

# Generating Efficient Safe Query Plan for Probabilistic Databases

Biao Qin<sup>1,2</sup>\*, Yuni Xia<sup>2</sup>

1. Dept. of Computer Science, Renmin University of China, P. R. China, 100872

2. Dept. of Computer and Information Science, Indiana University - Purdue University Indianapolis 46202

---

## Abstract

Managing uncertain information using probabilistic databases has drawn much attention recently in many fields such as information retrieval, multimedia database and sensor data management. Differing from conventional databases which maintain certain data, probabilistic databases manage data with probability. A query plan applicable to a conventional database may generate incorrect composite probability in the query result when applied to a probabilistic database. Finding *safe* query plans is the key to query evaluation on probabilistic databases. In this paper, we propose a new approach to generate safe plans for probabilistic databases. Adopting effective preprocessing and multiway split techniques, our algorithm can reduce the number of projection and avoid unnecessary cartesian-products in the query plans. Further, we extend existing transformation rules to allow the safe plans generated by the Safe-Plan algorithm [11] and the proposed Multi-way Split algorithm to be transformed between each other. We apply our approach on the TPC-H benchmark queries and show that safe query plans can be accurately generated. We also demonstrate through experiments that the query plans generated by our algorithm are significantly more efficient comparing to the mainstream approach.

*Key words:* Probabilistic databases; Safe plan; Query optimization

---

\* corresponding author.

*Email addresses:* bqin@cs.iupui.edu (Biao Qin<sup>1,2</sup>), yxia@cs.iupui.edu (Yuni Xia<sup>2</sup>).

## 1 Introduction

There exists a pressing need for integrating database (DB) and Information Retrieval (IR) techniques. While DB systems focus on accessing exactly matched data in fast response, IR systems often focus on retrieving inexactly matched data in a relatively lengthy process; for example, searching the images similar to a given sample. There are several aspects in the integration of DB and IR, for example, extending SQL to express search activities, or extending search by supporting direct data access. While all those efforts deserve to be encouraged, we believe the more practical way is to use database techniques to manage the growing amount of imprecise IR results, such as the annotations, properties or concepts of multimedia data with certain degrees of believes. This is exactly the capability of probabilistic database.

In a regular database, tuples in a relation represent assertions, and a query looks for exactly matched assertions. However, in a probabilistic database, tuples in a relation or in a query result bear two kinds of information: assertions and their probabilities, meaning that each assertion has only a certain degree of belief. Since a query has probabilistic semantics, its evaluation is constrained for preserving such semantics.

A query is evaluated based on its execution plan. A query plan is the combination of relational operators represented by a query tree. A query can have multiple execution plans, and there exist a set of primitive transformation rules for converting query plans from one another with preserved semantics. The purpose of query optimization is to choose an efficient execution plan. A query result against a probabilistic database is correct only if both assertions and their probability are correct. A query execution plan that can gain correct results is called a **safe** execution plan.

As probability could be conditional, we cannot assume that the tuples, or the relations, represent independent assertions thus can be manipulated independently. Therefore, a query plan that is correct without considering probability, could be incorrect, when composite probability is taken into account. In fact, the query plans having equal semantics without considering probability, may

not be all safe on a probabilistic database. Therefore, in regular database query evaluation, the goal is to find efficient execution plan; in probabilistic database query evaluation, the goal is to find safe as well as efficient execution plan. As probabilistic databases have become critical to many emerging technologies, the practical value of this research topic has been gradually realized.

To find safe as well as efficient execution is our objective. Our solution is based on query rewriting for generating safe plans efficiently, and making safety-preserved query optimization in a probabilistic database environment. The main contributions are as follows.

- We classify projections into simple ones and complex ones where a simple projection involves only one relation, and a complex projection involves two or more relations; then we prove that any execution plan for a probabilistic conjunctive query without complex projection is safe, that is, yield correct data as well as probability without requiring special treatment.
- We project out non-association attributes by preprocessing the queries involving complex projection. If several tables satisfy the split condition at the same time, we adopt the multiway split algorithm which can decrease the number of projections and avoid unnecessary cartesian-product in the safe plan.
- We extend existing transformation rules. With these new rules, the safe plans generated by the Safe-Plan [11] algorithm and our proposed Multiway Split algorithm can be transformed between each other.

The rest of this paper is organized as follows. Section 2 outlines the probabilistic database model and query evaluation. Section 3 discusses the extensional query evaluation semantics with respect to simple and complex projections. Section 4 describes the novel safe plan generation algorithm. Section 5 deals with safe plan optimization. The experimental results are shown in Section 6. Section 7 discusses related work. Section 8 concludes the paper.

## 2 Probabilistic database theory

An event may be probabilistic. The probability of an event can be independent, or conditional based on the probabilistic dependencies with other events. Managing such events is the goal of probabilistic databases [11,17,27].

### 2.1 Probabilistic database model

We call an independent probabilistic event an *atomic event*, and the composition of atomic events constructed with logic operators  $\wedge$ ,  $\vee$ , and  $\neg$  a complex event. We denote a probability function on event  $e$  as  $Pr(e) \rightarrow [0, 1]$ .

**Probabilistic database** A probabilistic database manages probabilistic events, where each tuple has a certain probability of belonging to the database.

**Probabilistic relation** A probabilistic relation is a relation with a distinguished event attribute  $E$  for complex events. We write  $R$  for a relation name,  $Attr(R)$  for the set of its attributes, and  $Key(R)$  for the key of  $R$ . Users "see" only  $R$ , but the system needs to access the event attribute  $R^p.E$ . The set of functional dependencies ( $FD$ )  $\Gamma^p$  always contains  $Attr(R) \rightarrow R^p.E$ . This ensures that we don't associate two different events  $e_1$  and  $e_2$  to the same tuple  $t$  (instead, we may want to associate  $e_1 \vee e_2$  to  $t$ ). In this paper, we do not consider self-join involving a probabilistic table.

Figure 1 shows a probabilistic database  $D^p$  with two relations,  $S^p$  and  $T^p$ : the tuples in  $S^p$  have probabilities 0.9 and 0.6, and the unique tuple in  $T^p$  has probability 0.5. We use the superscript  $p$  to emphasize that a relation or a database is probabilistic. We assume in this example that all the tuples represent independent probabilistic events.

### 2.2 Query evaluation on probabilistic databases

On a probabilistic database, a query answer is correct only if the resulting data and their probabilities are both correct. Therefore, query evaluation on a

probabilistic database includes not only *data computation* but also *probability computation*. There are two ways of probabilistic query processing: one couples data computation and probability computation together [11], the other does them separately [28]. We will focus on the former.

We further focus on conjunctive queries expressible by datalog or equivalently by select (distinct)-project-join (SPJ). Accordingly a query plan  $P$  is expressed using relational operators  $\sigma, \Pi, \bowtie (\times)$ . A probabilistic relation with atomic events which satisfies the  $FD R^p.E \rightarrow Attr(R)$  is called extensional. Otherwise, it is called intensional.

**Intensional query evaluation** Intensional query evaluation on probabilistic databases works by modifying relational operators to handle complex events [15] step by step in each intermediate result. Let  $\sigma^i, \Pi^i, \times^i$  be the modified operators, which have the following definitions [11]:  $\sigma^i$  acts like  $\sigma$  then copies the complex events from the input tuples to the output tuples;  $\Pi^i$  associates to a tuple  $t$  the complex event  $e_1 \vee \dots \vee e_n$  obtained from the complex events of all input tuples  $t_1, \dots, t_n$  that project into  $t$ ; and  $\times^i$  simply associates to a product tuple  $(t, t')$  the complex event  $e \wedge e'$ . The authors [11] proposed to compute for each possible tuple  $t$  a probability rank that  $t$  belonged to any answer, and sorted tuples sorted by this rank, which was denoted by  $q^{rank}(D^P)$ .

With intensional query evaluation, computation on complex event  $e(s_1, \dots, s_k)$  is exponential in  $k$ , since  $Pr(e)$  is #P-complete [33] even for complex events without negation. As a result, using intensional semantics to compute the rank probabilities [11] is impractical.

**Extensional query evaluation** Extensional query evaluation [11] is based on modifying the query operators to compute probabilities rather than complex events. It is called extensional since real numbers rather than event expressions are dealt with, and therefore efficient. The input tuples have independent events with  $Pr_p(t) \in [0, 1]$  for each tuple  $t$ .

Let  $\sigma^e, \Pi^e, \times^e$  be modified operators defined as follows [11]:  $\sigma^e$  acts like  $\sigma$  that propagates the probabilities of tuples from the input to the output,  $\Pi^e$  computes the probability of a tuples  $t$  as  $1 - (1 - p_1)(1 - p_2)\dots(1 - p_n)$  where

$p_1, \dots, p_n$  are the probabilities of all input tuples projecting to  $t$ , while  $\times$  computes the probability of each tuple  $(t, t')$  as  $p \times p'$ . Let  $P^e(D^p)$  be the result of applying plan  $P$  chosen for query  $q$ . If  $P^e(D^p) = q^{rank}(D^p)$  then we say that the extensional semantics of plan  $P$  is correct. A query plan is safe if its extensional semantics is correct for all probabilistic database instances of a given schema and FDs. The formal definition of safety is given in [11] as following.

**Definition 1** *Given a schema  $\bar{R}^p$  and the set of FDs  $\Gamma^p$ , a plan for a query  $q$  is **safe** if  $P^e(D^p) = q^{rank}(D^p)$  for all  $D^p$  of that schema.*

**Example 1** *Let us consider the database  $D^p$  described in Figure 1 and the query  $q$  shown below.*

$$q(D) : -S^p(A, B), T^p(C, D), B = C \quad (1)$$

*Consider the query plan  $P = \Pi_D^e(S^p \bowtie_{B=C}^e T^p)$  for (1). The meaning of a probabilistic database is a probability distribution on all database instances, which we call possible worlds, and denote  $pwd(D^p)$ . With the possible world semantic, the results of the running example is shown in Figure 2.*

*If we adopt extensional query evaluation method, the data item is 'p' and its probability is 0.615. So the data item in the result of the query plan  $P = \Pi_D^e(S^p \bowtie_{B=C}^e T^p)$  is correct while its probability is wrong. The reason is that the two tuples in  $S^p \bowtie_{B=C}^e T^p$  are not independent events, hence the formula used in the plan is wrong. However, let us consider an alternative plan  $P_1 = \Pi_D^e(\Pi_B^e(S^p) \bowtie_{B=C}^e T^p)$ . Now the data item in the result is 'p' and its probability is 0.48. This time both the data item and its probability are correct.*

This example indicates the importance of query plan selection on probability databases - plans with the same relational algebra semantics are not all correct from probability point of view.

### 2.3 Probabilistic query example

We test our query plan generation algorithms on the probabilistic relations generated dynamically from regular relations corresponding to approximate queries. As mentioned before, these algorithms can also be directly applied to evaluate queries on the probabilistic databases where the probability values of data are pre-prepared.

In this case, a SQL query with approximate matches is translated into a regular query over the generated probabilistic relations. The query results are ranked by probability. This can be illustrated by the following simple example from DBLP [13]

```
SELECT a.authorid as id, a.name, p.title  
  
FROM Authors a, Write w, Papers p  
  
WHERE a.authorid = w.authorid  
  
      AND w.paperid = p.paperid  
  
      AND a.name  $\approx$  'Jim Gray'  
  
      AND p.title  $\approx$  'database system'
```

where predicates on the *name* attribute of relation *Authors*, and on the *title* attribute of relation *Papers*, are uncertain.

Corresponding to the approximate query condition "a.name  $\approx$  'Jim Gray'", each tuple in the relation *Authors* is assigned a probability based on how well it matches the predicate "name  $\approx$  'Jim Gray'". This is done using a function based on the 3-gram distance strategy [24] for gaining the probability, which measures the similarities of two words by the number of triplets of consecutive letters common to both words. This function results in a probabilistic relation, denoted by *Author<sup>p</sup>*. Similarly, the uncertain predicate on *Papers* corresponds to a probabilistic relation *Papers<sup>p</sup>*. Then, we evaluate the following query that is obtained by dropping the similarity predicates from the original SQL query:

```

SELECT a.authorid as id, name,title
FROM Authorsp a, Write w, Papersp p
WHERE a.authorid = w.authorid

AND w.paperid = p.paperid

```

With the proposed algorithms, the result is shown in Table 1. The generated query plan will be shown in Section 6.2.

### 3 Simple and complex projections in extensional query evaluation

We classify projections into simple ones and complex ones where a simple projection involves only one relation, and a complex projection involves two or more relations. We can prove that any execution plan for a probabilistic conjunctive query without complex projection is safe, that is, it yields correct data as well as probability without requiring special treatment. This will allow us to focus and simplify our solution.

Recall that a probabilistic database schema  $\bar{R}^p$  may consist both of probabilistic relation names, which have an event attribute  $E$ , and deterministic relation names. For a conjunctive query  $q$ , we use the following notations as in [11]:

- $Rel_s(q) = \{R_1, \dots, R_k\}$  all relation names occurring in  $q$ . We assume that each relation name occurs at most once in the query.  $|Rel_s(q)|$  denotes the number of relations in the query  $q$ .
- $PRel_s(q) =$  the probabilistic relation names in  $q$ ,  $PRel_s(q) \subseteq Rel_s(q)$ .
- $Attr(q) =$  all attributes in all relations in  $q$ . To disambiguate, we denote attributes as  $R_i.A$ .
- $Head(q) =$  the set of attributes that are in the output of the query  $q$ .  $Head(q) \subseteq Attr(q)$ .

Let  $q$  be a conjunctive query. We define the *induced* functional dependencies  $\Gamma^p(q)$  on  $Attr(q)$  as [11]:

- Every FD in  $\Gamma^p$  is also in  $\Gamma^p(q)$ .
- For every join predicate  $R_i.A = R_j.B$ , both  $R_i.A \rightarrow R_j.B$  and  $R_j.B \rightarrow R_i.A$  are in  $\Gamma^p(q)$ .

Below we study the projection safety based on the following two theorems given in [11].

**Theorem 1** *Consider a database schema where all the probabilistic relations are tuple-independent. Let  $q$  and  $q'$  be conjunctive queries that do not share any relation name. Then,*

1.  $\sigma_c^e$  is always safe in  $\sigma_c(q)$ .
2.  $\times^e$  is always safe in  $q \times q'$ .
3.  $\Pi_{A_1, \dots, A_k}^e$  is safe in  $\Pi_{A_1, \dots, A_k}(q)$  iff for every  $R^p$  the following can be inferred from  $\Gamma^p(q)$ :

$$A_1, \dots, A_k, R^p.E \rightarrow \text{Head}(q). \quad (2)$$

**Theorem 2** *Let  $P$  be a safe relational algebra plan for a query  $q$  consisting of selects, projects and joins. Then, all operators in  $P$  are safe.*

Let  $A(R_i^p) = \{A_1, \dots, A_k, R_i^p.E\}$ . We start from the following definitions and corollaries.

**Definition 2** *Consider a database  $D^p$  where all probabilistic relations  $R_i^p \in D^p$  ( $i = 1, \dots, n$ ) are tuple-independent. Let a query be  $q = \Pi_{A_1, \dots, A_k}^e(q_i)$  and  $R_i^p$  be in  $q_i$ . Based on  $\Gamma^p(q)$ , the set of attributes which can be inferred from  $A_1, \dots, A_k, R_i^p.E$  is represented by  $\{A_1, \dots, A_k, R_i^p.E\}^+$  and we denote it by  $\text{InfAttr}_{A(R_i^p)}(q)$ , and let  $\text{InfAttr}(q) = \bigcap_{i=1}^n \text{InfAttr}_{A(R_i^p)}(q)$ .*

$\text{InfAttr}(q)$  may cover some relations in  $q = \Pi_{A_1, \dots, A_k}^e(q_i)$ . Maximal coverage set ( $\text{MCS}(q)$ ) of  $\text{InfAttr}(q)$  has the following property: For each  $R_i^p \in \text{Rels}(q_i)$ , if  $\text{InfAttr}(q) \supseteq \text{Attr}(R_i^p)$ , then  $\text{MCS}(q) = \text{MCS}(q) \cup \{R_i^p\}$  ( $i = 1, \dots, n$ ). Similarly, for each  $R_j \in \text{Rels}(q_i)$ , if  $\text{InfAttr}(q) \supseteq \text{Attr}(R_j)$ , then  $\text{MCS}(q) = \text{MCS}(q) \cup \{R_j\}$  ( $j = 1, \dots, m$ ). Let  $\text{MCS}(\text{Attr}(R_i^p)^+)$  denote the maximal coverage set of  $\text{Attr}(R_i^p)^+$ ,  $\text{MCS}(\text{Attr}(R_i^p)^+)$  has the following prop-

erty: For each  $R_j^p \in \text{Rels}(q)$ , if  $\text{Attr}(R_i^p)^+ \supseteq \text{Attr}(R_j^p)$ , then  $\text{MCS}(\text{Attr}(R_i^p)^+) = \text{MCS}(\text{Attr}(R_i^p)^+) \cup \{R_j^p\}$ . Similarly, for each  $R_j \in \text{Rels}(q)$ , if  $\text{Attr}(R_i^p)^+ \supseteq \text{Attr}(R_j)$ , then  $\text{MCS}(\text{Attr}(R_i^p)^+) = \text{MCS}(\text{Attr}(R_i^p)^+) \cup \{R_j\}$ . We use  $\text{Attr}(\text{MCS}(q))$  denoting all attributes of tables in  $\text{MCS}(q)$ ,  $\text{Attr}(\text{MCS}(q))^+$  denoting all attributes which can be inferred from  $\text{Attr}(\text{MCS}(q))$  and  $|\text{MCS}(q)|$  denoting the number of relations in  $\text{MCS}(q)$ .

**Corollary 1** *Suppose  $q = \Pi_{A_1, \dots, A_n}(q_i)$ , then  $\text{Attr}(\text{MCS}(q))^+ \subseteq \text{InfAttr}(q)$ .*

Proof. We prove the theorem by contradiction. Suppose  $\exists R_i^p.A \in \text{Attr}(\text{MCS}(q))^+$  but  $R_i^p.A \notin \text{InfAttr}(q)$ . There are two cases:

1.  $R_i^p.A \in \text{Attr}(\text{MCS}(q))$ : Because  $\text{Attr}(\text{MCS}(q)) \subseteq \text{InfAttr}(q)$ , we have  $R_i^p.A \in \text{InfAttr}(q)$ .
2.  $R_i^p.A \notin \text{Attr}(\text{MCS}(q))$ : Since  $R_i^p.A \in \text{Attr}(\text{MCS}(q))^+$ ,  $\exists R_j^p.B \in \text{Attr}(\text{MCS}(q))$  and there must be a join condition  $R_i^p.A = R_j^p.B$  in the query. Therefore, for any  $R_k^p \in q$ ,

$$\begin{aligned} A_1, \dots, A_n, R_k^p.E &\rightarrow \dots, \text{Attr}(\text{MCS}(q)), \dots \\ &\rightarrow \dots, \text{Attr}(\text{MCS}(q)), R_i^p.B, \dots \\ &\rightarrow \dots, \text{Attr}(\text{MCS}(q)), R_i^p.A, R_j^p.B, \dots \end{aligned}$$

Thus  $\text{InfAttr}(q) \supseteq \text{Attr}(\text{MCS}(q)) \cup \{R_i^p.A\}$ , which means  $R_i^p.A \in \text{InfAttr}(q)$ .

Both cases contradict the assumptions that  $\exists R_i^p.A \in \text{Attr}(\text{MCS}(q))^+$  but  $R_i^p.A \notin \text{InfAttr}(q)$ , therefore  $\text{Attr}(\text{MCS}(q))^+ \subseteq \text{InfAttr}(q)$ . Similarly, we can prove if  $\exists R_j.A \in \text{Attr}(\text{MCS}(q))^+$ , then  $R_j.A \in \text{InfAttr}(q)$ . Thus the corollary is proved.  $\square$

**Corollary 2** *Consider a database where all the probabilistic relations are tuple-independent. Then,*

1. *If it only includes  $\sigma_c^e$ ,  $\times^e$  or simple projections, the query  $q$  is safe.*
2. *If  $|\text{Rels}(q)| \geq 2$ ,  $q = \Pi_{A_1, \dots, A_k}^e(q_i)$  is safe iff  $\text{Head}(q_i) \subseteq \text{InfAttr}(q)$ .*

Proof. 1. Follows trivially from Theorem 1.

2. By Theorem 1,  $q = \Pi_{A_1, \dots, A_k}^e(q_i)$  is safe iff for every  $R_i^p, A_1, \dots, A_k, R_i^p.E \rightarrow \text{Head}(q_i)$  ( $i = 1, \dots, n$ ). Thus  $\text{Head}(q_i) \subseteq \bigcap_{i=1}^n \text{InfAttr}_{A(R_i^p)}(q)$ . Therefore,  $q = \Pi_{A_1, \dots, A_k}^e(q_i)$  is safe iff  $\text{Head}(q_i) \subseteq \text{InfAttr}(q)$ .  $\square$

With Corollary 1 and 2, we know that only complex projections influence the safety of SPJ queries. The left-deep join tree (L-tree) [16] is an useful tool to describe execute plan in the conventional relational database system. We call the tree to describe the safe execute plan safe tree (S-tree) in the probabilistic databases.

**Example 2** Continuing **Example 1**, suppose both  $S^p$  and  $T^p$  are tuple-independent probabilistic relations, and the execution plan for the query is  $q = \Pi_D^e(S^p \bowtie_{B=C}^e T^p)$ , whose L-tree is shown in Figure 3a. Let  $q_i = S^p \bowtie_{B=C}^e T^p$ ,  $\text{Head}(q_i) = \{A, B, C, D\}$  and  $\Gamma^p(q) = \{S^p.A, S^p.B \rightarrow S^p.E; T^p.C, T^p.D \rightarrow T^p.E; S^p.E \rightarrow S^p.A, S^p.B; T^p.E \rightarrow T^p.C, T^p.D; S^p.B \rightarrow T^p.C; T^p.C \rightarrow S^p.B\}$ . By Theorem 1,

$$\begin{aligned} T^p.D, S^p.E \rightarrow S^p.A, S^p.B, T^p.D \\ \rightarrow S^p.A, S^p.B, T^p.C, T^p.D \\ T^p.D, T^p.E \rightarrow T^p.C, T^p.D \\ \rightarrow S^p.B, T^p.C, T^p.D \end{aligned}$$

Then  $\text{InfAttr}(q) = \{T^p.D, S^p.E\}^+ \cap \{T^p.D, T^p.E\}^+ = \{S^p.B, T^p.C, T^p.D\}$ . Therefore,  $\text{Head}(q_i) \not\subseteq \text{InfAttr}(q)$ . By Corollary 2, the plan  $\Pi_D^e(S^p \bowtie_{B=C}^e T^p)$  is unsafe.

Assume that the query plan is  $q = \Pi_D^e(\Pi_B^e(S^p) \bowtie_{B=C}^e T^p)$ . Let  $q_j = \Pi_B^e(S^p) \bowtie_{B=C}^e T^p$ , then  $\text{Head}(q_j) = \{B, C, D\}$ . Hence  $\text{Head}(q_j) \subseteq \text{InfAttr}(q)$ . By Corollary 2, the plan  $\Pi_D^e(\Pi_B^e(S^p) \bowtie_{B=C}^e T^p)$ , whose S-tree is shown in Figure 3b, is safe. By the query plan, we can compute the correct probability.

## 4 The novel safe plan generation algorithm

In this section, we will describe the Multiway-Split algorithm, prove its correctness and show its efficiency.

#### 4.1 The preprocessing algorithm

Selection is one to one mapping, so it can not cause any error in computing probabilities. Projection for an original relation is many to one mapping, thus it does not cause any error in computing probabilities either. As a result, pushing down selection and projection related to each original relation does not bring errors in computing probability for a query  $q$ .

Before going to the algorithm, we first consider the attributes associated with join or projection in a query, and prove that a projection-join query plan of two relations involving only such attributes is safe.

**Definition 3** *If an attribute is associated with join conditions or  $Head(q)$  in a query  $q$ , we call it association attribute. An association attribution bag of query  $q$  is denoted by  $A^2B(q)$ . It includes the following attributes:*

- *If  $R_i^p.A \in Head(q)$ , then we put  $R_i^p.A$  into  $A^2B(q)$ ;*
- *If there is a join predicate  $R_i^p.B = T_j^p.C$ , then we put both  $R_i^p.B$  and  $T_j^p.C$  into  $A^2B(q)$ ;*

If all the duplicate attributes are eliminated from the association attribute bag of  $q$ , we call it the association attribute set, which is denoted by  $A^2S(q)$ . The association attribute set of a relation  $R_i^p$  is denoted by  $A^2S(R_i^p)$ . It is the intersection of  $A^2S(q)$  and  $Attr(R_i^p)$ , that is,  $A^2S(R_i^p) = A^2S(q) \cap Attr(R_i^p)$ . If the attribute of  $R_i^p$  can not be inferred from  $A^2S(R_i^p)$ , we call it non-association attribute. If  $A^2S(R_i^p) \not\supseteq Key(R_i^p)$ ,  $R_i^p$  has non-association attributes. Similarly, if the attribute of  $R_j$  can not be inferred from  $A^2S(R_j)$ , we call it non-association attribute. If  $A^2S(R_j) \not\supseteq Key(R_j)$ ,  $R_j$  has non-association attributes.

Our preprocessing algorithm is shown in Algorithm 1, which includes the following two steps.

1. We first push down selection predicate  $R_i^p.A\theta c$  and projection predicate  $\Pi_A(R_j^p)$ . Based on the new query  $q$ , we drive  $A^2S(q)$ .
2. For probabilistic relations, we project out the non-association attributes if

$A^2S(R_i^p) \not\subseteq \text{Key}(R_i^p)$ . For deterministic tables, we also project out the non-association attributes if  $A^2S(R_j) \not\subseteq \text{Key}(R_j)$ .

**Example 3** Consider the following query

$$q(U) : - L^p(A), J^p(B, C, D), R^p(F), A = B, D = F, S^p(U, U_1), U_1 = B.$$

Assume that  $U$  is a key for the relation  $S^p(U, U_1)$ . According to Definition 3,  $A^2B(q) = \{A, B, D, F, U, B, U_1\}$ ,  $A^2S(q) = \{A, B, D, F, U, U_1\}$  and  $\Gamma^p(q) = \{L^p.E \rightarrow L^p.A; L^p.A \rightarrow L^p.E; J^p.E \rightarrow J^p.B, J^p.C, J^p.D; J^p.B, J^p.C, J^p.D \rightarrow J^p.E; R^p.E \rightarrow R^p.F; R^p.F \rightarrow R^p.E; S^p.E \rightarrow S^p.U, S^p.U_1; S^p.U, S^p.U_1 \rightarrow S^p.E; L^p.A \rightarrow J^p.B; J^p.B \rightarrow L^p.A; J^p.D \rightarrow R^p.F; R^p.F \rightarrow J^p.D; S^p.U_1 \rightarrow J^p.B; J^p.B \rightarrow S^p.U_1; S^p.U \rightarrow S^p.U_1\}$ . Their association attribute sets are as follows.

$$\begin{aligned} A^2S(L^p) &= \text{Attr}(L^p) \cap A^2S(q) = \{A\} \\ A^2S(J^p) &= \text{Attr}(J^p) \cap A^2S(q) = \{B, D\} \\ A^2S(R^p) &= \text{Attr}(R^p) \cap A^2S(q) = \{F\} \\ A^2S(S^p) &= \text{Attr}(S^p) \cap A^2S(q) = \{U, U_1\} \end{aligned}$$

Because  $A^2S(L^p)$  is equal to  $\text{Attr}(L^p)$ , there is no non-association attribute in relation  $L^p$ , hence  $L^p$  need not project out any attribute. For the same reason, neither relation  $R^p$  nor  $S^p$  need project out any attribute. Because  $A^2S(J^p) \not\subseteq \text{Key}(J^p)$ ,  $J^p$  should project out the non-association attribute  $\{C\}$ , that is,  $J_1^p = \Pi_{BD}(J^p)$ . After preprocessing, we have the following query plan:

$$q(U) : - L^p(A), J_1^p(B, D), R^p(F), A = B, D = F, S^p(U, U_1), U_1 = B.$$

Then  $J^p.E \rightarrow J^p.B, J^p.C, J^p.D$  and  $J^p.B, J^p.C, J^p.D \rightarrow J^p.E$  in  $\Gamma^p(q)$  are replaced by  $J_1^p.E \rightarrow J_1^p.B, J_1^p.D$  and  $J_1^p.B, J_1^p.D \rightarrow J^p.E$ .

**Theorem 3** Let  $q = \Pi_{C_1, \dots, C_n}^e (R_i^p \bowtie_{A=B}^e T_j^p)$  be an execution plan for a query  $q$ . If both  $R_i^p$  and  $T_j^p$  only have association attributes, then  $\Pi_{C_1, \dots, C_n}^e (R_i^p \bowtie_{A=B}^e T_j^p)$  is safe.

Proof. We prove the theorem by contradiction. Let  $q_i = R_i^p \bowtie_{A=B}^e T_j^p$ . Suppose  $\exists R_i^p.D \in \text{Head}(q_i)$  but  $R_i^p.D \notin \text{InfAttr}(q)$ , since  $R_i^p.D$  is an association

attribute, it should be one of the following two cases:

1. If  $R_i^p.D \in \{C_1, \dots, C_m\}$ , then  $R_i^p.D \in (\{C_1, \dots, C_m\} \cup \text{Attr}(R_i^p))^+$  and  $R_i^p.D \in (\{C_1, \dots, C_m\} \cup \text{Attr}(T_j^p))^+$ . Therefore,  $R_i^p.D \in \text{InfAttr}_{A(R_i^p)}(q)$  and  $R_i^p.D \in \text{InfAttr}_{A(T_j^p)}(q)$ , which means  $R_i^p.D \in \text{InfAttr}(q)$ .
2. If  $R_i^p.D \in \{R_i^p.A, T_j^p.B\}$ , we have

$$\begin{aligned} C_1, \dots, C_m, R_i^p.E &\rightarrow C_1, \dots, C_m, \dots, R_i^p.A, T_j^p.B, \dots \\ C_1, \dots, C_m, T_j^p.E &\rightarrow C_1, \dots, C_m, \dots, R_i^p.A, T_j^p.B, \dots \end{aligned}$$

Thus  $R_i^p.D \in \text{InfAttr}_{A(R_i^p)}(q)$  and  $R_i^p.D \in \text{InfAttr}_{A(T_j^p)}(q)$ . Hence  $R_i^p.D \in \text{InfAttr}(q)$ .

Both cases contradict the assumptions that  $R_i^p.D \in \text{Head}(q_i)$  but  $R_i^p.D \notin \text{InfAttr}(q_i)$ . Therefore,  $\text{Head}(q_i) \subseteq \text{InfAttr}(q)$ . By Corollary 2,  $\Pi_{C_1, \dots, C_m}(R_i^p \bowtie_{A=B} T_j^p)$  is safe. Therefore, the theorem is proved.  $\square$

#### 4.2 The Multiway-Split algorithm

Before presenting the algorithm for finding a safe plan, we first introduce a few concepts. Assume  $|\text{MCS}(q)| = n$ , there are the following two cases:

- If  $n > 0$  and  $n < |\text{Rels}(q)|$ ,  $\text{Rels}(q)$  is partitioned into  $n+1$  sets. Each of the first  $n$  relation set  $R_{S_i}$  only includes one relation of  $\text{MCS}(q)$ ; whereas the last relation set  $R_{S_{n+1}}$ , which is called the splitting relation set, includes all relations of  $\text{Rels}(q) - \text{MCS}(q)$ . Thus  $R_{S_1}, R_{S_2}, \dots, R_{S_{n+1}}$  form a separation for query  $q$ .
- If  $n$  is equal to  $|\text{Rels}(q)|$ ,  $\text{Rels}(q)$  is partitioned into  $n$  sets. Each relation set  $R_{S_i}$  only includes one relation of  $\text{MCS}(q)$ . Thus  $R_{S_1}, R_{S_2}, \dots, R_{S_n}$  form a separation for query  $q$ .

Let  $\text{Attr}(R_S)$  denote all attributes in all relations in  $R_S$ . For the attributes belonging to splitting relation set  $R_{S_{n+1}}$ , if they can join the attributes in other relation sets, we call them the partition join attributes. These attributes form

a set called partition join attribute set and denoted by  $PJAS(R_{S_{n+1}})$ . Furthermore, partition association attribute set denoted by  $PA^2S(R_{S_{n+1}})$  is the union of  $PJAS(R_{S_{n+1}})$  and  $(Head(q) \cap Attr(R_{S_{n+1}}))$ , that is,  $PA^2S(R_{S_{n+1}}) = PJAS(R_{S_{n+1}}) \cup (Head(q) \cap Attr(R_{S_{n+1}}))$ .

**Definition 4** Suppose  $q = \Pi_{Head(q)}^e(R_1^p \bowtie^e R_2^p \dots \bowtie^e R_n^p)$  and each relation  $R_i^p$  has no non-association attribute, if  $q$  is split into  $q = \Pi_{Head(q)}^e(q_i)$  and  $q_i = q_1 \bowtie^e q_2 \dots \bowtie^e q_m$ , then  $InfAttr(q) \supseteq Head(q_i)$ . We call this split a safe split.

If a cartesian-product can be avoided by query optimization in the safe plan, we call it unnecessary cartesian-product, otherwise, we call it necessary cartesian-product.

**Theorem 4** Consider a database where all the probabilistic relations are tuple-independent. After preprocessing, let  $q = \Pi_{Head(q)}^e(R_1^p \bowtie^e R_2^p \bowtie^e \dots \bowtie^e R_n^p)$  ( $Head(q) \neq \{\}$   $\wedge n > 1$ ), its  $MCS(q)$  includes  $m$  ( $1 \leq m \leq n$ ) relations (for example  $R_1^p, \dots, R_m^p$ ), then:

1. If  $m < n$ , the split  $\Pi_{Head(q)}^e(R_1^p \bowtie^e R_2^p \dots \bowtie^e R_m^p \bowtie^e \Pi_{PA^2S(R_{S_1})}(R_{m+1}^p \dots \bowtie^e R_n^p))$  is safe.
2. If  $m$  is equal to  $n$ ,  $\Pi_{Head(q)}^e(R_1^p \bowtie^e R_2^p \dots \bowtie^e R_m^p)$  is safe.

Proof. We prove the theorem as follows.

1. Let  $q_j = R_1^p \bowtie^e R_2^p \dots \bowtie^e R_m^p$ ,  $R_{S_1} = \{R_{m+1}, \dots, R_n\}$ . Then  $q_i = q_j \bowtie^e \Pi_{PA^2S(R_{S_1})}(R_{m+1}^p \bowtie^e \dots \bowtie^e R_n^p)$  and  $Head(q_i) = Attr(q_j) \cup PA^2S(R_{S_1}) = Attr(q_j) \cup PJAS(R_{S_1}) \cup (Head(q) \cap Attr(R_{S_1}))$ . For any  $R_i^p.B \in Head(q_i)$ ,

- If  $R_i^p \in MCS(q)$ , then  $InfAttr(q) \supseteq Attr(R_i^p)$ . Therefore,  $R_i^p.B \in InfAttr(q)$ .
- If  $R_i^p.B \in (Head(q) \cap Attr(R_1))$ , then  $R_i^p.B \in InfAttr(q)$ .
- If  $R_i^p.B \in PJAS(R_{S_1})$ , then there must exist a relation  $R_j^p \in MCS(q)$  and a join condition  $R_j^p.C = R_i^p.B$ . Thus  $R_i^p.B \in Attr(MCS(q))^+$ . By Corollary 1,  $Attr(MCS(q))^+ \subseteq InfAttr(q)$ , therefore  $R_i^p.B \in InfAttr(q)$ .

Thus  $Head(q_i) \subseteq InfAttr(q)$ . Hence the split is safe.

2. If  $m$  is equal to  $n$ , we have

$$\begin{aligned} \text{InfAttr}(q) &\supseteq \text{Attr}(\text{MCS}(q))^+ \\ &\supseteq \text{Attr}(\text{MCS}(q)) \\ &= \text{Head}(q_i) \end{aligned}$$

Hence  $\Pi_{\text{Head}(q)}^e(R_1^p \bowtie^e R_2^p \dots \bowtie^e R_m^p)$  is safe.  $\square$

**Corollary 3** *Consider a database where all the probabilistic relations are tuple-independent. After preprocessing, let  $q = \Pi_{\text{Head}(q)}^e(R_1^p \bowtie^e R_2^p \bowtie^e \dots \bowtie^e R_n^p)$  and  $(\text{Head}(q) \neq \{\})$ . If  $\text{MCS}(q)$  is an empty set, the query has no safe plan.*

Proof. We prove the corollary by contradiction. Assume that the query has a safe plan  $q = \Pi_{\text{Head}(q)}^e(q_j)$ , then  $\text{Head}(q_j) \subseteq \text{InfAttr}(q)$ . Let  $q_i = R_1^p \bowtie^e R_2^p \bowtie^e \dots \bowtie^e R_n^p$ . Because  $\text{MCS}(q) = 0$ ,  $\text{InfAttr}(q) \subset \text{Attr}(q_i) = \text{Head}(q_i)$ ,  $\text{Head}(q_j) \subset \text{Head}(q_i)$ . Then there must exist a safe split in  $q$  and  $q_j = R_1^p \bowtie^e \dots \bowtie^e R_m^p \bowtie^e \Pi^e(R_{m+1}^p \bowtie^e \dots \bowtie^e R_n^p)$  ( $0 < m < n$ ). Therefore,  $\text{MCS}(q) = (n - m) > 0$ , which contradicts the assumption that  $\text{MCS}(q)$  is an empty set. Hence the query has no safe plan. This proves the corollary.  $\square$

Algorithm 2 is our Multiway-Split algorithm, which includes the following steps:

1. If  $q$  includes no complex projection, any plan is safe. If  $\text{Head}(q) = \{\}$ , use the Safe-Plan algorithm [11].
2. We preprocess the query plan  $q$  and then invoke the Split algorithm. Thereafter, either a safe plan is returned or an error is returned.

Algorithm 3 is the Split algorithm, which includes the following steps:

1. If there is no complex projection in  $q$ , return any plan for  $q$ . If there is only one  $\bowtie$  in  $q$ , return the present plan for  $q$ .
2. We calculate  $\text{InfAttr}(q)$  and  $\text{MCS}(q)$ . Let  $|\text{MCS}(q)| = n$ . If  $n$  is equal to 0, no safe plans exists. If  $n < |\text{RelS}(q)|$ , we can split  $q$  into  $n + 1$  relation sets, then we calculate their safe plans recursively. If  $n$  is equal to  $|\text{RelS}(q)|$ , the

present plan is safe.

**Example 4** Continuing **Example 3**. After pre-process, we have

$$q(U) : - L^p(A), J_1^p(B, D), R^p(F), A = B, D = F, S^p(U, U_1), U_1 = B.$$

Let  $q_1 = L^p \bowtie_{A=B}^e J_1^p \bowtie_{U_1=B}^e S^p \bowtie_{D=F}^e R^p$ . By Theorem 1, we have

$$\begin{aligned} L^p.E, S^p.U &\rightarrow S^p.U, S^p.U_1, J_1^p.B, L^p.A \\ J_1^p.E, S^p.U &\rightarrow S^p.U, S^p.U_1, J_1^p.B, J_1^p.D, L^p.A \\ R^p.E, S^p.U &\rightarrow R^p.F, S^p.U, S^p.U_1, J_1^p.B, L^p.A \\ S^p.E, S^p.U &\rightarrow S^p.U, S^p.U_1, J_1^p.B, L^p.A \end{aligned}$$

Thus  $\text{InfAttr}_{A(L^p)}(q) = \{S^p.U, S^p.U_1, J_1^p.B, L^p.A\}$ ,  $\text{InfAttr}_{A(J_1^p)}(q) = \{S^p.U, S^p.U_1, J_1^p.B, J_1^p.D, L^p.A\}$ ,  $\text{InfAttr}_{A(R^p)}(q) = \{R^p.F, S^p.U, S^p.U_1, J_1^p.B, L^p.A\}$  and  $\text{InfAttr}_{A(S^p)}(q) = \{S^p.U, S^p.U_1, J_1^p.B, L^p.A\}$ . So  $\text{InfAttr}(q) = \text{InfAttr}_{A(L^p)}(q) \cap \text{InfAttr}_{A(J_1^p)}(q) \cap \text{InfAttr}_{A(R^p)}(q) \cap \text{InfAttr}_{A(S^p)}(q) = \{S^p.U, S^p.U_1, J_1^p.B, L^p.A\}$ .

Then  $\text{MCS}(q) = \{L^p, S^p\}$ , because  $\text{MCS}(q)$  includes two relations  $L^p$  and  $S^p$ , we will split query  $q$  into three relation sets:  $\{L^p\}$ ,  $\{S^p\}$  and  $\{J_1^p, R^p\}$ . Hence  $q = \Pi_U^e(L^p \bowtie_{A=U_1}^e S^p \bowtie_{U_1=B}^e \Pi_B(J_1^p \bowtie_{D=F}^e R^p))$ .

Finally,  $q_2$  only includes one relation  $L^p$ ,  $q_3$  only includes one relation  $S^p$  and  $q_4 = \Pi_B^e(J_1^p \bowtie_{D=F}^e R^p)$ . Because  $q_2$ ,  $q_3$  and  $q_4$  are all safe, the query plan  $P = \Pi_U^e(L^p \bowtie_{A=U_1}^e S^p \bowtie_{U_1=B}^e \Pi_B^e(J_1^p \bowtie_{D=F}^e R^p))$ , whose  $S$ -tree is shown in Figure 4, is safe.

Next, we will show the correctness of the proposed algorithm.

**Theorem 5** The Multiway-Split algorithm is sound, that is, any plan it returns is safe.

Proof. We will prove the theorem by induction. The algorithm returns a safe plan in the following three cases:

Case 1. It returns at Line 2. In this case, there are only joins, selections, or simple projections in the plan. By Corollary 2, the plan is safe. If  $Head(q) = \{\}$ , the Safe-Plan algorithm ensures that either the returned plan is safe or no plan returns.

Case 2. The main process of the Split algorithm is to split the query recursively. If it returns at Line 3, it is safe by Corollary 2. If it returns at Line 6, it is safe by Theorem 3.

Case 3. The Split algorithm returns at Line 14.  $q_1, q_2, \dots, q_{n+1}$  are all smaller than  $q$ . By Theorem 4 item 1, the split is safe. By induction, they are all safe plans. These plans are connected by join operators, thus the returned plan is safe. If the Split algorithm returns at Line 16, it is safe by Theorem 4 item 2.

This proves the theorem.  $\square$

**Theorem 6** *The Multiway-Split algorithm is complete.*

Proof. We prove the theorem in the following cases.

Case 1. If the query has no complex projection, our algorithm can return a safe plan for it.

Case 2. If  $Head(q) = \{\}$ , the Safe-Plan algorithm ensures that the returned plan is safe or no plan returns.

Case 3. After preprocessing, the query becomes  $q = \Pi_{A_1, \dots, A_n}^e (R_1^p \bowtie^e \dots \bowtie^e R_n^p)$ . Let  $q_i = R_1^p \bowtie^e \dots \bowtie^e R_n^p$ .

Subcase 1. The query has complex projection but only has one join operation in the complex projection. Our algorithm can return a safe plan for it.

Subcase 2. The query has complex projection but has more than one join operations.

- If  $((|MCS(q)| > 0) \wedge (|MCS(q)| = |RelS(q)|))$ , Theorem 4 item 2 ensures the present plan of the query is safe.

- If  $((|MCS(q)| > 0) \wedge (|MCS(q)| < |Rels(q)|))$ , Theorem 4 item 1 ensures the query has a safe split  $q = \Pi_{A_1, \dots, A_n}^e(q_1 \bowtie^e q_2 \dots \bowtie^e q_m)$ . Our algorithm recursively finds safe plan for each  $q_i$  until either it finds a safe plan or no safe plan can be found.

Hence if the query is safe, our algorithm can find a safe plan for it.  $\square$

**Theorem 7** *Algorithm Multiway-Split can avoid unnecessary cartesian-product in the safe plans.*

Proof. We prove the theorem by contradiction. Let  $Head(q_1) = \{B_1, B_2, \dots, B_n\}$  and  $Head(q_2) = \{A_1, A_2, \dots, A_m\}$ . There are the following two cases:

Case 1. Let the safe plan  $P = \Pi_{B_1, B_2, \dots, B_n}^e(q_i \bowtie_{A=B}^e \Pi_{A_1, A_2, \dots, A_m}(R_1^p \bowtie^e R_2^p \bowtie^e \dots R_k^p \bowtie^e R))$  has unnecessary cartesian-product between deterministic table and probabilistic table, for example,  $R$  has cartesian-product with other relations. Because the plan  $P$  is safe, then  $Attr(R) \subseteq Head(q_2)$  or  $Key(R) \subseteq Head(q_2)$ , and for any  $T_i^p \in Rels(q_i)$ ,

$$\begin{aligned} Head(q_1), T_i^p.E &\rightarrow Head(q_i) \cup Head(q_2) \\ &\rightarrow Head(q_i) \cup Head(q_2) \cup Attr(R) \\ Head(q_1), Head(q_2) &\rightarrow Head(q_i) \cup Head(q_2) \\ &\rightarrow Head(q_i) \cup Head(q_2) \cup Attr(R) \end{aligned}$$

Thus  $R \in MCS(q)$ . According to the proposed algorithm, the returned safe plan is  $P_1 = \Pi_{B_1, B_2, \dots, B_n}^e(q_i \bowtie^e R \bowtie^e \Pi_{C_1, C_2, \dots, C_l}^e(R_1^p \bowtie^e R_2^p \bowtie^e \dots R_k^p))$ , which does not have unnecessary cartesian-product between deterministic table and probabilistic table.

Case 2. Let the safe plan  $P = \Pi_{B_1, B_2, \dots, B_n}^e(q_i \bowtie_{A=B}^e \Pi_{A_1, A_2, \dots, A_m}(R_1^p \bowtie^e R_2^p \bowtie^e \dots R_k^p))$  has unnecessary cartesian-product between probabilistic tables, for example,  $R_j^p$  ( $1 \leq j \leq k$ ) has cartesian-product with other relations. Because the plan  $P$  is safe,  $Attr(R_j^p) \subseteq Head(q_2)$  or  $Key(R_j^p) \subseteq Head(q_2)$ , and for any  $T_i^p \in Rels(q_i)$ ,

$$\begin{aligned} Head(q_1), T_i^p.E &\rightarrow Head(q_i) \cup Head(q_2) \\ &\rightarrow Head(q_i) \cup Head(q_2) \cup Attr(R_j^p) \end{aligned}$$

$$\begin{aligned} \text{Head}(q_1), \text{Head}(q_2) &\rightarrow \text{Head}(q_i) \cup \text{Head}(q_2) \\ &\rightarrow \text{Head}(q_i) \cup \text{Head}(q_2) \cup \text{Attr}(R_j^p) \end{aligned}$$

Thus  $R_j^p \in MCS(q)$ . According to the proposed algorithm, if  $j < k$ , the safe plan is  $P_1 = \Pi_{B_1, B_2, \dots, B_n}^e(q_i \bowtie^e R_j^p \bowtie^e \Pi_{C_1, C_2, \dots, C_l}^e(R_1^p \bowtie^e R_2^p \dots \bowtie^e R_{j-1}^p \bowtie^e R_{j+1}^p \dots \bowtie^e R_k^p))$ , which does not have unnecessary cartesian-product between probabilistic tables. If  $j$  is equal to  $k$ , the returned safe plan is  $P_1 = \Pi_{B_1, B_2, \dots, B_n}^e(q_i \bowtie^e R_j^p \bowtie^e \Pi_{C_1, C_2, \dots, C_l}^e(R_1^p \bowtie^e R_2^p \dots \bowtie^e R_{j-1}^p))$ , which does not have unnecessary cartesian-product between probabilistic tables either.

Therefore, the proposed algorithm can avoid unnecessary cartesian-product.  $\square$

## 5 Query optimization

There are five transformation rules given in [11] as follows.

**Rule 1:** [*Join Commutativity*] Extensional joins are commutative

$$R \bowtie^e S \Leftrightarrow S \bowtie^e R$$

**Rule 2:** [*Join Associativity*] Extensional joins are associative

$$R \bowtie^e (S \bowtie^e T) \Leftrightarrow (R \bowtie^e S) \bowtie^e T$$

**Rule 3:** [*Cascading Projections*] Successively eliminating attributes from a relation is equivalent to simply eliminating all but the attributes retained by the last projection

$$\Pi_A^e(\Pi_{A \cup B}^e(R)) \Leftrightarrow \Pi_A^e(R)$$

**Rule 4:** [*Pushing Projections Below a Join*] A projection can be pushed below a join if it retains all the attributes used in the join.

$$\Pi_A^e(R \bowtie^e S) \Rightarrow (\Pi_{A_1}^e(R)) \bowtie^e (\Pi_{A_2}^e(S))$$

Where  $A_1$  and  $A_2$  are the attributes in  $R$  and  $S$  retained by the projection.

**Rule 5:** [*Lifting Projections Up a Join*] A projection can not always be lifted up a join. The following transformation rule can be applied only when the top  $\Pi^e$  operator in the resulting plan satisfies the Eq. 2 of Thm. 1 in this paper.

$$(\Pi_A^e(R)) \bowtie^e S \Rightarrow \Pi_{A \cup \text{Attrs}(S)}^e(R \bowtie^e S)$$

Furthermore, the following theorem is given in [11]:

**Theorem 8** *Let  $P_1$  and  $P_2$  be two safe plans for a query  $q$ . Then,  $P_1 \iff^* P_2$ .*

We call the safe plan generated by the Multiway-Split algorithm MSPlan and the safe plan generated by the Safe-Plan algorithm SPlan.

**Example 5** *By the Safe-Plan algorithm, the SPlan of the query in **Example 3** is  $P_1 = \Pi_U(L^p \bowtie_{A=B}^e \Pi_{BU}(S^p \bowtie_{U_1=B}^e \Pi_B(\Pi_{BD}(J^p) \bowtie_{D=F}^e R^p)))$ . The corresponding tree is shown in Figure 5. From **Example 4**, the MSPlan is  $P_2 = \Pi_U(L^p \bowtie_{A=B}^e S^p \bowtie_{U_1=B}^e \Pi_B^e(\Pi_{BD}^e(J^p) \bowtie_{D=F}^e R^p))$ . Comparing Figure 4 to Figure 5, we find that Figure 4 has less projections. By Theorem 8,  $P_1 \iff^* P_2$ .*

However, from the five transformation rules in [11], we can not prove  $P_1 \iff^* P_2$ . Based on the application, we extend the transformation rules as follows.

**Theorem 9** *Consider two plans  $P_1 = \Pi_C^e(q_3 \bowtie_{D_1=D_2}^e \Pi_A^e(q_1 \bowtie_{E_1=E_2}^e q_2))$  and  $P_2 = \Pi_C^e(q_3 \bowtie_{D_1=D_2}^e \Pi_{(A \cup B) \cap \text{Head}(q_1)}^e(q_1) \bowtie_{E_1=E_2}^e \Pi_{(A \cup B) \cap \text{Head}(q_2)}^e(q_2))$ , where  $q_1$  and  $q_2$  are queries,  $A^+ = \text{Head}(q_1) \cup \text{Head}(q_2)$  and  $B = E_1 \cup E_2$ :*

1. *the two plans have the same data results;*
2. *If  $P_1$  is a safe plan, then  $P_2$  is a safe plan;*
3. *If  $P_2$  is a safe plan, then  $P_1$  is a safe plan.*

Proof. Let  $\text{Key}(q)$  denotes all keys of tables in  $q$ . Because  $\text{Head}(q_1) \cup \text{Head}(q_2) =$

$A^+, (Key(q_1) \cup Key(q_2)) \subseteq A$ .

1. Let  $|q|$  denotes the cardinality of the result of  $q$ , since  $Key(q_1) \subseteq Head(q_1)$  and  $Key(q_1) \subseteq ((A \cup B) \cap Head(q_1))$ ,  $|q_1| = |\Pi_{(A \cup B) \cap Head(q_1)}(q_1)|$ . Similarly,  $|q_2| = |\Pi_{(A \cup B) \cap Head(q_2)}(q_2)|$ . Then

$$\begin{aligned} |\Pi_{(A \cup B) \cap Head(q_1)}(q_1) \bowtie_{E_1=E_2} \Pi_{(A \cup B) \cap Head(q_2)}(q_2)| &= |q_1 \bowtie_{E_1=E_2} q_2| \\ &= |\Pi_A(q_1 \bowtie_{E_1=E_2} q_2)| \end{aligned}$$

Thus the two plans have the same data results.

2. Because  $P_1$  is a safe plan,  $q_1$ ,  $q_2$  and  $q_3$  are all safe. For each  $R^p \in PRels(q_1)$ ,

$$\begin{aligned} C, R^p.E &\rightarrow Head(q_3) \cup A \\ &\rightarrow Head(q_3) \cup Head(q_1) \cup Head(q_2) \\ &\rightarrow Head(q_3) \cup ((A \cup B) \cap Head(q_1)) \cup ((A \cup B) \cap Head(q_2)) \end{aligned}$$

Similarly, for each  $R^p \in PRels(q_2)$  and  $R^p \in PRels(q_3)$  we can prove that  $C, R^p.E \rightarrow Head(q_3) \cup ((A \cup B) \cap Head(q_1)) \cup ((A \cup B) \cap Head(q_2))$ . Furthermore, for each  $T^p \in PRels(q_1)$

$$\begin{aligned} (A \cup B) \cap Head(q_1), T^p.E &\rightarrow (Key(q_1) \cup Key(q_2)) \cap Head(q_1), T^p.E \\ &\rightarrow Key(q_1), T^p.E \\ &\rightarrow Head(q_1), T^p.E \\ &\rightarrow Head(q_1) \end{aligned}$$

Thus  $\Pi_{(A \cup B) \cap Head(q_1)}^e(q_1)$  is safe. Similarly,  $\Pi_{(A \cup B) \cap Head(q_2)}^e(q_2)$  is safe. This demonstrates that  $P_2$  is a safe plan.

3. Because  $P_2$  is safe,  $q_1$ ,  $q_2$  and  $q_3$  are all safe. For each  $R^p \in PRels(q_1)$

$$\begin{aligned} C, R^p.E &\rightarrow Head(q_3) \cup ((A \cup B) \cap Head(q_1)) \cup ((A \cup B) \cap Head(q_2)) \\ &\rightarrow Head(q_3) \cup ((Key(q_1) \cup Key(q_2)) \cap Head(q_1)) \\ &\quad \cup ((Key(q_1) \cup Key(q_2)) \cap Head(q_2)) \\ &\rightarrow Head(q_3) \cup Key(q_1) \cup Key(q_2) \\ &\rightarrow Head(q_3) \cup Head(q_1) \cup Head(q_2) \\ &\rightarrow Head(q_3) \cup A \end{aligned}$$

Similarly, for each  $R^p \in PRels(q_2)$  and  $R^p \in PRels(q_3)$  we can prove that  $C, R^p.E \rightarrow Head(q_3) \cup A$ . Furthermore, for each  $T^p \in PRels(q_1)$

$$\begin{aligned} A, T^p.E &\rightarrow Head(q_1) \cup Head(q_2), T^p.E \\ &\rightarrow Head(q_1) \cup Head(q_2) \end{aligned}$$

Similarly, for each  $T^p \in PRels(q_2)$  we can prove that  $A, T^p.E \rightarrow Head(q_1) \cup Head(q_2)$ . Thus  $\Pi_A^e(q_1 \bowtie^e q_2)$  is safe. This proves that  $P_1$  is a safe plan.  $\square$

**Rule 6:** [*Extension of Pushing Projections Below a Join*] A projection can be pushed below a join if  $A^+ = Head(q_1) \cup Head(q_2)$  and  $B$  contains all the attributes used in the join between  $q_1$  and  $q_2$ .

$$\Pi_C^e(q_3 \bowtie^e \Pi_A^e(q_1 \bowtie^e q_2)) \Leftrightarrow \Pi_C^e(q_3 \bowtie^e \Pi_{(A \cup B) \cap Head(q_1)}^e(q_1) \bowtie^e \Pi_{(A \cup B) \cap Head(q_2)}^e(q_2))$$

**Example 6** Continuing **Example 5**, let  $q_1 = S^p$ ,  $q_2 = \Pi_B^e(\Pi_{BD}^e(J^p) \bowtie^e R^p)$  and  $q_3 = L^p$ . Then  $\{B, U\}^+ = \{B, U, U_1\} = Head(q_1) \cup Head(q_2)$ ,  $(\{B, U\} \cup \{B, U_1\}) \cap Head(q_2) = \{B, U, U_1\} \cap \{B\} = \{B\}$ , and  $(\{B, U\} \cup \{B, U_1\}) \cap Head(q_1) = \{B, U, U_1\} \cap \{U, U_1\} = \{U, U_1\}$ . By Rule 6,  $P_1 \Leftrightarrow P_2$ .

The differences between the Multiway-Split algorithm and the Safe-Plan algorithm lie in the following aspects:

- The Safe-Plan algorithm adopts binary split on non-leaf nodes. As shown in the experiments, it may bring more projection operations than necessary and may have unnecessary cartesian-product in the safe plan. Our algorithm adopts multiway split on non-leaf nodes, thus avoids these problems.
- The Safe-Plan algorithm does not perform preprocessing, while the Multiway-Split algorithm adopts preprocessing to project out non-association attributes.

The similarities between the Multiway-Split algorithm and the Safe-Plan algorithm is in the way they process  $\Pi_{\Omega}(q_i)$ .

**Corollary 4** Suppose  $P_1$  is the safe plan generated by the Multiway-Split algorithm and  $P_2$  is the one by the Safe-Plan algorithm, then  $P_1 \Leftrightarrow^* P_2$ .

Proof. For each recursive step of the two algorithms,  $P_1 = \Pi_C^e(q_3 \bowtie^e \Pi_A^e(q_1 \bowtie^e q_2))$  and  $P_2 = \Pi_C^e(q_3 \bowtie^e \Pi_{(A \cup B) \cap \text{Head}(q_1)}^e(q_1) \bowtie^e \Pi_{(A \cup B) \cap \text{Head}(q_2)}^e(q_2))$ , where  $q_1$ ,  $q_2$  and  $q_3$  are queries,  $\text{Head}(q_1) \cup \text{Head}(q_2) = A^+$  and  $B$  contain all the attributes used in the join between  $q_1$  and  $q_2$ . Then  $P_1$  is generated by the Safe-Plan algorithm and  $P_2$  is generated by the Multiway-Split algorithm. By Rule 6,  $P_1 \Leftrightarrow P_2$ . After the algorithms terminates,  $P_1 \Leftrightarrow^* P_2$ . Thus the corollary is proved.

## 6 Experiments

### 6.1 The prototype

We have developed a prototype for probabilistic query evaluation. Our system is implemented as a middleware, which can work on top of any relational database engine, and we have chosen Postgre SQL 8.1 [1] as the underlying DBMS. Using the prototype, we have conducted an extensive set of experiments for comparing the performance of the Multiway-Split algorithm with the Safe-Plan algorithm here. On average, the time for generating a safe plan is about 300ms, which is about the same as the Safe-Plan algorithm.

We used the TPC-H benchmark [10], with a database of 0.1GB as in [11]. We modified all queries by replacing all the predicates in the WHERE clause with uncertain matches. We also remove the top-level aggregations as [11]. Furthermore, the probabilistic relations are generated using the 3-gram distance function [25]. All of the following experiments were carried on the first 10 of the 22 TPC-H queries. Other queries are not very interesting for applying uncertain predicates, since most of them involve complex aggregates. Out of the 10 TPC-H queries, 8 turned out to have safe plans.  $Q_7$  and  $Q_8$  fail to find a safe plan by both the Multiway-Split algorithm and the Safe-Plan algorithm.

## 6.2 Probabilistic query plan example

As mentioned early, probabilistic relations are generated dynamically from regular relations corresponding to approximate queries. A query with uncertain predicates on deterministic relations is transformed into a query over the generated probabilistic relations.

Continuing the query shown in Section 2.3, we further explain the process of its plan generation. The probabilistic relations are generated using the 3-gram distance function [24]. Assume we have the user-defined function (UDF) **DIST** that returns a degree of match for two author names, then the following SQL phrase underlies the generation of the probabilistic relation **Authors**.

```
SELECT a.name, DIST(a.name, 'Jim Gray')
```

```
From Authors a
```

Similarly, a probabilistic relation **Papers** can be generated. The above query with uncertain predicates is rewritten into a new SQL query embedded with all the probability calculations. This rewritten SQL query can be directly executed by the database engine to return the tuples ranked by probability. More exactly, using the proposed algorithm, we obtain the following safe plan for the query:

$$P = \Pi_{authorid,name,title}^e(\Pi_{authorid,name,paperid}^e(\Pi_{name,authorid}^e(Authors^p) \bowtie_{authorid}^e \Pi_{authorid,paperid}^e(Write)) \bowtie_{paperid}^e \Pi_{title,paperid}^e(Papers^p)).$$

This plan corresponds to the following SQL query with probability calculations. Note that  $PROBAGG(A) \equiv POW-ER(10, SUM(LOG(A)))$ .

```
SELECT P1.authorid, P1.name, P4.title,
```

```
1 - PROBAGG(1.0 - P1.prob * P4.prob) AS prob
```

```
FROM
```

```
(SELECT P2.authorid, P2.name, P3.paperid,
```

```

1 - PROBAGG(1.0 - P2.prob * P3.prob) AS prob

FROM

(SELECT name, authorid, 1 - PROBAGG(1.0 -

    DIST(name,'Jim Gray')) AS prob

FROM authors

GROUP BY name, authorid

) AS P2

(SELECT paperid, authorid, 1 AS prob

FROM write

GROUP BY paperid, authorid

) AS P3

WHERE P2.authorid = P3.authorid

GROUP BY P2.name, P3.paperid

)AS P1

(SELECT title, paperid, 1 - PROBAGG(1.0 -

    DIST(title,'database system')) AS prob

FROM papers

GROUP BY title, paperid

) AS P4

WHERE P1.paperid = P4.paperid

GROUP BY P1.name, P4.title

```

From the above example, we find we must perform **aggregation** on the probability and **group by** on the select list attributes when we perform projection in the probabilistic databases. So projection is more expensive in probabilistic databases than in traditional databases.

### 6.3 Performance

We measure the running times for the eight queries that have safe plans. The results are shown in Figure 6a and 6b. The first column is the running time of the SPlan. The second column is the running time of the MSPlan. We call a safe plan without taking into account the computation time for the uncertain predicate a bare query. Figure 6a shows the time for running bare queries, and Figure 6b is for running safe plans. Comparing the two graphs, we find that the query executor spends most of time on computing uncertain predicates for safe plans.

From Figure 6b, we can see that for  $Q1$  and  $Q2$ , their respective MSPlans and SPlans take almost the same amount of time to execute. Since they are both simple projections and any plans for them are safe, there is little room for further optimization.

From Figure 6b, we find the following phenomena. The MSPlan takes one sixth of the execution time of the SPlan for  $Q2$ , one thirtieth for  $Q4$  and one eighth for  $Q9$ . What is common among these three queries is that the SPlans have one more projection operator than the MSPlans. For  $Q2$ , we must create a view  $tmp$ . Its MSPlan is  $\Pi_{Head(Q2)}^e(\Pi_A^e(\Pi_B(supplier \bowtie partsupp \bowtie nation \bowtie tmp) \bowtie^e region^p) \bowtie^e part_1^p)$ , where  $part_1^p = \Pi_{A^2S(part^p)}^e(part^p)$ ,  $A$  and  $B$  denote the corresponding association attribute set respectively. The SPlan is  $\Pi_{Head(Q2)}^e(part_1^p \bowtie^e \Pi_C(\Pi_D(supplier \bowtie partsupp \bowtie nation) \bowtie^e \Pi_E(region^p \bowtie^e tmp)))$ , where  $C$ ,  $D$  and  $E$  denote the corresponding association attribute set respectively. For  $Q4$ , the MSPlan is  $\Pi_{Head(Q4)}^e(orders^p \bowtie^e lineitem_1^p)$ , in which  $lineitem_1^p = \Pi_{A^2S(lineitem^p)}^e(lineitem^p)$ . The SPlan is  $\Pi_{Head(Q4)}^e(orders_1^p \bowtie^e lineitem_1^p)$ , where  $orders_1^p = \Pi_{A^2S(orders^p)}^e(orders^p)$ . For  $Q9$ , the MSPlan is  $\Pi_{Head(Q9)}^e(part^p \bowtie^e \Pi_B(supplier \bowtie lineitem \bowtie partsupp \bowtie$

$orders \bowtie nation$ ). The SPlan is  $\Pi_{Head(Q_9)}^e(\Pi_{A^2S(part^p)}^e(part^p \bowtie \Pi_B(supplier \bowtie lineitem \bowtie partsupp \bowtie orders \bowtie nation)))$ , where  $B$  denotes the corresponding association attribute set. The one more projection operator in the SPlans causes the query executor to execute **aggregation** on the probability and **group by** on the select list attributes one more time. From these three query plans, we find that the cost of projection operator is much higher in probabilistic databases than in conventional databases.

Figure 6b also indicates that for  $Q3$  and  $Q5$ , the MSPlans have substantial performance gain over the SPlans. The execution time for the MSPlan is only about one third of the SPlan for  $Q3$  and one sixth for  $Q5$ . The reason is similar - the SPlans have more projection operators than the MSPlans. The query executor has to spend more time in computing the uncertain predicates and has less opportunity to optimize the plan.

We observe that on  $Q10$ , the performance of the MSPlan is slightly better than the SPlan, as shown in Figure 6b. The MSPlan is  $\Pi_{Head(Q10)}^e(customer \bowtie^e nation \bowtie^e orders^p \bowtie^e lineitem_1^p)$ , in which  $lineitem_1^p = \Pi_{A^2S(lineitem)}^e(lineitem^p)$ ,  $orders_1^p = \Pi_{A^2S(orders^p)}^e(orders^p)$ ,  $nation_1 = \Pi_{A^2S(nation)}^e(nation)$  and  $customer_1 = \Pi_{A^2S(customer)}^e(customer)$ . The SPlan is  $\Pi_{Head(Q10)}^e(customer_1 \bowtie^e \Pi_A(orders_1^p \bowtie^e \Pi_B(lineitem^p \bowtie^e nation))))$ , where  $A$  and  $B$  denote the corresponding association attribute set respectively. Because cartesian-product between deterministic table ( $nation$ ) and probabilistic table ( $lineitem$ ) exists in the SPlan, the performance is unsatisfactory. Thus an alternative SPlan is chosen:  $\Pi_{Head(Q10)}^e(nation_1 \bowtie^e \Pi_C(customer_1 \bowtie^e \Pi_D(orders_1^p \bowtie^e lineitem_1^p)))$ , where  $C$  and  $D$  denote the corresponding association attribute sets. Although for the bare query, the performance of the MSPlan is much worse than the SPlan as shown in Figure 6a, the performance of the MSPlan is slightly better than the SPlan when considering the cost of uncertain predicates.

From the experiments, we learn that the performance is inferior if unnecessary cartesian-products exist in the query plan. Our algorithm can avoid the unnecessary cartesian-product, while the Safe-Plan algorithm can not. The Multiway-Split algorithm can also reduce the number of projects in the query plans. When the number of projections is decreased, the gain is two-fold: First,

it takes less time for the query executor to compute uncertain predicates. Second, the query executor has more opportunities to select a better execution plan.

## 7 Related work

Probabilistic information has been studied for decades under different names: probabilistic, uncertain, approximate, fuzzy, incomplete, imprecise and so on. Most of the previous work is theoretical [4,8,9,15,17,22,23,29], there is increased interest in building systems recently, for example, Trio [34], Orion [30], MystiQ [5], for querying uncertain data [5,30,34], integrating inconsistent data sources [18] and bridging IR and DB fields [19].

One kind of mechanisms used in probabilistic database research is intensional [15,25,27,35], which is based on the possible worlds semantics to deal with symbolic events rather than instant probability. Examples in the area include combined reliability [27], probabilistic relational algebra [15], probabilistic first-order language [35]. In [28], an efficient strategy was developed for query evaluation over probabilistic databases by casting the query processing problem as an inference problem in an appropriately constructed probabilistic graphical model.

Another kind of mechanisms is extensional, which deals with instant probability on data. Techniques in this category are more system oriented [3,7,14,21]. However, they have to be based on certain assumptions for simplifying the implementation, such as limiting the tuples in a relation to represent disjoint events [7], allowing attributes to be inaccurate but requiring tuples to be independent [3], forcing projections to contain keys [14], requiring strategies from users to combine probabilities [21]. As those restrictions are rigid, the systems built around them can hardly handle queries flexibly.

Recently, the research focus in this field has shifted to rewriting query plans to search for one with correct extensional semantics. [11] indicates that this efficient method can avoid the complexity of the intensional method and the

restrictions of the extension method. There also five transformation rules proposed for converting query plans, with equality as well as safety preserved. Our work is based on this technique and further extend and optimize it.

There are much researchers effort launched recently on probabilistic databases. For example, [26] and [31] focus on efficient top-k query evaluation on probabilistic databases. In [32] and [12], Dalvi and Suciu discuss challenges, open problems, and future research directions in probabilistic databases. Bravo and Ramakrishnan [6] present a broad class of aggregate queries, called MPF queries, inspired by the work on probabilistic inference in statistics and machine learning. [2] introduces I-SQL, an analog to SQL for the case of incomplete information. Kimelfeld and Sagiv [20] have studied the problem of maximally joining probabilistic relational data.

## 8 Conclusions and future works

Managing uncertain information using probabilistic databases has become a new trend for supporting a number of emerging technologies. On a probabilistic database, a query can be approximate, and the query result should contain records with correct probabilities. Finding efficient safe plans is vital for probabilistic database query evaluation. We propose and develop a multiway split algorithm for generating safe plans in a probabilistic database. The major advancement is adopting preprocessing and multiway split to avoid unnecessary cartesian-products and reduce projections in the safe plan. Furthermore, we extend existing transformation rules [11] to allow the safe plans generated by the Safe-Plan algorithm and our proposed Multi-way Split algorithm to be transformed between each other.

We limit our discussion to conjunctive queries thus far. We plan to extend these techniques to  $\cup$ ,  $-$ ,  $\gamma$  (union, difference, groupby-aggregate). We will also study the merge of relational model and probability model for a more generalized way towards the integration of DB and IR.

## 9 Acknowledgments

The authors would like to thank Dan Suciu and Nilesch Dalvi for the source code of their safe plan algorithm. We would like to thank Qiming Chen for his comments on earlier drafts of this paper. This work is supported by National Natural Science Foundation of China under Grant No. 60503038.

## References

- [1] Postgre SQL 8.1. <http://www.postgresql.org/>.
- [2] L. Antova, C. Koch, D. Olteanu, From Complete to Incomplete Information and Back, In Proc. SIGMOD 2007, pp.713-724.
- [3] D. Barbara, H. Garcia-Molina, D. Porter. The management of Probabilistic Data, IEEE Trans. Knowl. Data Eng., 1992, 4(5): 487-502.
- [4] R. S. Barga, C. Pu. Accessing imprecise data: An approach based on intervals, IEEE Data Engineering Bulletin, 1993, 16(2): 12-15.
- [5] J. Boulos, N. Dalvi, B. Mandhani and S. Mathur and C. Re and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In Proc. SIGMOD 2005, pp.891-893.
- [6] H. C. Bravo, R. Ramakrishnan. OPTimizing MPF Queries: Decision Support and Probabilistic Inference. In Proc. SIGMOD 2007, pp.701-712.
- [7] R. Cavallo, M. Pittarelli. The theory of probabilistic databases. In Proc. VLDB 1987, pp.71-81.
- [8] R. Cheng, D. V. Kalashnikov, S. Prabhakar. Evaluating probabilistic queries over imprecise data. In Proc. SIGMOD, 2003, pp.551-562.
- [9] D. Chu, A. Deshpande, J. Hellerstein, W. Hong. Approximate Data Collection in Sensor Networks using Probabilistic Models, In Proc. ICDE, 2006, pp.633-644.
- [10] Transaction Processing Performance Council. TPC *BENCHMARK<sup>TM</sup>* H Standard Specification, Revision 2.6.0, url = "http://www.tpc.org/tpch/spec/tpch2.6.0.pdf". 2006.10.

- [11] N. Dalvi, D. Suciu. Efficient Query Evaluation on Probabilistic Database. VLDB J. 2007, 16(4), pp.523-544.
- [12] N. Dalvi, D. Suciu. Management of Probabilistic Data Foundations and Challenges. In Proc. PODS 2007, pp.1-12.
- [13] DBLP. <http://www.informatik.uni-trier.de/~ley/db/>.
- [14] D. Dey, S. Sarkar. A probabilistic relational model and algebra. ACM. Trans. Database System, 1996, 21(3), pp.339-369.
- [15] N. Fuhr, T. Rolleke, A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. ACM Transactions on Information Systems, 1997, 15(1), pp.32-66.
- [16] H. Garcia-Molina and J. D. Ullman and J. Widom. Database System Implementation, Prentice Hall, 2000.
- [17] L. Getoor, N. Friedman, D. Koller, B. Taskar. Learning Probabilistic Models of Relational Structure. In Proc. ICML 2001, pp.170-177.
- [18] Z. G. Ives, N. Khandelwal, A. Kapur, M. Cakir. Orchestra: Rapid, collaborative sharing of dynamic data. In Proc. CIDR 2005, pp.41-46.
- [19] E. Kandogan, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan and H. Zhu. Avatar Semantic Search: A Database Approach to Information Retrieval, In Proc. SIGMOD 2006, pp.790-792.
- [20] B. Kimelfeld, Y. Sagiv. Maximally Joining Probabilistic Data, In Proc. SIGMOD 2007, pp.303-312.
- [21] L. V. S. Lakshmanan, N. Leone, R. Ross and V. S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. ACM TODS, 1997, 22(3) pp.419-469.
- [22] I. Lazaridis, S. Mehrotra. Approximate selection queries over imprecise data. In Proc. ICDE 2004, pp.140-152.
- [23] K. -L. Liu, R. Sunderraman. Indefinite and maybe information in relational databases. ACM Transactions on Databases Systems, 1990, 15(1), pp.1-39.
- [24] G. Navarro. A guided tour approximated string matching. ACM Computing Surveys. 2001, 33(1), pp. 31-38.
- [25] J. Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

- [26] C. Ré, N. Dalvi, D. Suciu. Efficient Top-k Query Evaluation on Probabilistic Database. In Proc. ICDE 2007, pp.886-895.
- [27] F. Sadri. Integrity constraints in the information source tracking method. IEEE Transactions on Knowledge and Data Engineering, 1995, 7(1), pp.106-119.
- [28] P. Sen, A. Deshpande. Representing and Querying Correlated Tuples in Probabilistic Databases. In Proc. ICDE 2007, pp.596-605.
- [29] M. Shapcott, S. McClean, B. Scotney. Aggregation of imprecise and uncertain information in databases. IEEE Trans. on Knowledge and Data Engineering, 2001, 13(6), pp.902-912.
- [30] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, S. Hambrusch. Indexing Uncertain Categorical Data. In Proc. 2007, pp.616-625.
- [31] M. A. Soliman, Ihab F. Ilyas, Kevin C. Chang. Top-k Query Processing in Uncertain Databases. In Proc. ICDE 2007, pp.896-905.
- [32] D. Suciu, N. Dalvi. Foundations of probabilistic answers to queries. SIGMOD 2005, pp.963.
- [33] L. Valiant. The complexity of enumeration and reliability problems. SIAM J. Comput.,1979, pp.410-421.
- [34] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In Proc. CIDR 2005, pp.262-276.
- [35] E. Zimanyi. Query evaluation in probabilistic databases. Theoretical Computer Science, 1997, 171(1-2), pp.179-219.