Indeterministic Handling of Uncertain Decisions in Deduplication

FABIAN PANSE, University of Hamburg, Germany
MAURICE VAN KEULEN, University of Twente, the Netherlands
NORBERT RITTER, University of Hamburg, Germany

In current research and practice, deduplication is usually considered as a deterministic approach in which database tuples are either declared to be duplicates or not. In ambiguous situations, however, it is often not completely clear-cut, which tuples represent the same real-world entity. In deterministic approaches, many realistic possibilities may be ignored, which in turn can lead to false decisions. In this article, we present an indeterministic approach for deduplication by using a probabilistic target model including techniques for proper probabilistic interpretation of similarity matching results. Thus, instead of deciding for one of the most likely situations, all realistic situations are modeled in the resultant data. This approach minimizes the negative impact of false decisions. Moreover, the deduplication process becomes almost fully automatic and human effort can be largely reduced. To increase applicability, we introduce several semi-indeterministic methods that heuristically reduce the set of indeterministically handled decisions in several meaningful ways. We also describe a full-indeterministic method for theoretical and presentational reasons.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications

General Terms: Theory, Algorithms, Experimentation

Additional Key Words and Phrases: Deduplication, Probabilistic Data, Uncertainty

ACM Reference Format:
Panse, F., van Keulen, M., and Ritter, N. 2013. Indeterministic handling of uncertain decisions in deduplication. *ACM J. Data Inform. Quality* 4, 2, Article 9 (March 2013), 25 pages.
DOI: http://dx.doi.org/10.1145/2435221.2435225

1. INTRODUCTION

In a variety of commercial and scientific situations, for example, in healthcare [Taddei et al. 2008] or bioinformatics [Goble and Stevens 2008], data from different sources need to be combined. For that reason, data integration [Lenzerini 2002] has become an important area of research. Nevertheless, data sets to be integrated may contain duplicates. Working with non-duplicate-free data, however, can do serious economic damage or can lead to incorrect conclusions in scientific applications. Therefore, deduplication [Elmagarmid et al. 2007; Naumann and Herschel 2010] is an important component in an integration process. Due to deficiencies like missing data, typos, data obsolescence or misspellings, real-life data are often incorrect and incomplete. Hence, it often cannot be determined with absolute certainty from the data itself that two or more database tuples belong to the same real-world entity. This principally hinders deduplication and is a crucial source of uncertainty.

Author's address: F. Panse, University of Hamburg, Vogt-Koelln-Strasse 30, 22527 Hamburg, Germany; email: panse@informatik.uni-hamburg.de.

DOI: http://dx.doi.org/10.1145/2435221.2435225

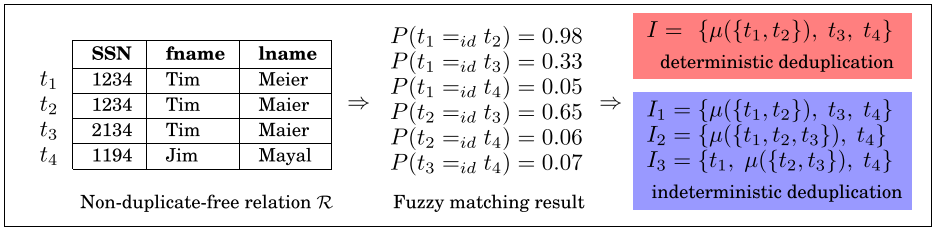


Fig. 1. Deterministic vs. indeterministic deduplication.

Most current deduplication approaches acknowledge many kinds of uncertainty and often apply fuzzy matching techniques, but in the end they still are deterministic: finally an absolute decision needs to be taken either by (1) deferring the situation to domain experts (clerical review) which is expensive and time consuming, or (2) choose one of the most likely configurations thereby risking a wrong choice with all consequences this may have.

By using probabilistic target models, however, such determinism is not necessary. Instead any kind of uncertainty arising in duplicate decisions can be accurately modeled in the resultant data. In this way, all significantly likely duplicate mergings find their way into the database, hence any query answer or other derived data will reflect the inherent uncertainty. This concept may protect against negative impact resulting from false duplicate decisions. Moreover, expensive tuning of thresholds and the even more expensive clerical reviews can be avoided. Due to the fact that no decisions are taken (the decisions are handled indeterministically), we denote this concept as an indeterministic deduplication.[1]

In traditional deduplication approaches "old" tuples are usually kept around. However, this means that they are stored separately and hence cannot be uniformly queried with the deduplicated relation. This may be very well for applications needing concrete results (whether or not certain), but is not adequate for a lot of other applications as consistent query answering or data mining tasks (see Section 6).

1.1. Motivating Example

As an illustration, consider relation $\mathcal{R}$ of Figure 1. All 4 tuples bear some resemblance, so, in theory, there are 15 possible ways to deduplicate this table. By making a few realistic assumptions, however, we can easily reduce this number. For example, it is quite certain that t_4 represents a different entity here. This leaves 4 possible worlds for which the 4th can also be rejected, because it can be realistically argued that the situation where t_1 and t_3 represent the same real-world entity (notation $t_1 =_{id} t_3$) while t_2 is not, cannot be true: For $t_1 =_{id} t_3$, two typos each one in the *SSN* and the *lname* need to have occurred. Nevertheless, each of these typos individually would have resulted in t_2. Thus, the depicted three worlds I_1-I_3 are the only realistic ones.

Typically, fuzzy matching techniques do expose more than one possibility, in our example by assigning significant probabilities to $t_1 =_{id} t_3$ and $t_2 =_{id} t_3$ as well. Deterministic deduplication approaches, however, need to defer such ambiguous situations to expensive clerical reviews or simply decide on the most likely situation, which is I_1 in our example. However, only merging t_1 and t_2 (notation $\mu(\{t_1, t_2\})$) may be a false decision. By ignoring the other two possibilities, any use of this result or data derived

[1]Note, only the deduplication result is of an indeterministic nature, the deduplication process itself is deterministic, that is, it always produces the same result if same configuration settings and same input data are given.

from it may be wrong as well. Moreover, this may go unnoticed for a long time. In contrast to deterministic approaches where always a single world results, by using an indeterministic approach, we are able to maintain all three realistic database instances I_1, I_2 and I_3.

1.2. Contribution

Although data integration is regarded as an important application area for probabilistic databases, much work focuses on modeling the uncertainty in schema matching [Dong et al. 2009] or in the merge result of conflicting tuples [Tseng et al. 1993]. Modeling the uncertainty around possible duplicate representations, is much less researched. Exceptions can be found in Beskales et al. [2009], and Ioannou et al. [2010]. Both approaches store the probabilistic result in a special data model. Beskales et al. [2009] embrace the concept of indeterministic deduplication for data cleaning. They define a tailor-made data model that is restricted if the data should be further processed such as in subsequent integration steps. Ioannou et al. [2010] produce indeterministic deduplication results at query time by processing the entity data along with probabilistic linkages both stored in a *probabilistic linkage database*.

In general, deduplication is required in many application areas and hence the resultant data should be further processed in a variety of ways. Using a tailor-made data model, however, is usually too specific to cover all these ways. For that reason, we propose to model ambiguous situations within the possible world semantics and store the resultant data using a traditional probabilistic databases like Trio [Benjelloun et al. 2006] or MayBMS [Koch 2009]. In this way, the resultant data are modeled in a more general fashion and we benefit from the strong querying power supported by these databases, which has been extensively studied in the past.

Our approach is, in essence, a generic graph-based process that starts from a graph representing matching similarities and that results in a set of graphs representing multiple possible worlds that are subsequently stored in a probabilistic database. Due to the inherent complexity of a full-indeterministic approach (an approach in which each uncertain decision is handled indeterministically) is usually not manageable, we introduce some semi-indeterministic methodes which reduce the volume of resultant uncertain data to a manageable size making the indeterministic deduplication feasible in practice. Moreover, we present techniques for probabilistic interpretation of similarity matching results. Although our approach is generic, the final step of modeling the resultant uncertainty in probabilistic data depends on the used probabilistic data model. In this article, we use the ULDB model from the Trio database as a representative, but using another model, for example, MayBMS, is also possible.

1.3. Outline

The article is structured as follows. First, we examine related work in Section 2. Then, we discuss deterministic techniques for deduplication (Section 3.1), outline probabilistic data models (especially, the ULDB model) and show how data lineage can be used to faithfully model tuple dependencies (Section 3.2). In Sections 4 and 5, we propose our indeterministic approach. This is done by first clarifying the problem, then presenting a theoretical full-indeterministic method, which is finally refined to several semi-indeterministic methods. Moreover, we discuss sources of matching probabilities. In Section 6, we consider querying indeterministic deduplication results. Finally, in Section 7, we show by some experiments on a real data set how efficient and effective our approach can be if semi-indeterministic methods are used. Section 8 concludes the article and gives an outlook on future research.

2. RELATED WORK

In general, duplicate detection is already handled in several works (for overviews, see Elmagarmid et al. [2007], Naumann and Herschel [2010], and Talburt [2011]). However, even though in the most of these works uncertainty in tuple matching is considered by using different measures of similarity, the decision whether two tuples are duplicates or not is always made in a deterministic way.

There are several approaches using probabilistic data models for handling uncertainties in tuple merging. In de Keijzer and van Keulen [2008], a semi-structured probabilistic model is used for handling ambiguities in deduplication of XML data. Tseng et al. [1993] already used probabilistic values to resolve conflicts between two or more certain relational values. None of the studies, however, handle the uncertainty of duplicate decisions in detecting relational duplicates.

A probabilistic handling of uncertain duplicate decisions is proposed by Beskales et al. [2009]. In this approach, deduplication is considered as a data cleaning task and uncertainty in duplicate decisions is handled by using a set of possible repairs. In contrast to our graph-based approach using the possible world semantics, the authors restrict to hierarchical tuple clusterings. Thus, our approach is more general, which can be specialized to the hierarchical clustering approach by using an HC-restriction (see Section 5.3). Moreover, for representing possible repairs, Beskales et al. [2009] define a new and specific uncertain data model. In contrast, since our approach is based on the possible world semantics, several existing traditional probabilistic data model as ULDB or MayBMS can be used. As we think, this increases the reusability of the resultant data, especially if deduplication is considered as a step in a data integration process.

Another indeterministic approach was introduced in Ioannou et al. [2010]. Based on linkage information, Ioannou et al. [2010] decide at query time (on the fly) which of the query-relevant tuples are duplicates and have to be merged. Although, this and our approach are similar to a large extent, there are quite some differences: First, Ioannou et al. [2010] use their own probabilistic data model (called *probabilistic linkage database*) that supports only simple queries with projections and selections. In contrast, our approach is based on the usage of traditional probabilistic databases for which efficient querying is already researched in an exhaustive way [Dalvi and Suciu 2007; Koch 2008]. Thus, complex queries with joins, grouping, aggregations, subqueries etc. can be efficiently performed on the resultant data (see Section 6). Moreover, using a linkage database or using a traditional probabilistic database results in two fundamental different approaches for handling uncertain decisions in deduplication. Ioannou et al. [2010] perform an offline tuple matching, store the most uncertain duplicate decisions (linkages) for single tuple pairs persistently in their special database and finally perform the possible world creation at query time. In contrast, similar to Beskales et al. [2009], we do not store the uncertain tuple matching results persistently, but immediately perform the possible world creation on the given tuple matchings and finally store the resultant worlds in a probabilistic database. Thus, in the approach of Ioannou et al. [2010], queries are applied to stored linkages, and in our approach, queries are applied to stored possible worlds.

Furthermore, in their on-the-fly entity-aware query processing only a small number of linkages exists. Since in reality for a lot of tuple pairs a nonlinkage (probability of 0) cannot be taken with absolute certainty, we understand their approach in the way that they only consider linkages with high probability and then perform the transitive closure of them for creating factors. This guarantees that factors are always very small, which makes the querying efficient, but also poses two kinds of problems: (1) they cannot ensure that always the most probable worlds result and (2) they disregard

Input: tuple pair (t_i, t_j), comparison vector $\vec{c}_{ij} = [c_1^{ij}, \ldots, c_n^{ij}]$

1. Execution of the matching function $\varphi(\vec{c}_{ij})$
   $\Rightarrow$ Result: $sim(t_i, t_j) \in \mathbb{R}$
2. Classification of (t_i, t_j) into $\{M, U\}$ based on $sim(t_i, t_j)$
   $\Rightarrow$ Result: $(t_i, t_j) \mapsto \{M, U\}$

Output: Decision whether (t_i, t_j) is a duplicate or not

Fig. 2. General representation of decision models.

negative information (certain non-duplicate decisions), which makes the result more inaccurate and which makes an introduction of negative expertise during the initially offline performed linkage creation impossible. For an illustration of these drawbacks, we include an example in Appendix A.

Finally, Ioannou et al. do not discuss the sources of their linkage probabilities as we will do in Section 5.2. On the other hand, they consider uncertainty on data level from which we currently abstract.

3. BACKGROUND

Due to it is required background for our proposed approach, we shortly present deterministic deduplication and probabilistic data models, especially, the ULDB model.

3.1. Deterministic Deduplication

Traditional approaches for deterministic deduplication are based on pairwise tuple comparisons and consist of four main phases [Naumann and Herschel 2010]:

(1) *Attribute Value Matching*. First, for each tuple pair the similarities of their attribute values are measured. Despite data preparation, syntactic as well as semantic irregularities remain. Thus, attribute value similarity is quantified by syntactic (e.g., edit-based distance or token-based distance [Elmagarmid et al. 2007]) and semantic (e.g., glossaries or ontologies) means. From comparing two tuples, we obtain a normalized *comparison vector* $\vec{c} = [c_1, \ldots, c_n]$, where $c_i \in [0, 1]$ represents the similarity of the values from the ith attribute.

(2) *Decision Model*. The comparison vector is input for a decision model that classifies a given tuple pair into matching tuples (M) or unmatching tuples (U). Common decision models [Elmagarmid et al. 2007] are based on probability theory [Fellegi and Sunter 1969; Newcombe et al. 1959], identification rules [Hernández and Stolfo 1995; Wang and Madnick 1989], distance measures [Koudas et al. 2004] or learning techniques [Ravikumar and Cohen 2004].
In general, the decision whether a tuple pair (t_i, t_j) is a match or an unmatch can be decomposed into two steps (see Figure 2). In the tuple matching step (Step 1), based on the comparison vector a single similarity degree $sim(t_i, t_j)$ is determined by a *matching function*:

$$\varphi : [0, 1]^n \to \mathbb{R} \qquad sim(t_i, t_j) = \varphi(\vec{c}_{ij}). \tag{1}$$

In the classification step (Step 2), based on the similarity $sim(t_i, t_j)$ the tuple pair is assigned to M or U. To minimize the number of uncertain decisions, in most approaches a third set of possibly matching tuples (P) is intermediately introduced. Each tuple pair originally classified to P is later manually assigned to M or U by domain experts (clerical reviews). Usually, the classification is based on two tuple similarity thresholds T_λ and T_μ that demarcate the boundaries between the sets M, P, and U (see Figure 6).

	name	firm	$p(t)$
t_1	Tim	Oracle	0.3
	Jim	Nokia	0.7
t_2	Tim	IBM	0.6

?

	firm	industry	$p(t)$
t_3	IBM	software	1.0
t_4	Oracle	software	1.0
t_5	Nokia	cell-phone	1.0

Fig. 3. X-relation $\mathcal{R}_1$ and x-relation $\mathcal{R}_2$.

(3) *Duplicate Clustering*. Based on the decisions taken for individual tuple pairs a globally consistent result is achieved by using a duplicate clustering technique. Simplest, clustering can be achieved by using the transitive closure of detected matches. More complex, but also more promising approaches are presented in Naumann and Herschel [2010] and Hassanzadeh et al. [2009].
(4) *Tuple Merging*. After detecting multiple duplicates, these various representations have to be merged (also known as fusion [Bleiholder and Naumann 2008]) into a single tuple. In our work, we focus on handling uncertainty in duplicate decisions and abstract from merging details. In the following, we assume an associative and idempotent merging function μ, where $t = \mu(T)$ represents the tuple resulting from merging the tuples of set T. Since, we use a probabilistic target model, merging cannot be only realized by conflict resolution, but also by creating a probabilistic tuple with all the base-tuples as alternatives.
For reasons of clarity and comprehensibility, in following examples, the index of a merged tuple is an ordered concatenation of the indexes of the tuples it is merged from. For example, the result of $\mu(\{t_1, t_2, t_3\})$ is denoted by t_{123}.

In summary, the outcome of using a deterministic approach is only one of several (maybe equally probable) possible worlds. On the contrary, the usage of probabilistic data models allows for constructing and later querying all these worlds simultaneously. Matching of attribute values and tuple merging is required in both concepts and hence will be reused in our indeterministic approach as it is.

3.2. Probabilistic Data Models

Theoretically, a probabilistic database is defined as $PDB = (W, P)$ where $W = \{I_1, \ldots, I_n\}$ is the set of possible worlds and $P : W \rightarrow (0, 1]$, $\sum_{I \in W} P(I) = 1$ is the probability distribution over these worlds. Because the data of individual worlds often considerably overlaps and it is sometimes even impossible to store them separately, a succinct representation has to be used.

In probabilistic relational models, uncertainty is modeled on two levels: (a) each tuple t is assigned with a probability $p(t) \in (0, 1]$ denoting the likelihood that t belongs to the corresponding relation, and (b) alternatives for attribute values are given. In earlier approaches, alternatives of different attribute values are considered to be independent (e.g., Barbará et al. [1992]). In these models, each attribute value can be considered as a separate random variable with its own probability distribution. Newer models like ULDB [Benjelloun et al. 2006] or MayBMS [Koch 2009] support dependencies by introducing new concepts like ULDB's x-tuple and MayBMS's U-relation. Both models support all the concepts required for our purpose. Since we are more familiar with the ULDB model, we consider it as a representative throughout this paper. Nevertheless, we shortly discuss a usage of MayBMS in Appendix B.

3.2.1. ULDB: A Model for Uncertainty and Lineage. For modeling dependencies between attribute values, in the ULDB model the concept of x-tuples is introduced. An x-tuple t consists of one or more alternatives $(t^1, \ldots, t^n)$ which are mutually exclusive. *Maybe* x-tuples (tuples for which non-existence is possible, that is, for which the probability sum of the alternatives is smaller than 1) are indicated by "?". Relations containing

	name	industry	$p(t)$		
t_6	Jim	cell-phone	0.7	?	$\lambda(6,1) = ((1,2) \wedge (5,1))$
t_7	Tim	software	0.9	?	$\lambda(7,1) = ((1,1) \wedge (4,1)) \vee ((2,1) \wedge (3,1))$

Fig. 4. X-relation $\mathcal{R}_3$ with its lineage information.

one or more x-tuples are called x-relations. As an example, see the x-relations $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ in Figure 3 and Figure 4.

Besides data uncertainty, the ULDB model supports the concept of data lineage. The lineage of a data item contains information about its derivation. In the ULDB model, lineage is considered at the granularity of x-tuple alternatives and is defined as a Boolean function λ over the presence of other alternatives.

An example of lineage is shown in Figure 4. Relation $\mathcal{R}_3$ results from a natural join of $\mathcal{R}_1$ with $\mathcal{R}_2$ and a subsequent projection on the attributes *name* and *industry*. Let (i,j) denote the jth alternative of x-tuple t_i (outside of lineage also shortly noted as t_i^j). The lineage formula $\lambda(7,1) = ((1,1) \wedge (4,1)) \vee ((2,1) \wedge (3,1))$ for the single alternative of t_7 expresses the information that this alternative is derived from the first alternatives of t_1 and t_4 or from the first alternatives of t_2 and t_3.

An interesting and useful feature of lineage is that the probability of an alternative can be computed from the probabilities of the data items in its lineage. Moreover, an x-tuple alternative with lineage can belong to a possible world only, if its lineage condition is satisfied by the presence of the referenced alternatives in the considered world. As a consequence, lineage imposes restrictions on possible worlds. For example, if the alternative t_6^1 is not present in a possible world I_1 then alternative 1 must be chosen for x-tuple t_1, and hence x-tuple t_7 must be present in I_1.

3.2.2. Tuple Dependencies. A general framework for modeling tuple dependencies in probabilistic data is proposed in Sen and Deshpande [2007]. For representing the indeterministic deduplication result, however, only a modeling of a specific kind of mutual exclusion, which we denote as *complementation*, is required.

Definition 1 (*Complementation*). The tuple sets $\mathcal{A} = \{A_1, \ldots, A_k\}$ are complementing (short $\mathfrak{cpl}(\mathcal{A})$) in a probabilistic database $PDB = (W, P)$, iff their existence are jointly exhaustive and mutually exclusive events in W. Naturally speaking either all tuples of A_1 exist and none of $A_2, \ldots, A_k$ that is not in A_1, or all tuples of A_2 exists and none of $A_1, A_3, \ldots, A_k$ that is not in A_2, and so on: $\mathfrak{cpl}(\mathcal{A}) \Leftrightarrow (\forall I \in W) : (\exists A_i \in \mathcal{A}) : A_i \subseteq I \wedge (\forall t \in ((\bigcup \mathcal{A}) - A_i)) : t \notin I$.

3.2.3. Modeling Tuple Dependencies in ULDB. For modeling complementations within the ULDB model, we use the concept of data lineage and create a specific catalog relation called *tuple dependency-indicator* (short $\mathcal{I}_{td}$). In more detail, for modeling the complementation $\mathfrak{cpl}(\{A_1, \ldots, A_k\})$ of k x-tuple sets, we create one indicator x-tuple $i \in \mathcal{I}_{td}$ with k alternatives so that $\sum_{j \in \{1,\ldots,k\}} p(i^j) = 1$. Whereas the x-tuples of the first set have a lineage to the first alternative of i, the x-tuples of the second set have a lineage to its second alternative, and so forth. Note, in such cases, the new lineage conditions hold for the whole x-tuple and hence for all of its alternatives. Thus, we consider lineage on x-tuple granularity.

Because the alternatives of i are mutually exclusive, this dependency holds for the x-tuple sets, too. Due to the fact that i is not maybe, one of the x-tuple sets exists for sure. In other words, we use the complementation of x-tuple alternatives to model complementing sets of x-tuples.

Since in data lineage the presence of alternatives can be negated (e.g., $\neg(i,1)$), theoretically instead of k only $k-1$ indicator alternatives are sufficient. In this case,

x-tuple	lineage
t_1	$\lambda(t_1) = (i_1, 1)$
t_2	$\lambda(t_2) = (i_1, 1)$
t_{12}	$\lambda(t_{12}) = \neg(i_1, 1)$

indicator		
id	**val**	$p(i)$
i_1	1	0.4

Fig. 5. Modeling $\mathfrak{cpl}(\{\{t_1, t_2\}, \{t_{12}\}\})$ in $\mathcal{R}_X$ (left) with the indicator relation $\mathcal{I}_{td}$ (right).

the indicator tuple becomes maybe and the fact that one x-tuple set must exists is modeled by the used negation. Nevertheless, for query-processing reasons, it is desirable to minimize the complexity of lineage formulas. Thus, we use negation only for modeling mutual exclusions between two tuple sets ($k = 2$). Otherwise, we always use k indicator alternatives.

As an example, we consider two certain tuples (x-tuples with one alternative) t_1 and t_2 of a relation $\mathcal{R}$, which are duplicates with a probability of 60%. To model the two possible worlds resulting from this uncertain duplicate decision, we have to ensure that either the tuples t_1 and t_2 or the merged tuple $t_{12} = \mu(\{t_1, t_2\})$ belong to the resultant x-relation $\mathcal{R}_X$. To represent this complementation, we need an indicator x-tuple i_1 of the catalog relation $\mathcal{I}_{td}$ having the single alternative $i_1^1 = 1$ with a probability of 40%. By creating the lineages $\lambda(t_1), \lambda(t_2)$ and $\lambda(t_{12})$ as depicted in Figure 5, we can guarantee that always one of these x-tuple sets exist, but we can exclude that both x-tuple sets belong to a same possible world. In our case, all source tuples are certain. Thus, the probabilities of t_1, t_2 and t_{12} result in $p(t_1) = p(t_2) = p(i_1^1) = 0.4$ and $p(t_{12}) = 1 - p(i_1^1) = 0.6$.

3.2.4. Querying Probabilistic Data. Querying probabilistic data has been extensively studied in the past and is still an active field of research [Dalvi and Suciu 2007; Koch 2008, 2009; Sarma et al. 2008; Suciu et al. 2011]. In general, queries on probabilistic data can be evaluated in an intensional or an extensional manner [Suciu et al. 2011]. The principle of intensional query semantics is to build a propositional formula (like the ULDB's lineage) for each result tuple during query evaluation and then to compute probabilities based on these formulas and the source data in a subsequent step. In contrast, in extensional query semantics, probability computation is directly included in query evaluation. This, semantics is more efficient, but correct probabilities can only be ensured for some specific classes of queries (see concept of "safe queries" in Dalvi and Suciu [2007]). Probability computation of intensional queries is based on probabilistic inference and can be performed exactly, for example, by variable elimination [Dechter 1996], or roughly by approximation techniques [Koch 2008] (e.g., by Monte-Carlo estimation [Karp and Luby 1983]). In our research, we focus on the two probabilistic databases Trio [Widom 2009] and MayBMS [Koch 2009] and hence base ourselves on the manuals for TriQL[2] (The Trio Query Language) and MayBMS-SQL[3]. Trio computes probabilities by exploiting the result tuples' lineages and hence uses intensional query semantics [Sarma et al. 2008]. MayBMS provides techniques for both semantics (intensional and extensional) [Koch 2009]. We consider querying indeterministic deduplication results in Section 6.

4. PROBLEM DESCRIPTION AND MOTIVATION

The use of deterministic deduplication presented in Section 3.1 results in an elementary problem to solve (for illustration see Figure 6): The greater the distance between the two thresholds T_λ and T_μ, the lower is the number of false decisions (sum of yellow

[2]http://infolab.stanford.edu/~widom/triql.html
[3]http://maybms.sourceforge.net/

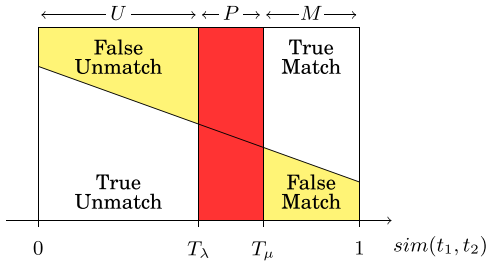


Fig. 6. Trade-off between effectiveness and human effort [Batini and Scannapieco 2006].

areas), but the higher is the number of possible matches which have to be resolved by domain experts (red area). In general, for financial and processing-time-based reasons, clerical reviews have to be reduced to a minimum. Nevertheless, data of high quality result only from an effective deduplication. As a consequence, in existing approaches a trade-off between the effectiveness of the deduplication process and the human effort resulting from clerical reviews has to be accepted. This trade-off, however, is not required if a probabilistic target model is used. Instead, uncertain decisions can be handled indeterministically and both, the number of false decisions as well as human effort, can be largely reduced.

This problem goes hand in hand with three other challenges:

(1) In many applications (e.g., dynamic data integration) a full-automatic deduplication is required ($T_\lambda = T_\mu$). Whatever a value for T_λ is used, always a single world results which is selected without the help of human expertise and hence must not be the most probable. As shown in Figure 6, the number of false decisions grows extremely in this situation. In contrast, by using an indeterministic approach the deduplication process can be fully automatized without accepting such a high rate of false (un)matches as it results from a deterministic one, because multiple possible worlds are considered.
(2) In general, resolving uncertain decisions is extremely costly in terms of time. As a rough guide, 10% of duplicate decisions cause 90% of the required processing time. Thus, the whole integration process need not become blocked because of a small amount of ambiguous matches that need clerical review. By using an indeterministic handling of uncertain decisions, the uncertainty of the ambiguous matches is intermediately modeled in the resultant data and can be resolved later after the integration process is finished (see the concept of good-is-good-enough integration in de Keijzer and van Keulen [2008]).
(3) In a deterministic approach, a domain expert is always forced to decide. All-knowing experts, however, are not a realistic assumption. In contrast, even experts are often not aware about the real-world state. As a consequence, in ambiguous situations, experts only have the choice between making an uncertain decision or to spend more time for further investigations. Both options, however, are not desirable in many deduplication processes.

5. INDETERMINISTIC DEDUPLICATION

The problems we have stated in the previous section arise, because in decision models (see Section 3.1) uncertainty is ignored during the classification of tuple pairs into M, U (or P) (Step 2). Such classifications, however, are not enforced, if a probabilistic target model is used. In contrast, if similarity between tuples can be mapped to the probability that both tuples are duplicates (in the following, denoted as *matching probability*), probabilities of possible worlds can be derived.

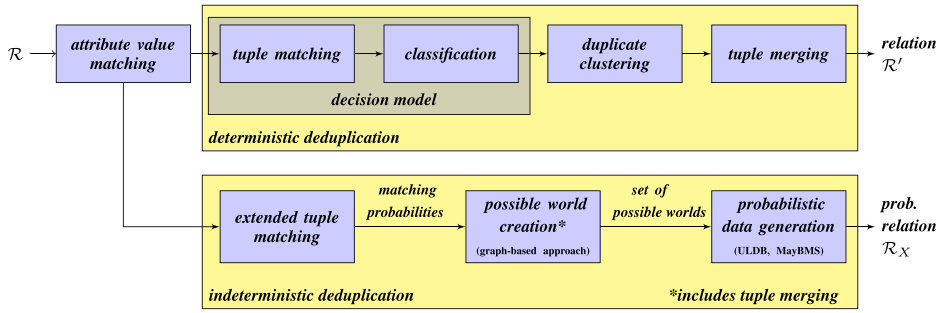


Fig. 7. Execution phases of a(n) (in)deterministic deduplication process.

Input: tuple pair (t_i, t_j), comparison vector $\vec{c}_{ij} = [c_1^{ij}, \ldots, c_n^{ij}]$

1. Calculation of tuple similarity by the matching function $\varphi(\vec{c}_{ij})$
   $\Rightarrow$ Result: $sim(t_i, t_j) \in \mathbb{R}$
2. Mapping from similarity to probability by $\rho(sim(t_i, t_j))$
   $\Rightarrow$ Result: $p(t_i, t_j) \in [0, 1]$

Output: Probability whether (t_i, t_j) is a duplicate

Fig. 8. General representation of the extended tuple matching phase.

As intuitively known and as formally examined in Appendix D, due to the high number of resultant possible worlds an indeterministic handling of all uncertain decisions (full-indeterministic approach) is usually not manageable. For that reason, we introduce some semi-indeterministic strategies that carefully reduce the number of indeterministically handled decisions in Section 5.3. Since such strategies can be seen as restrictions on the full-indeterministic approach, we present the latter first.

5.1. Full-Indeterministic Approach

In the full-indeterministic approach, the three phases, decision model, duplicate clustering and tuple merging, are replaced by three other phases (see Figure 7). Similar to the first decision model step, initially for each tuple pair a tuple matching is applied, where after similarity computation a matching probability is computed (Phase 1). Based on the computed matching probabilities a set of possible worlds is derived (Phase 2). Finally, depending on the used target model, a probabilistic result relation representing all these worlds needs to be generated (Phase 3). Tuple merging is included in the phase of possible world creation.

5.1.1. Extended Tuple Matching (Phase 1). In the tuple matching phase, two tuples are first matched by computing their tuple similarity (Figure 8, Step 1). As known from the first decision model step (see Figure 2), the similarity of two tuples t_i and t_j results from applying a *matching function* $\varphi(\vec{c}_{ij})$.

Since we want to interpret matching results as the probability that both tuples are duplicates ($P(t_1 =_{id} t_2)$, short $p(t_i, t_j)$), a mapping from tuple similarity to matching probability (*sim2p-mapping*) is required (Figure 8, Step 2). In the following, the function used for the *sim2p-mapping* is denoted as ρ:

$$\rho : \mathbb{R} \rightarrow [0, 1] \qquad p(t_i, t_j) = \rho(sim(t_i, t_j)). \tag{2}$$

In several decision models, for example, identification rules (e.g., Hernández and Stolfo [1995]), the similarity of two tuples is defined as the certainty that both tuples

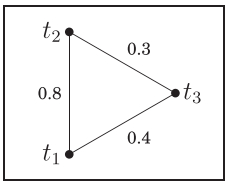


Fig. 9. The sample *M-graph* $M = (N, E, \gamma)$ with $N = \{t_1, t_2, t_3\}$, $E = \{\{t_1, t_2\}, \{t_1, t_3\}, \{t_2, t_3\}\}$, $\gamma = \{\{t_1, t_2\} \mapsto 0.8, \{t_1, t_3\} \mapsto 0.4, \{t_2, t_3\} \mapsto 0.3\}$.

are duplicates. Thus, in these cases, tuple similarity can be directly used as matching probability. Other sources of matching probabilities are discussed in Section 5.2.

5.1.2. Possible World Creation (Phase 2). In the second phase, a set of possible worlds is derived from the matching probabilities in four steps (algorithm for some of the individual steps are presented in Appendix C). For reasons of presentation, we define possible world creation as a graph-based process. For this purpose, we define two kinds of graphs: a *matching-graph* representing tuple matching results and *world-graphs* each representing a conceivable world.

Generation of the Initial Matching-Graph. A *matching-graph* is a weighted undirected graph, where each node represents a base-tuple. Two nodes are connected with an edge, if the corresponding tuples have been matched in the tuple matching phase.[4] The weight of an edge denotes the probability that the connected tuples are duplicates. An exemplary *matching-graph* is shown in Figure 9.

Definition 2 (*Matching-Graph*). A *matching-graph* (short M-graph) is a triple $M = (N, E, \gamma)$ where N is a set of nodes, $E \subseteq \{\{a, b\} \mid a, b \in N\}$ is a set of undirected edges and γ is a weight function $\gamma : E \rightarrow [0, 1]$ denoting matching probabilities.

We call an edge to be uncertain, if its weight is between 0 and 1 ($\gamma \in (0, 1)$). The set of *definite positive edges* ($\gamma = 1$) is denoted by E^+, the set of *definite negative edges* ($\gamma = 0$) is denoted by E^- and the set of *uncertain edges* is denoted by $E^?$.

Generation of World-graphs. A *world-graph* is an unweighted undirected graph representing one conceivable world where edges denote that the associated tuples are declared to be duplicates.

Definition 3 (*World-Graph*). A *world-graph* (short *W-graph*) is a triple $G = (N, E, P)$ where N is a set of nodes, $E \subseteq \{\{a, b\} \mid a, b \in N\}$ is a set of undirected edges and P is the probability of the corresponding world.

From the initial *M-graph* a set of *W-graphs* can be derived by removing all *definite negative edges* and by eliminating each *uncertain edge* by either removing it or replacing it by a *definite positive edge*. The process of *W-graph* generation is formalized by the mapping $\nu : \mathcal{M} \mapsto 2^{\mathcal{G}}$, where $\mathcal{M}$ is the set of all possible *matching-graphs* and $2^{\mathcal{G}}$ is the power set of all possible *world-graphs*. Let $M = (N, E, \gamma)$ be the initial *M-graph*, the mapping ν is defined as:

$$\nu(M) = \bigcup_{K \in 2^{E^?}} \{(N, E^+ \cup K, \prod_{e \in K} \gamma(e) \prod_{e \in E^? \setminus K} (1 - \gamma(e)))\} \quad (3)$$

More illustrative, we exactly create one *W-graph* for each possible combination $K \in 2^{E^?}$ of uncertain edges. All *W-graphs* have the same nodes (the set N) as the initial *M-graph* and contain each edge $e \in E^+$. The probability of each *W-graph* results from the

[4]In processes without search space reduction, the *matching-graph* is complete.

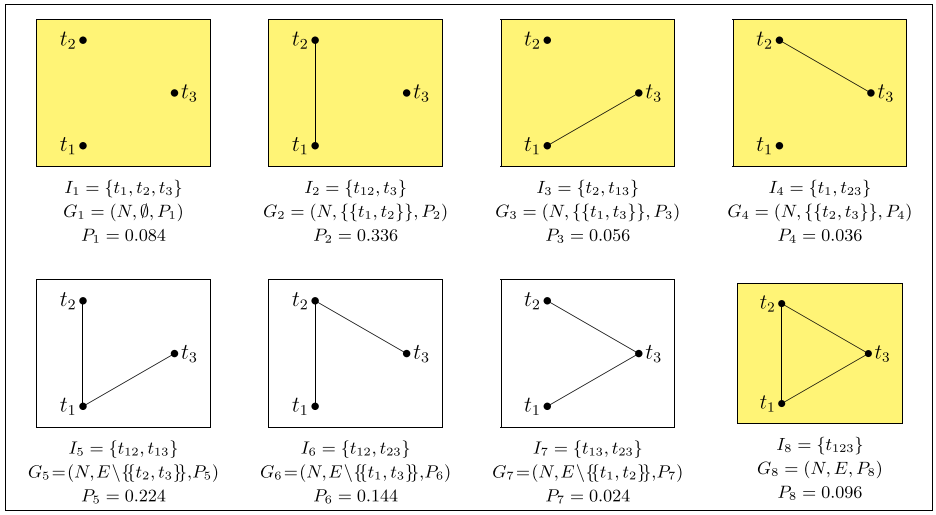


Fig. 10. The worlds I_1-I_8 with their corresponding *W-graphs*.

weights of the edges belonging to this *W-graph* and the inverse weights of the edges not belonging to this *W-graph*.

As an example, we consider the *M-graph* M from Figure 9. The base-tuples t_1, t_2 and t_3 are pairwise compared with each other and have the matching probabilities $p(t_1, t_2) = 0.8$, $p(t_1, t_3) = 0.4$, and $p(t_2, t_3) = 0.3$. Based on these probabilities, eight worlds along with their corresponding *W-graphs* can be derived (see Figure 10).

Removing Inconsistent World-Graphs. By definition, identity is a transitive relation. Worlds in which transitivity is not valid are considered impossible.

Definition 4 (*Possible World*). A world I is possible, if and only if $(\forall t_1, t_2, t_3 \in I) : t_1 =_{id} t_2 \wedge t_1 =_{id} t_3 \Rightarrow t_2 =_{id} t_3$.

A *W-graph* is called consistent, if it represents a possible world. An *M-graph* is consistent, if at least one consistent *W-graph* can be derived from it.

THEOREM 1. *A* W-graph $G = (N, E, P)$ *that represents world* I *is consistent, if and only if* G *is equivalent to its transitive closure:* $G = G^*$.

PROOF. ($\Rightarrow$) Assumption: $G \neq G^*$, but G is consistent.
$\Rightarrow (\exists t_1, t_2, t_3 \in N) : \{t_1, t_2\}, \{t_1, t_3\} \in E \wedge \{t_2, t_3\} \notin E$
$\Rightarrow (\exists t_1, t_2, t_3 \in I) : t_1 =_{id} t_2 \wedge t_1 =_{id} t_3 \wedge \neg(t_2 =_{id} t_3)$
$\Rightarrow$ the world I is impossible
$\Rightarrow G$ is inconsistent □

PROOF. ($\Leftarrow$) Assumption: G is inconsistent, but $G = G^*$.
$\Rightarrow$ the world I is impossible
$\Rightarrow (\exists t_1, t_2, t_3 \in I) : t_1 =_{id} t_2 \wedge t_1 =_{id} t_3 \wedge \neg(t_2 =_{id} t_3)$
$\Rightarrow (\exists t_1, t_2, t_3 \in N) : \{t_1, t_2\}, \{t_1, t_3\} \in E \wedge \{t_2, t_3\} \notin E$
$\Rightarrow G \neq G^*$ □

THEOREM 2. *An* M-graph $M = (N, E, \gamma)$ *is consistent, if and only if* $(\forall t_1, t_2, t_3 \in N) : \gamma(\{t_1, t_2\}) = \gamma(\{t_1, t_3\}) = 1 \Rightarrow \gamma(\{t_2, t_3\}) > 0$.

PROOF. ($\Rightarrow$) Assumption: $(\exists t_1, t_2, t_3 \in N) : \gamma(\{t_1, t_2\}) = \gamma(\{t_1, t_3\}) = 1$
$\wedge\, \gamma(\{t_2, t_3\}) = 0$, but M is consistent.
$\Rightarrow (\forall G = (N', E', P) \in \nu(M)) : \{t_1, t_2\}, \{t_1, t_3\} \in E' \wedge \{t_2, t_3\} \notin E'$
$\Rightarrow (\forall G \in \nu(M)) : G$ is inconsistent
$\Rightarrow M$ is inconsistent □

PROOF. ($\Leftarrow$) Assumption: M is inconsistent, but
$(\forall t_1, t_2, t_3 \in N) : \gamma(\{t_1, t_2\}) = \gamma(\{t_1, t_3\}) = 1 \Rightarrow \gamma(\{t_2, t_3\}) > 0$.
$\Rightarrow (\forall G \in \nu(M)) : G$ is inconsistent
$\Rightarrow (\forall G = (N', E', P) \in \nu(M)) : (\exists t_1, t_2, t_3 \in N') : \{t_1, t_2\}, \{t_1, t_3\} \in E' \wedge \{t_2, t_3\} \notin E'$
$\Rightarrow (\exists t_1, t_2, t_3 \in N) : \gamma(\{t_1, t_2\}) = \gamma(\{t_1, t_3\}) = 1 \Rightarrow \gamma(\{t_2, t_3\}) = 0$ □

In the tuple-matching phase, tuple pairs are matched independently. Thus, worlds are created from independent considerations and hence can be impossible. Each inconsistent *W-graph* represents an impossible world and hence is removed from the set of considered graphs. In other words, dependencies between individually taken duplicate decisions are introduced by only considering consistent *W-graphs*.

We consider the example from Figure 10. Due to the transitivity of identity is violated, three ($\{I_5, I_6, I_7\}$) of the eight worlds are definitely not the true world and hence the *W-graphs* G_5, G_6, and G_7 have to be removed from further considerations.

After removing inconsistent *W-graphs* (impossible worlds), the probabilities of the remaining *W-graphs* (worlds) no longer sum up to 1. Therefore, the probabilities of the remaining *W-graphs* are conditioned with the event B that the true world must be a possible world (the probability of B is the overall probability of all remaining *W-graphs*). For instance, in our example, the conditioned probability of G_1 (and hence I_1) results in:

$$P(G_1 \mid B) = P(G_1)/P(B) = 0.084/0.608 = 0.138.$$

Generation of Possible Worlds. Finally, from each *W-graph* exactly one possible world has to be derived. Since all considered *W-graphs* are consistent, each *W-graph* $G = (N, E, P)$ can be divided into m maximally connected components $\{G_1, \ldots, G_m\}$. A component with only one node represents a base-tuple that is apparently not a duplicate, hence, it is included in the resultant world as it is (e.g., tuple t_3 in I_2). The tuples associated with a component consisting of multiples nodes have to be merged into one result tuple by using the merging function μ (e.g., in I_2 the tuples t_1 and t_2 are merged to t_{12}). Thus, from a given component $G_i = (N_i, E_i)$ with $N_i = \{t_1, \ldots, t_k\}$, the tuple $t_{G_i} = \mu(\{t_1, \ldots, t_k\})$ is derived.

Since all possible worlds (set W) are mutually exclusive and one of these worlds must exist, the tuple dependency $\mathfrak{cpl}(W)$ is implicitly given.

5.1.3. Generation of Probabilistic Data (Phase 3). In the last phase, a probabilistic database representing the resultant set of possible worlds needs to be generated. That generation, however, depends on the used target model.

As described in Section 3.2.2, for representing the resultant set of possible worlds within the ULDB model, we use an indicator tuple $i \in \mathcal{I}_{td}$ with $|W|$ alternatives. The resultant x-relation $\mathcal{R}_X$ contains each tuple belonging to at least one possible world. The new lineage for each of these tuples results in the disjunction of the indicator's alternatives representing the worlds this tuple belongs to. A complete algorithm for x-relation generation is shown in Appendix C.

For our example, we create an indicator x-tuple $i_1 \in \mathcal{I}_{td}$ with one alternative for each of the five possible worlds $\{I_1, I_2, I_3, I_4, I_8\}$ and generate the lineage for each tuple in the resultant x-relations $\mathcal{R}_X$ as described in Section 3.2.2 (see Figure 11).

x-tuple	lineage
t_1	$\lambda(t_1) = (i_1,1) \vee (i_1,4)$
t_2	$\lambda(t_2) = (i_1,1) \vee (i_1,3)$
t_3	$\lambda(t_3) = (i_1,1) \vee (i_1,2)$
t_{12}	$\lambda(t_{12}) = (i_1,2)$
t_{13}	$\lambda(t_{13}) = (i_1,3)$
t_{23}	$\lambda(t_{23}) = (i_1,4)$
t_{123}	$\lambda(t_{123}) = (i_1,5)$

indicator		
id	val	$p(i)$
i_1	1	0.138
	2	0.553
	3	0.092
	4	0.059
	5	0.158

Fig. 11. X-relation $\mathcal{R}_X$ (left) and x-relation $\mathcal{I}_{td}$ (right).

5.2. Sources of Matching Probabilities

The effectiveness of the indeterministic deduplication essentially depends on the taken matching probabilities. Nevertheless, most often, deriving adequate probabilities from tuple similarities is not trivial. In many cases, tuple similarity is directly derived from the similarities of their attribute values. The similarity $sim(a_1, a_2) = 0.5$ of two attribute values a_1 and a_2, however, does not necessarily imply that both values represent the same real-world property with a probability of 50%. In contrast, often the opposite is true. For example, it is very unlikely that "Sabine" and "Janina" both represent the first name of the same person. Using the normalized Levenshtein-distance, however, the similarity of both words is 0.5.

In general, the more similar two tuples are, the higher is the probability that they are duplicates. Thus, a *sim2p-mapping* must be monotonically nondecreasing.

Some possible sources of matching probabilities are the following.

(1) *Specifications Based on Empirical Analyses*. To receive adequate mappings from tuple similarity to matching probability, statistics can be used. For example, the probability that the tuples t_i and t_j are duplicates can be defined as the conditional probability $P(t_i =_{id} t_j | sim(t_i, t_j))$, which can result from empirical analysis on labeled sample (training) data. Since the resultant function should be nondecreasing some further curve-fitting modification steps need to be applied. Nevertheless, this approach is only possible if labeled sample data is available. Moreover, the resultant *sim2p-mapping* is extremely domain-dependent. An example of a mapping function resulting from an empirical analysis on a labeled data set is depicted in Figure 15(ii).

(2) *Specifications Based on Threshold Distances*. If the indeterministical handling is only applied to possible matches (see P-restriction in Section 5.3), only tuple pairs with a similarity $sim(t_i, t_j) \in (T_\lambda, T_\mu]$ need to be considered. In this case, matching probability can be automatically derived from the distances of the tuple similarity to the two thresholds T_λ and T_μ:

$$p(t_i, t_j) = 1 - \frac{T_\mu - sim(t_i, t_j)}{T_\mu - T_\lambda}. \tag{4}$$

(3) *Manual Specifications*. In cases, clerical reviews are used to evaluate possible matches, but domain experts do not know with certainty whether tuples are duplicates or not, the matching probabilities can be manually specified by these experts during their review (see manual-restriction in Section 5.3).

All the methods we have discussed here derive matching probabilities from single-value tuple similarities. However, uncertainty in deduplication does not only arise in the classification step by not knowing which threshold classifies best, but is also present in choosing methods for matching attribute values and/or computing tuple similarities. Consequently, instead of deriving the matching probability of a tuple pair

from a single similarity value, we could derive it from a set of similarity values, where each value is weighted by its believability. These similarity values as well as their believabilities can be computed by the weighted use of several attribute-value matching methods and/or several tuple-similarity computation methods. The matching probability of a specific tuple pair can then be finally derived by applying a single threshold classification or a two-threshold classification to each of the pair's similarity values.

5.3. Semi-Indeterministic Approaches

To make the indeterministic approach feasible in practice, we propose five semi-indeterministic approaches in which only the most probable worlds are taken into account. In the first four approaches, the initial *M-graph* is modified. Thus, the number of resultant worlds is downsized by reducing the set of uncertain edges and hence by reducing the set of indeterministically handled decisions. In contrast, in the fifth approach, the number of *W-graphs* is reduced by modifying the *W-graph*-generation mapping ν. An important point is that these approaches are not considered to be competitors, but are designed for different scenarios and partially can be combined with each other.

In the end, the probabilities of all worlds must sum up to 1. Thus, the actual probabilities of the resultant worlds are conditioned and hence may be distorted. However, the result is still more accurate than the one world resulting from a deterministic approach.

The five semi-indeterministic approaches are:

(1) (α, β)-*Restrictions*. In order to filter out the most improbable worlds, only the most uncertain duplicate decisions have to be considered in an indeterministic way. The uncertainty whether two tuples are duplicates is maximal, if their matching probability is 0.5. For that reason, we define the two thresholds $\alpha < 0.5$ and $\beta \geq 0.5$. Probabilities lower than α are then initially mapped to 0 and probabilities greater or equal to β are initially mapped to 1. Thus, only decisions with probabilities between α and β are handled indeterministically. In contrast, decisions with probabilities outside this range are quite evident and can be deterministically handled without running a high risk of failure. On the whole, depending on α and β, the number of uncertain decisions (and hence the number of *uncertain edges* in corresponding *M-graphs*) can be effectively reduced by this way (see our experimental results in Section 7). To make sure that the most probable worlds result, we always use $\beta = 1 - \alpha$.

(2) *P-Restrictions*. In this approach, we limit the indeterministic deduplication on tuple pairs classified into the set of possible matches (P). Matching probability can be suitably computed by regarding T_λ and T_μ (see Section 5.2). Naturally, the effectiveness and correctness of a P-restriction is lower than evaluating the tuple pairs in P by clerical reviews. However, a P-restriction is a full-automatic approach and hence no effort of domain experts is required. The main difference to (α, β)-restrictions is that at first the set of indeterministically handled decisions is restricted and then probabilities are computed. On the contrary (α, β)-restrictions are based on probability values and hence require a *sim2p-mapping* resulting from empirical analyses. Note, by considering tuple similarity as matching probability ($p(t_i, t_j) = sim(t_i, t_j)$), a P-restriction produces the same worlds (but not the same probabilities) as an (α, β)-restriction with $\alpha = T_\lambda$ and $\beta = T_\mu$.

(3) *Manual-Restrictions*. During clerical reviews, it could happen that responsible experts do not know with certainty whether two tuples are duplicates or not. In such cases, experts can consider both choices by handling the decision

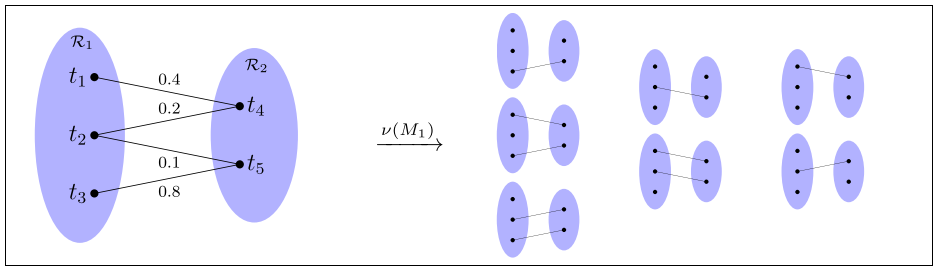


Fig. 12. Bipartite *M-graph* M_1 resulting from a context-based restriction on the two duplicate-free relations $\mathcal{R}_1$ and $\mathcal{R}_2$ (left) and the resultant set of its bipartite consistent *W-graphs* (right).

indeterministically. By doing so, the indeterministic approach is only applied to individual tuple pairs and the number of resultant worlds remains low.

(4) *Context-Based-Restrictions*. During an integration process additional context information on the given sources can be available. Sometimes, this information can be used to restrict the set of *uncertain decision*. For instance, one or more source relations can be known (no heuristic, because certain information is used) or assumed (heuristic) to be duplicate-free. Thus, with respect to these relations instead of intrasource duplicates only intersource duplicates need to be detected. In this case, tuples originating from same sources do not need to be compared and corresponding matching probabilities can be automatically set to 0. This enormously decreases the number of worlds that have to be considered in the subsequent steps. As an example, we consider the two relations $\mathcal{R}_1 = \{t_1, t_2, t_3\}$ and $\mathcal{R}_2 = \{t_4, t_5\}$. Both relations are known to be duplicate-free. For that reason, in the corresponding *M-graph* M_1 (see Figure 12) all edges connecting two nodes representing tuples of the same source are weighted with 0 (for simplification, these edges are removed in Figure 12). Thus, by using the available context information the initial *M-graph* can be restricted to a bipartite graph. As a consequence, each *W-graph* is a bipartite graph, too, which is only consistent, if each node is only connected with at most one other node. This in turn reduces the number of considered *W-graphs* enormously.

(5) *HC-Restrictions*. Restrictions on hierarchical tuple clustering are already known from Beskales et al. [2009]. In our approach, such restrictions can be achieved by modifying the original *W-graph*-generation mapping ν that we have introduced in Eq. (3). One of several possible HC-restrictions is to consider an uncertain edge only then, if all other edges having a weight greater or equal than the edge's weight have been considered as well, instead of generating one *W-graph* for each possible combination of uncertain edges. Let $M = (N, E, \gamma)$ be an *M-graph*. The previously mentioned HC-restriction can be realized by introducing the parameter $\tau \in \{\gamma(e)|e \in E\}$. For each τ, a *W-graph* is generated by only regarding edges having a weight greater than or equal to τ. A corresponding *W-graph*-generation mapping ν_{HC} is defined as:

$$\nu_{\text{HC}}(M) = \bigcup_{\tau \in \{\gamma(e)|e \in E\}} \{(N, K = \{e \in E | \gamma(e) \geq \tau\}, \prod_{e \in K} \gamma(e) \prod_{e \in E \setminus K} (1 - \gamma(e)))\}.$$

Using this HC-restriction strategy, from the *M-graph* M shown in Figure 9 only the *W-graphs* $\{G_1, G_2, G_5, G_8\}$ are derived. As a consequence, the hierarchical clustering with the three consistent *W-graphs* $\{G_1, G_2, G_8\}$, as illustrated in Figure 13 results. Instead of a final conditioning, the range of τ leading to a specific *W-graph* could be taken as the graph's resultant probability (e.g. $P_1 = 0.2$, $P_2 = 0.5$, and $P_8 = 0.3$).

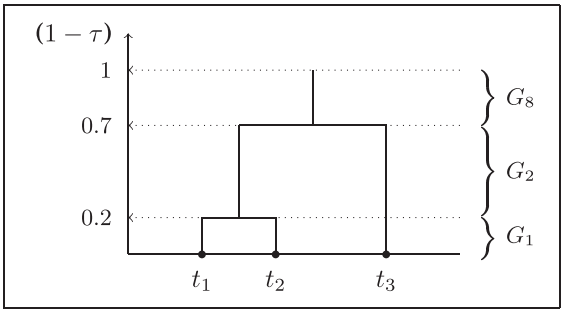


Fig. 13. Hierachical tuple clustering.

Table I. Use Cases of the Different Semi-Indeterministic Approaches

Approach	**Source of Probability**	**Use Case**
(α, β)-restriction	empirical analyses	full-automatic with training data
P-restriction	threshold distance	full-automatic without training data
Manual-restriction	manual specification	semi-automatic with clerical reviews
Context-based-restriction	ALL	ALL with additional context information
HC-restriction	ALL	ALL

The main benefit of restricting the indeterministic deduplication result to hierarchical clusterings is its low computation complexity. By using the mapping ν for *W-graph*-generation, from an *M-graph* with k uncertain edges 2^k *W-graph* result. In contrast, the mapping ν_{HC} used for creating hierarchical clusterings generates at most $k+1$ *W-graphs*. Thus, an HC-restriction performs well even for *M-graphs* (or *partial M-graphs*, see Section 5.3.2) with many uncertain edges (see experimental results in Appendix F).
Since an HC-restriction concerns the *W-graph*-generation mapping instead of changing matching probabilities, it can be combined with other restriction techniques, as for example, an (α, β)-restriction.

Note, by using P-restrictions or manual-restrictions the classification step is additionally included in the extended tuple matching phase.

As mentioned previously, these strategies do not compete with each other, but are possible alternatives, each having a different best use case as listed in Table I. (α, β)-restrictions are based on given matching probabilities and hence are particularly suitable for cases where training data were given for empirical analyses. In contrast, due to matching probabilities can be completely derived by threshold distances, deduplication processes using P-restrictions can be performed automatically without labeled training data. Manual restrictions are designed for unburden experts to come to ultimate decisions in clerical reviews and hence to avoid doubtful decisions in semi-automatic deduplication processes. HC-restrictions can be combined with any other strategy to improve efficiency. Thus, they can be used in every scenario, independent from the source of matching probabilities. Context-based-restrictions enable the inclusion of additional context information and hence can be used to improve effectiveness and efficiency. They can be combined with any other strategy.

5.3.1. Consistency. By using a semi-indeterministic approach, deterministically taken decisions can be contradictory. In general, in an (α, β)-restriction, the closer α and β,

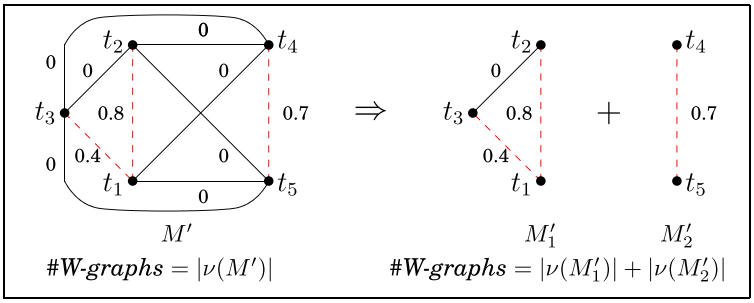


Fig. 14. Decomposition of an *M-graph* M' in its independent *partial M-graphs* M'_1 and M'_2.

the higher is the probability that the initial *M-graph* is inconsistent and hence all resultant worlds are per se impossible. In such cases, repair operations are required for ensuring the consistency of the resultant *M-graph* with minimal effort and minimal decision modifications (see future goals in Section 8).

Let $M_1 = (N_1, E_1, \gamma_1)$ be a consistent initial *M-graph* that has been restricted to the inconsistent *M-graph* $M_2 = (N_2, E_2, \gamma_2)$. A simple method to repair M_2 is to sort all definite edges $e \in (E_2^+ \cup E_2^-)$ by the certainty of their original weight $c(e) = |\gamma_1(e) - 0.5|$ in ascending order and then modify edge by edge from the top of the sorted list from definite to uncertain until the modified *M-graph* is consistent.

5.3.2. Decomposition of Matching-Graphs. The more the set of indeterministically handled decisions is restricted, the larger is the proportion of edges weighted with 0. As a consequence, the usage of a semi-indeterministic approach enables a splitting of the initial *M-graph* into multiple independent subgraphs (called *partial M-graphs*[5]). In this case, for each of the *partial M-graphs* the *W-graph*-generation mappings ν (or ν_{HC} respectively) can be applied independently. Thus, the number of resultant *W-graphs* can be dramatically downsized and hence the resultant possible worlds are represented in a more succinct way. Since the decisions of the individual subgraphs are independent to each other, instead of complementations of whole worlds, only complementations of small parts of these worlds result. This in turn extremely reduces the number of required indicator alternatives. An example for decomposing an initial *M-graph* is shown in Figure 14. Note that $|\nu(M')| = |\nu(M'_1)| \times |\nu(M'_2)| = 3 \cdot 2 = 6$ and $|\nu(M'_1)| \times |\nu(M'_2)| > |\nu(M'_1)| + |\nu(M'_2)|$.

5.3.3. Complexity. All proposed techniques for semi-indeterministic restrictions may, in the worst case, eliminate not even one uncertain edge and hence both full- and semi-indeterministic approaches have theoretically the same complexity. However, as we will see in the experimental results presented in Section 7, even marginal restrictions come nowhere near the worst case, because they eliminate the *majority* of the uncertain edges rather than none at all. Since the number of uncertain edges k is the dominant factor in the complexity formulas (see Appendix D), it means that, in practice, a semi-indeterministic approach moves us into an entirely different area of complexity curve, an area we show is well manageable in practice.

[5]Each *partial M-graph* can be considered as a graphical representation of a factor, as defined in Sen and Deshpande [2007]

6. QUERYING INDETERMINISTIC DEDUPLICATION RESULTS

We identified four classes of applications for which querying indeterministic deduplication results is of use. Each of them is based on a different type of queries:

(1) *Applications Needing Concrete Absolute Query Results*. Concreteness can be achieved by querying the most probable world. This is similar to querying a deterministic deduplication result. $Q1$ is a sample TriQL-query computing the most probable world by using horizontal subqueries on indicator tuples:

```
Q1: SELECT *
    FROM R_X t, Ind i
    WHERE Lineage(t,i)
      AND (i.id,i.val) IN (SELECT *
                           FROM Ind i
                           WHERE Conf(i) = [max(Conf(*))]);
```

(2) *Applications Interested in Query Results that Are Dead Certain* (see the concept of consistent query answering [Arenas et al. 1999]). By querying deterministic deduplication results, such a certainty cannot be ensured, because maybe some of the duplicate decisions were actually not made with absolute certainty, but the system does not know which of them. $Q2$ is a sample TriQL-query computing all persons who definitely have a nonunique name. For this query, a consideration of uncertainty is especially important in the evaluation of the subquery predicate EXISTS, if the result should be absolutely certain:

```
Q2: SELECT *
    FROM (SELECT *
          FROM Person t1
          WHERE EXISTS (SELECT *
                        FROM Person t2
                        WHERE t1.tid != t2.tid
                          AND t1.name = t2.name))
    WHERE Conf(*) = 1.0;
```

(3) *Applications Where It Is Useful to Present/Visualize the Uncertainty Around Certain Data Items*, may pose queries to return uncertain results (e.g., by using the TriQL-predicate *Conf*()). Moreover, in some applications, it could be of interest to distinguish certain duplicate decisions from ambiguous duplicate decisions. $Q3$ is a sample TriQL-query returning all tuples that are involved in duplicate decisions with less certainty (the probability of each choice is lower than 0.6), ordered by the probabilities of the corresponding choices:

```
Q3: SELECT *
    FROM R_X t, Ind i
    WHERE Lineage(t,i)
      AND i.id IN (SELECT id
                   FROM Ind
                   WHERE [max(Conf(*))] < 0.6)
    ORDER BY Confidences ASC;
```

(4) *Analytical Applications (such as data mining)* are statistical in nature themselves, so they can process the uncertain data directly. For that purpose, computing aggregation values (e.g., the expected number) can be required. $Q4$ is a sample

TriQL-query computing the expected value, the minimal value (worst case), and the maximal value (best case) of the sales for a company's products:

```
Q4: SELECT   Esum(value) AS ExpSale, Lsum(value) AS MinSale, Hsum(value) AS MaxSale
    FROM     Sale
    GROUP BY product;
```

Note, traditional (vertical) subqueries and subquery-predicates are not supported by the currently available Trio version, but all these queries can be also formulated by using joins and auxiliary tables. However, we use these constructs in our examples, because they are part of the TriQL manual and we want to define queries as short as possible. Equivalent and working queries are listed in Appendix E.

7. EXPERIMENTAL EVALUATIONS

To demonstrate the efficiency and effectiveness of semi-indeterministic approaches, we run two experiments each with different (α, β)-restrictions on an online cd dataset[6] with 9,763 items. To obtain matching probabilities, we split the data into two parts. The first part (5000 items) was used as labeled sample data to determine an adequate *sim2p-mapping* (see Section 5.2). The second part (4763 items) was used as actual source data. To match attribute values, we used the normalized Levenshtein distance. To compute tuple similarity, we used an ordinary distance function based on the similarities of the values of the three attributes a_1=*dtitle*, a_2=*artist*, and a_3=*category*:

$$sim(t_i, t_j) = 0.5 \cdot \vec{c}_{ij}(a_1) + 0.4 \cdot \vec{c}_{ij}(a_2) + 0.1 \cdot \vec{c}_{ij}(a_3).$$

7.1. Experiments on Efficiency

To evaluate the efficiency also for very small restrictions (e.g., $\alpha = 0.01$), we only use 2000 items of the second data part for these experiments. The experimental results are shown in Table II and are graphically presented in Figure 15. We use *M-graph* decomposition to improve efficiency, but perform all experiments by using the nonhierarchical *W-graph*-generation mapping ν. Experimental results of using an HC-restriction for improving efficiency further on are reported in Appendix F.

As depicted in Figure 15(i), the similarity of most tuple pairs was very low (98.7% were lower than 0.35 and only 0.002% were higher than 0.7). Moreover, as shown in Figure 15(ii), only high similarity implied an appreciable value of matching probability (almost all duplicates of the labeled sample data had a similarity higher than 0.7). The number of considered worlds and the number of *W-graphs* could be drastically downsized by only taking the most uncertain decisions into account. For example, only a small restriction of the area of indeterministically handled decisions from $(0, 1)$ to $(.1, .9)$ was required to decrease the number of uncertain edges by almost a factor of one hundred thousand (see Figure 15(iii)).

As expected, the complexity decreased with a shrinking area of indeterministically handled decisions. A $(0, 1)$-restriction is a full-indeterministic approach having an unmanageable complexity, even if not as complex as the worst case predicted in Appendix D (only each fourth edge was uncertain). In contrast, a $(.5, .5)$-restriction is equal to a full-deterministic approach. Therefore, naturally no uncertain edge and hence only one *W-graph* as well as only one possible world resulted. Since 16 duplicates were detected, the resultant x-relation contains 1984 tuples.

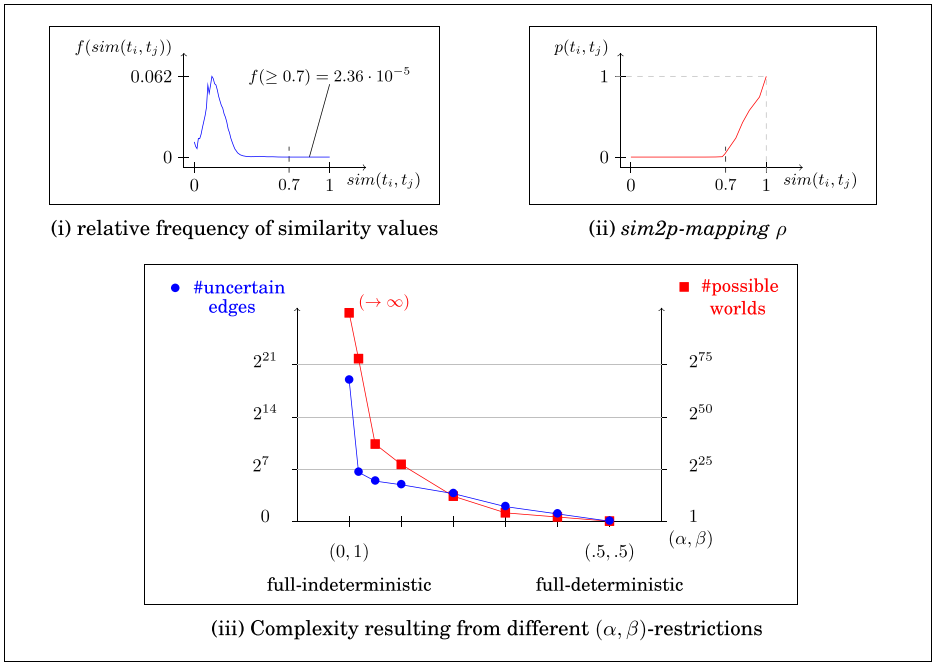
In general, most edges had low weights. Thus, the number of uncertain edges decreased dramatically, if the area of indeterministically handled decisions was marginally reduced. In contrast, a restriction of this area from $\alpha = 0.05$ to $\alpha = 0.4$

[6]http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cd_datasets.html

Table II. Complexity and Uncertainty of Several $(\alpha, 1-\alpha)$-Restrictions with *M-graph* Decompostion

α	#unc. edges	#possible worlds	#world-graphs	#result tuples	#ind. altern.	Dens.	Dec.	runtime [sec.]
0	577004	$\rightarrow \infty^*$	$\rightarrow \infty^*$	$\rightarrow \infty^*$	$\rightarrow \infty^*$	$\rightarrow 1^*$	$\rightarrow 0^*$	$\rightarrow \infty^*$
.0165	98	$3.6 \cdot 10^{23}$	133296	5739	1194	0.0151	0.9915	18.519
.0175	95	$1.0 \cdot 10^{23}$	4288	2349	227	0.0153	0.9911	10.549
.05	42	$1.7 \cdot 10^{11}$	2011	2036	51	0.0087	0.9939	2.519
.1	30	$1.7 \cdot 10^{8}$	1996	2019	33	0.0067	0.9951	1.845
.4	2	4	1985	1987	2	0.0005	0.9994	0.969
.5	0	1	1984	1984	0	0	1	0.966

*Due to our limited resources, processing a full-indeterministic approach was not feasible.



(i) relative frequency of similarity values

(ii) *sim2p-mapping* ρ

(iii) Complexity resulting from different (α, β)-restrictions

Fig. 15. Experimental results of several (α, β)-restrictions with *M-graph* decomposition.

only insignificantly reduced this number further on. The number of resultant possible worlds imploded exponentially with a shrinking indeterministic area. In contrast, the number of resultant tuples and the number of required indicator alternatives decreased proportionally with a decreasing number of uncertain edges (see Table II).

Runtime (we measured runtime starting from tuple matching results) increased noticeably for very small restrictions ($\alpha < 0.02$), but was still of an acceptable size for all experiments. Note that all experiments were performed by a prototypical implementation that was not tuned so far. Thus, absolute runtime values do not bring any scientific value and can be only used to give a feeling of complexity.

As shown by these results, the number of edges weighted with 0 increased enormously, if a semi-indeterministic approach is used. As mentioned in Section 5.3.2, the more edges were weighted with 0, the more *partial M-graphs* the initial *M-graph* could be decomposed into. For example, already a small restriction to $\alpha = 0.05$ was

sufficient to decompose the initial *M-graph* into a high number of subgraphs (1952 *partial M-graphs*). The most of these *partial M-graphs* (1913) were single nodes. For that reason, only 2011 partial *W-graphs* resulted. This in turn reduced the required number of indicator tuple alternatives from $1.7 \cdot 10^{11}$ (the number of possible worlds) to 51. In contrast, in a full-indeterministic approach instead of 1913 only 12 tuples could definitely be excluded to be duplicates. In general, in a full-indeterministic approach, the initial *M-graph* can be only limitedly decomposed.

To score the uncertainty of the probabilistic result, we adopted the two measures *Uncertainty Density* (Dens.) and *Answer Decisiveness* (Dec.) from de Keijzer and van Keulen [2007] by considering each *partial M-graph* as a choice point and its consistent *partial W-graphs* as its mutually exclusive alternatives. The *Uncertainty Density* (*Answer Decisiveness*) is evaluated to 0 (1) for a databases that contains no uncertainty. As you can see, for $\alpha \geq 0.05$, the uncertainty was very low, because the majority of base-tuples could be classified as nonduplicates with high certainty. Even for $\alpha = 0.0165$, the resultant uncertainty was still of a manageable size.

In conclusion, these results demonstrate that the complexity and uncertainty of an indeterministic approach is already manageable, if the area of indeterministically handled decisions is only marginally restricted.

More detailed results on our experiments on efficiency are listed in Appendix F.

7.2. Experiments on Effectiveness

To evaluate the effectiveness of several $(\alpha, 1-\alpha)$-restrictions (FAID_1-FAID_3), we took all 4763 items of the second data part as input. Existing adaptations to recall and precision, such as van Keulen and de Keijzer [2009], insufficiently capture what is intuitively better for these applications. In general, the meaning of quality depends on the intended use. Thus, we compared the number of resultant false decisions,[7] which, for example, is an appropriate measure for the quality of consistent query results, with those resulting from four processes of two ordinary deterministic deduplication approaches: (1) A process of a full-automatic approach for deduplication (FADD_1) as a benchmark having a single threshold $T_\lambda = T_\mu$. To obtain an adequate benchmark, we took the threshold $T_\lambda = 0.78$ leading to the best F_1-*score* in the labeled sample data. (2) Three processes of a semi-automatic approach (SADD_1-SADD_3) each producing a temporary set of possible matches requiring clerical reviews. We considered several threshold settings (each with center 0.78) each resulting in a set P having a realistic number of clerical reviews. To be independent of the experts' competence, we assumed each manual decision to be correct.

As presented in Table III, in comparison to FADD_1, the number of false decisions decreased with a growing size of P. The share of false decisions taken by FADD_1, which was correctly assigned to M or U (denoted as improvement) by SADD_1-SADD_3, was up to 50%. However, in reverse, the number of clerical reviews increased as well. For example, given the setting $P = (0.63, 0.93]$, 2735 clerical reviews were required. Take into consideration that these were 0.022% of all matches, which was not much in our experiment, but which can be tremendous for large data sets (c.a. $1.1M$ reviews for 100,000 source items). In contrast, by using one of the indeterministic processes FAID_1-FAID_3, the number of false decisions was reduced to a large extent without any clerical review. Even an $(.4, .6)$-reduction had an improvement of around 23%. A restriction with $\alpha = 0.05$ which was still performing efficiently (see Section 7.1) improved the result of 66.7%, which was higher than the best result of the semi-automatic deterministic approaches.

[7]We consider a decision to be false, if it was false with absolute certainty, that is, if it was false in every possible world.

Table III. Effectiveness of Several Deduplication Approaches Scored by Number of False Unmatches (FU), Number of False Matches (FM), Number of Clerical Reviews, Number of Indeterministically Handled Decisions (IhD), and the Resultant Amount of Certainty (Measured in *Answer Decisiveness*)

Full-Automatic Deterministic Deduplication (FADD), $T_\lambda = T_\mu$

	Settings	FU	FM	Improv.	Reviews	IhD	Certainty
FADD_1	$T_\lambda = T_\mu = 0.78$	25	35	–	0	0	1.0

Semi-Automatic Deterministic Deduplication (SADD), $P = (T_\lambda, T_\mu]$

	Settings	FU	FM	Improv.	Reviews	IhD	Certainty
SADD_1	$T_\lambda = 0.73, T_\mu = 0.83$	17	26	28.3%	70	0	1.0
SADD_2	$T_\lambda = 0.68, T_\mu = 0.88$	14	20	43.3%	389	0	1.0
SADD_3	$T_\lambda = 0.63, T_\mu = 0.93$	12	17	51.6%	2735	0	1.0

Full-Automatic Indeterministic Deduplication (FAID), $(\alpha, 1-\alpha)$**-Restriction**

	Settings	FU	FM	Improv.	Reviews	IhD	Certainty
FAID_1	$\alpha = 0.4$	26	20	23.3%	0	13	0.9973
FAID_2	$\alpha = 0.2$	22	13	41.7%	0	44	0.9966
FAID_3	$\alpha = 0.05$	15	5	66.7%	0	137	0.9957

Of course, by using an indeterministic approach, the resultant data was more accurate, but also more uncertain than by using a deterministic one. To score certainty, we took the *Answer Decisiveness* as used in our experiments before. As you can see, the certainties of our indeterministic results were only marginally reduced. In general, the extent of indeterministically handled decisions is always a trade-off between accuracy and certainty and thus, the best configuration setting differs from case to case.

These results demonstrate the effectiveness of our indeterministic approach in general, but also show the high potential of using an adequate *sim2p-mapping*. From the used sample data, we got a *sim2p-mapping* of good quality (we have to confess that this was not always the case). Thus, even if all indeterministically handled decisions would be pass to a domain expert, the number of clerical reviews for FAID_3 (137 reviews) would be lower than by using SADD_2 (389 reviews) despite the fact that the number of false decisions is reduced as well (improvement 66.7% to 43.3%).

8. CONCLUSION

In this article, we propose an efficient indeterministic approach for deduplication including how to represent possible mergings using x-relations [Benjelloun et al. 2006]. The latter requires careful construction of data lineage to faithfully represent the inherent tuple dependencies. For reasons of generality and illustration, we first present a theoretical full-indeterministic approach that we then refine into a set of more practical semi-indeterministic approaches. For the same reasons, we model the fundamental part using a graph-based approach and possible world semantics.

The main contributions of this article are: An indeterministic approach for deduplication that is based on the possible world semantics. The approach minimizes the negative impact (i.e., loss of data quality) resulting from false decisions in ambiguous circumstances and avoids human effort during the duplicate decisions – clerical reviews and tuning of thresholds are not that crucial anymore. Moreover, it enables the usage of traditional probabilistic databases as Trio, which increases the usability of the resultant data (e.g., for further integration processes) and enables a more powerful querying than in other indeterministic approaches. Finally, we present techniques for proper probabilistic interpretation of similarity values.

We have shown by experiments that even very small and cautious restrictions of the set of indeterministically handled decisions already reduce the volume of resultant uncertain data to a manageable size, making indeterministic deduplication feasible in practice. Moreover, we demonstrate that the number of false decisions resulting from deterministic approaches with or without clerical reviews can be reduced to a large extent, if an indeterministic approach is used.

Although feasible, better scalability of deduplication remains a direction for future research. Another direction is to identify effective and efficient repair strategies for dealing with an inconsistent initial *M-graph* (see Section 5.3.1). Furthermore, if deduplication is to be a step in a larger data integration process, it needs to be extended to probabilistic source data. Moreover it should respect fundamental properties such as idempotence if data is duplicate-free. Another important aspect is to analyze the correlation between the amount of uncertainty modeled in the database and the response time of different kind of queries. Finally, an essential point of future work is the design of new well-defined quality metrics for probabilistic data. These are required for (1) capturing the benefits and drawbacks of probabilistic data with respect to certain data, (2) working out the most effective parameter settings (e.g. used similarity measures, *matching functions* or *sim2p-mappings*), and (3) better comparing the effectiveness of full-indeterministic, semi-indeterministic and deterministic approaches for deduplication.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

Arenas, M., Bertossi, L. E., and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of PODS*. 68–79.

Barbará, D., Garcia-Molina, H., and Porter, D. 1992. The management of probabilistic data. *IEEE Trans. Knowl. Data Eng. 4*, 5, 487–502.

Batini, C. and Scannapieco, M. 2006. *Data Quality: Concepts, Methodologies and Techniques*. Springer.

Benjelloun, O., Das Sarma, A., Halevy, A., and Widom, J. 2006. ULDBs: Databases with uncertainty and lineage. In *Proceedings of VLDB*. 953–964.

Beskales, G., Soliman, M. A., Ilyas, I. F., and Ben-David, S. 2009. Modeling and querying possible repairs in duplicate detection. *PVLDB 2*, 1, 598–609.

Bleiholder, J. and Naumann, F. 2008. Data fusion. *ACM Comput. Surv. 41*, 1.

Dalvi, N. N. and Suciu, D. 2007. Efficient query evaluation on probabilistic databases. *VLDB J. 16*, 4, 523–544.

Das Sarma, A., Theobald, M., and Widom, J. 2008. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In *Proceedings of ICDE*. 1023–1032.

de Keijzer, A. and van Keulen, M. 2007. Quality measures in uncertain data management. In *Proceedings of SUM*. 104–115.

de Keijzer, A. and van Keulen, M. 2008. IMPrECISE: Good-is-good-enough data integration. In *Proceedings of ICDE*. 1548–1551.

Dechter, R. 1996. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of UAI*. 211–219.

Dong, X. L., Halevy, A. Y., and Yu, C. 2009. Data integration with uncertainty. *VLDB J. 18*, 2, 469–500.

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

Fellegi, I. and Sunter, A. 1969. A theory for record linkage. *J. Amer. Statist. Assoc. 64*, 1183–1210.

Goble, C. A. and Stevens, R. 2008. State of the nation in data integration for bioinformatics. *J. Biomed. Informat. 41*, 5, 687–693.

Hassanzadeh, O., Chiang, F., Miller, R. J., and Lee, H. C. 2009. Framework for evaluating clustering algorithms in duplicate detection. *PVLDB 2*, 1, 1282–1293.

Hernández, M. A. and Stolfo, S. J. 1995. The merge/purge problem for large databases. In *Proceedings of the SIGMOD Conference*. 127–138.

Ioannou, E., Nejdl, W., Niederée, C., and Velegrakis, Y. 2010. On-the-fly entity-aware query processing in the presence of linkage. *PVLDB 3*, 1, 429–438.

Karp, R. M. and Luby, M. 1983. Monte-Carlo algorithms for enumeration and reliability problems. In *Proceedings of FOCS*. 56–64.

Koch, C. 2008. Approximating predicates and expressive queries on probabilistic databases. In *Proceedings of PODS*. 99–108.

Koch, C. 2009. MayBMS: A system for managing large uncertain and probabilistic databases. In *Proceedings of Managing and Mining Uncertain Data*. Springer.

Koudas, N., Marathe, A., and Srivastava, D. 2004. Flexible string matching against large databases in practice. In *Proceedings of VLDB*. 1078–1086.

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proceedings of PODS*. 233–246.

Naumann, F. and Herschel, M. 2010. *An Introduction to Duplicate Detection*. Morgan & Claypool Publishers.

Newcombe, H. B., Kennedy, J. M., Axford, S. J., and James, A. P. 1959. Automatic linkage of vital records. *Science 130*, 954–959.

Ravikumar, P. D. and Cohen, W. W. 2004. A hierarchical graphical model for record linkage. In *Proceedings of UAI*. 454–461.

Rota, G. 1964. The number of partitions of a set. *Amer. Math. Monthly 71*, 5, 498–504.

Sen, P. and Deshpande, A. 2007. Representing and querying correlated tuples in probabilistic databases. In *Proceedings of ICDE*. 596–605.

Suciu, D., Olteanu, D., Ré, C., and Koch, C. 2011. *Probabilistic Databases*. Morgan & Claypool Publishers.

Taddei, A., Dalmiani, S., Vellani, A., Rocca, E., Piccini, G., Carducci, T., Gori, A., Borghini, R., Marcheschi, P., Mazzarisi, A., Salvatori, C., and Macerata, A. 2008. Data integration in cardiac surgery health care institution: Experience at G. Pasquinucci Heart Hospital. In *Computers in Cardiology*. 287–290.

Talburt, J. R. 2011. *Entity Resolution and Information Quality*. Morgan-Kaufmann.

Tseng, F. S.-C., Chen, A. L. P., and Yang, W.-P. 1993. Answering heterogeneous database queries with degrees of uncertainty. *Distrib. Parall. Datab. 1*, 3, 281–302.

van Keulen, M. and de Keijzer, A. 2009. Qualitative effects of knowledge rules and user feedback in probabilistic data integration. *VLDB J. 18*, 5, 1191–1217.

Wang, Y. R. and Madnick, S. E. 1989. The inter-database instance identification problem in integrating autonomous systems. In *Proceedings of ICDE, 1989*. IEEE Computer Society, 46–55.

Widom, J. 2009. Trio: A system for data, uncertainty, and lineage. In *Managing and Mining Uncertain Data*. Springer.

Received December 2010; revised August 2011, December 2011; accepted February 2012