ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20191123110408
Original scientific paper

Query with Assumptions for Probabilistic Relational Databases

Caicai ZHANG, Zhuolin MEI, Bin WU, Zhiqiang ZHAO, Jing YU, Qingqing WANG

Abstract: Users may have prior knowledge about a probabilistic database. They prefer to query over a probabilistic database on their prior knowledge which cannot be written as component clauses of conventional SQL queries. A naive approach is to query over a new database version, which is generated by transforming the original probabilistic database to satisfy users' prior knowledge; however, it is impractical to generate a different probabilistic database version for each prior knowledge. In this paper, we propose the concept of the query with assumptions which allow users to describe their prior knowledge with a newly introduced ASSUMPTION clause of SQL. We also propose an approach to obtain the result of a query based on assumption clauses. The experimental studies show our approach has better performance compared to the naive approach.

Keywords: probabilistic database; prior knowledge; query with assumptions

1 INTRODUCTION

Probabilistic relational databases play important role in many applications involving large data sets with uncertainties [1-3], for example, data integration [4-6], data publication [7-9], tracking moving objects [10, 11], blockchain technology [12], sensor networks [13-15], stream optimization [16, 17] and so on. The semantic of a probabilistic database is a probability distribution over a set of all possible worlds, which are deterministic database instances [18].

In applications for probabilistic databases, it is common that users would have *prior knowledge* (defined as C) from other sources [19]. Users could not obtain what they really want from a probabilistic database PDB by conventional queries in such situations. The probability inference of a conventional query is a priori probability $P(Q)$, while what users need is a posteriori probability, the conditional probability $P(Q|C)$. The problem is thus how to evaluate a query Q given C.

Let's consider the following example.

Example 1: Assume a probabilistic table $PT1$ in Tab. 1 records information of suspicious persons related to a crime.

Table 1 $PT1$

Rid	Name	Color	Sex	f
$t1$	Jim	yellow	M	$v1$
$t2$	Lily	yellow	F	$v2$
$t3$	Dan	black	M	$v1$
$t4$	Jone	black	F	$v1 \wedge v2$

Table V_P in Tab. 2 records a set of mutually independent Boolean variables, each associated with a probability being true.

Table 2 V_P

V	P
$v1$	0.5
$v2$	0.6

According to their degree of suspicion and correlation, each suspicious person associates a probability computed by the logical expression in column f being true. For example, since the logical expression in f for tuple $t1$ is $v1$ with 0.5 being true, Jim is responsible for the crime with probability 0.5. As the logical expressions of tuple $t1$ and $t3$ are mutually exclusive, Dan and Jim did not participate in the crime together.

Each assignment of variables in V_P represents a possible instance of $PT1$: the instance containing all the tuples whose logical expressions are true with the given assignment. A probabilistic database is a joint probability distribution over the assignments of these variables. So $PT1$ includes four possible worlds (See Tab. 3). $wi(x1, x2)$ donates the i-th possible world with $v1$,$v2$ taking values $x1$, $x2$ separately and $P(wi)$ donates the probability of wi.

Table 3 Four possible worlds for $PT1$

W	$v1$	$v2$	$PT1$	P
$w1$	1	1	$\{t1, t2, t4\}$	0.30
$w2$	1	0	$\{t1\}$	0.20
$w3$	0	1	$\{t2, t3\}$	0.30
$w4$	0	0	$\{t3\}$	0.20

Suppose a detective has an own point of view based on experience or research when checking information of criminals with black hair. For example, the detective suspected Jim participated in the crime, which means the detective has a prior knowledge about $PT1$ (the tuple $t1$ must be present in $PT1$). Consider the following conventional query $Q1$, *select name from* $PT1$ *where color = 'black'*.

$Q1$ evaluated one very possible world of $PT1$ separately according to the semantics of probabilistic query evaluation. The result is another set of possible results instances with the same probability distribution. The final result of the probabilistic query is a union of all the possible result tuples, and the probability of each result tuple is the sum of the probabilities of all results instances that contain it. In this example, Dan and Jone is the result of $Q1$ executing on $PT1$ (See Fig. 1).

Result of $Q1$	
Name	P
Dan	0.5
Jone	0.3

Result of AQ	
Name	P
Jone	0.6

Figure 1 Result of $Q1$ and Result of AQ

However, results obtained by the conventional query $Q1$ actually do not meet the demand of the detective, because the result is obtained based on all the four possible worlds of $PT1$, two of which do not satisfy the detective's

prior knowledge. What's more, the prior knowledge cannot be described in WHERE clause.

Therefore, we propose *query with assumptions* so that users with prior knowledge can describe the assumption in the query, the *query with assumptions AQ* in this example can be written as follows:

select name
*from PT*1 *where color='black'*
assumption exist *name='Jim';*

where the *assumption* is the keyword to describe the user's prior knowledge.

The result of this *query with assumptions* is shown in Fig. 1, namely, the suspect degree of *Jone* is 0.6 and *Dan* did not participate in the crime. Let $P(t \in Q1)$ be the probability of tuple t in the result of $Q1$, C be the assumption about *PT*1 "*exist name='Jim'*", and AQ be $Q1$ executed based on assumption C. Therefore, $P(t \in AQ) = P(t \in Q1|C)$, by applying the Bayesian theory of conditional probability,

$$P(t \in AQ) = P(t \in Q1|C) = P(t \in Q1 \wedge C)/P(C) \quad (1)$$

In this example, only two possible words $w1$, $w2$ satisfy the given assumption, therefore, $Q1$ is only evaluated against these two possible words. Jone is the result from $w1$ denoted as *t_Jone*, and empty is the result from $w2$. Thus, $P(t\_Jone \in Q1 \wedge C) = P(w1) = 0.3$, $P(C) = P(w1) + P(w2) = 0.5$.

By Eq. (1), the probability of *t_Jone* to be in the result of AQ, namely, the degree of Jone to be a criminal is, $P(t\_Jone \in AQ) = P(t\_Jone \in Q1 \wedge C)/P(C) = 0.6$.

The information obtained is based on the probability database and the detective's prior knowledge, which meets the detective's demand.

Example 1 is all above.

Prior knowledge may change a probability distribution of information in a probabilistic relational database, thus, it cannot be processed in a conventional query. What users need is not just the result of a conventional query, but the result under the condition of a prior knowledge. Fig. 2 shows the research problem considered in this paper.

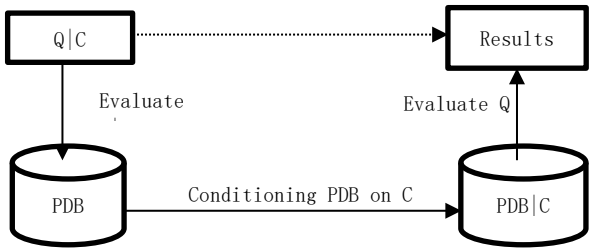

Figure 2 Query Q with the assumption C over the probabilistic database PDB

The *conditioning_based approach* is to execute conventional queries over posteriori probabilistic relational databases. The posteriori probabilistic database is the result of conditioning probabilistic relational databases. However, different users may have different prior knowledge. In *example 1*, the other detectives may have different opinions or assumptions about the criminals while a detective suspected *Jim* must be involved in that crime. It is too heavy cost to generate a new probabilistic database version for each query with different assumptions, and then delete the new database version after the query.

Our aim of this study is to enable users to obtain the result of a query on a priori knowledge, and do not produce a new version of the probabilistic database. In *Example 1*, the *query with assumptions* makes the detective obtain the degree of suspicion of criminals on an assumption, and the other detectives can obtain information based on their different views or assumptions at the same time.

The main contributions of this paper are as follows:

(1) A new ASSUMPTION clause is introduced to SQL syntax. At the end of a conventional SQL query, multiple ASSUMPTION clauses can be added, and in each ASSUMPTION clause, users' prior knowledge can be described as either existence or non-existence of tuples satisfying specifying conditions.
(2) We propose a new *lineage_based* evaluation approach for processing ASSUMPTION clause. The result of *query with assumptions* is obtained based on the result of the conventional query and the conditional probabilities. The conditional probability of the result of the conventional query under the given prior knowledge is calculated for the result of *query with assumptions*. We also provide an improved method for calculating conditional probabilities by incorporating probability calculations into the lineage computation so that shared sub-expressions are not re-evaluated all the time.
(3) We conduct an experimental study of the algorithm presented in this study. Experimental results show that the *lineage_based* approach obtains the correct result for *query with assumptions* and is more efficient than the *conditioning_based* approach.

2 RELATED WORKS

2.1 Probabilistic Relational Database

The studies on uncertain data representation can be divided into two categories [20, 21], one is based on simple correlation assumption [22, 23], which associates existence probabilities with individual tuples. The tuples in the probabilistic relational databases are mutually independent or exclusive while the other can express complex correlations between tuples [24, 25].

Approaches to query evaluation in probabilistic relational databases can also be divided into two categories [26]. One is to evaluate the query and calculate the probability results separately [27, 28]. Lineage expressions of result tuples can be used for correct confidence computation, without restricting the specific query plans. Another approach integrates the probabilistic inference with the query evaluation step [29]. Standard data management techniques can be used to speed up the processing of probabilistic inference. But it is suitable for only the queries that have safety plans. The first approach is more suitable for the evaluation of *query with assumptions*, since the answer tuples of *query with assumptions* are computed based on the conventional query, and the confidences of the result tuples are computed based on the conditional probability of the lineage under a given assumption.

2.2 Conditioning

As the authors claim, [30-32] are the only three existing works on conditioning probabilistic relational databases. Conditioning probabilistic relational databases remove possible worlds that violate the additional knowledge. Our work is different from conditioning probabilistic databases in two aspects.

Firstly, the scenarios are different. Conditioning probabilistic database is mainly useful in scenarios that the administrations add in some new evidence to a database of priori probabilities, and update it to a posteriori probabilistic database taking the evidence into account, so it focuses on how to get the posteriori probabilistic database after conditioning. However, *query with assumptions* is useful in scenarios when different ordinary users query over a probabilistic database with their different prior knowledge or assumptions, and prefer the result taking their assumption into account without affecting the probabilistic relational database. Secondly, the approach of conditioning probabilistic database is not appropriate for solving *query with assumptions*, because it is impractical to generate a posteriori probabilistic database for each query with different assumptions.

3 QUERY WITH ASSUMPTIONS

Definition 1: A *query with assumptions* $AQ_{PDB}(Q, C)$ over a probabilistic database *PDB*: $AQ_{PDB}(Q, C)$ is a query Q over a probabilistic database *PDB* whose possible worlds only include all possible worlds satisfying assumption C, where Q is a conventional query over the probabilistic database, C is an assumption about presence or absence of tuples in the *PDB*.

Supposed $\{RT_{AQ}, P_{AQ}\}$ is the result of *query with assumptions* $AQ_{PDB}(Q, C)$, where RT_{AQ} is a set of tuples in the result, P_{AQ} is the present probability of tuple in RT_{AQ}. Then RT_{AQ} is the set of result tuple of Q executed over all possible worlds of *PDB* satisfying assumption C. Let $W\{w_1, \ldots, w_n\}$ be the set of possible worlds of *PDB*. The set of result tuple $RT_{AQ} = \{t|w_i \in W, C \sim w_i, t \in Q(w_i)\}$, where $C \sim wi$ means possible world wi satisfy assumption C.

For tuple $t \in RT_{AQ}$, $P_{AQ}(t) = P(t \in Q|C) = P(t \in Q \wedge C)/P(C)$, $P(t \in Q \wedge C)$ represents the sum probability of all possible worlds which include t in the result of query Q and satisfy C, $P(C)$ represents the sum probability of all possible worlds satisfying assumption C.

3.1 Syntax

The assumption supported in this paper is limited to that which will not introduce new possible worlds on the basis of the probabilistic database. Since the presence or absence of each tuple in the probabilistic database can determine a possible world, the assumption can be converted into presence or absence of several tuples. So the assumption in the query can be presence or absence of tuples satisfying specifying conditions. Since assumption on the constraint of the number of present tuples is inconvenient for users to convert into presence or absence of several tuples, an interface for count assumption is provided, which will be automatically converted into the presence or absence of tuples. Based on this consideration, the syntax for assumption in the query is defined as follows:

C::=[**assumption**<*C1*>] …[**assumption**<*Cn*>]
<*Ci*>::=**<*c_exist*>** | **<*c_count*>**
<*c_exist*>::=<exp_1>
[**and**|**or**<exp_2>]…[**and**|**or**<exp_m>]
<exp_i>::=[(] <*exp_exist*> | <*exp_not_exist*> [)]
<exp_exist>::=**exist**<*table_name*>.<*attribute*><*compa--rison*>[<*where_clause*>]
<*exp_not_exist*>::=**not**<*exp_exist*>
<*c_count*>::=<*table_name*>**. count** =<*int_count*> [<*where_clause*>]
<*where_clause*>::=**where**<condition>
<*comparison*>::= **>**| **<**| **=** | **>=** | **<=**<value>
<*int_count*>::= natural number

Example 2： In *example 1*, if detective believes the criminals are male.

assumption not exist *PT1. Sex='F'*

Example 3： In *example 1*, if detective believes Jim participated in the crime and Dan did not.

assumption exist *PT1. Name='Jim'* ***and not exist*** *PT1.Name='Dan'*

Example 4： In *example 1*, if detective believes only one criminal is male.

assumption *PT1.COUNT=1* ***where*** *PT1. Sex='M'*

Example 5： In *example 1*, if detective believes only a man participates in the crime.

assumption not exist *PT1. Sex='F'* ***assumption*** *PT1.COUNT=1*

3.2 Transformed Assumption Expression

Assumption clauses need to be transformed to a logical expression of tuple identifiers in the probabilistic database (denoted as C_E), which represents all possible worlds satisfying assumption in *PDB*. The following is the process of transforming assumption clause. Let $C_E(Ci)$ be the transformed expression from each <*Ci*>.

<Ci>::=<c_exist>.

Let $C_E(exp_j)$ be the transformed expression from each <exp_j>.

The tuples related to the assumption can be obtained by the following query,

select RID from <table_name> where <attribute><comparison>[*and <condition>*]

Suppose $R\{t_1, \ldots, t_k\}$ is the set of result tuples,

$$C_E(\exp_j) = \begin{cases} \vee_{1 \le i \le k} t_i & <\exp_j>=<\exp\_exist> \\ \wedge_{1 \le i \le k} \neg t_i & <\exp_j>=<\exp\_not\_exist> \end{cases}$$

C_E(Ci) = $C_E(exp_1)[\wedge|\vee C_E(exp_2)]\ldots[\wedge|\vee C_E(exp_m)]$, where $\wedge$ means **and** in <*c_exist*>, $\vee$ means **or** in <*c_exist*>.

Example 6: Expression C_E of assumption in *example2* is $C_E = \neg t2 \wedge \neg t4$.

Example 7: Expression C_E of assumption in *example3* is $C_E = t1 \wedge \neg t3$.

<Ci>::=<c_count>.

The tuples related to the assumption can be obtained by the following query,

select RID from <table_name> [*where <condition>*]

Suppose $R\{t_1, \ldots, t_k\}$ is the set of result tuples, s = <*int_count*>, $p = k*(k-1)*\ldots*(k+s-1)/s!$, then C_E is a disjunctive normal form which contains x conjunctive clauses as the following:

$C_E(C_i) = \vee_{1\le i\le p} M_i$

where Mi is a conjunctive clause that contains either t_j or ┐t_j for each tuple t_j in R. The number of t_j without negative form in M_i is equal to s. So M_i is defined as the following:

$M_i = \wedge_{1\le j\le k} f(t_j, i)$, $M_i \neq M_j$ (if $i \neq j$),

where $f(t_j, i)$ is the identifier form of tuple t_j in M_i which can be t_j or ┐t_j.

Since in each M_i, the number of identifiers without negative form should be s, let $h(t_j, i)$ be 1 when $f(t_j, i)$ take value of t_j, $h(t_j, i)$ be 0 when $f(t_j, i)$ take value of ┐t_j and the sum of $h(t_j, i)$ for each i be s,

$$h(t_j, i) = \begin{cases} 1 & f(t_j, i) = t_j \\ 0 & f(t_j, i) = \neg t_j \end{cases} \text{ and } \sum_{1\le j\le k} h(t_j, i) = s$$

Example 8: Expression C_E of assumption in *example 4* is $C_E = (t1 \wedge$ ┐$t3) \vee$ (┐$t1 \wedge t3$).

When there are a number of n assumption keywords in the ASSUMPTION clause, each assumption Ci can be transformed separately, then the expression for the assumption clause can be obtained by the conjunction of each $C_E(Ci)$.

$C_E = \vee_{1\le i\le n} C_E(C_i)$

Example 9: Expression C_E of assumption in *example 5* is $C_E = (t1 \wedge$ ┐$t2 \wedge$ ┐$t3 \wedge$ ┐$t4) \vee$ (┐$t1 \wedge$ ┐$t2 \wedge t3 \wedge$ ┐$t4$).

3.3 Process Procedure of a Query with Assumptions

Conditioning_based approach: For a *query with assumptions* $AQ_{PDB}(Q, C)$, the *conditioning_based* approach is to evaluate the conventional query Q over a posteriori probabilistic relational database version. The posteriori probabilistic relational database is generated by conditioning.

But for different assumption clauses, different posteriori probabilistic databases need to be generated and deleted after the query with assumptions, we consider an alternative way to process the query with assumptions.

Since $AQ_{PDB}(Q, C)$ is a conventional query Q executed under specifying assumption C and the result of Q over PDB can be obtained by existing methods [9], we study the correlation between Q and $AQ_{PDB}(Q, C)$ and obtain the result of $AQ_{PDB}(Q, C)$ based on the result of Q and their correlation.

Theorem 1: Given a *query with assumptions* $AQ_{PDB}(Q, C)$, if $\{RT_{AQ}, P_{AQ}\}$ is the result of a *query with assumptions* $AQ_{PDB}(Q, C)$, where RT_{AQ} is the set of tuples in the result, P_{AQ} is the existence probability of tuple in RT_{AQ}; if $\{RT_Q, L\}$ is the result of evaluating Q without the probability inference, which RT_Q is the set of result tuple, L is the lineage of a result tuple, then

(1) $RT_{AQ} \subseteq RT_Q$

(2) $\forall t \in RT_{AQ}$, $P(t \in RT_{AQ}) = P(L(t)|C_E)$

where $L(t)$ is the lineage of tuple t, C_E is a transformed expression for assumption C in the *query with assumptions*.

Proof: Suppose $PDB\{W, P\}$ is a probabilistic database, where $W = \{w1, ..., wn\}$ is a set of possible worlds, n is the number of possible worlds, P is the probability distribution over W. Suppose $PDB'\{W', P'\}$ is the probabilistic database transformed from PDB to satisfy assumption C [11]:

$W' = \{wi | wi \in W, C \sim wi\}$,

$P'(wi) = P(wi)/P(C_E)$,

$P(C_E) = \sum_{wi\in W, C\sim wi} P(wi)$,

$AQ_{PDB}(Q, C)$ is equal to executing the conventional query Q over $PDB'\{W', P'\}$.

$RT_{AQ} = \{t | w_i \in W', t \in Q(w_i)\} = \{t | w_i \in W, C \sim w_i, t \in Q(wi)\}$

$P(t \in RT_{AQ}) = \sum_{wi\in W', t\in Q(wi)} P'(wi)$

$RT_Q = \{t | w_i \in W, t \in Q(wi)\}$

For $\forall t \in RT_{AQ}$, then $t \in RT_Q$, so we have

$$RT_{AQ} \subseteq RT_Q \tag{2}$$

For $\forall t \in RT_Q$, given a specify possible world, if $L(t)$ is true, then t is in the result of Q over this possible world.

Namely, when $L(t) \sim wi$, $t \in Q(wi)$

For $\forall t \in RT_{AQ}$

$P(t \in RT_Q) = P(L(t)) = \sum_{L(t)\sim wi,\ wi\in W} P(wi)$

$P(t \in RT_{AQ}) = \sum_{L(t)\sim wi,\ wi\in W'} P'(wi)$

$P(L(t) \wedge C_E) = \sum_{L(t)\sim wi,\ C\sim wi,\ wi\in W} P(wi)$

$P(L(t) | C_E) = \sum_{L(t)\sim wi,\ C\sim wi,\ wi\in W} P(wi)/P(C_E) = \sum_{L(t)\sim wi,\ wi\in W'} P'(wi)$

So we have

$$\forall t \in RT_{AQ}, P(t \in RT_{AQ}) = P(L(t) | C_E) \tag{3}$$

End proof

By *Theorem 1*, a *query with assumptions* $AQ_{PDB}(Q, C)$ can be processed as follows.

(1) Evaluate the conventional query Q over PDB without probability computation, and obtain the set of result tuple RT_Q and their lineage L by Eq. (*2*).

(2) Compute the conditional probability for each tuple in RT_Q by Eq. (3).

(3) Return the tuples with probability greater than 0.

The algorithm we proposed for *query with assumptions* is shown in Fig. 3.

```
Assumption_Query(AQ_PDB(Q, C))
(RT_Q, L) ← Q(PDB)
C_E ← expression of assumption clause C;
Compute P(C_E)
if P(C_E) > 0 then
    for each rs ∈ RT_Q do
        P(rs) = P(L(rs) | C_E)
        if P(rs) > 0 then
            Add (rs, P(rs)) into (RT_AQ, P_AQ)
        end if
    end for
    return (RT_AQ, P_AQ)
end if
print("assumption error")
return
```

Figure 3 Algorithm of query with assumptions

Example 10: Given a *query with assumptions* as follows over *PT1* in *example 1*,

*select name from PT*1 *where color='black' and Sex='F'**assumption exist** name='Jim' **and not exist** name='Dan';*

First, the following conventional query $Q3$ in this *query with assumptions* will be evaluated,

select name from PT1 where color='black' and Sex='F'

We obtain *Jone*(*t4*) in the result.

The transformed expression for assumption： $C = t1 \wedge \neg\, t3$.

$P(t4 \mid t1 \wedge \neg\, t3) = P(t4 \wedge t1 \wedge \neg\, t3)/P(t1 \wedge \neg\, t3)$

$= P(v1 \wedge v2 \wedge v1 \wedge \neg\, (\neg\, v1))/P(v1 \wedge \neg\, (\neg\, v1))$

$= P(v1 \wedge v2)/P(v1) = 0.6$

3.4 Improved Conditional Probability Calculation

We next give an algorithm for computing $P(L(rs)|C_E)$, the existence probability of a result tuple *rs* of a *query with assumptions*, where $L(t)$ is the lineage of tuple t, C_E is a transformed expression for assumption C in a *query with assumptions.* Both of $L(t)$ and C_E is a logical expression of tuple identifiers in the probabilistic database.

Since the true value of $f(t)$ determines the presence or absence of the tuple t, and the probability of $f(t)$ to be true or false is defined by probabilities associated with the variables of which it is composed, we compute the probability of a logical expression of tuple identifiers by replacing the tuple identifier t with $f(t)$.

The existence probability inference of a result tuple *rs* of a conventional query Q is a priori probability $P(L(rs))$, whereas the existence probability of the result tuple *rs* of a *query with assumptions* $AQ_{PDB}(Q, C)$ is a posteriori probability, the conditional probability $P(L(rs)|C_E)$. Our goal is to make $P(L(rs)|C_E)$ have the same time complexity with $P(L(rs))$.

Definition 2: Expression of Variables EV: Supposed X is a logical expression of tuple identifiers in a *PDB*, $EV(X)$ is to transform X by replacing every tuple identifier t in X with $f(t)$.

Definition 3: Set of Variables SV: Supposed Y is a logical expression of variables in a PDB, $SV(Y)$ is a set of variables that appeared in Y.

Definition 4: Set of Variables in an Expression of tuple identifiers SVT: Supposed X is a logical expression of tuple identifiers in a *PDB*, $SVT(X) = SV(EV(X))$.

$P(L(rs)|C_E)$ can be computed in different ways in different cases.

(1) $VST(L(t)) \cap VST(C_E) = \Phi$,

$P(L(t)|C_E) = P(L(t) \wedge C_E)/P(C_E)$

$P(L(t) \wedge C_E) = P(L(t) = 1, C_E = 1) = P(L(t) = 1) * P(C_E = 1)$

$P(L(t)|C_E) = P(L(t))$

(2) $VST(L(t)) \supseteq VST(C_E)$

Let $V\{e1, e2, \ldots, ex\} = VST(L(t))$, and vj， $j \in [1, 2^x]$ be a joint assignment of all variables in V.

$P(vj) = \prod_{ei \in V} P(ei \sim vj)$, where $ei \sim vj$ represents the assignment of the variable ei in vj.

$P(L(t)|C_E) = P(L(t) \wedge C_E)/P(C_E)$

$P(L(t) \wedge C_E) = \sum_{L(t)=1,\, CE=1, vj \in V} P(vj)$

(3) $VST(L(t)) \cap VST(C_E) \neq \Phi$， $VST(C_E) - VST(L(t)) \neq \Phi$,

Let $IS\{e1, e2, \ldots, ey\} = VST(L(t)) \cap VST(C_E)$,

$P(L(t) \wedge C_E) = P(L(t) = 1, C_E = 1) = \sum_{\{e1, e2, \ldots, ey\}} P(L(t) = 1, C_E = 1 \mid e1, e2, \ldots, ey) * P(e1, e2, \ldots, ey)$

The assignments of $L(t)$ and C_E are conditionally independent given the assignment of $IS\{e1, e2, \ldots, ey\}$, namely,

$P(L(t) = 1, C_E = 1 \mid e1, e2, \ldots, ey) = P(L(t) = 1 \mid e1, e2, \ldots, ey) * P(C_E = 1 \mid e1, e2, \ldots, ey)$

$P(L(t) \wedge C_E) = \sum_{\{e1, e2, \ldots, ey\}} P(L(t) = 1 \mid e1, e2, \ldots, ey) * P(C_E = 1 \mid e1, e2, \ldots, ey) * P(e1, e2, \ldots, ey)$

4 DISCUSSION OF TIME COMPLEXITY OF ALGORITHM FOR QUERY WITH ASSUMPTIONS

For a conventional query Q over a tuple-correlated probabilistic database *PDB*, let $T(Q_{PDB})$ be the time for evaluating the set of result tuples and their lineage, and $T(P_Q)$ be the time for computing the probabilities of tuples.

Given a *query with assumptions* $AQ(Q, C)$ over a probabilistic database *PDB*, We compare the *lineage_based* approach with the *conditioning_based* approach (mentioned in section 3.3). We also compare the time for $AQ(Q, C)$ with Q.

4.1 Conventional Query Q

Let $\{RT, L\}$ be the result of Q without probability inference.

To compute the probability of an arbitrary logical expression being true, the inference method is to enumerate the assignments of involved variables and sum up the probability of assignments that can make the logical expression true, thus the time complexity of probability inference is exponential complexity over the number of involved variables.

For each $t \in RT_Q$, the time complexity of computing the probability of t, $P(L(t))$, is $O(2^{|SVT(L(t))|})$, where $SVT(L(t))$ donates the set of variables appeared in $L(t)$ after replacing each tuple identifier t_i in $L(t)$ with $f(t_i)$.

$T(\mathrm{P_Q}) = \sum_{t \in RT} O(2^{|SVT(L(t))|})$

Therefore, the time for the conventional query is:

$T(Q_{PDB}) + \sum_{t \in RT} \mathrm{O}(2^{|SVT(L(t))|})$

4.2 *AQ*(*Q*, *C*) by our Approach

(1) *Naive probability inference method*

According to the algorithm *Assumption_Query* (mentioned in section 3.3), our approach firstly evaluates the conventional query Q over the probabilistic database *PDB* without probability inference, then computes $P(L(t)|C)$ as the probability of each result tuple t. The first step takes $T(Q_{PDB})$ time as a conventional query does.

$P(L(t)|C) = P(L(t) \wedge C)/P(C)$.

Computing $P(C)$ takes $O(2^{|SVT(C)|})$ time, where $SVT(C)$ donates the set of variables that appeared in C after replacing each tuple identifier t_i in C with $f(t_i)$. $P(C)$ is computed only once, then it can be used in the probability calculation of each result tuple.

A naive inference method for computing $P(L(t) \wedge C)$ is to enumerate the assignments of variables in $SVT(L(t) \wedge C)$. The time complexity of the naive inference method to compute $P(L(t) \wedge C)$ is $O(2^{|SVT(L(t) \wedge C)|})$, where $SVT(L(t) \wedge C)$

donates the set of variables appeared in $L(t) \wedge C$ after replacing each tuple identifier t in $L(t)$ and C with $f(t)$.

Thus, the time for $AQ(Q, C)$ by the naive inference method is:

$T(Q_{PDB}) + \sum_{t \in RT} O(2^{|SVT(L(t) \wedge C)|})$

(2) *Improved probability inference method*

The improved conditional probability calculation algorithm *Probability_Compute* (mentioned in section 3.4) calculates $P(L(t)|C)$ according to the intersection of $SVT(L(t))$ and $SVT(C)$.

If $VST(L(t)) \cap VST(C) = \Phi$, $P(L(t)|C) = P(L(t))$, the time complexity of *Probability_Compute* algorithm to compute $P(L(t)|C)$ is $O(2^{|SVT(L(t))|})$.

If $VST(L(t)) \supseteq VST(C)$, $P(L(t)|C) = P(L(t) \wedge C)/P(C)$. Since $SVT(L(t) \wedge C)$ is equal to $SVT(L(t))$, the time complexity of *Probability_Compute* algorithm to compute $P(L(t)|C)$is $O(2^{|SVT(L(t))|})$.

If $VST(L(t)) \cap VST(C) \neq \Phi$, $VST(C) - VST(L(t)) \neq \Phi$, let $IS = VST(L(t)) \cap VST(C_E)$, the *Probability_Compute* algorithm calculates $P(L(t)|C)$ by enumerating the assignments of IS and then computing $P(L(t))$ and $P(C)$ separately. The time complexity of *Probability_Compute* algorithm to compute $P(L(t)|C_E)$ is:

$O(2^{|IS|}*(2^{|SVT(L(t))|-|IS|}+2^{|SVT(C)|-|IS|})) = O(2^{|SVT(L(t))|} +2^{|SVT(C)|}) = O(2^{\max\{|SVT(L(t))|, |SVT(C)|\}})$

In the worst case, when $|SVT(C)|$ contains all variables in *PDB*, computing $P(L(t)|C_E)$ takes $O(2^{|E|})$ time.

Thus, the time for $AQ(Q, C)$ by the improved inference method is:

$T(Q_{PDB})+ \sum_{t \in RT} T(P(t))$

If $|VST(L(t))| < |VST(C)| <= |E|$ and $VST(L(t)) \cap VST(C) \neq \Phi$, then $T(P(t)) = O(2^{|SVT(C)|})$; otherwise, $T(P(t)) = O(2^{|SVT(L(t))|})$.

4.3 *AQ*(Q, *C*) by the *conditioning_based* Approach

The *conditioning_based* approach needs a preprocessing step to generate a posteriori probabilistic database PDB' and evaluate the conventional query Q over the posteriori probabilistic database. Since the posteriori probabilistic database PDB' has the same set of tuples R as the original database *PDB*, the time of Q over PDB' without probability inference is the same as Q over *PDB*. The preprocessing step takes $O(|R|*2^{|E|})$ time, where $|R|$ is the number of tuples and $|E|$ is the number of variables in the probabilistic relational database.

Let $\{RT, L'\}$ be the result of Q over PDB' without probability inference. Let $T(P'_Q)$ be the time for computing the probabilities of result tuples.

The set of variables in *PDB*' and the logical formula for each tuple may be different from *PDB*. Therefore, $T(P'_Q)$ may be more or less than $T(P_Q)$.

Thus, the time for $AQ(Q, C)$ by the *conditioning_based* approach is:

$O(|R|*2^{|E|})+ T(Q_{PDB})+ T(P'_Q)$

$T(P'_Q) = \sum_{t \in RT} O(2^{|SVT(L'(t))|})$

4.4 Analysis

For a *query with assumptions*, the *lineage_based* approach and the *conditioning_based* approach take the same time in the step of evaluating conventional query without probability inference, while the preprocessing of generating a posteriori probabilistic database in the *conditioning_based* approach takes a huge time cost more than time for probability inference of result tuples in our approach. Furthermore, the *lineage_based* approach avoids generating a new probabilistic database version.

Next, we compare the time for a *query with assumptions* $AQ(Q, C)$ by our approach with the conventional query Q.

The naive probability inference method takes more time for the probability calculation of result tuples than the conventional query does. While the improved probability inference method will not take more time than the conventional query unless there is a result tuple t satisfying the condition $|VST(L(t))| < |VST(C)|$ and $VST(L(t)) \cap VST(C) \neq \Phi$.

5 EXPERIMENTS

In this section, we evaluate the efficiency of the *Lineage_based* approach for answering *queries with assumptions* over probabilistic relational databases.

5.1 Experiment Setup

Probabilistic databases: The data set consists of a variables table V_P and tuple-correlated probabilistic databases. V_P contains a set of mutually independent boolean variables $\{e_1, e_2, \ldots, e_{10}\}$, whose probability distributions are chosen at random. Tuple-correlated probabilistic databases are obtained from relational databases produced by TPC-H 2.14.4, where each tuple t is associated with a logical formula $f(t)$ that is composed of variables in V_P.

5.2 *Lineage_based* Approach, the *Conditioning_based* Approach

The *conditioning_based* approach incurs a cost in terms of generating a posteriori probabilistic database and processing a conventional query. We generate posteriori probabilistic databases by transforming the probabilistic database at 0.01 *TPC-H* scale for the two assumption clauses $C1$, $C2$. The posteriori *PDB* (with 6 variables) for $C1$ includes 4 less variables than the original probabilistic database, while that (with 14 variables) for $C2$ includes 4 more variables than the original probabilistic database.

Table 4 Comparison: Lineage_based approach, conditioning_based approach for AQ

$\|RT\|$	$AQ(Q, C)$	conditioning_based / ms	lineage_based / ms
100	$AQ(Q, C1)$	28681050	35344
	$AQ(Q, C2)$	28795261	35391
1000	$AQ(Q, C1)$	28681274	419266
	$AQ(Q, C2)$	29835872	496328
10000	$AQ(Q, C1)$	28731512	5749466
	$AQ(Q, C2)$	32548765	5749587

Tab. 4 shows that the running time of $AQ(Q, C1)$ and $AQ(Q, C2)$ by the *lineage_based approach* and the *conditioning_based approach* over the probabilistic database at each TPC-H scale.

For a $AQ(Q, C)$, the time cost of the *conditioning_based* approach increases extremely faster than the lineage_based approach when the database scales, since it needs much more time to generate the posteriori *PDB*. When $|RT|$ grows, the time cost of the lineage_approach increases and still much less than the *conditioning_based* approach. Although when $C = C1$, the lineage for result tuples L for probability computing in the *conditioning_based approach* includes less variables than that in the *lineage_based approach*, the time cost during generating the posteriori probabilistic database is much more than all the time for processing $AQ(Q, C)$ by the *lineage_based approach*. It demonstrates that the *conditioning_based* approach costs much more time than the *lineage_based approach* for all $AQ(Q, C)$, no matter when $C = C1$ or $C = C2$. This is because, in the *conditioning_based* approach, the posteriori probabilistic database re-computes the logical formula for each tuple in the PDB in $O(|R|*2^{|E|})$ time complexity, and the total cost is $O(|R|*2^{|E|}) + T(Q_{PDB}) + \sum_{t\in RT}O(2^{|SVT(L'(t))|})$, while the total cost of the *lineage_based* approach is $T(Q_{PDB}) + \sum_{t\in RT}O(2^{|SVT(L(t) \wedge C)|})$. $O(|R|*2^{|E|})$ is much more than $\sum_{t\in RT}O(2^{|SVT(L(t) \wedge C)|})$. Thus, the *conditioning_based* approach always costs much more time than the *lineage_based approach* no matter the posteriori *PDB* contains more or less variables than the original *PDB*.

5.3 Query with Assumptions, Conventional Query

Based on the analysis in *section* 4.4, a *query with assumptions* $AQ_{PDB}(Q, C)$ by *lineage_based approach* has the same time complexity as the conventional query Q on the step of query evaluation without probability inference, donated as $T(Q_{PDB})$. And when the number of variables in assumption $C_E|SVT(C_E)|$ is less than that in the lineage of each tuple $SVT(L)$, probability inference of $AQ_{PDB}(Q, C)$ also has the same time complexity as that of Q, otherwise, probability inference of $AQ_{PDB}(Q, C)$ costs more time than that of Q.

5.3.1 $|SVT(L)| >= |SVT(C_E)|$

(1) $|RT| = 100$, $|SVT(L)| = 8$, $D\_Scale = \{0.01, 0.05, 0.10\}$, $|SVT(C_E)| = \{6,8\}$

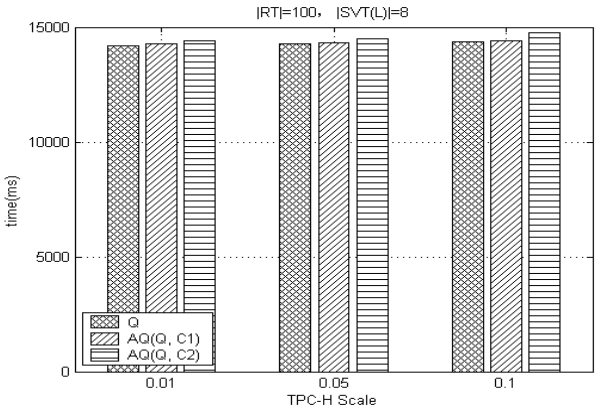

Figure 4 Comparison: *AQ*, *Q*. $|SVT(C1)| < |SVT(C2)| = |SVT(L)|$

Fig. 4 demonstrates the running time of Q, $AQ(Q, C1)$ and $AQ(Q, C2)$ over different scales of probabilistic databases when $|SVT(C1)| = 6$, $|SVT(C2)| = 8$, the number of result tuples of Q $|RT|$ is fixed of 100, and the number of variables in the lineage of each result tuple $|SVT(L)|$ is fixed of 8, where Q represents the conventional query with $|RT|$ is 100 and $|SVT(L)|$ of each result tuple is 8 at each scale of the probabilistic database.

(2) $|RT| = 100$, $|SVT(L)| = 10$, $D\_Scale = \{0.01, 0.05, 0.10\}$, $|SVT(C_E)| = \{6,8,9,10\}$

Fig. 5 demonstrates the running time of Q, $AQ(Q, C1)$, $AQ(Q, C2)$, $AQ(Q, C3)$, $AQ(Q, C4)$ over different scales of probabilistic databases when $|SVT(C3)| = 9$, $|SVT(C4)| = 10$, the number of result tuples of $Q|RT|$ is fixed at 100, and the number of variables in the lineage of each result tuple $|SVT(L)|$ is fixed ta 10, where Q represents the conventional query with $|RT|$ is 100 and $|SVT(L)|$ of each result tuple is 10 at each scale of the probabilistic database.

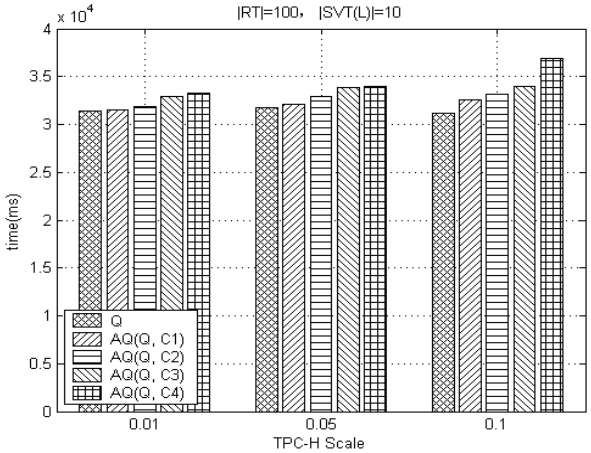

Figure 5 Comparison: *AQ*, *Q*. $|SVT(C1)| < |SVT(C2)| < |SVT(C3)| < |SVT(C4)| = |SVT(L)|$

Fig. 4 and Fig. 5 show that at each scale of probabilistic databases, each query with assumptions $AQ(Q, C)$ does not perform much worse than the conventional query Q. This is because $AQ(Q, C)$ and Q share the same time cost for evaluation of Q, $T(Q_{PDB})$, while the time cost of probability calculation of result for Q is $\sum_{t\in RT}O(2^{|SVT(L)|})$ time complexity and for $AQ(Q, C)$ is $\sum_{t\in RT}O(2^{\max\{|SVT(L)|, |SVT(C)|\}})$. When $|SVT(L)| >= |SVT(C_E)|$, $\max\{|SVT(L)|, |SVT(C)|\} = |SVT(L)|$, thus, $AQ(Q, C)$ and Q have same time complexity.

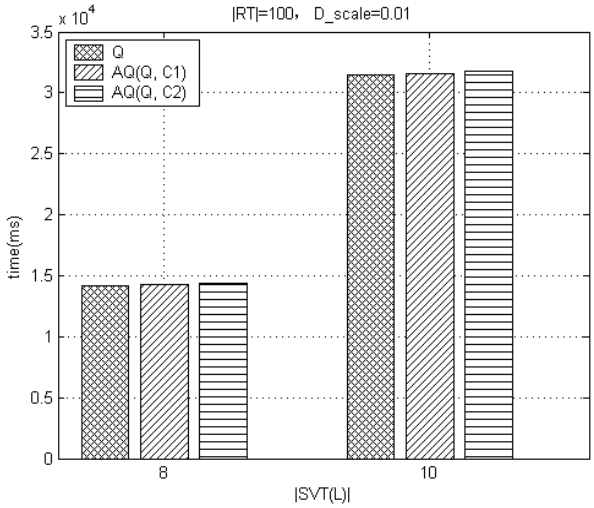

Figure 6 Comparison: Comparison: *AQ*, *Q*

The time costs of queries with the same $|RT|$ hardly increase when the probabilistic database scales up. This is because the time cost of evaluation of query $T(Q_{PDB})$ is much less than that of probability calculation of results of $Q\sum_{t\in RT}O(2^{|SVT(L)|})$ in a probabilistic database. Therefore, if the number of result tuples of queries $|RT|$ and the number of variables in the lineage of result tuples $|SVT(L)|$ remain unchanged, the total time cost of queries does not increase largely as the probabilistic database scales up.

(3) $|RT| = 100$, $|SVT(L)| = \{8,10\}$, $D\_Scale = 0.01$, $|SVT(C_E)| = \{2,8\}$

Fig. 6 demonstrates the running time of Q, $AQ(Q, C1)$, $AQ(Q, C2)$ over the probabilistic database at 0.01 *TPC-H* scale. Fig. 6 shows when the *D_scale* and $|RT|$ is fixed, Queries with $|SVT(L)| = 10$ cost much more time than those with $|SVT(L)| = 8$. This is because when |SVT(L)| increases, the time cost of probability calculation of results grows exponentially. And the time cost of probability calculation accounts for a large proportion of the total time cost of query processing. Therefore, when $|SVT(L)|$ increases, the total time cost of queries grows significantly.

5.3.2 $|SVT(L)| < |SVT(C_E)|$

(1) $|RT| = 100$, $|SVT(L)| = 8$, $D\_Scale = \{0.01, 0.05, 0.10\}$, $|SVT(C_E)| = \{9, 10\}$

Fig. 7 demonstrates the running time of Q, $AQ(Q, C3)$, $AQ(Q, C4)$ over different scales of probabilistic databases when $|SVT(C3)| = 9$, $|SVT(C4)| = 10$, the number of result tuples of Q $|RT|$ is fixed at 100, and the number of variables in lineage of each result tuple $|SVT(L)|$ is fixed at 8, where Q represents the conventional query with $|RT|$ is 100 and $|SVT(L)|$ of each result tuple is 8 at each scale of the probabilistic database.

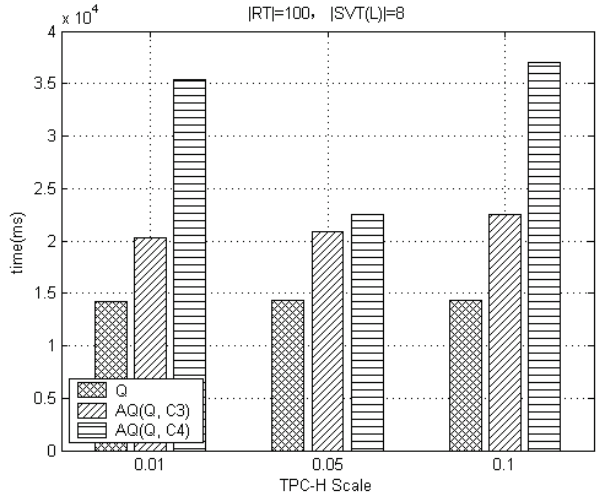

Figure 7 Comparison: Q, AQ. $|SVT(L)| < |SVT(C3)| < |SVT(C4)|$

The cost time of Q, $AQ(Q, C3)$, $AQ(Q, C4)$ with the same $|RT|$ over the same scale of probabilistic database successively increases when |SVT(L)| is fixed at 8. This is because although Q, $AQ(Q, C3)$, $AQ(Q, C4)$ share the same time complexity of evaluation of Q, their time of probability computing for result tuples is $\sum_{t\in RT}O(2^{|SVT(L)|})$, $\sum_{t\in RT}O(2^{|SVT(C3)|})$, $\sum_{t\in RT}O(2^{|SVT(C4)|})$ respectively and $|SVT(L)| < |SVT(C3)| < |SVT(C4)|$. Thus, the total cost time of processing Q, $AQ(Q, C3)$, $AQ(Q, C4)$ successively increases.

6 DISCUSSION

For assumption queries, the *lineage_based* approach is more efficient than the *conditioning_based* approach, even when the generated posteriori probabilistic database includes less variables than the original probabilistic database. Because in the *lineage_based* approach, the time saved by avoiding generating a new probabilistic database version covers the time of probability computation for result tuples. The *lineage_based* approach will not take more time for the query with assumption than the conventional query as long as the number of variables in the transformed expression of the assumption is less than that in the lineage expressions of any result tuples of the convention query.

7 CONCLUSIONS

When users have prior knowledge about a probabilistic database, they cannot obtain data on users' additional knowledge from the probabilistic database by conventional queries. Users' prior knowledge is difficult to be described in the component clauses of a conventional query statement. Therefore, we propose *query with assumptions*, the conventional query based on a given assumption, which makes users able to obtain information from the probabilistic database based on their prior knowledge.

The *conditioning_based* approach generates a posteriori probabilistic database for each *query with assumptions*, which is too resource consuming. Our approach obtains the result of *query with assumptions* from the original probabilistic relational database directly without conditioning.

The experimental results show that our approach has much better performance than the *conditioning_based* approach. A *query with assumptions* by our approach has approximately performance with a conventional query when the transformed expression of assumption does not contain more variables than the lineage of any tuple in the result of the convention query.

The assumption supported in this paper is limited to that which will not introduce new possible worlds based on the probabilistic database. As future work, we plan to consider the assumption which will introduce new possible worlds.

Acknowledgment

This research was supported by the National Natural Science Foundation of China [Nos. 61762055 and 61962029]; the Jiangxi Provincial Natural Science Foundation of China [No. 20181BAB202014]; the Key Scientific and Technological Research Project of Jiangxi Provincial Education Department of China [No. GJJ190899]; the Humanities and Social Sciences Foundation of Colleges and Universities in Jiangxi Province [No.TQ18111]; and the Science and Technology Research Project of Jiangxi Education Department [No. GJJ180904].

8 REFERENCES

[1] Friedman, T. & Broeck, G. V. D. (2019). On Constrained Open-World Probabilistic Databases. In IJCAI 2019. https://doi.org/10.24963/ijcai.2019/793

[2] Leung, C. K., MacKinnon, R. K., & Jiang, F. (2017). Finding efficiencies in frequent pattern mining from big uncertain data. *World Wide Web, 20*(3), 571-594. https://doi.org/10.1007/s11280-016-0411-3

[3] Fuentes-Andino, D., Beven, K., Halldin, S., Xu, C. Y., & Baldassarre, G. D. (2017). Reproducing an extreme flood with uncertain post-event information. *Hydrology and Earth System Sciences, 21*(7), 3597-3618. https://doi.org/10.5194/hess-21-3597-2017

[4] Zhang, W., Zhang, Z., Chao, H. C., & Tseng, F. H. (2018). Kernel mixture model for probability density estimation in Bayesian classifiers. *Data Mining and Knowledge Discovery, 32*(3), 675-707. https://doi.org/10.1007/s10618-018-0550-5

[5] Soares, A., Nunes, R. & Azevedo, L. (2017). Integration of Uncertain Data in Geostatistical Modelling. *Mathematical Geosciences, 49*(2), 253-273. https://doi.org/10.1007/s11004-016-9667-5

[6] Wang, X., Yang T. L., Wang, Y., Ren, L., & Deen, M. J. (2020). ADTT: A Highly-Efficient Distributed Tensor-Train Decomposition Method for IIoT Big Data. *IEEE Transactions on Industrial Informatics, 2020*. https://doi.org/10.1109/TII.2020.2967768

[7] Cui, Z., Wu, Z., Zhou, C., Gao, G., Yu, J., Zhao, Z., & Wu, B. (2016). An efficient subscription index for publication matching in the cloud. *Knowledge-Based Systems, 110*, 110-120. https://doi.org/10.1016/j.knosys.2016.07.017

[8] Wu, Z., Wang, R., Li, Q., Lian, X., Xu, G., Chen, E., & Liu, X. (2020). A Location Privacy-Preserving System Based on Query Range Cover-up for Location-Based Services. *IEEE Transactions on Vehicular Technology*. https://doi.org/10.1109/TVT.2020.2981633

[9] Shi, J., Huang C., He, K., & Shen, X. (2019). ACS-HCA: An Access Control Scheme under Hierarchical Cryptography Architecture. *Chinese Journal of Electronics, 28*, 56-65. https://doi.org/10.1049/cje.2018.10.002

[10] Zhao, Z., Feng, P., Guo, J., Yuan, C., Wang, T., Liu, F., Zhao, Z., Cui, Z., & Wu, B. (2018). A hybrid tracking framework based on kernel correlation filtering and particle filtering. *Neurocomputing, 297*, 40-49. https://doi.org/10.1016/j.neucom.2018.02.043

[11] Robertson, C. & Feick, R. (2018). Inference and analysis across spatial supports in the big data era: Uncertain point observations and geographic contexts. *Transactions in GIS, 22*(2), 455-476. https://doi.org/10.1111/tgis.12321

[12] Ju, S. & Jung., L. (2020). Blockchain Technology and Its Applications: Case Studies. *Journal of System and Management Sciences, 10*(1), 83-93.

[13] Muzammal, M., Gohar, M., Rahman, A. U., Qu, Q., Ahmad, A., & Jeon, G. (2017). Trajectory Mining Using Uncertain Sensor Data. *IEEE Access, 6*, 4895-4903. https://doi.org/10.1109/ACCESS.2017.2778690

[14] Zhang, Z., Zhang, W., Chao, H. C., & Lai, C. F. (2016). Toward belief function-based cooperative sensing for interference resistant industrial wireless sensor networks. *IEEE Transactions on Industrial Informatics, 12*(6), 2115-2126. https://doi.org/10.1109/TII.2016.2558464

[15] Ren, L., Cheng, X., Wang, X., Cui, J. & Zhang, L. (2018). Multi-Scale Dense Gate Recurrent Unit Networks for Bearing Remaining Useful Life Prediction. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2018.12.009

[16] Chi, L., Li, B., Zhu, X., Pan, S., & Chen, L. (2018). Hashing for Adaptive Real-time Graph Stream Classification with Concept Drifts. *IEEE Transactions on Cybernetics, 48*(5), 1591-1604. https://doi.org/10.1109/TCYB.2017.2708979

[17] Peng, H., Deng, C., & Wu, Z. (2019). Best neighbor guided artificial bee colony algorithm for continuous optimization problems. *Soft computing, 23*(18), 8723-8740. https://doi.org/10.1007/s00500-018-3473-6

[18] Liu, H., Zhang, X., & Zhang, X. (2018). Possible world based consistency learning model for clustering and classifying uncertain data. *Neural Networks, 102*, 48-66. https://doi.org/10.1016/j.neunet.2018.02.012

[19] Sung W. K. & Song E. H. (2018). Environmental Information Disclosure in the Hotel Sector: Global Reporting Initiative Application. *Journal of Environmental Assessment Policy & Management, 19*(1), 1750019. https://doi.org/10.1142/S1464333217500193

[20] Li, L., Wang, H., Li, J., & Gao, H. (2018). A survey of uncertain data management. *Frontiers of Computer Science, 14*(1), 162-190. https://doi.org/10.1007/s11704-017-7063-z

[21] Li, Y., Chen, J., & Feng, L. (2013). Dealing with uncertainty: a survey of theories and practices. *IEEE Transactions on Knowledge and Data Engineering, 25*(11), 2463-2482. https://doi.org/10.1109/TKDE.2012.179

[22] Gullo, F., Ponti, G., Tagarelli, A., & Greco, S. (2017). An information-theoretic approach to hierarchical clustering of uncertain data. *Information Sciences, 402*, 199-215. https://doi.org/10.1016/j.ins.2017.03.030

[23] Zhang, Z., Wei, X., Xie, X., Pan, H., & Miao, Y. (2018). An Efficient Optimization Approach for Top-k Queries on Uncertain Data. *International Journal of Cooperative Information Systems, 27*(1), 174-1002. https://doi.org/10.1142/S0218843017410027

[24] Sen, P., Deshpande, A., & Getoor, L. (2009). PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB Journal, 18*(5), 1065-1090. https://doi.org/10.1007/s00778-009-0153-2

[25] Wang, L., Wang, L., & Peng, Z. (2017). Probabilistic object deputy model for uncertain data and lineage management. *Data & Knowledge Engineering, 109*(1), 70-84. https://doi.org/10.1016/j.datak.2017.03.005

[26] Fink, R. & Olteanu, D. (2016). Dichotomies for queries with negation in probabilistic databases. *Acm Transactions on Database Systems, 41*(1), 4-47. https://doi.org/10.1145/2877203

[27] Qin, B. (2015). Efficient queries evaluation on block independent disjoint probabilistic databases. *Proceedings of Database Systems for Advanced Applications.* https://doi.org/10.1007/978-3-319-18123-3_5

[28] Chen, L., Gao, Y., Zhong, A. et al. (2017). Indexing metric uncertain data for range queries and range joins. *The VLDB Journal, 26(*4), 585-610. https://doi.org/10.1007/s00778-017-0465-6

[29] Fink, R., Olteanu, D., & Rath, S. (2011). Providing support for full relational algebra in probabilistic databases. *ICDE, 2011*. https://doi.org/10.1109/ICDE.2011.5767912

[30] Zhu, H., Zhang, C., Cao, Z., & Tang, R. (2016). On efficient conditioning of probabilistic relational databases. *Knowledge-based Systems, 95*(1), 112-126. https://doi.org/10.1016/j.knosys.2015.10.017

[31] Koch, C. & Olteanu, D. (2008). Conditioning probabilistic databases. *PVLDB, 1*(1), 313-325. https://doi.org/10.14778/1453856.1453894

[32] Tang, R. M., Cheng, R., Wu, H. Y., & Bressan, S. (2012). A Framework for Conditioning Uncertain Relational Data. *Database and Expert Systems Applications*. https://doi.org/10.1007/978-3-642-32597-7_7

Contact information:

Caicai ZHANG, PhD
School of Information Science and Technology, Jiujiang University,
No. 551, Qianjin East Road, Jiujiang, Jiangxi 332005, China
E-mail: caicaizhng@gmail.com

Zhuolin MEI, PhD
School of Information Science and Technology, Jiujiang University,
No. 551, Qianjin East Road, Jiujiang, Jiangxi 332005, China
E-mail: meizhuolin@126.com

Bin WU, Lecturer, PhD
School of Information Science and Technology, Jiujiang University,
No. 551, Qianjin East Road, Jiujiang, Jiangxi 332005, China
E-mail: wubincs@gmail.com

Zhiqiang ZHAO, Lecturer, PhD
School of Information Science and Technology, Jiujiang University,
No. 551, Qianjin East Road, Jiujiang, Jiangxi 332005, China
E-mail: zqzhao2000@foxmail.com

Jing YU, PhD candidate
(Corresponding author)
School of Information Science and Technology, Jiujiang University,
No.551, Qianjin East Road, Jiujiang, Jiangxi 332005, China
School of Business Administration, Wonkwang University,
No.460, Iksandae-ro, Iksan, Jeonbuk 54538, Korea
E-mail: yujingellemma@gmail.com

Qingqing WANG, BE
School of Information Science and Technology, Jiujiang University,
No. 551, Qianjin East Road, Jiujiang, Jiangxi 332005, China
E-mail: 1959743535@qq.com