], rule-based [11] (see [5] for a survey). As indicated by Fan et al. [7], no matter what approaches to use, it is essential to decide "what attributes to compare" and "how to compare them", known as *matching keys*. While matching keys can typically assure high matching accuracy, the determination and tuning of such matching rules is highly non-trivial, often requires extremely high manual effort from the human experts [5]. Recently, great efforts have been made on enriching matching rules by reasoning over a given set of matching keys [7, 6]. Owing to the existence of possibly redundant semantics (as illustrated in the following Example 1), analogous to conventional keys, a special group of matching keys are concerned where the number of compared attributes is minimized, namely *relative candidate keys* (RCKs) [7]. However, redundancy issues exist not only w.r.t. "what attributes to compare", but also in "how to compare them". We note that the distance comparisons could be redundant in different RCKs on the same attributes as shown in the following example.

Proceedings of the VLDB Endowment, Vol. 7, No. 12

Table 1: An example instance of staff

ssn	name	address	department	
234***	Jason Smith	Mark Road	Social Science	t_1
2****3	J Smith	Mark Rd	Social Science	t_2
862***	W J Smith	Park St	Social Science	t_3
862***	Will J Smith	Park Street	Social Science	t_4
0****5	C Green	Mark Road	Computing	t_5
0****5	C Green	Mark Rd	Computing	t_6

Example 1. *Consider a relation for collecting staff information in Table 1. Since different digits of* ssn *are hidden (denoted by *) for privacy issues from various data sources, we need to determine whether 234*** and 2****3 denote the same* ssn *of a staff. Let*

$$\psi_1 : \quad (\mathsf{name}, \mathsf{address} \parallel [0,4], [0,2])$$

be a matching key relative to ssn *declared on attributes* name *and* address*, where* $[0,4]$ *and* $[0,2]$ *denote the restrictions of edit distances*[1] *on attributes* name *and* address*, respectively. It states that for any tuples* t_i, t_j *in a relation instance of* staff*, if their distance on attribute* name *is in the range of* $[0,4]$*, i.e.,* ≥ 0 *and* ≤ 4*, and the distance on* address *is in* $[0,2]$*, their* ssn *must be identified.*

*Consequently, the identification of 234*** in* t_1 *and 2****3 in* t_2 *in Table 1 can be implied, since they have* name *distance equal to* 4 *and* address *distance equal to* 2*, in the range of* $[0,4]$ *and* $[0,2]$*, respectively.*

Similar to the demand of minimal keys, redundancy exists among matching keys. A matching key ψ *is said* redundant *w.r.t. a relation if all the tuple pairs that can be identified by*

[1]Our proposed techniques below are independent to the selection of similarity/distance metrics (refer to [19] for selection of best similarity functions). Without loss of generality, we use edit distance by default in the following examples.

the matching key ψ can also be identified by another matching key ψ'. For example, given ψ_1, the following ψ_2 with additional restrictions on department *is unnecessary.*

$$\psi_2 : \quad (\mathsf{name}, \mathsf{address}, \mathsf{department} \parallel [0,4], [0,2], [0,0])$$

For ψ_1, since the number of compared attributes is minimized, i.e., not able to remove an attribute such that the remaining ones can still identify ssn*, this ψ_1 is an* RCK.

Owing to the presence of various distance restrictions, matching keys on the same attributes could be redundant as well. For example, the following ψ_3 with $[0,0]$ on name *overlaps with ψ_1 having $[0,4]$.*

$$\psi_3 : \quad (\mathsf{name}, \mathsf{address} \parallel [0,0], [0,2])$$

Although ψ_3 is an RCK *as well (i.e., minimal w.r.t. the number of compared attributes), any tuple pair agreeing on ψ_3 with* name *distance in $[0,0]$ (say t_5, t_6 in Table 1 for instance) will always satisfy $[0,4]$ of ψ_1, i.e.,* ***redundancy*** *among matching keys on the same attributes.*

Such redundancy obviously increases the overhead of record matching. It is not necessary to consider the redundant ψ_2, ψ_3 to detect (t_5, t_6) again, since they have already been identified by ψ_1.

In this study, rather than proposing a new record matching technique or another notation of matching rules, we employ the existing matching keys [7] and focus on choosing concise sets of matching keys with *high quality* and *less redundancy*. By providing a proper set of matching rules, it complements the existing record matching methods.

To evaluate the quality of matching keys, following the same line of discovering data dependencies and keys from data [14], we consider a data instance where the same real-world entities in attribute Y are pre-identified, e.g., the matching tuple pairs $(t_1, t_2), (t_3, t_4), (t_5, t_6)$ on ssn in Table 1. The *support* measure [3] evaluates the number of tuple pairs that can be covered by matching keys relative to Y, and *confidence* indicates the proportion of covered tuple pairs that correspond to true identifications on Y (see more explanations in Example 4).

The concise set discovery problem is to find the optimal set of matching keys relative to a given Y, which has the minimum set size (less redundancy) and can still meet the quality requirements on support and confidence (refer to Example 5 for more details).

Contributions. While [7] focuses on deducing a set of RCKs from a given set of matching keys, this paper is dedicated to discover a concise set of matching keys from data. Our main contributions are summarized as follows.

(1) Recognizing the NP-hardness of discovering the optimal matching key set, we devise approximation algorithms to efficiently find near optimal solutions in Section 3. A bound of approximation ratio on the set size introduced by the approximation is discussed (Proposition 1). Most importantly, we show that the matching keys discovered by the algorithm must be relative candidate keys (RCKs) [7] (Proposition 5) and minimal w.r.t. distance restrictions (Proposition 3).

(2) We develop advanced pruning strategies to further improve the efficiency in Section 4. Unqualified candidates of matching keys are filtered out during the set discovery computation (Propositions 6 and 8).

Table 2: Notations

Symbol	Description
ψ	matching key
Ψ	matching key set
$(t_1, t_2) \asymp \psi$	Two tuples agree on a key
$C_1[A] \lessdot C_2[A]$	Subsumption of distance restrictions
$\psi_1 \prec \psi_2$	Dominating between keys
$\mathsf{agree}(\psi)$	Set of tuple pairs in r agreeing on a ψ
η_c	Minimum requirement of confidence
η_s	Minimum requirement of support
FDs	Functional dependencies [12]
RCKs	Relative candidate keys [7]

(3) We report an extensive experimental evaluation in Section 5. The experiments demonstrate the effectiveness of choosing concise RCK sets for record matching, and the efficiency of the proposed discovery algorithms.

2. PRELIMINARIES

In this section, we introduce the formal syntax of matching keys [7], and the corresponding statistical measures [3] over a given relation instance. Table 2 lists the frequently used notations in this paper.

2.1 Syntax

Let A be an attribute in a relation scheme $\mathcal{R}$ and $\mathsf{dom}(A)$ denote a finite domain of A. We consider one distance metric d_A for each attribute A, denoted by $d_A : \mathsf{dom}(A) \times \mathsf{dom}(A) \to \mathcal{D}$, where $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$ is a finite set of distance values. It satisfies non-negativity, $d_A(a,b) \geq 0$; identity of indiscernibles, $d_A(a,b) = 0$ iff $a = b$; and symmetry, $d_A(a,b) = d_A(b,a)$, where $a, b \in \mathsf{dom}(A)$. For example, we can use edit distance [15] or cosine similarity [4]. It is worth noting that the selection of best distance metrics is not the focus of this study, please refer to [19] for a discussion. Our proposed techniques are fully compatible with any other distance metrics having the aforesaid properties.

Matching Keys. A *distance restriction* is a range of metric distances in the form of $[d_v, d_u]$, where $d_v, d_u \in \mathcal{D}$ and $d_v \leq d_u$. It specifies the restriction on distance between two values from A. We say two values $a, b \in \mathsf{dom}(A)$ satisfy the restriction $[d_v, d_u]$ if $d_v \leq d_A(a,b) \leq d_u$.

A *matching key* ψ relative to Y is in the form of $(X \parallel C)$, where X, Y are attribute sets in $\mathcal{R}$, and C is a pattern of distance restrictions on X. Each $C[A]$ denotes the distance restriction on an attribute $A \in X$. It states that if the X values of two tuples satisfy the distance restrictions C on X, their Y values should be identified, i.e., $(X \parallel C) \to (Y \rightleftharpoons)$.

Let t_1, t_2 be two tuples from a relation instance r of $\mathcal{R}$. We say that t_1, t_2 *agree on* the matching key $\psi : (X \parallel C)$, denoted by $(t_1, t_2) \asymp \psi$, if their distances on attributes X satisfy the distance restrictions C. That is, for each attribute $A \in X$, the distance between $t_1[A]$ and $t_2[A]$ satisfies the corresponding distance restriction $C[A] = [d_v, d_u]$, having $d_v \leq d_A(t_1[A], t_2[A]) \leq d_u$ or simply $d_v \leq d_A(t_1, t_2) \leq d_u$.

A dependency $\psi \to (Y \rightleftharpoons)$ requires that $(t_1, t_2) \asymp \psi$ implies $t_1[Y] \rightleftharpoons t_2[Y]$. If they agree on ψ, their Y values should be identified, i.e., denoting the same real-world entity.

Example 2 (Example 1 continued). *Consider the matching key* $\psi_1 : (\mathsf{name}, \mathsf{address} \parallel [0,4], [0,2])$ *relative to* ssn*, denoted by* $\psi_1 \rightarrow (\mathsf{ssn} \rightleftharpoons)$*. According to* ψ_1*,* t_1 *and* t_2 *in Table 1 should have* ssn *values identified, referring to their* name *distance* $0 \leq d_{\mathsf{name}}(t_1, t_2) = 4 \leq 4$*, i.e., in the range of* $[0, 4]$*, and* address *distance in the range of* $[0, 2]$ *having* $0 \leq d_{\mathsf{address}}(t_1, t_2) = 2 \leq 2$.

Relative Candidate Keys. Key $\psi : (X \parallel C)$ is a *relative candidate key* (RCK) if there is no other key $\psi' : (X' \parallel C')$ relative to Y such that (1) $X' \subset X$, and (2) for each $A \in X'$, $C'[A]$ of ψ' is exactly $C[A]$ of ψ. That is, no proper subset of distance restrictions can still form a valid matching key.

Let $d_{|\mathcal{D}|} = d_{\max}$ be the maximum value of distances in $\mathcal{D}$ and $d_1 = 0$ be the minimum distance value. We call $[d_1, d_{|\mathcal{D}|}] = [0, d_{\max}]$ an *unlimited distance restriction*, since any distance values will always be in the range from 0 to $d_{\max}$. For the attributes not specified in an RCK, it implicates unlimited distance restrictions. By appending unlimited restrictions, we can represent matching keys declared on different attributes *equivalently* by a unified *standard form* with the same attributes $X = \mathcal{R} \setminus Y$.

Example 3 (Example 1 continued). *We say that* ψ_1 *is in a non-standard form, with* $X \subset \mathcal{R} \setminus Y$*. There is no restriction on attribute* department *specified by* ψ_1*, i.e., unlimited on* department*. By appending the unlimited* $[0, d_{\max}]$ *on* department*,* ψ_1 *is equivalent to*

$$\psi_1^* : (\mathsf{name}, \mathsf{address}, \mathsf{department} \parallel [0,4], [0,2], [0, d_{\max}]).$$

This $\psi_1^* : (X \parallel C)$ *with* $X = \mathcal{R} \setminus Y$ *is a standard form of* ψ_1.

In the following of this paper, matching keys are considered in the standard form with $X = \mathcal{R} \setminus Y$ by default.

2.2 Measures

Suppose that the same real-world entities are pre-identified in a given data instance, e.g., Table 1 with all matching pairs $(t_1, t_2), (t_3, t_4), (t_5, t_6)$ identified on ssn. To discover reasonable matching keys from the data instance, we first need to evaluate the quality of a key, i.e., how the matching key can identify Y (ssn) values in the given data instance.

Evaluating a Single Matching Key. In light of coverage and validity in record matching, we study the following *support* and *confidence* measures defined on tuple pairs [3].

$$\mathsf{supp}(\psi) = \frac{|\{t_1, t_2 \in r \mid (t_1, t_2) \asymp \psi\}|}{|\{t_1, t_2 \in r\}|} \quad (1)$$

$$\mathsf{conf}(\psi) = \frac{|\{t_1, t_2 \in r \mid (t_1, t_2) \asymp \psi, t_1[Y] \rightleftharpoons t_2[Y]\}|}{|\{t_1, t_2 \in r \mid (t_1, t_2) \asymp \psi\}|} \quad (2)$$

Intuitively, given a relation instance r of $\mathcal{R}$, the *support* of a ψ is the proportion of tuple pairs in r whose values agree on ψ. It denotes the "coverage" of the matching key. The *confidence* is the ratio of tuple pairs whose values agree on ψ also having identified Y values, i.e., the "validity" of identifying Y by the matching key ψ. Matching keys with higher support and confidence are preferred.

Evaluating a Set of Matching Keys. Let Ψ denote a set of matching keys relative to the same Y. As discussed in the introduction, a tuple pair may agree on (be covered by) several keys $\psi \in \Psi$, i.e., redundancy in Ψ. If we simply add $\mathsf{supp}(\psi)$ of all $\psi \in \Psi$ as the support of the set Ψ, a tuple pair may be counted more than once due to the redundancy. To avoid duplicate counting, we should consider the distinct tuple pairs that are covered by a set of matching keys.

We say that t_1, t_2 *agree on* a set Ψ of matching keys, denoted by $(t_1, t_2) \asymp \Psi$, if there exists at least one $\psi \in \Psi$ such that $(t_1, t_2) \asymp \psi$. The *set support* of a set Ψ is defined as the proportion of distinct tuple pairs that agree on at least one of the matching keys in the set Ψ, i.e.,

$$\mathsf{supp}_s(\Psi) = \frac{|\{t_1, t_2 \in r \mid (t_1, t_2) \asymp \Psi\}|}{|\{t_1, t_2 \in r\}|}. \quad (3)$$

Moreover, to evaluate the validity of a set Ψ, we need to justify the confidence of each individual key in Ψ. Intuitively, in order to approach high accuracy in entity matching, it is expected that each applied matching key in the set should have high confidence. The minimum confidence of all the keys thus reflects the confidence of the set Ψ,

$$\mathsf{conf}_s(\Psi) = \min_{\psi \in \Psi} \mathsf{conf}(\psi). \quad (4)$$

Example 4 (Example 2 continued). *Given the data instance in Table 1 with truth matching* $(t_1, t_2), (t_3, t_4), (t_5, t_6)$*, we consider* $\psi_5 : (\mathsf{name}, \mathsf{address} \parallel [0,4], [0,4])$ *relative to* ssn*. There are four pairs of tuples* $(t_1, t_2), (t_2, t_3), (t_3, t_4), (t_5, t_6)$ *agreeing on* ψ_2 *and three pairs of tuples* $(t_1, t_2), (t_3, t_4), (t_5, t_6)$ *with identified* ssn*. Considering all the 15 tuple pairs in Table 1, we have* $\mathsf{supp}(\psi_2) = 4/15$ *and* $\mathsf{conf}(\psi_2) = 3/4$.

There are only two tuple pairs $(t_1, t_2), (t_5, t_6)$ *that can be covered by* $\psi_1 : (\mathsf{name}, \mathsf{address} \parallel [0,4], [0,2])$*. Indeed, it is already the key with the highest support, i.e.,* $\mathsf{supp}(\psi_1) = 2/15$*, when the confidence is required to be at least 1. We cannot achieve a larger support by any individual key.*

For a set $\Psi_1 = \{\psi_1, \psi_4\}$ *relative to* ssn*, where* (t_3, t_4) *agree on* $\psi_4 : (\mathsf{name}, \mathsf{address} \parallel [0,4], [4,4])$*. We have* $\mathsf{supp}_s(\Psi_1) = 3/15$*. Given* $\mathsf{conf}(\psi_1) = \mathsf{conf}(\psi_4) = 1$*, it follows* $\mathsf{conf}_s(\Psi_1) = 1$*. That is,* Ψ_1 *can* correctly *(with confidence 1) address* all *(a support of 3/15) the tuple pairs with identified* Y.

2.3 Problem Statement

Intuitively, we want to discover matching keys with high quality (high confidence) and address identifications as many as possible (high support). While a high confidence is always preferred, maximizing the support is not necessary during the discovery, since it may "overfit" the data [10]. Following the same line, we propose to find a set Ψ of matching keys relative to Y with the minimum requirements of support η_s and confidence η_c.

The selection of η_c for confidence is analogous to the requirement of matching accuracy. The support requirement η_s could be chosen based on the user's knowledge about how many duplicates exit. If η_s and η_c are set too high, it may not be able to find a feasible solution even when we consider all the possible matching keys as the answer set. Therefore, the first question is whether there exists a set Ψ such that the minimum requirements of support and confidence could be achieved. If yes, we find the minimum set Ψ_o^* which has less redundancy and can still satisfy η_s and η_c.

Problem 1. *The matching key set determination problem is: given a relation instance* r *of* $\mathcal{R}$*, a* Y *over* $\mathcal{R}$*, a constant* k*, and the minimum requirements of support* η_s *and confidence* η_c*, to decide whether there exists a set* Ψ *of matching keys such that* $\mathsf{supp}_s(\Psi) \geq \eta_s, \mathsf{conf}_s(\Psi) \geq \eta_c$*, and the size of the set is* $|\Psi| \leq k$.

This problem of deciding whether a feasible matching key set exists is found to be NP-complete [1].

Considering possible redundant semantics among matching keys, it naturally leads us to discover the most concise set of matching keys with less redundancy that can still meet the measure requirements of support η_s and confidence η_c.

The corresponding optimization problem is *to find the optimal set Ψ_o^* of matching keys such that the set size of Ψ_o^* is minimized, and the set still satisfies $\mathsf{supp}_s(\Psi_o^*) \geq \eta_s$ and $\mathsf{conf}_s(\Psi_o^*) \geq \eta_c$, if exists.*

Example 5 (Example 4 continued)**.** *Given $\eta_c = 1, \eta_s = 3/15$, a set $\Psi_2 = \{\psi_1, \psi_3, \psi_4\}$ with $\mathsf{conf}(\Psi_2) = 1, \mathsf{supp}(\Psi_2) = 3/15$ is feasible but not optimal, since $\Psi_1 = \{\psi_1, \psi_4\}$ is also a feasible solution that meets the η_c, η_s requirements but with a smaller size. Ψ_1 is more concise without the redundant ψ_3.*

3. APPROXIMATION METHOD

In this section, we present a greedy algorithm for efficiently approximating the desired set. In particular, we show that the discovered set of matching keys are always RCKs, and most importantly, minimal w.r.t. distance restrictions.

3.1 Candidates of Matching Keys

We consider all the potential matching keys relative to Y in standard form which specify distance restrictions on $X = \mathcal{R} \setminus Y$. Recall that a finite set of all distance values $\mathcal{D}$ is defined based on the $\mathsf{dom}(A)$ of an attribute A. With this $\mathcal{D}$, we can enumerate intervals of distance restrictions.

For each attribute A, we define $\mathcal{C}_A$ to be the set of all distance restrictions $[d_v, d_u]$ over A, i.e.,

$$\begin{aligned}\mathcal{C}_A =& \{[d_v, d_u] \mid d_1 \leq d_v \leq d_u \leq d_{|\mathcal{D}|}\}\\ =& \{[d_1, d_1], \ldots, [d_1, d_{|\mathcal{D}|-1}], [d_1, d_{|\mathcal{D}|}],\\ & [d_2, d_2], \ldots, [d_2, d_{|\mathcal{D}|}],\\ & \ldots, \quad [d_{|\mathcal{D}|}, d_{|\mathcal{D}|}]\}\end{aligned}$$

where $d_1, \ldots, d_{|\mathcal{D}|}$ denote all the distance values in $\mathcal{D}$ (see the following Figure 1 for instance). The size of $\mathcal{C}_A$ is $\mathcal{O}(|\mathcal{D}|^2)$.

Consider m attributes in $X = \mathcal{R} \setminus Y = \{A_1, \ldots, A_m\}$. Let Ψ_c be the set of all the potential matching keys,

$$\Psi_c = \{(X \parallel C) \mid C \in \mathcal{C}_{A_1} \times \cdots \times \mathcal{C}_{A_m}\}. \tag{5}$$

The size of Ψ_c is $\mathcal{O}(|\mathcal{D}|^{2m})$. In the worst case, the number of distinct distance values can be $|\mathcal{D}| = |\mathsf{dom}(A)|^2$. Let c be the size of $\mathsf{dom}(X)$, having $c = |\mathsf{dom}(A)|^m$. It follows $|\Psi_c| = \mathcal{O}(c^4)$.

For any potential matching key ψ, let $\mathsf{agree}(\psi)$ denote the set of all the tuple pairs $t_i, t_j \in r$ that agree on ψ, i.e.,

$$\mathsf{agree}(\psi) = \{(t_i, t_j) \mid (t_i, t_j) \asymp \psi, t_i, t_j \in r\}, \tag{6}$$

which is utilized to compute

$$\begin{aligned}\mathsf{supp}(\psi) =& \frac{|\mathsf{agree}(\psi)|}{|\{t_i, t_j \in r\}|},\\ \mathsf{conf}(\psi) =& \frac{|\{(t_i, t_j) \in \mathsf{agree}(\psi) \mid t_i[Y] \rightleftharpoons t_j[Y]\}|}{|\mathsf{agree}(\psi)|}.\end{aligned}$$

According to the set confidence in formula (4), the requirement of $\mathsf{conf}_s(\Psi) \geq \eta_c$ for any set Ψ is equivalent to $\mathsf{conf}(\psi) \geq \eta_c, \forall \psi \in \Psi$. In other words, only those matching keys with confidence $\geq \eta_c$ can be considered as candidates.

Algorithm 1 presents the discovery of a set Ψ of potential matching keys from a relation instance r, whose confidences are no less than η_c. Since the support and confidence measures are defined on the pairs of tuples in a relation instance, the algorithm considers all the tuple pairs in r. Specifically, according to formula (5), Line 3 goes through the candidates Ψ_c for each tuple pair $t_i, t_j \in r$. Line 8/5 creates/maintains $\mathsf{agree}(\psi)$ in formula (6), which is utilized to compute the support and confidence measures. Finally, those candidates with confidence satisfying the minimum requirement η_c are returned in Ψ.

Algorithm 1 Candidate set generation $\mathbf{CS}(r, \eta_c)$

Input: data instance r, minimum confidence requirement η_c
Output: A set Ψ of matching keys whose confidences are no less than η_c

```
1: Ψ := ∅
2: for each tuple pair t_i, t_j ∈ r do
3:   for each candidate ψ ∈ Ψ_c s.t. (t_i, t_j) ≍ ψ do
4:     if ψ ∈ Ψ then
5:       insert (t_i, t_j) to agree(ψ)
6:       update conf and supp of ψ to Ψ
7:     else
8:       agree(ψ) := {(t_i, t_j)}
9:       compute conf and supp of ψ, insert ψ to Ψ
10: return {ψ ∈ Ψ | conf(ψ) ≥ η_c}
```

Note that the **for** statement in Line 2 of Algorithm 1 adds a specific pair (t_i, t_j) to a certain $\mathsf{agree}(\psi)$ exactly once. Let n be the number of tuples in r. According to $|\Psi_c| = \mathcal{O}(c^4)$, the generation algorithm runs in $\mathcal{O}(n^2 c^4)$ time.

3.2 Greedy Algorithm

Now, we study the greedy algorithm for approximating a near optimal matching key set in polynomial time.

Let Ψ be a candidate set of matching keys (obtained in the above candidate generation), and let Ψ_o denote the near optimal set to discover. Intuitively, the greedy algorithm removes a candidate ψ with the maximum support from Ψ in each iteration, adds it into Ψ_o, and does not stop until the minimum support requirement η_s is satisfied or all the valid candidates in Ψ are added to Ψ_o.

Note that during the generation of candidates, a distinct tuple pair (t_i, t_j) may be included in $\mathsf{agree}(\psi)$ of several ψ in Ψ. However, according to formula (3), when we compute the support of a set Ψ_o, each tuple pair should be counted towards $\mathsf{supp}_s(\Psi_o)$ only once. To follow this principle, in each iteration of processing the current ψ, we need to remove the tuple pairs that agree on ψ (covered by ψ) from $\mathsf{agree}(\psi')$ for all the remaining candidates ψ' in Ψ. That is, we conduct the deduction operation,

$$\mathsf{agree}(\psi') = \mathsf{agree}(\psi') \setminus \mathsf{agree}(\psi),$$

to avoid counting a tuple pair more than once.

Algorithm 2 presents the discovery of a near optimal set Ψ_o of matching keys from the candidate set Ψ. Line 7 adds a ψ to Ψ_o in each iteration. The deduction operation $\mathsf{agree}(\psi') = \mathsf{agree}(\psi') \setminus \mathsf{agree}(\psi)$ in Line 10 deducts all the tuple pairs in $\mathsf{agree}(\psi)$ from $\mathsf{agree}(\psi')$ of the remaining $\psi' \in \Psi$. Line 11 re-calculates the $\mathsf{supp}(\psi')$ of ψ' by using the updated $\mathsf{agree}(\psi')$. By ensuring that tuple pairs in $\mathsf{agree}(\psi)$ have not been counted towards $\mathsf{supp}_s(\Psi_o)$ in previous steps, we can directly add $\mathsf{supp}(\psi)$ to $\mathsf{supp}_s(\Psi_o)$ in Line 8.

Algorithm 2 Greedy algorithm $\mathbf{GA}(\Psi, \eta_s)$

Input: candidate set Ψ, minimum support requirement η_s
Output: a near optimal set Ψ_o

```
1: Ψ_o := ∅
2: supp_s(Ψ_o) := 0
3: while Ψ ≠ ∅ and supp_s(Ψ_o) < η_s do
4:    ψ := arg max_{ψ∈Ψ} supp(ψ)
5:    if supp(ψ) = 0 then
6:       break
7:    move ψ from Ψ to Ψ_o
8:    supp_s(Ψ_o) += supp(ψ)
9:    for each ψ' ∈ Ψ do
10:      agree(ψ') := agree(ψ') \ agree(ψ)
11:      update supp of ψ' to Ψ
12: if supp_s(Ψ_o) < η_s then
13:    return ∅
14: else
15:    return Ψ_o
```

Proposition 1. *The greedy algorithm is* $1+2\ln|r|$ *approximation, having* $|\Psi_o|/|\Psi_o^*| \leq 1+2\ln|r|$*, where* $|\Psi_o|$ *is the size of the returned result set and* $|\Psi_o^*|$ *is the optimal set size.*

Proof. We employ the *k-partial set cover* problem: given a set of N elements $E = \{E_1, E_2, \ldots, E_N\}$, a collection S of subsets of E, $S = \{S_1, S_2, \ldots, S_M\}$, a cost function of S, and a k, to find a minimum cost sub-collection of S that covers at least k elements of E. Our discovery problem can be modeled as the k-partial set cover problem as follows. Each E_i denotes a tuple pair and each S_j denotes a ψ in our problem. The k corresponds to the minimum support η_s, and the cost function counts the number of subsets, i.e., analogous to $|\Psi_o|$. Consequently, to find a set with the minimum size, it is equivalent to find a minimum sub-collection of S. According to [8], the greedy algorithm is a $\ln N + 1$ approximation for the partial covering problem, which equivalently holds for our discovery problem, where $N = |r|^2$ is the total number of tuple pairs. □

Note that the **for** statement in Line 2 of Algorithm 1 adds a specific tuple pair to a certain $\mathsf{agree}(\psi)$ exactly once (in Line 5 or 8), i.e., $\mathcal{O}(n^2c^4)$. Subsequently, the **while** statement in Line 3 of Algorithm 2 removes a specific tuple pair from a certain $\mathsf{agree}(\psi')$ at most once (in Line 10). Therefore, the GA complexity is also $\mathcal{O}(n^2c^4)$, where n is the number of tuples in r.

3.3 Redundancy Free Results

In the following, we show that the matching keys returned by Algorithm 2 are redundancy free, i.e., RCKs and minimal w.r.t. distance restrictions.

Definition 1. *For any two distance restrictions* $[d_v, d_u]$ *and* $[d_g, d_h]$*, if* $d_v \leq d_g$ *and* $d_h \leq d_u$*, we say that* $[d_v, d_u]$ subsumes $[d_g, d_h]$*, denoted by* $[d_v, d_u] \lessdot [d_g, d_h]$*.*

Consider two matching keys ψ_1 and ψ_2.

Definition 2. *If* $C_{\psi_1}[A] \lessdot C_{\psi_2}[A]$ *for all attributes* $\forall A \in X$*, we say* ψ_1 dominates ψ_2*, denoted by* $\psi_1 \prec \psi_2$*.*
If there exists one attribute $\exists A \in X$ *having* $C_{\psi_1}[A] \lessdot C_{\psi_2}[A]$*, we say* ψ_1 partially *dominates* ψ_2*, denoted by* $\psi_1 \prec^p \psi_2$*.*

Referring to the support definition, we derive the following dominating relationships between matching keys.

Lemma 2. *For any* ψ_1 *dominating* ψ_2*, i.e.,* $\psi_1 \prec \psi_2$*, we have* $\mathsf{agree}(\psi_1) \supseteq \mathsf{agree}(\psi_2)$ *and* $\mathsf{supp}(\psi_1) \geq \mathsf{supp}(\psi_2)$*.*
When $\mathsf{agree}(\psi_1) = \mathsf{agree}(\psi_2)$*, we say that* ψ_1 *and* ψ_2 *are equivalent, having* $\mathsf{supp}(\psi_1) = \mathsf{supp}(\psi_2)$*.*

Proof. For each attribute $A \in X$, let $C_{\psi_2}[A] = [d_g, d_h]$ and $C_{\psi_1}[A] = [d_v, d_u]$. Consider a tuple pair (t_i, t_j) in $\mathsf{agree}(\psi_2)$, i.e., having $d_g \leq d_A(t_i, t_j) \leq d_h$ for any attribute $A \in X$. According to $\psi_1 \prec \psi_2$, we have $d_v \leq d_g \leq d_A(t_i, t_j) \leq d_h \leq d_u$, that is, $d_A(t_1, t_2)$ satisfies $C_{\psi_1}[A]$ for each attribute A as well. In other words, all the tuple pairs in $\mathsf{agree}(\psi_2)$ are also contained in $\mathsf{agree}(\psi_1)$, i.e., $\mathsf{agree}(\psi_2) \subseteq \mathsf{agree}(\psi_1)$. Referring to the support definition, we have $\mathsf{supp}(\psi_1) \geq \mathsf{supp}(\psi_2)$.

Moreover, since we always have $\mathsf{agree}(\psi_2) \subseteq \mathsf{agree}(\psi_1)$, it follows $\mathsf{supp}(\psi_1) = \mathsf{supp}(\psi_2)$ if and only if $\mathsf{agree}(\psi_2) = \mathsf{agree}(\psi_1)$. That is, ψ_1 and ψ_2 cover exactly the same tuple pairs. Referring to the greedy algorithm, there is no difference between the candidates ψ_1 and ψ_2, i.e., equivalent. □

When both matching keys are valid (with confidences $\geq \eta_c$), we say that ψ_2 is *redundant* w.r.t. ψ_1, since any tuples with Y values identified by ψ_2 are identified by ψ_1 as well according to $\psi_1 \prec \psi_2$.

A matching key ψ is *minimal*, if there does not exist any ψ' such that $\psi' \prec \psi$ and $\mathsf{conf}(\psi) \geq \eta_c$. That is, ψ is not redundant w.r.t. any other possible matching keys.

Proposition 3. *The matching keys in* Ψ_o *discovered by* ***GA*** *algorithm are always* minimal*, i.e.,* $\forall \psi \in \Psi_o$*, there does not exist any* $\psi' \in \Psi_c$ *such that* $\psi' \prec \psi$ *and* $\mathsf{conf}(\psi) \geq \eta_c$*.*

Proof. According to Lemma 2, Algorithm 2 always selects ψ with higher support in Line 4. For the remaining ψ', by conducting $\mathsf{agree}(\psi') := \mathsf{agree}(\psi') \setminus \mathsf{agree}(\psi)$ in Line 10, we have $\mathsf{agree}(\psi') = \emptyset$ as $\mathsf{agree}(\psi) \supseteq \mathsf{agree}(\psi')$. Since there is no remaining support on ψ', ψ' will not contribute and thus cannot be selected into Ψ_o. □

It is worth noting that the "minimal" definition is more strict than the RCK definition. While RCKs require not existing any other key with a subset of same distance restriction, the minimal matching keys ensure no other key with distance restrictions dominating the minimal matching key.

Lemma 4. *Minimal matching keys are always* RCKs*.*

Proof. Let ψ be a minimal matching key but not relative candidate key. That is, there exists a $\psi' : (X' \parallel C')$ relative to Y such that $X' \subset X$, and for each $A \in X'$, $C'[A]$ of ψ' is exactly $C[A]$ of ψ. By representing both keys in standard form, we have $C'[B] : [0, d_{\max}] \lessdot C[B], \forall B \in X \setminus X'$. It follows $\psi' \prec \psi$, according to the definition of dominating, i.e., ψ is not minimal. □

Consequently, following the same line of Proposition 3, we can show that the results are always RCKs.

Proposition 5. *The matching keys in* Ψ_o *discovered by* ***GA*** *algorithm are always relative candidate keys (*RCKs*).*

Since equality is considered as a special case of distance restriction, i.e., $[0, 0]$, the relationship between traditional super key and candidate key can also be interpreted by the dominating relationship (see examples below).

Example 6 (Example 5 continued)**.** *Consider* $\psi_1 : (\mathsf{name}, \mathsf{address} \parallel [0, 4], [0, 2])$ *and* $\psi_3 : (\mathsf{name}, \mathsf{address} \parallel [0, 0], [0, 2])$*, having* $[0, 4] \lessdot [0, 0]$ *on* name *and consequently* $\psi_1 \prec \psi_3$*. For the tuples in Table 1, we have* $\mathsf{agree}(\psi_1) = \{(t_1, t_2), (t_5, t_6)\}$*, while* ψ_3 *can only cover one of them, i.e.,* (t_5, t_6)*.*

Since both ψ_1 and ψ_3 can identify ssn *of (t_5, t_6), ψ_3 is redundant w.r.t. ψ_1. **GA** algorithm first selects ψ_1 with higher support* $\mathsf{supp}(\psi_1) = 2/15$. *The deduction operation in Line 10, i.e.,* $\mathsf{agree}(\psi_3) = \mathsf{agree}(\psi_3) \backslash \mathsf{agree}(\psi_1)$, *eliminates (t_5, t_6) that has been addressed by ψ_1 from* $\mathsf{agree}(\psi_3)$ *and makes it empty. That is, there is no remaining tuples supporting ψ_3. In other words, ψ_3 cannot further contribute and will never be returned as results. By further processing ψ_4, the returned matching keys $\{\psi_1, \psi_4\}$ are minimal.*

A super key $\{\mathsf{name}, \mathsf{address}, \mathsf{department}\}$ *w.r.t.* FDs *represented by* $\psi_s : (\mathsf{name}, \mathsf{address}, \mathsf{department} \parallel [0,0], [0,0], [0,0])$ *is not minimal, since there is a candidate key* $\{\mathsf{name}, \mathsf{address}\}$ *with a subset of attributes, denoted by* $\psi_c : (\mathsf{name}, \mathsf{address}, \mathsf{department} \parallel [0,0], [0,0], [0, d_{\max}])$. *We have $\psi_c \prec \psi_s$.*

4. PRUNING APPROACH

The greedy discovery of a near optimal set is still very costly. This is because the greedy step requires scanning all the possible candidates of matching keys in Ψ. Remarkably, as introduced below, not all the possible matching keys are necessary to be considered in candidate generation as Algorithm 1 does. Moreover, in the greedy step, after moving a key to Ψ_o, it is not necessary either to update the agree information for all the remaining candidates in Ψ. In the rest of this section, we propose several pruning strategies for the candidate set generation and the greedy step, respectively.

4.1 Pruning in Candidate Set

Based on the aforesaid subsumption/dominating relationships among distance restrictions, we are already able to discern some **redundant** matching keys in the candidate set Ψ, without further evaluation on agree sets of tuple pairs.

To understand the relationships more clearly, we represent the matching keys as follows. Based on the subsumption relationship, all the distance restrictions on an attribute can be represented by a directed acyclic graph. For example, as shown in Figure 1(a), each dot node denotes a possible distance restriction. An arrow from node a to b denotes $a \lessdot b$. The transitivity relationship is naturally implied, in other words, $a \lessdot b$ and $b \lessdot c$ indicate $a \lessdot c$ as well. In the figure, we only plot the most tight subsumption relationships, i.e., for each $a \lessdot b$ there does not exist another c such that $a \lessdot c \lessdot b$, and omit the subsumption that can be inferred.

Consequently, there is a triangle structure that specifies all the possible distance restrictions corresponding to an attribute A, e.g., Figure 1(b) for attribute A_2, Figure 1(c) for attribute A_3, etc. The root node in the triangle structure, $[d_1, d_{|\mathcal{D}|}]$, is the unlimited restriction. Recall that we have $X = \mathcal{R} \setminus Y$ in the standard form. Thereby, each ψ (in the standard form) consists of exactly one node (a distance restriction) from each triangle (each attribute in X).

Identify Unqualified Candidates

Now, we study the pruning techniques for reducing the number of candidates, before the greedy step is conducted. Recall that distance values $\mathcal{D}$ of an attribute A are defined with respect to the domain of A, $\mathsf{dom}(A)$. Therefore, some of the distance values (say d_v) in $\mathcal{D}$ of an attribute A may not appear in a given relation instance r. However, according to the candidate set generation algorithm, these distance values $d_v \in \mathcal{D}$ are still considered as *bounds* in intervals of distance restrictions in candidate matching keys. We study the pruning strategies based on these non-appearing distance values.

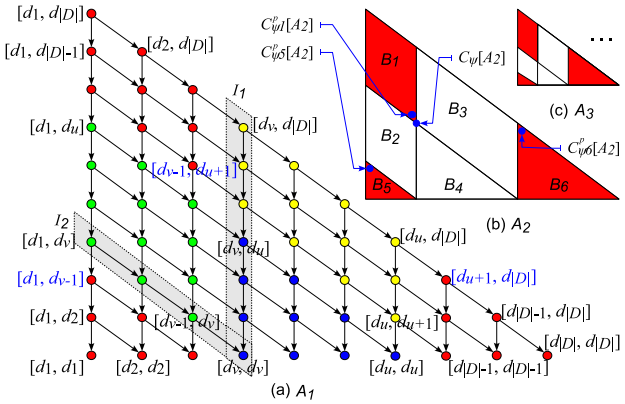

Figure 1: Relationship among distance restrictions

Intuitively, since the distance value d_v does not appear in attribute A in the relation instance r, all the candidates ψ containing the distance restriction $C_\psi[A] = [d_v, d_v]$ on A should have an empty $\mathsf{agree}(\psi)$ set and can be ignored. Moreover, let us consider some other ψ with distance restrictions like $\psi[A] = [d_v, d_{v+1}]$ on A. We prove that there always exists another ψ' (such as $\psi'[A] = [d_{v+1}, d_{v+1}]$ on A) having $\mathsf{agree}(\psi) = \mathsf{agree}(\psi')$. According to Lemma 2, ψ is equivalent to ψ' and can be pruned. We define these prunable candidates with certain distance restrictions below.

Proposition 6. *Consider a distance value $d_v \in \mathcal{D}$ of an attribute $A \in X$, which does not appear in the attribute A of any tuple pairs in the relation instance r. Then all the candidates $\psi \in \Psi$ that contain the following distance restrictions on attribute A, i.e., $C_\psi[A] \in (I_1 \cup I_2)$, can be pruned from the candidate set Ψ.*

$$I_1 = \{[d_v, d_{v+u}] \mid u = 0, 1, 2, \dots\}$$
$$I_2 = \{[d_{v-u}, d_v] \mid u = 0, 1, 2, \dots\}$$

Proof sketch. For each candidate ψ having $C_\psi[A] \in I_1$, we can always find a ψ_2 with $C_{\psi_2}[A] = [d_{v+1}, d_{v+u}]$, which is equivalent to ψ, i.e., $\mathsf{agree}(\psi) = \mathsf{agree}(\psi_2)$. Thus, candidate ψ can be pruned as redundancy. Similarly, for any ψ with $C_\psi[A] \in I_2$, we can find a ψ_1 with $C_{\psi_1}[A] = [d_{v-u}, d_{v-1}]$, as redundancy of ψ. □

Example 7. *Suppose that the distance value d_v does not appear in any tuple pair of r on attribute A_1. Then, the sets of distance restrictions, $I_1 = \{[d_v, d_v], [d_v, d_{v+1}], \dots, [d_v, d_{|\mathcal{D}|}]\}$ and $I_2 = \{[d_1, d_v], [d_2, d_v], \dots, [d_v, d_v]\}$ marked by shade areas in Figure 1(a), can be ignored in attribute A_1 during the candidate generation. That is, the candidate set pruning (*CSP*) method removes all the candidates from Ψ whose distance restrictions come from I_1 or I_2 on attribute A_1.*

4.2 Pruning in Greedy Step

The major cost of each greedy step originates from the update of tuple pairs in $\mathsf{agree}(\psi')$ after moving a ψ with the highest support to Ψ_o. We propose pruning rules to reduce the number of updates, and remove redundant candidates.

Let ψ be the currently selected candidate in a greedy step. Let $[d_v, d_u]$ denote the distance restriction of ψ on attribute A, i.e., $C_\psi[A] = [d_v, d_u]$. For each attribute A, we divide all the distance restrictions into 6 blocks according to the

subsumption relationship on $[d_v, d_u]$ as follows.

$$B_1[A] = \{C[A] \mid C[A] \preccurlyeq [d_{v-1}, d_{u+1}]\}$$
$$B_2[A] = \{C[A] \mid [d_1, d_u] \preccurlyeq C[A] \preccurlyeq [d_{v-1}, d_v]\}$$
$$B_3[A] = \{C[A] \mid [d_v, d_{|\mathcal{D}|}] \preccurlyeq C[A] \preccurlyeq [d_u, d_{u+1}]\}$$
$$B_4[A] = \{C[A] \mid [d_v, d_u] \preccurlyeq C[A]\}$$
$$B_5[A] = \{C[A] \mid [d_1, d_{v-1}] \preccurlyeq C[A]\}$$
$$B_6[A] = \{C[A] \mid [d_{u+1}, d_{|\mathcal{D}|}] \preccurlyeq C[A]\}$$

Among the above 6 blocks, $B_1[A]$ represents the distance restrictions on A that subsume $C_\psi[A]$; $B_4[A]$ denotes all the restrictions on A that are subsumed by $C_\psi[A]$; $B_5[A]$ and $B_6[A]$ are the restrictions that have no overlap with $C_\psi[A]$; $B_2[A]$ and $B_3[A]$ are the restrictions that have overlap with $C_\psi[A]$. For example, in Figure 1(b), we illustrate the 6 blocks of all the distance restrictions in attribute A_2 based on $C_\psi[A_2]$ of the current ψ.

Identify Qualified Candidates

We first identify the set of candidates ψ' whose agree set is not updated by the deduction operation $\mathsf{agree}(\psi') = \mathsf{agree}(\psi') \setminus \mathsf{agree}(\psi)$ even in the original greedy algorithm (Algorithm 2). Intuitively, the deduction operation has no effect on those distance restrictions, which do not subsume any other restrictions that are subsumed by $C_\psi[A]$, i.e., distance restrictions in $B_5[A]$ and $B_6[A]$ having no overlap with $C_\psi[A]$.

Lemma 7. *After inserting a ψ with the maximum support into Ψ_o, the following sets of candidates ψ' are **not** updated.*

$$K_5 = \{\psi' \mid \exists A \in X, C_{\psi'}[A] \in B_5[A]\}$$
$$K_6 = \{\psi' \mid \exists A \in X, C_{\psi'}[A] \in B_6[A]\}$$

Proof sketch. For any $\psi' \in K_5$ or K_6, we can prove that the intersection of agree sets of ψ and ψ' is $\mathsf{agree}(\psi') \cap \mathsf{agree}(\psi) = \emptyset$. The operation $\mathsf{agree}(\psi') = \mathsf{agree}(\psi') \setminus \mathsf{agree}(\psi)$ takes no effect on candidate ψ'. Thus ψ' is not updated. □

According to Lemma 7, all the candidates with distance restrictions from B_5 or B_6 on *any* attribute will not be updated in the current iteration. In other words, only those candidates will be updated, which have distance restrictions from B_1, B_2, B_3, B_4 on **all** the attributes, i.e., $\Psi \setminus (K_5 \cup K_6)$.

Identify Unqualified Candidates

Now, we study the pruning in $\Psi \setminus (K_5 \cup K_6)$ to avoid the updates on unqualified candidates. Based on the subsumption and dominating relationships, we propose to filter out the following two types of candidates:

(1) Those candidates ψ' having $\mathsf{agree}(\psi') = \emptyset$ after the $\mathsf{agree}(\psi') = \mathsf{agree}(\psi') \setminus \mathsf{agree}(\psi)$ operation. Candidates with empty agree set will have no contribution to the result, and thus can be pruned without conducting the updating operation. Intuitively, those candidates with distance restrictions in B_4, i.e., subsumed by $C_\psi[A]$, may fall into this category.

(2) Those candidates ψ' that always have another ψ_1 in K_5 or K_6 having $\mathsf{agree}(\psi') = \mathsf{agree}(\psi_1)$, i.e., equivalent, after the $\mathsf{agree}(\psi') = \mathsf{agree}(\psi') \setminus \mathsf{agree}(\psi)$ operation. For example, a candidate ψ' with distance restrictions in B_2 may be considered to find an equivalent ψ_1 with distance restriction in B_5, such that $C_{\psi'}[A]$ subsumes $C_{\psi_1}[A]$. Since this ψ_1 is reserved in Ψ without updating in the current iteration, the equivalent one ψ' can be pruned as redundancy.

Formally, we define the candidates that can be directly pruned from the candidate set Ψ as follows.

Proposition 8. *After inserting the current ψ with the maximum support into Ψ_o, the following set of candidates K_p can be pruned from Ψ.*

$$K_p = \{\psi' \mid \forall A \in X, C_{\psi'}[A] \in (B_2[A] \cup B_3[A] \cup B_4[A])\}$$

Proof sketch. For a candidate $\psi' \in K_p$, the distance restriction $C_{\psi'}[A]$ of any attribute A comes from either $B_2[A]$, $B_3[A]$ or $B_4[A]$. Let

$$K_2 = \{\psi' \mid \exists A \in X, C_{\psi'}[A] \in B_2[A], \psi' \in K_p\}$$
$$K_3 = \{\psi' \mid \exists A \in X, C_{\psi'}[A] \in B_3[A], \psi' \in K_p\}$$
$$K_4 = \{\psi' \mid \forall A \in X, C_{\psi'}[A] \in B_4[A], \psi' \in K_p\}$$

having $K_p = K_2 \cup K_3 \cup K_4$.

For any $\psi' \in K_2$ or K_3, we prove that there always exists a candidate in the remaining candidate sets (say $\psi_1 \in K_5$ or $\psi_2 \in K_6$) which is equivalent to ψ' after the current deduction step. Thus, candidate ψ' can be pruned as duplicates.

For any candidate $\psi' \in K_4$, by proving that $\mathsf{agree}(\psi') = \emptyset$ after the current deduction step, ψ' can be pruned. □

According to the definition of K_p, $\psi' \in K_p$ contains only distance restrictions from B_2, B_3, B_4 on all the attributes $A \in X$. In other words, $\psi' \in K_p$ does not contain distance restrictions from B_1, B_5, B_6 on all the attributes. Let

$$K_1 = \{\psi' \mid \exists A \in X, C_{\psi'}[A] \in B_1[A]\}.$$

Then, we can also represent K_p by $K_p = \Psi \setminus (K_1 \cup K_5 \cup K_6)$.

Definition 3. *Let $\psi_1^p, \psi_5^p, \psi_6^p$ be three* pivot *candidates such that, $\forall A \in X$, $C_{\psi_1^p}[A] = [d_{v-1}, d_{u+1}]$, $C_{\psi_5^p}[A] = [d_1, d_{v-1}]$ and $C_{\psi_6^p}[A] = [d_{u+1}, d_{|\mathcal{D}|}]$, where $C_\psi[A] = [d_v, d_u]$.*

For instance, $C_{\psi_1^p}[A_2]$, $C_{\psi_5^p}[A_2]$ and $C_{\psi_6^p}[A_2]$ on attribute A_2 are illustrated in Figure 1(b) w.r.t. the current $C_\psi[A_2]$. According to the partial dominating $\prec^p$ in Definition 2, we rewrite $K_1 = \{\psi' \mid \psi' \prec^p \psi_1^p\}$, $K_5 = \{\psi' \mid \psi_5^p \prec^p \psi'\}$ and $K_6 = \{\psi' \mid \psi_6^p \prec^p \psi'\}$ by $\psi_1^p, \psi_5^p, \psi_6^p$.

According to Proposition 8, we only need to update the candidates ψ' in $K_1 \cup K_5 \cup K_6$, i.e., $\psi' \prec^p \psi_1^p$ **or** $\psi_5^p \prec^p \psi'$ **or** $\psi_6^p \prec^p \psi'$, while the remaining candidates $K_p = \Psi \setminus (K_1 \cup K_5 \cup K_6)$ can be safely pruned.

Greedy Algorithm with Pruning

Finally, we present the greedy algorithm with pruning (namely GAP). Rather than removing each tuple pair from possible $\mathsf{agree}(\psi)$ exactly once in the original greedy algorithm, we prune the candidates that belong to the aforesaid K_p.

Algorithm 3 presents the pruning steps in the greedy computation. As illustrated in Line 4, we greedily select a candidate ψ in each step. Line 9 computes the pruning pivots $\psi_1^p, \psi_5^p, \psi_6^p$ in Definition 3. According to Proposition 8, those candidates in K_p can be safely pruned, which are identified by using the pivots $\psi_1^p, \psi_5^p, \psi_6^p$. That is, we only conduct the deduction operation (in Line 12) on those candidates ψ' such that $\psi' \prec^p \psi_1^p$ **or** $\psi_5^p \prec^p \psi'$ **or** $\psi_6^p \prec^p \psi'$, while the other candidates (belonging to K_p) are directly removed in Line 15. Finally, the greedy iteration terminates if either the requirement η_s of support is reached or all the candidates have been evaluated (i.e., $\Psi = \emptyset$ in Line 3).

Example 8 (Example 6 continued)**.** *Suppose that ψ_1^* : (*name, address, department $\parallel$ $[0, 4], [0, 2], [0, d_{\max}]$*) is the currently selected candidate in Line 4 in Algorithm 3. We show that ψ_3^* : (name, address, department $\parallel$ $[0, 0], [0, 2], [0, d_{\max}]$) can*

Algorithm 3 Greedy algorithm with pruning $\mathbf{GAP}(\Psi, \eta_s)$

Input: candidate set Ψ, minimum support requirement η_s
Output: a near optimal set Ψ_o

```
1: Ψ_o := ∅
2: supp_s(Ψ_o) := 0
3: while Ψ ≠ ∅ and supp_s(Ψ_o) < η_s do
4:    ψ := arg max_{ψ∈Ψ} supp(ψ)
5:    if supp(ψ) = 0 then
6:       break
7:    move ψ from Ψ to Ψ_o
8:    supp_s(Ψ_o) += supp(ψ)
9:    calculate ψ_1^p, ψ_5^p, ψ_6^p from ψ
10:   for each candidate ψ' ∈ Ψ do
11:      if ψ' ≺^p ψ_1^p or ψ_5^p ≺^p ψ' or ψ_6^p ≺^p ψ' then
12:         agree(ψ') := agree(ψ') \ agree(ψ)
13:         update conf and supp of ψ' to Ψ
14:      else
15:         remove ψ' from Ψ
16: if supp_s(Ψ_o) < η_s then
17:    return ∅
18: else
19:    return Ψ_o
```

be safely pruned without further evaluation on its agree *set (as it does in Example 6).*

First, for the attribute name*, we have* $C_{\psi_1^*}[\mathsf{name}] = [d_v, d_u] = [0, 4]$. *Since* $d_v = 0$ *is already the minimum distance value in* $\mathcal{D}$*, blocks* $B_1[\mathsf{name}]$ *and* $B_5[\mathsf{name}]$ *are empty according to the definition. Referring to Definition 3, we have* $C_{\psi_6^p}[\mathsf{name}] = [5, d_{\max}]$. *That is,* $C_{\psi_3^*}[\mathsf{name}]$ *does not belong to* $B_6[\mathsf{name}]$ *either, and thus must be in blocks* B_2, B_3 *or* B_4.

For the other attributes with $C_{\psi_1^*}[\mathsf{address}] = C_{\psi_3^*}[\mathsf{address}] = [0, 2]$ *and* $C_{\psi_1^*}[\mathsf{department}] = C_{\psi_3^*}[\mathsf{department}] = [0, d_{\max}]$*, we directly conclude that* $C_{\psi_3^*}[\mathsf{department}]$ *and* $C_{\psi_3^*}[\mathsf{address}]$ *belong to* B_4 *of* ψ_1^* *on* department *and* address*, respectively.*

Finally, since none of $C_{\psi_3^*}[A]$ *belong to* B_1, B_5, B_6*, we have all* $\psi_3^* \prec^p \psi_1^p, \psi_5^p \prec^p \psi_3^*, \psi_6^p \prec^p \psi_3^*$ *equal to* **false**.

Let δ $(0 \leq \delta \leq 1)$ be the pruning rate on average, i.e., δ percentage of candidates can be avoided to perform the $\mathsf{agree}(\psi') = \mathsf{agree}(\psi') \setminus \mathsf{agree}(\psi)$ operation. Calculating the pivot candidates $\psi_1^p, \psi_5^p, \psi_6^p$ for K_p is in constant time. The complexity of Algorithm 3 with pruning is $\mathcal{O}((1-\delta)n^2c^4)$, where n is the number of tuples in r. As illustrated in the experiments, GAP can always improve time performance in practice. According to the results in Figures 7(f), 8(f) and 9(f), the prune rate is greater than 0.8 (80%) in most tests.

Since the pruning methods do not affect the results, the conclusions about RCKs and minimal matching keys of greedy algorithm are still valid.

5. EXPERIMENTAL EVALUATION

We report experiments on evaluating the proposed techniques in two aspects. 1) Since this study is to complement the existing record matching techniques by providing proper matching keys, we implement a rule-based matching method [11], and compare the matching effectiveness by using our concise RCK set and the RCKs return by the existing findRCKs approach [7]. 2) We compare the efficiency of various proposed techniques for finding the concise RCK sets.

Benchmark datasets for evaluating record linkage are employed,[2] including two real datasets *Cora* (with 864 tuples) and *Restaurant* (with 1295 tuples) for evaluating the matching accuracy and a synthetic *UIS* database generator (with 10,000 tuples generated) for efficiency evaluation. To study the accuracy of record matching, we use the standard *f-measure* of *precision* and *recall* [17].

[2]http://www.cs.utexas.edu/users/ml/riddle/data.html

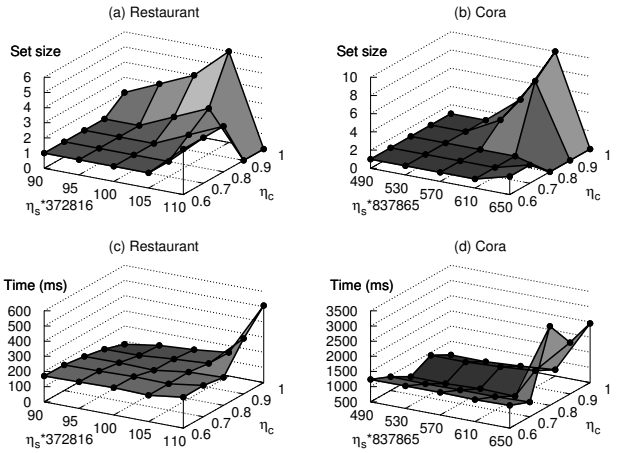

Figure 2: Concise RCK sets with various η_s and η_c

Exp1: Evaluating Concise RCK Sets. The first experiment, in Figure 2, observes the sizes of returned RCK sets under various η_s and η_c settings. With the increase of the minimum support requirement η_s, we need to add more matching keys into the set Ψ_o in order to increase the support, and thus the Ψ_o size increases as well. On the other hand, if the minimum confidence requirement η_c is large, as mentioned, some candidates with high support but low confidence may become invalid. We have to seek some other second-highest support candidates to meet the η_s requirement, and the Ψ_o set size increases consequently.

When both η_s and η_c are too high, there does not exist any matching key set with support and confidence greater than the requirements even by considering all the possible candidates, denoted by size 0. It is the worst case of considering all the candidates, and thus the corresponding time costs are extremely high (see more results in the following experiments on efficiency).

Exp2: Comparing RCKs in Record Matching. To evaluate the quality of matching keys, we employ the existing rule-based matching method [11]. In particular, the experiments in Figure 3 compare the performance of using all the matching keys under certain confidence guarantees η_c, the top five RCKs by findRCKs as evaluated in [7], and the concise RCK sets with various support commitment η_s (each line e.g. 90 in Figure 3 denotes a $\eta_s = 90/(864*863/2) = 90/372816$).

As shown in Figures 3(c) and (d), the recall is high by using all the matching keys. However, many irrational matching keys are probably included that overfit the data, and thus the corresponding precision is relatively low as illustrated in Figures 3(a) and (b). Indeed, owing to the large number of irrational matching keys, the time costs of using all matching keys are extremely high in Figures 3(g) and (h). By choosing high quality matching keys, our concise RCK sets (with various support guarantees) show comparable recall and higher precision compared with all matching keys. Generally, the higher the η_s is, the better the recall

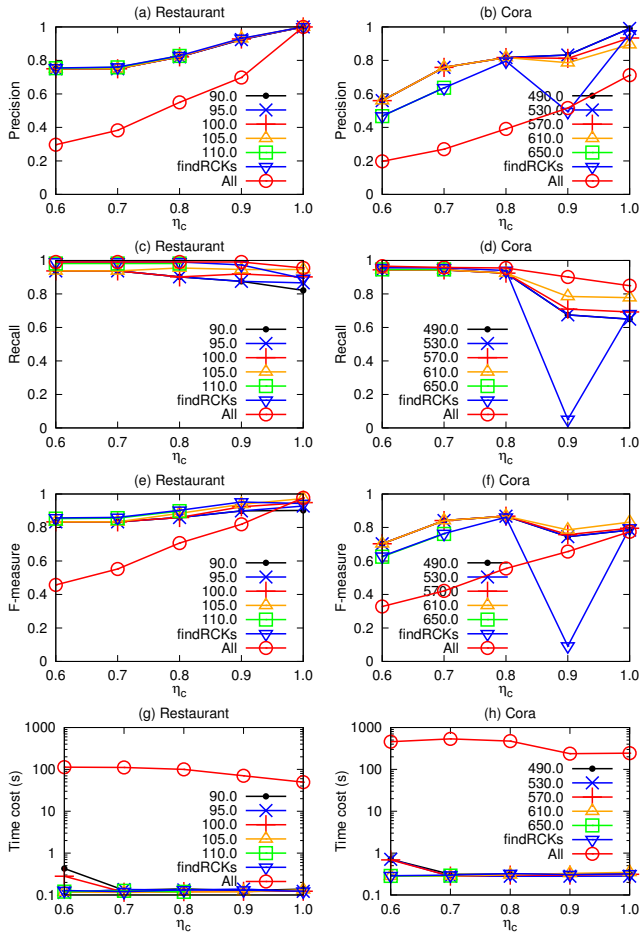

Figure 3: Record matching by various matching keys

will be. The corresponding time costs of concise RCK sets are significantly lower than that of all matching keys, with 3 orders of magnitude improvement.

The RCKs returned by findRCKs [7] do not show stable accuracy as illustrated in Figures 3(b), (d) and (f). The rationale behind is that findRCKs considers only the number of attributes and the lengths of attribute values in choosing matching keys, while our approach investigates the more precise support evaluation. To demonstrate clearly the superiority, we conduct another experiment in Figure 4 to compare the top-k RCKs of findRCKs and the first k RCKs of our GA algorithm. Both approaches achieve high precision in Figure 4(a), while the precision of our GA is higher in all the k tests in Figure 4(b). Although the recall of findRCKs increases (by finding matching keys similar to our concise set), no better results are reported compared with GA. These results verify again the effectiveness of finding high quality RCKs by the proposed methods.

Exp3: Comparing Record Matching Methods. We report another group of experiments on comparing the proposed technique with other record matching approaches.

For machine learning approaches, we implement a logistic regression (LR)-based approach that performs record matching as classification [21] and a SVM-based approach [2] that uses SVM to learn how to merge the matching results for individual fields of the records. For the constraint optimization method [13], the rules should reflect absolute

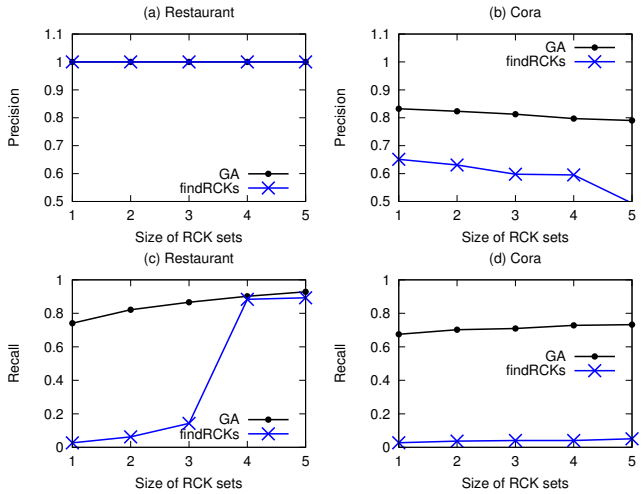

Figure 4: Varying the number of RCKs in matching

truths and serve as functional dependencies (FD). We employ the widely used level-wise algorithm [12] to discover the optimal/minimal FDs. For the distance-based approach, we implement [4] that treats a record as a long field and uses one distance metric to determine which records are similar.

Figure 5 illustrates the results given various sizes of training data. As shown, the recall accuracy of our GA approach is significantly higher than SVM and Distance-based methods, especially when the training size for learning/discovering is quite limited. The rationale behind is that our GA algorithm can still discover a number of RCKs even over a very limited size of training data. Although FD and LR can also achieve a relatively high recall, their corresponding precision accuracies are much lower owing to the weakness in expressing precise matching criteria. Consequently, the overall f-measure accuracy of our GA approach is higher.

To further evaluate the performance on a limited size of training data, we evaluate the above approaches in the active learning framework [16]. The idea of active learning is to interactively identify challenging training pairs for labeling in each round, and thus the learned model is expected to be gradually improved.

Figure 6 reports the results over various rounds of active learning. The initial round 0 has 20 training pairs, and each succeeding round has 10 challenging pairs labeled that are identified by the uncertainty [16]. Again, the accuracy of GA is high even in the initial round 0, and keeps higher accuracy in succeeding rounds. Generally, the active learning makes the training more efficient. For instance, for the GA approach, round 4 (with total 20+10*4=60 labeled pairs) in Figure 6(f) already achieves an accuracy as high as that of 200 training pairs in Figure 5(f).

Exp4: Efficiency of RCK Set Discovery. We study time performance of proposed algorithms, including the original candidate set generation (CS), the original greedy algorithm (GA), the candidate set generation with pruning (CSP), and the greedy algorithm with pruning (GAP). Figures 7, 8 and 9 show time performance over 2 real data sets Restaurant, Cora, and another larger synthetic data UIS, respectively. For each data set, e.g., Restaurant in Figures 7, we test 4 experiments (a), (b), (c) and (d) with different η_s and η_c requirements: test (a) with a large η_c, test (c) with a large

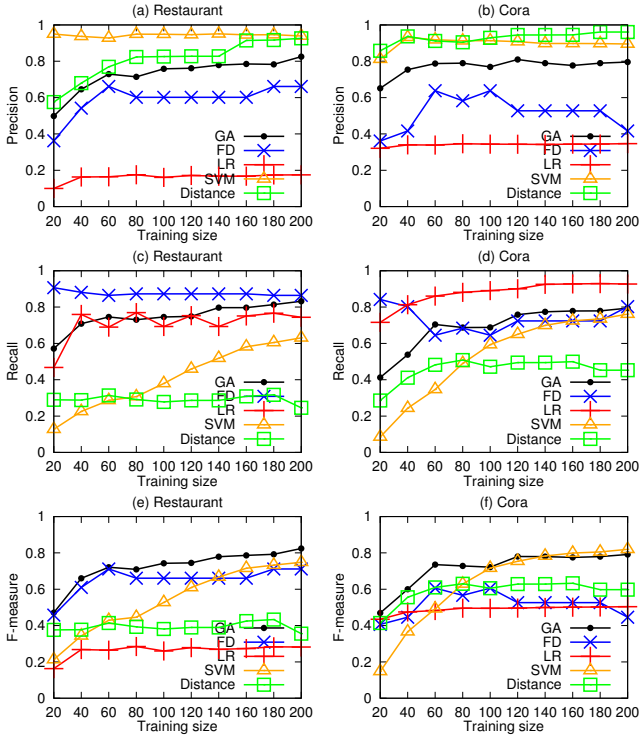

Figure 5: Comparing record matching techniques

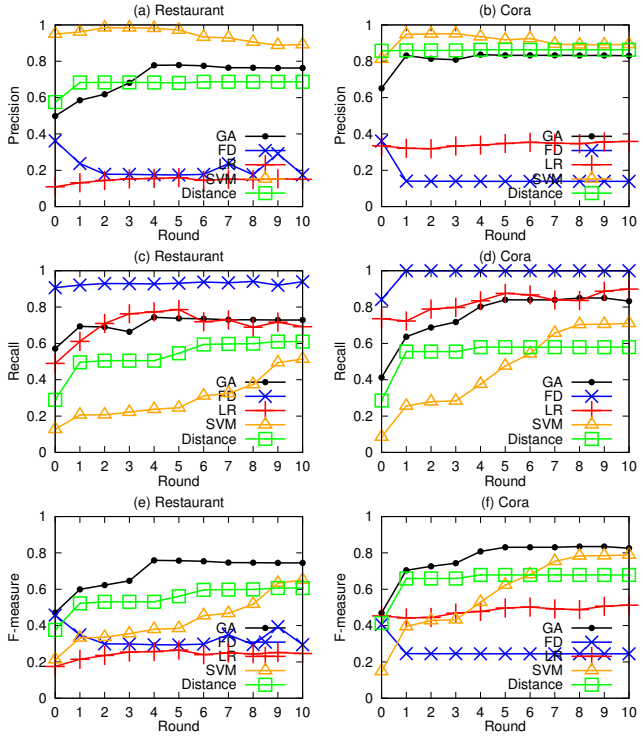

Figure 6: Comparison under active learning

η_s, test (b) having both large η_s and η_c, and test (d) having small η_s and η_c. The time costs of various tests are reported in sub-figures (a), (b), (c) and (d).

First, as observed in Figure 2 as well, the time performance of greedy algorithms heavily relies on the size of the returned set Ψ_o. If the size of the result set Ψ_o is large, we need to search more candidates in order to assemble such a large set and thus higher time costs. Since the confidence measure is not monotonic w.r.t. the size of data, the corresponding Ψ_o sizes (given certain confidence requirements) may vary under different data sizes.

As mentioned, the returned set size is affected by the η_s and η_c requirements, and thereby the time performance of approaches is affected by different η_s and η_c. As shown in Figures 7(e), 8(e) and 9(e), test (d) with smaller n_s and small n_c yields smaller set sizes, compared with tests (a), (b) and (c). Consequently, it is not surprising that the corresponding time costs are lower in test (d) in Figures 7(d), 8(d) and 9(d). When both the support and confidence requirements are large, in tests (b), the algorithm needs to seek a large number of candidates in order to satisfy the η_s and η_c requirements. The corresponding Ψ_o sizes and times costs of tests (b) are large. Recall that when both the η_s and η_c requirements are set too large, there might not exist any set that can achieve such high requirements, e.g., set size 0 under 10k tuples of test (b) in Figure 9(e). It is the worst case to traverse all the candidates. Thereby, as presented in Figure 9(b), the time cost on 10k is the highest.

Both the CSP and GAP techniques can improve the time performance compared with the original CS and GA respectively. In Figure 7, CSP works well and keeps low time cost even when CS requires about 5 times larger cost in the same environment, e.g., 900 tuples in Figure 7(b). Note that the pruning of CSP relies on the distance values that do not appear in the given data. When most possible values appear, e.g., in Figures 8 and 9, CSP does not work. Nevertheless, the GAP algorithm performs well (either with or without CSP). The CSP+GAP approach always achieves the best time performance and shows up to 2 orders of magnitude improvement (e.g., test of 1100 tuples in Figure 8).

To illustrate whether the ratio is practical, we design a group of new experiments on calculating the upper bound of approximation ratio in Figures 7(g), 8(g) and 9(g). In particular, although computing the exact minimum set is difficult (NP-hard as shown in [1]), it is possible to efficiently identify a lower bound of the minimum set size. The idea is to compute a set Ψ_u of matching keys with highest supports, whose support summation is greater than η_s (that is, exactly Algorithm 2 by ignoring Line 10 of reduction operation). Since the reduction operation is not considered, the set could be redundant and needs more matching keys to meet the support requirement η_s. In other words, the true minimum set Ψ_o^* may be greater than the computed set Ψ_u. Let Ψ_o be the set computed by the Greedy algorithm. We have approximation ratio $|\Psi_o|/|\Psi_o^*| \leq |\Psi_o|/|\Psi_u|$, where $|\Psi_o|/|\Psi_u|$ serves as an upper bound of approximation ratio. As shown in Figures 7(g), 8(g) and 9(g), the upper bound of approximation ratio is no greater than 6 in all the tests, where the actual approximation could be lower than the upper bound. In particular, the bound of UIS with up to 10k records is no greater than 3 in Figure 9(g). These results demonstrate that the approximation ratio is practical and much lower than the theoretical bound.

6. RELATED WORK

Record Matching Technique. A variety of record matching methods have been devised (see [5] for a survey). Although matching keys [7] are not able to support more com-

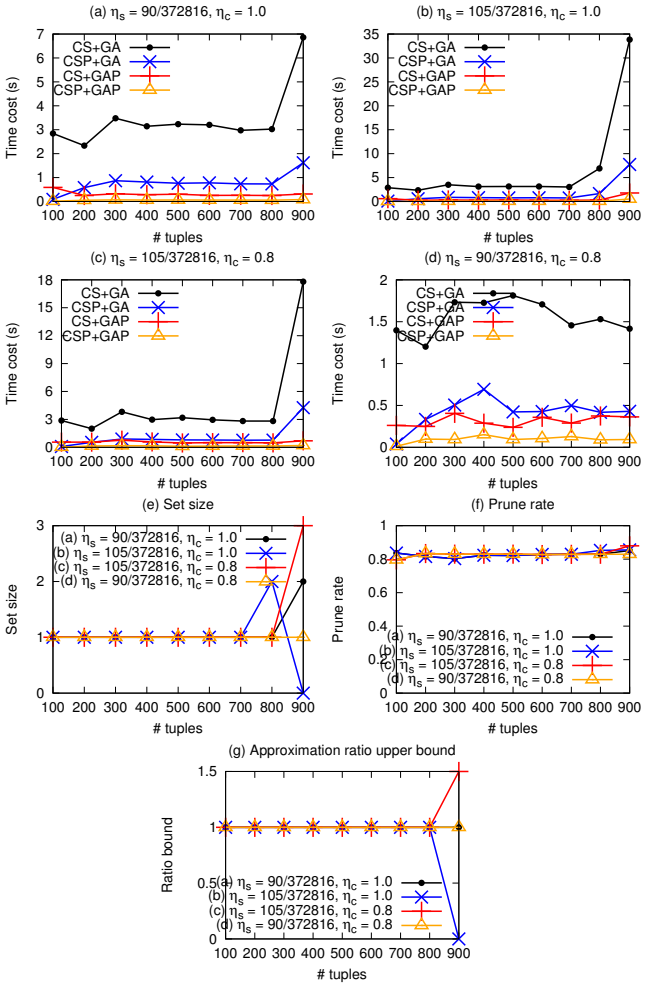

Figure 7: Discovery performance on *Restaurant*

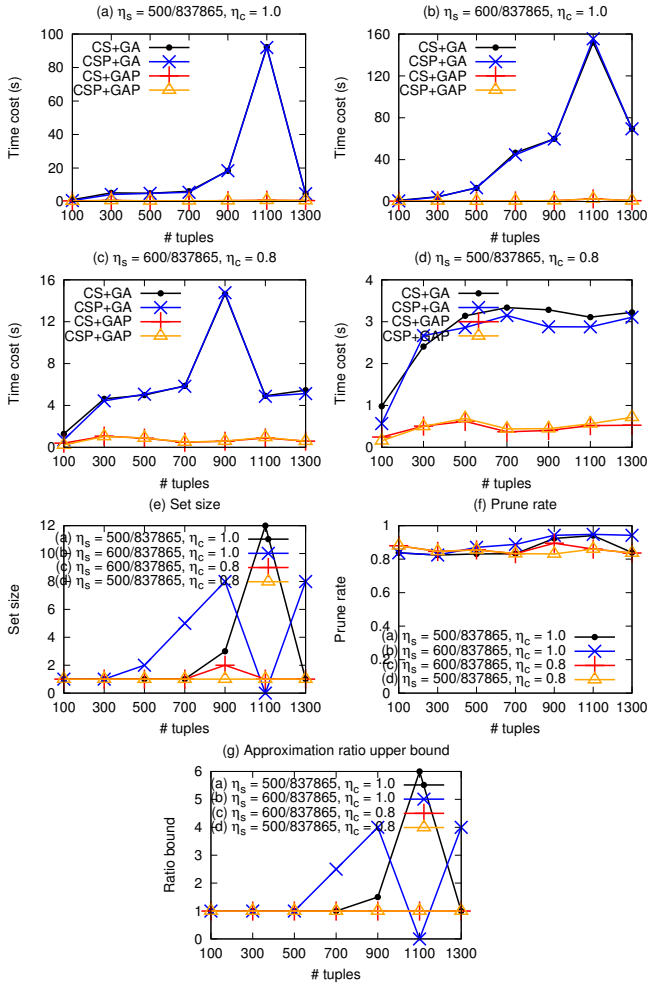

Figure 8: Discovery performance on *Cora*

plex Boolean expressions like disjunctions or negations, it is not the focus of this work to propose another matching algorithm or invent another type of rules. Instead, following the same line of [7], our study complements existing matching algorithms by providing proper matching keys. Nevertheless, matching keys naturally support one distance function for multiple attributes. Indeed, when combining all the attributes together in a matching key with one single distance function, it is equivalent to the distance-based approach [4]. We compare the Distance-based approach in Figures 5 and 6. As shown, the GA approach of matching keys (concerning distances separately in different attributes) has higher accuracy than the Distance one (with one distance function for multiple attributes).

Winkler [20] has shown that computerized record linkage procedures can significantly reduce the resources needed for identifying duplicates in comparison with methods that are primarily manual. As one of the computerized record linkage approach (see [5] for a survey), we believe that our approach also benefits from the effort reduction. To illustrate the efficient manual effort reduction, we also add a new group of experiments on comparing the approaches under the active learning framework in Figure 6, where the manual labels are dynamically fed round by round. Nevertheless, besides the contribution of improving (recall/overall) accuracy in record matching, it is always promising to reduce the time costs of discovery/construction, without loss of quality of the returned matching keys. Figures 7–9 demonstrate our contribution in improving the time performance of discovery.

Constraint Optimization. In the constraint optimization-based record matching [13], the rules should reflect absolute truths and serve as functional dependencies (FD). As one of the motivations of this study, the employed constraints are expected to be optimal/minimal/concise. A variety of methods have been proposed for discovering the optimal integrity constraints from a given data instance (see [14] for a survey). To compare our proposed techniques with the constraint optimization-based approach, we employ the widely used level-wise algorithm [12] to discover the optimal/minimal FDs. Owing to the strict equality-based relationships of conventional integrity constraints, the expressiveness is limited compared with the matching keys [7] where more rich metric distance restrictions are considered. Consequently, as the experimental results reported in Figures 5 and 6, the accuracy of our proposed GA (with matching keys) is higher than that of constraint-based approach (FD).

7. CONCLUSIONS

Matching keys specify *what attributes to compare* and *how to compare them* for record matching. Owing to the exis-

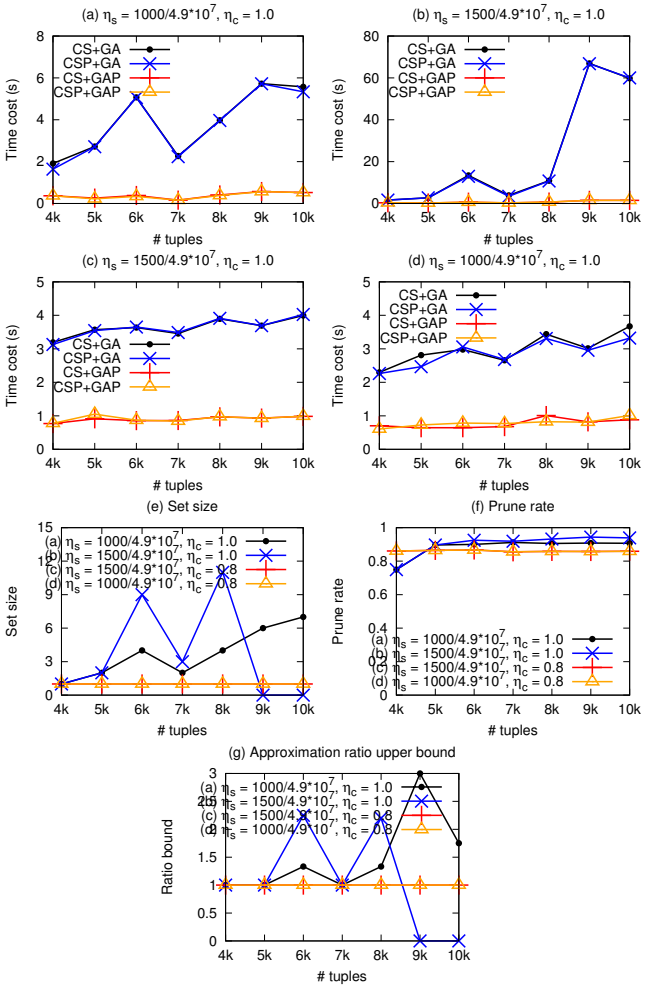

Figure 9: Discovery performance on *UIS*

tence of redundant semantics among matching keys, it is highly demanded to explore the most concise set of matching keys with less redundancy. While relative candidate keys (RCKs) can clear up redundant semantics w.r.t. "what attributes to compare" (minimal on the number of compared attributes), redundancy issues may still exist among RCKs on the same attributes about "how to compare them". In this paper, we introduce the greedy discovery algorithm with a bound on approximation ratio. To ensure the quality of matching keys, the return results are guaranteed to be RCKs (minimal w.r.t. attributes), and also minimal w.r.t. distance restrictions (i.e., redundancy free w.r.t. "how to compare the attributes"). Experiment results demonstrate that our concise RCK set is more effective in the evaluation over the existing record matching methods. Moreover, the proposed pruning techniques can significantly improve the efficiency of concise RCK set discovery.

We believe that the proposed pruning technique can be applied in solving other similar problems. For example, a sequential dependency [9] SD: ($date \rightarrow_{[20,40]} price$) identifies stock prices that rapidly increase from day to day (by at least 20 points but no greater than 40). The interval *[20,40]* denotes a range of distance between two tuples on attribute price, which has the same meaning as the distance interval of matching keys studied in this paper. By adapting the algorithm for dealing with the order relationships on attribute date, the proposed pruning technique can be applied in determining the proper interval on price for SDs.

Acknowledgment. This work is supported in part by China NSFC Grant No. 61202008, 61232018, 61370055, Hong Kong RGC/NSFC Project No. N_HKUST637/13, National Grand Fundamental Research 973 Program of China Grant No. 2012-CB316200, Microsoft Research Asia Gift Grant, Google Faculty Award 2013, Hong Kong RGC GRF Project No. CUHK 411211, 411310, CUHK Direct Grant No. 4055015.

8. REFERENCES

[1] Full version. *http://ise.thss.tsinghua.edu.cn/sxsong/doc/mkey.pdf.*

[2] M. Bilenko, R. J. Mooney, W. W. Cohen, P. Ravikumar, and S. E. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

[3] T. Calders, R. T. Ng, and J. Wijsen. Searching for dependencies at multiple abstraction levels. *ACM Trans. Database Syst.*, 27(3):229–260, 2002.

[4] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD Conference*, pages 201–212, 1998.

[5] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

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

[7] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *PVLDB*, 2(1):407–418, 2009.

[8] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

[9] L. Golab, H. J. Karloff, F. Korn, A. Saha, and D. Srivastava. Sequential dependencies. *PVLDB*, 2(1):574–585, 2009.

[10] L. Golab, H. J. Karloff, F. Korn, D. Srivastava, and B. Yu. On generating near-optimal tableaux for conditional functional dependencies. *PVLDB*, 1(1):376–390, 2008.

[11] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *SIGMOD Conference*, pages 127–138, 1995.

[12] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Efficient discovery of functional and approximate dependencies using partitions. In *ICDE*, pages 392–401, 1998.

[13] E.-P. Lim, J. Srivastava, S. Prabhakar, and J. Richardson. Entity identification in database integration. In *ICDE*, pages 294–301, 1993.

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

[15] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[16] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, pages 269–278, 2002.

[17] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[18] V. S. Verykios, G. V. Moustakides, and M. G. Elfeky. A bayesian decision model for cost optimal record matching. *VLDB J.*, 12(1):28–40, 2003.

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

[20] W. E. Winkler. Matching and record linkage. In *Business Survey Methods*, pages 355–384. Wiley, 1995.

[21] W. E. Winkler. Overview of record linkage and current research directions. Technical report, BUREAU OF THE CENSUS, 2006.