Set Similarity Join on Probabilistic Data

Xiang Lian and Lei Chen
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong, China
{xlian, leichen}@cse.ust.hk

ABSTRACT

Set similarity join has played an important role in many real-world applications such as data cleaning, near duplication detection, data integration, and so on. In these applications, set data often contain noises and are thus uncertain and imprecise. In this paper, we model such probabilistic set data on two uncertainty levels, that is, set and element levels. Based on them, we investigate the problem of *probabilistic set similarity join* (PS^{2}J) over two probabilistic set databases, under the *possible worlds* semantics. To efficiently process the PS^{2}J operator, we first reduce our problem by condensing the possible worlds, and then propose effective pruning techniques, including Jaccard distance pruning, probability upper bound pruning, and aggregate pruning, which can filter out false alarms of probabilistic set pairs, with the help of indexes and our designed synopses. We demonstrate through extensive experiments the PS^{2}J processing performance on both real and synthetic data.

1. INTRODUCTION

Recently, set similarity join has become an increasingly important tool in many real-world applications such as data cleaning [7], near duplication detection [23], data integration [12], and so on. As an example, in the application of detecting near duplicate Web pages [23], each Web page contains a set of tokens (such as words or shingles [6]), and the similarity of any two Web pages can be evaluated by a Jaccard similarity between their corresponding sets of tokens. Large similarity measure indicates a high likelihood that these two pages are duplicates. Similarly, in the application of data integration, based on the set similarity of tokens, similar documents from multiple sources can be also identified and merged.

Formally, given two set databases R and S containing sets of elements and a similarity threshold $\gamma \in (0, 1]$, a *set similarity join* returns all pairs of sets $r \in R$ and $s \in S$ such that $sim(r, s) \geq \gamma$, where $sim(\cdot, \cdot)$ is a function measuring the similarity of two sets.

In the aforementioned applications, owing to reasons such as entry typos, data integration from unreliable sources, or inaccurate information extraction from unstructured documents, the obtained data are often uncertain and imprecise. It is reported by recent statistics that even enterprises typically have approximately 1%-5% erroneous data [20]. Such unreliable data are propagated by copier, truncated, updated, or merged with other data. As a result, the same data entity might have different versions, resulting from different sources. Given another example, in the case of information extraction [13], the city information can be automatically extracted from an unstructured address string "52-A Goregaon West Mumbai 400 076". However, due to different segmentations, the city name can be either "Mumbai" or "West Mumbai", each with certain confidence.

To describe the uncertainty in the set data, in this paper, we model imprecise sets of tokens (e.g., extracted from Web pages or documents) as probabilistic data [11]. We tackle the problem of set similarity join over probabilistic data, namely *probabilistic set similarity join* (PS^{2}J). Specifically, we first formalize the probabilistic set models on two levels, that is, set and element levels, and then define PS^{2}J over two set databases R^P and S^P, retrieving those pairs of probabilistic sets such that they are similar to each other with the probability above a given threshold. Intuitively, this probabilistic threshold guarantees the confidence of join results over probabilistic set data.

Although there are many previous works [7, 3, 23] on manipulating similarity join over precise set data, to the best of our knowledge, no prior work dealt with the set similarity join problem in the context of probabilistic set data. Compared to the join over precise set data, the manipulation of probabilistic data usually considers the *possible worlds* semantics, where each possible world is a materialized instance of probabilistic data that can occur in the real world. The PS^{2}J problem is equivalent to first conducting the join operator in each possible world and then aggregating the join results from all the possible worlds. Since the number of possible worlds is exponentially large with respect to the database size, directly applying the join method on precise set data to our PS^{2}J scenario can be computationally expensive. Therefore, it is important yet challenging to efficiently obtain the PS^{2}J results under the possible worlds semantics.

In order to tackle the efficiency obstacle of performing PS^{2}J, in this paper, we propose effective pruning techniques to filter out false pairs of probabilistic sets, and reduce the PS^{2}J search space. In particular, while previous works on join over precise set data usually conduct linear scan on raw set data or simple signatures, we utilize an M-tree index built upon probabilistic set data to facilitate the PS^{2}J processing. We design a synopsis for summarizing probabilistic set data, which can be seamlessly integrated into the constructed index and help the pruning with the probabilistic threshold. Further, we present efficient procedure of PS^{2}J processing that integrates our proposed pruning techniques.

Specifically, we make the following contributions in this paper.

1. We formally define the models of probabilistic set data, on both set and element levels in Section 2.

Proceedings of the VLDB Endowment, Vol. 3, No. 1

probabilistic set, r_i	set instance, r_{ik}	existence prob., $r_{ik}.p$
r_1	$r_{11} = \{A, B, C\}$	0.8
	$r_{12} = \{B, C, D\}$	0.2
r_2	$r_{21} = \{A, B\}$	0.6
	$r_{12} = \{C, D\}$	0.3

possible world, $pw^{SL}(R^P)$	appearance prob., $Pr\{pw^{SL}(R^P)\}$
$pw_1^{SL}(R^P) = \{r_{11}\}$	$0.8 \times (1 - 0.6 - 0.3) = 0.08$
$pw_2^{SL}(R^P) = \{r_{12}\}$	$0.2 \times (1 - 0.6 - 0.3) = 0.02$
$pw_3^{SL}(R^P) = \{r_{11}, r_{21}\}$	$0.8 \times 0.6 = 0.48$
$pw_4^{SL}(R^P) = \{r_{11}, r_{22}\}$	$0.8 \times 0.3 = 0.24$
$pw_5^{SL}(R^P) = \{r_{12}, r_{21}\}$	$0.2 \times 0.6 = 0.12$
$pw_6^{SL}(R^P) = \{r_{12}, r_{22}\}$	$0.2 \times 0.3 = 0.06$

Table 1: Set-level probabilistic set DB and its possible worlds (6).

2. We propose the problem of *probabilistic set similarity join* (PS^{2}J) over two probabilistic set databases in Section 2, and reduce the complex problem under the possible worlds semantics by condensing the possible worlds in Section 3.

3. We design effective pruning techniques to reduce the search space of PS^{2}J, and propose a synopsis to facilitate the filtering of false alarms in Section 4.

4. We demonstrate through extensive experiments the efficiency of our proposed approaches for PS^{2}J processing in Section 6.

In addition, Section 7 reviews previous works on set similarity join over certain set databases, and probabilistic join processing with different probabilistic data types and measures. Finally, Section 8 concludes this paper.

2. PROBLEM DEFINITION

In this section, we propose probabilistic set data model, and define the problem of *probabilistic set similarity join* (PS^{2}J).

2.1 Probabilistic Set Models

Set-Level Probabilistic Set Database. Set-level probabilistic set database is useful in many real applications, such as integrating near duplicate documents from multiple data sources. In particular, a data source (e.g., a Web site) may contain some near duplicate documents, which correspond to a document entity, and each document can be associated with a probability to indicate its correctness in reality. Thus, in this case, the document entity can be modeled as a set-level probabilistic set consisting of several set instances (near duplicate documents). When we do the integration, we may want to find and merge similar document entities (i.e., matching set-level probabilistic sets) from different sources.

Formally, a set-level probabilistic set database R^P consists of a number of probabilistic sets, denoted as r_i ($1 \leq i \leq a$). Each probabilistic set r_i can be explicitly represented by l_i set instances r_{i1}, r_{i2}, ..., and r_{il_i}. All the set instances r_{ik} (for any $1 \leq k \leq l_i$) of a probabilistic set r_i are *mutually exclusive* (i.e., they cannot appear in the real world at the same time); moreover, each instance r_{ik} is associated with an *existence probability* $r_{ik}.p \in (0, 1]$, where $\sum_{k=1}^{l_i} r_{ik}.p \leq 1$ (note: the inequality indicates the absence of this probabilistic set r_i).

Table 1 depicts an example of a set-level probabilistic set database, which contains two probabilistic sets r_1 and r_2. In particular, r_1 has two set instances $r_{11} = \{A, B, C\}$ and $r_{12} = \{B, C, D\}$, with existence probabilities $r_{11}.p = 0.8$ and $r_{12}.p = 0.2$, respectively, where symbols A, B, C, and D are set elements.

After defining the set-level probabilistic database, we immediately give its *possible worlds* semantics.

DEFINITION 2.1. (*Possible Worlds of Set-Level Probabilistic Set Databases, $pw^{SL}(R^P)$) Given a set-level probabilistic set database R^P containing a number of probabilistic sets $\{r_1, r_2, ..., r_a\}$, a possible world, $pw^{SL}(R^P)$, of R^P is a subset of probabilistic set database R^P, where each probabilistic set r_i contributes either 0 or 1 instance r_{ik}. The appearance probability $Pr\{pw^{SL}(R^P)\}$ of possible world $pw^{SL}(R^P)$ is given by:*

$$Pr\{pw^{SL}(R^P)\} = \prod_{\forall r_{ik} \in pw^{SL}(R^P)} r_{ik}.p \cdot \prod_{\forall r_i \notin pw^{SL}(R^P)} (1 - \sum_{k=1}^{l_i} r_{ik}.p). \quad (1)$$

where r_{ik} is the k-th set instance of probabilistic set r_i.

In Definition 2.1, each probabilistic set r_i in the database R^P has either zero or one (i.e., r_{ik}) set instance appearing in a possible world $pw^{SL}(R^P)$. The appearance probability, $Pr\{pw^{SL}(R^P)\}$, of possible world $pw^{SL}(R^P)$ is given by multiplying probabilities that set instances exist or do not exist in the possible world.

In Table 1, we also show the 6 possible worlds of the previously discussed set-level probabilistic set database.

Element-Level Probabilistic Set Database. Different from the set-level probabilistic set, the element-level probabilistic set has a finer uncertainty level. In the application of information extraction from unstructured sources [13], tokens can be extracted from each sentence of a document. However, due to different segmentation methods, different tokens can be obtained for the same sentence. Thus, we can model the sentence as a probabilistic element, and tokens extracted from this sentence as its instances (each associated with a probability to be correct). As a result, the entire document (element-level probabilistic set) consists of a set of probabilistic elements. With such a model, we can detect near duplicate documents from different data sources, which exactly corresponds to our PS^{2}J problem defined later.

Specifically, an element-level probabilistic set database R^P consists of a number of probabilistic sets, denoted as r_i. Rather than explicitly representing instances of a probabilistic set r_i, each r_i is now expressed by m_i *probabilistic elements* $\{r_i[1], r_i[2], ..., r_i[m_i]\}$[1]. In particular, the k-th probabilistic element $r_i[k]$ can have u_{ik} (mutually exclusive) values $r_i^1[k]$, $r_i^2[k]$, ..., and $r_i^{u_{ik}}[k]$, each $r_i^u[k]$ associated with an existence probability $r_i^u[k].p \in (0, 1]$, where $\sum_{u=1}^{u_{ik}} r_i^u[k].p \leq 1$.

As an example in Table 2, we have an element-level probabilistic set database consisting of two probabilistic sets r_1 and r_2. In particular, the set r_1 has at most two elements, that is, the first element position $r_1[1]$ can be either token A ($= r_1^1[1]$) with existence probability 0.4, or B ($= r_1^2[1]$) with probability 0.6; the second position $r_1[2]$ can be D with probability 0.3, or have no element (with probability 0.7).

Note that, in fact, the element-level probabilistic set can be expanded and explicitly transformed to the set-level one, by enumerating all possible set instances. For example, the probabilistic set r_1 in Table 2 can be expanded to 4 instances $\{A\}$, $\{B\}$, $\{A, D\}$, and $\{B, D\}$. However, in the worst case, such an expansion can incur exponential space cost (w.r.t. the maximum number of elements in set instances), and in turn high computational cost. This is also the reason that we try to find a different solution to our element-level PS^{2}J problem (defined later). Due to the more compressed set representation in the element-level model, we need to process PS^{2}J without materializing all the probabilistic sets, which is thus more complex compared to the set-level one.

Similar to the set-level probabilistic set database, the element-level probabilistic set database can be defined as follows.

DEFINITION 2.2. (*Possible Worlds of Element-Level Probabilistic Set Databases, $pw^{EL}(R^P)$) Given an element-level probabilistic set database R^P, a possible world, $pw^{EL}(R^P)$, of R^P is a*

[1]Note that, although there is no order among probabilistic elements, here we number them only for the ease of illustration.

probabilistic set, r_i	probabilistic element, $(r_i[k], r_i^u[k].p)$
r_1	$r_1[1] = \{(A, 0.4), (B, 0.6)\}$ $r_1[2] = \{(D, 0.3)\}$
r_2	$r_2[1] = \{(A, 1)\}$ $r_2[2] = \{(C, 0.6), (D, 0.4)\}$

possible world, $pw^{EL}(R^P)$	appearance prob., $Pr\{pw^{EL}(R^P)\}$
$pw_1^{EL}(R^P) = \{\{A\}, \{A, C\}\}$	$0.4 \times (1 - 0.3) \times 1 \times 0.6 = 0.168$
$pw_2^{EL}(R^P) = \{\{A\}, \{A, D\}\}$	$0.4 \times (1 - 0.3) \times 1 \times 0.4 = 0.112$
$pw_3^{EL}(R^P) = \{\{B\}, \{A, C\}\}$	$0.6 \times (1 - 0.3) \times 1 \times 0.6 = 0.252$
$pw_4^{EL}(R^P) = \{\{B\}, \{A, D\}\}$	$0.6 \times (1 - 0.3) \times 1 \times 0.4 = 0.168$
$pw_5^{EL}(R^P) = \{\{A, D\}, \{A, C\}\}$	$0.4 \times 0.3 \times 1 \times 0.6 = 0.072$
$pw_6^{EL}(R^P) = \{\{A, D\}, \{A, D\}\}$	$0.4 \times 0.3 \times 1 \times 0.4 = 0.048$
$pw_7^{EL}(R^P) = \{\{B, D\}, \{A, C\}\}$	$0.6 \times 0.3 \times 1 \times 0.6 = 0.108$
$pw_8^{EL}(R^P) = \{\{B, D\}, \{A, D\}\}$	$0.6 \times 0.3 \times 1 \times 0.4 = 0.072$

Table 2: Element-level probabilistic set DB and its 8 possible worlds.

subset of probabilistic set database R^P, where each position $r_i[k]$ of probabilistic set r_i has either none or 1 element value $r_i^u[k]$. The appearance probability $Pr\{pw^{EL}(R^P)\}$ of possible world $pw^{EL}(R^P)$ is given by:

$$Pr\{pw^{EL}(R^P)\} = \prod_{\forall r_i[k] \in pw^{EL}(R^P)} \prod_{\forall r_i^u[k] \in r_i[k]} r_i^u[k].p \cdot \prod_{\forall r_i[k] \notin pw^{EL}(R^P)} (1 - \sum_{u=1}^{u_{ik}} r_i^u[k].p). \quad (2)$$

In Table 2, we present the 8 possible worlds of the element-level probabilistic set database.

2.2 PS^{2}J Definition

Probabilistic Set Similarity Join.

DEFINITION 2.3. (*Probabilistic Set Similarity Join, PS^{2}J*) *Given two probabilistic set databases R^P and S^P, a similarity threshold $\gamma \in (0, 1]$, and a probabilistic threshold $\alpha \in (0, 1]$, a probabilistic set similarity join (PS^{2}J) obtains all the pairs (r_i, s_j) from R^P and S^P with probability greater than or equal to threshold α, that is,*

$$Pr\{sim(r_i, s_j) \geq \gamma\} \geq \alpha, \quad (3)$$

where $sim(\cdot, \cdot)$ is a similarity function to evaluate the degree of similarity between two sets.

Note that, the choice of the similarity function $sim(\cdot, \cdot)$ in Eq. (3) highly depends on the application domain. Examples of such a choice include *Jaccard similarity*, *cosine similarity*, *overlap similarity*, and so on. Nonetheless, as mentioned in [3, 23], the aforementioned 3 measures are inter-related, and can be converted into each other via some variation. Therefore, in this paper, we will focus on one popular set similarity measure, Jaccard similarity:

$$sim(x, y) = J(x, y) = \frac{|x \cap y|}{|x \cup y|}. \quad (4)$$

Probability Computation for Set-Level PS^{2}J. Under the *possible worlds* semantics over the set-level probabilistic set data (in Definition 2.1), $Pr\{sim(r_i, s_j) \geq \gamma\}$ in Eq. (3) can be obtained by:

$$Pr\{sim(r_i, s_j) \geq \gamma\} = \sum_{\forall pw^{SL}(R^P): r_i \in pw^{SL}(R^P)} \left(\sum_{\forall pw^{SL}(S^P): s_j \in pw^{SL}(S^P)} Pr\{pw^{SL}(R^P)\} \cdot Pr\{pw^{SL}(S^P)\} \cdot \left(\sum_{\forall r' \in r_i, s' \in s_j: r' \in pw^{SL}(R^P) \wedge s' \in pw^{SL}(S^P)} \chi(sim(r', s') \geq \gamma) \right) \right) \quad (5)$$

where r' and s' are set instances of r_i and r_j, respectively; and $\chi(z)$ is a function s.t. $\chi(z) = 1$ if z is true; $\chi(z) = 0$, otherwise.

Intuitively, the probability computation of $Pr\{sim(r_i, s_j) \geq \gamma\}$ in Eq. (5) checks all the possible world combinations of R^P and S^P, $pw^{SL}(R^P)$ and $pw^{SL}(S^P)$, and sums up the appearance probabilities of those combinations in which set instances (i.e., r' and s') of r_i and s_j occur and satisfy the condition in χ function.

Probability Computation for Element-Level PS^{2}J. Similarly, according to the possible worlds semantics with the element-level probabilistic set model (as given by Definition 2.2), the probability $Pr\{sim(r_i, s_j) \geq \gamma\}$ in Eq. (3) can be rewritten as:

$$Pr\{sim(r_i, s_j) \geq \gamma\} = \sum_{\forall pw^{EL}(R^P): r_i \in pw^{EL}(R^P)} \sum_{\forall pw^{EL}(S^P): s_j \in pw^{EL}(S^P)} Pr\{pw^{EL}(R^P)\} \cdot Pr\{pw^{EL}(S^P)\} \cdot \chi\left(sim(r', s') \geq \gamma, \text{for } r' = argmax_{\forall r' = \{r'[1], \ldots, r'[l_i]\} \in r_i \wedge r' \in pw^{EL}(R^P)} |r'|, \text{ and } s' = argmax_{\forall s' = \{s'[1], \ldots, s'[l_j]\} \in s_j \wedge s' \in pw^{EL}(S^P)} |s'|\right), \quad (6)$$

where r' and s' are the materialized set instances of r_i and r_j, respectively, converted from element level, and $|x|$ is set x's size.

Similar to the set-level case, Eq. (6) calculates appearance probabilities of possible world combinations, where the materialized element-level set instances (i.e., r' and s') of r_i and s_j appear and satisfy the condition in function χ.

Straightforward Method for Processing PS^{2}J. One straightforward approach to solve the PS^{2}J problem (given by Definition 2.3) on either set or element level is to compute the probability (i.e., $Pr\{sim(r_i, s_j) \geq \gamma\}$) for every pair of probabilistic sets, (r_i, s_j), in a nested loop manner. However, this nested loop method incurs $O(a \cdot b)$ complexity, which is clearly not efficient for PS^{2}J processing in terms of both CPU time and I/O cost, where a and b are the numbers of probabilistic sets in R^P and S^P, respectively. Furthermore, as given in Eqs. (5) and (6), the probability computations on both set and element levels, respectively, have to consider exponential number of possible worlds. Thus, the cost of direct computation by enumerating all possible worlds is very expensive.

Thus, to tackle the efficiency problem of PS^{2}J processing, in the sequel, we will propose to condense the possible worlds, and reduce our PS^{2}J problem to the one on probabilistic data themselves in Section 3. Then, we will provide effective pruning techniques in Section 4, which can filter out false alarms of probabilistic set pairs that violate the PS^{2}J condition in Eq. (3). Further, to enable the pruning, we carefully design synopses for summarizing probabilistic sets on either set or element level, which can be integrated into a tree-based index on probabilistic set data and facilitate the pruning. We will discuss the details of efficient PS^{2}J processing over indexes constructed on probabilistic set databases. We summarize the commonly used symbols in this paper in Table 3, Appendix A.

3. PROBLEM REDUCTION

In the sequel, we aim to condense possible worlds, and simplify formulae of probability computation on both set and element levels.

Reduction of Set-Level Probability Computation. We first give the reduction of our PS^{2}J problem on the set level.

LEMMA 3.1. (*Probability Computation on the Set Level*) *The probability computation of $Pr\{sim(r_i, s_j) \geq \gamma\}$ on the set level in Eq. (5) can be simplified as:*

$$Pr\{sim(r_i, s_j) \geq \gamma\} = \sum_{\forall r' \in r_i} \sum_{\forall s' \in s_j} r'.p \cdot s'.p \cdot \chi(sim(r', s') \geq \gamma) \quad (7)$$

Proof. Please refer to Appendix B. □

Lemma 3.1 reduces probabilistic computation on exponential number of possible worlds in Eq. (5) to the one that only considers instances (r' and s') of probabilistic sets in Eq. (7). The time complexity of computing Eq. (7) is $O(l_i \cdot l_j)$, where l_i and l_j are numbers of set instances in probabilistic sets r_i and s_j, respectively.

Reduction of Element-Level Probability Computation. Next, we consider the PS^{2}J problem reduction on the element level.

LEMMA 3.2. (*Probability Computation on the Element Level*) *The probability computation of* $Pr\{sim(r_i, s_j) \geq \gamma\}$ *on the element level in Eq. (6) can be simplified as:*

$$\begin{aligned} & Pr\{sim(r_i, s_j) \geq \gamma\} \\ = & \sum_{\forall r' \in r_i} \sum_{\forall s' \in s_j} \prod_{\forall r'^u[k] \in r'} r'^u[k].p \cdot \prod_{\forall r'[k] \notin r'} (1 - \sum_{\forall r'^u[k] \in r'[k]} r'^u[k].p) \\ & \cdot \prod_{\forall s'^v[k] \in s'} s'^v[k].p \cdot \prod_{\forall s'[k] \notin s'} (1 - \sum_{\forall s'^v[k] \in s'[k]} s'^v[k].p) \\ & \cdot \chi(sim(r', s') \geq \gamma) \end{aligned} \quad (8)$$

where $r'^u[k]$ *and* $s'^v[k]$ *are the values of the* k*-th element positions* $r'[k]$ *and* $s'[k]$*, respectively.*

Proof. Please refer to Appendix C. □

Lemma 3.2 reduces the problem of computing the probability over possible worlds for element-level PS^{2}J to the one directly on probabilistic set elements, which is similar to the set-level case. However, the time complexity of directly computing the probability in Eq. (8) can still be exponential, that is, $O(\prod_k u_{ik} \cdot \prod_k v_{jk})$, where u_{ik} and v_{jk} are the numbers of possible values for the k-th element position $r_i[k]$ and $s_j[k]$, respectively. Therefore, it is still challenging and computationally expensive to compute all pairs of probabilistic sets from the two databases. Inspired by this, we aim to avoid checking those false alarms of some pairs via pruning techniques on either set or element level. This way, the computational cost of both cases can be greatly reduced.

4. PRUNING TECHNIQUES

4.1 Jaccard Distance Pruning

In this subsection, we present the Jaccard distance pruning method, which utilizes the property of Jaccard similarity measure. Specifically, although Jaccard similarity $J(r_i, s_j)$ itself is not a metric function, the *Jaccard distance*,

$$J\_dist(r_i, s_j) = 1 - J(r_i, s_j),$$

is a metric distance function, which follows the *triangle inequality*. Thus, the basic idea of our Jaccard distance pruning is to use the property of triangle inequality in Jaccard distance to prune those probabilistic set pairs that are definitely dissimilar.

Without loss of generality, for the set-level probabilistic set r_i (or s_j), we can select a *pivot set* piv_{r_i} (piv_{s_j}) that minimizes the summed Jaccard distance to all other set instances of r_i (or s_j), that is, achieving the minimum

$$L(r_i, piv_{r_i}) = \max_{\forall r_{ik} \in r_i} J\_dist(piv_{r_i}, r_{ik})$$

(or $L(s_j, piv_{s_j}) = \max_{\forall s_{jk} \in s_j} J\_dist(piv_{s_i}, s_{jk})$). Similarly, for the element-level probabilistic set r_i (or s_j), we can also select one pivot set piv_{r_i} (piv_{s_j}) with the same criterion, considering different set instances materialized from probabilistic elements.

Then, for any two probabilistic sets r_i and s_j, we have the following pruning lemma.

LEMMA 4.1. (*Jaccard Distance Pruning*) *Given two probabilistic sets* r_i *and* s_j*, and their selected pivot sets* piv_{r_i} *and* piv_{s_j}*, respectively, and a similarity threshold* $\gamma \in (0, 1]$*, if it holds that:*

$$J\_dist(piv_{r_i}, piv_{s_j}) - L(r_i, piv_{r_i}) - L(s_j, piv_{s_j}) > 1 - \gamma, \quad (9)$$

then the probabilistic set pair (r_i, s_j) *can be safely pruned.*

Proof. Please refer to Appendix D. □

4.2 Probability Upper Bound Pruning

The second pruning method we propose is to utilize the probabilistic threshold α (as mentioned in Definition 2.3) to filter out those probabilistic set pairs with confidence below α. Intuitively, if the probability upper bound (denoted as $UB\_P(r_i, s_j)$) of the probability $Pr\{sim(r_i, s_j) \geq \gamma\}$ in Eq. (3) is smaller than α, then the pair (r_i, s_j) can be safely pruned. The following lemma summarizes the probability upper pruning.

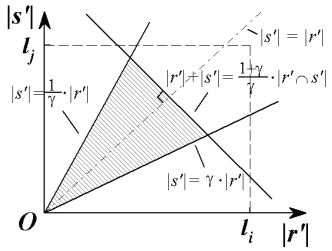


Figure 1: A visualization for the output of $\chi(\cdot)$ function.

LEMMA 4.2. (*Probability Upper Bound Pruning*) *Let* $UB\_P(r_i, s_j)$ *be the probability upper bound of probability* $Pr\{sim(r_i, s_j) \geq \gamma\}$ *given in Eq. (3). Then, given a probabilistic threshold* $\alpha \in (0, 1]$ *specified by PS*^{2}*J, if it holds that:*

$$UB\_P(r_i, s_j) < \alpha, \quad (10)$$

we can safely discard the probabilistic set pair (r_i, s_j)*.*

Proof. Please refer to Appendix E. □

Below, we address the non-trivial and challenging issue on how to obtain probability upper bound, $UB\_P(r_i,s_j)$, in Lemma 4.2.

Derivation of Set-Level Probability Upper Bound. We next aim to derive the set-level probability upper bound from Eq. (7). In particular, due to the equivalent form

$$J(x, y) = \frac{|x \cap y|}{|x| + |y| - |x \cap y|}$$

of Jaccard similarity, we can replace the equivalent condition in function $\chi(\cdot)$ of Eq. (7) as follows.

$$\begin{aligned} & Pr\{sim(r_i, s_j) \geq \gamma\} \quad (11) \\ = & \sum_{\forall r' \in r_i} \sum_{\forall s' \in s_j} r'.p \cdot s'.p \cdot \chi(|r' \cap s'| \geq \frac{\gamma}{1+\gamma} \cdot (|r'| + |s'|)) \end{aligned}$$

Since it holds that $|r'| \geq |r' \cap s'|$ and $|s'| \geq |r' \cap s'|$, from Eq. (11), we have:

$$\begin{aligned} & Pr\{J(r_i, s_j) \geq \gamma\} \quad (12) \\ = & \sum_{\forall r' \in r_i} \sum_{\forall s' \in s_j} r'.p \cdot s'.p \cdot \chi\Big(|r'| \geq \frac{\gamma}{1+\gamma} \cdot (|r'| + |s'|) \\ & \wedge |s'| \geq \frac{\gamma}{1+\gamma} \cdot (|r'| + |s'|) \wedge |r' \cap s'| \geq \frac{\gamma}{1+\gamma} \cdot (|r'| + |s'|)\Big) \\ = & \sum_{\forall r' \in r_i} \sum_{\forall s' \in s_j} r'.p \cdot s'.p \cdot \chi\Big(\gamma \cdot |s'| \leq |r'| \leq \frac{1}{\gamma} \cdot |s'| \\ & \wedge |r' \cap s'| \geq \frac{\gamma}{1+\gamma} \cdot (|r'| + |s'|)\Big). \end{aligned}$$

Note that, the first term in the $\chi(\cdot)$ function on the RHS of Eq. (12) is a necessary condition of the second term (i.e., the second term subsumes the first one).

Based on Eq. (12), we can visualize the cases where $\chi(\cdot)$ function may output 1 during the probability calculation. As illustrated in Figure 1, we consider a 2D space, where the horizontal axis corresponds to the size, $|r'|$, of a set instance $r' \in r_i$, and the vertical axis is the size, $|s'|$, of a set instance $s' \in s_j$, where $|r'| \leq l_i$ and $|s'| \leq l_j$. In this 2D space, we draw 3 lines, 1) $|s'| = \gamma \cdot |r'|$; 2) $|s'| = \frac{1}{\gamma} \cdot |r'|$; and 3) $|r'| + |s'| = \frac{1+\gamma}{\gamma} \cdot |r' \cap s'|$, where the first two lines correspond to the first term in the $\chi(\cdot)$ function of Eq. (12) (when taking the equalities), and the third line correspond to the second term (taking the equality). These three lines form a shaded region (as shown in Figure 1), exactly indicating the case where the $\chi(\cdot)$ function may output 1 (other white region corresponds to output of 0).

Therefore, let $max\_size(|r_i \cap s_j|)$ be the maximum possible size of the set intersection $(r' \cap s')$, for any $r' \in r_i$ and $s' \in s_j$. We can obtain an upper bound of the probability in Eq. (12) via $max\_size(|r_i \cap s_j|)$ as follows:

$$
\begin{aligned}
& Pr\{J(r_i, s_j) \geq \gamma\} \qquad (13)\\
\leq & \sum_{\forall r' \in r_i} \sum_{\forall s' \in s_j} r'.p \cdot s'.p \cdot \chi\left(\gamma \cdot |s'| \leq |r'| \leq \frac{1}{\gamma} \cdot |s'| \right.\\
& \left. \wedge max\_size(|r_i \cap s_j|) \geq \frac{\gamma}{1+\gamma} \cdot (|r'| + |s'|)\right).\\
= & \; UB\_P(r_i, s_j)
\end{aligned}
$$

In order to further simplify $UB\_P(r_i, s_j)$ in Eq. (13), without loss of generality, we assume that set instances r_{ik} (or s_{jk}) of r_i (s_j) have their sizes in non-descending order, that is, $|r_{i1}| \leq |r_{i2}| \leq ... \leq |r_{il_i}|$ (or $|s_{j1}| \leq |s_{j2}| \leq ... \leq |s_{jl_j}|$). Correspondingly, we denote their *cumulative probability vector* as CPV_{r_i} (or CPV_{s_j}), where $CPV_{r_i}[w]$ indicates the existence probability that r_i has sizes of instance sets smaller than or equal to w, that is, $CPV_{r_i}[w] = \sum_{\forall k, |r_{ik}| \leq w} r_{ik}.p$.

We have the following lemma to derive probability upper bound used for pruning over set-level probabilistic sets (in Lemma 4.2).

LEMMA 4.3. (*Derivation of Set-Level Probability Upper Bound Pruning*) *Let* $min\_len(|r_{ik}|) = \gamma \cdot |r_{ik}|$, *and* $max\_len(|r_{ik}|) = \min\{l_j, \frac{1}{\gamma} \cdot |r_{ik}|, \frac{1+\gamma}{\gamma} \cdot max\_size(r_i \cap s_j) - |r_{ik}|\}$. *Then, we have:*

$$
\begin{aligned}
& UB\_P(r_i, s_j) \qquad (14)\\
= & \sum_{k=1}^{l_i} r_{ik}.p \cdot \begin{cases} CPV_{s_j}[\lceil max\_len(|r_{ik}|)\rceil] - CPV_{s_j}[\lfloor min\_len(|r_{ik}|)\rfloor] \\ \qquad\qquad \textit{if } min\_len(|r_{ik}|) \leq max\_len(|r_{ik}|); \\ 0 \qquad\qquad \textit{otherwise.} \end{cases}
\end{aligned}
$$

Proof. Please refer to Appendix F. □

Derivation of Element-Level Probability Upper Bound. With the element-level probabilistic set model, the basic idea of deriving the element-level probability upper bound is the same to that of the set-level one (as discussed above in Lemma 4.3). However, there are two obstacles to tackle, which are the differences from the set-level computation. In brief, due to the element-level uncertainty, we need to compute the probability that the materialized set instances of a probabilistic set have sizes 1) equal to or 2) smaller than an integer, which have their counterparts, $r_{ik}.p$ and $CPV_{s_j}[\cdot]$, respectively, in Eq. (14).

Let $F(r_i, N, n)$ be the probability that, among N element positions we have seen so far, there are exactly n elements appearing in set instances. Thus, we can recursively compute $F(r_i, N, n)$ by:

$$
\begin{aligned}
F(r_i, N, n) &= \sum_{\forall u} r_i^u[N].p \cdot F(r_i, N-1, n-1)\\
&\quad + (1 - \sum_{\forall u} r_i^u[N].p) \cdot F(r_i, N-1, n)\\
F(r_i, N, 0) &= \prod_{k=1}^{N} (1 - \sum_{\forall u} r_i^u[k].p)\\
F(r_i, n, n) &= \prod_{k=1}^{n} \sum_{\forall u} r_i^u[k].p
\end{aligned}
$$

Therefore, to compute the probability that instance has size w, $Pr\{|r'| = w\}$, we simply let it be $F(r_i, l_i, w)$. Thus, correspondingly, the cumulative probability $CPV_{r_i}[w] = Pr\{|r'| \leq w\}$ (for $1 \leq w \leq l_i$) can be easily obtained.

We have the lemma below to derive probability upper bound used for pruning on element-level probabilistic sets (in Lemma 4.2).

LEMMA 4.4. (*Derivation of Element-Level Probability Upper Bound Pruning*) *Let* $min\_len(w) = \gamma \cdot w$, *and* $max\_len(w) = \min\{l_j, \frac{1}{\gamma} \cdot w, \frac{1+\gamma}{\gamma} \cdot max\_size(r_i \cap s_j) - w\}$. *Then, we have:*

$$
\begin{aligned}
UB\_P(r_i, s_j) &= \sum_{w=1}^{l_i} F(r_i, l_i, w) \qquad (15)\\
&\quad \cdot \begin{cases} CPV_{s_j}[\lceil max\_len(w)\rceil] - CPV_{s_j}[\lfloor min\_len(w)\rfloor] \\ \qquad\qquad \textit{if } min\_len(w) \leq max\_len(w); \\ 0 \qquad\qquad \textit{otherwise.} \end{cases}
\end{aligned}
$$

Proof. Please refer to Appendix G. □

5. PS²J PROCESSING APPROACH

5.1 Synopsis Design

Index. As mentioned in Section 4.1, the Jaccard distance is a metric measure that follows the triangle inequality. Thus, we can utilize this property to index the probabilistic set database via any metric-space index. In this paper, we adopt one popular metric index, M-tree [10]. Nonetheless, since our proposed methodology does not rely on the choice of metric index, our pruning techniques can be easily applied to other indexes in the metric space. Specifically, in the M-tree, for each probabilistic set r_i, we select a pivot set piv_{r_i}, associated with the maximum deviation $L(piv_{r_i}, r_i)$ from any instance of r_i to pivot. Then, the probabilistic sets are recursively grouped (via standard criteria for metric-space M-tree construction) until one final node (root) is obtained.

Synopses. Next, we focus on the synopsis design for probabilistic sets to facilitate the index pruning. In particular, within each intermediate node e of the M-tree index, we store a synopsis, $Syn(e)$, to describe the information for probabilistic sets rooted from this node. Each synopsis $Syn(e)$ consists of max size probability vectors SPV^e_{max}, min/max cumulative probability vectors CPV^e_{min} and CPV^e_{max}, max element probability vectors EPV^e_{max}, max element count vectors ECV^e_{max}, and max set sizes $Size^e_{max}$.

Specifically, the w-th position of SPV^e_{max} stores the maximum probabilities, that probabilistic sets under e have instances of exactly size w; CPV^e_{min} and CPV^e_{max} are the min/max cumulative probability vectors w.r.t. $SPV_.$. Moreover, each position in EPV^e_{max} (or ECV^e_{max}) corresponds a unique element value, and stores min/max existence probability (or count) for this element in probabilistic sets under node e. Finally, $Size^e_{max}$ is the maximum size of probabilistic set instances under node e.

5.2 Node Level Pruning

Similar to the data-level pruning in Lemmas 4.1 and 4.2, we give the pruning below on the node level via Jaccard distance pruning and probability upper bound pruning, respectively.

LEMMA 5.1. (*Node-Level Jaccard Distance Pruning*) *Given two nodes* e_1 *and* e_2, *as well as their pivots* piv_{e_1} *and* piv_{e_2}, *respectively, and a similarity threshold* $\gamma \in (0, 1]$, *if it holds that:*

$$
J\_dist(piv_{e_1}, piv_{e_2}) - L(e_1, piv_{e_1}) - L(e_2, piv_{e_2}) > 1 - \gamma, \qquad (16)
$$

then the node pair (e_1, e_2) *can be safely pruned.*

LEMMA 5.2. (*Node-Level Probability Upper Bound Pruning*) *Let* $UB\_P(e_1, e_2)$ *be the probability upper bound of probability* $Pr\{sim(e_1, e_2) \geq \gamma\}$ *given in Eq. (3). Then, given a probabilistic threshold* $\alpha \in (0, 1]$ *specified by PS²J, if it holds that:*

$$
UB\_P(e_1, e_2) < \alpha, \qquad (17)
$$

we can safely discard the probabilistic set pair (e_1, e_2), *where on either set or element level, we have:*

$$
\begin{aligned}
UB\_P(e_1, e_2) &= \sum_{w=1}^{Size^{e_1}_{max}} SPV^{e_1}_{max}[w] \qquad (18)\\
&\quad \cdot \begin{cases} CPV_{s_j, max}[\lceil max\_len(w)\rceil] - CPV_{s_j, min}[\lfloor min\_len(w)\rfloor] \\ \qquad\qquad \textit{if } min\_len(w) \leq max\_len(w); \\ 0 \qquad\qquad \textit{otherwise.} \end{cases}
\end{aligned}
$$

Furthermore, $max\_size(e_1, e_2)$ *used for computing* $max\_len(\cdot)$ *in Eq. (18) is the upper bound size of intersection* $r_i \cap s_j$ *for any* $r_i \in e_1$ *and* $s_j \in e_2$. *We let* $max\_size(e_1, e_2) = \min\{Size^{e_1}_{max}, Size^{e_2}_{max}, \sum_{\forall w} \min\{ECV^{e_1}_{max}[w], ECV^{e_2}_{max}[w]\}\}$.

Node-Level Aggregate Pruning. We notice that in Lemma 5.2, we always uses the maximum size $max\_size(e_1, e_2)$ to compute the probability upper bound, which might have lower pruning ability for higher level tree nodes (as they contain more probabilistic

sets). Therefore, in order to enhance the pruning power, we additionally propose another probability upper bound $UB\_P(e_1, e_2)$ by exploring the probability aggregates stored in the synopses.

Since $UB\_P(e_1, e_2)$ is the maximum probability that any two probabilistic sets $r_i \in e_1$ and $s_j \in e_2$ are similar, our basic idea is to compute an upper bound probability that the intersection between r_i and s_j has size w ($1 \leq w \leq max\_size(e_1, e_2)$).

Specifically, according to vectors $EPV^{e_1}_{max}$, $EPV^{e_2}_{max}$, $ECV^{e_1}_{max}$, and $ECV^{e_2}_{max}$, we can identify those elements that may have intersection between sets from e_1 and e_2 (i.e., both positions in $ECV^{e_1}_{max}$ and $ECV^{e_2}_{max}$ have nonzero counts). Without loss of generality, we denote them as $elem_1$, ..., $elem_n$, in non-increasing order of their (multiplied) corresponding probabilities (denoted as $elem_i.p$) in $EPV^{e_1}_{max}$ and $EPV^{e_2}_{max}$. Then, the upper bound probability that any intersection has size $w \in [1, max\_size(e_1, e_2)]$, can be given by $\prod_{i=1}^{w} elem_i.p$. Therefore, we can obtain another probability upper bound via aggregates, that is,

$$UB\_P(e_1, e_2) = \sum_{w=1}^{max\_size(e_1,e_2)} \prod_{i=1}^{w} elem_i.p. \quad (19)$$

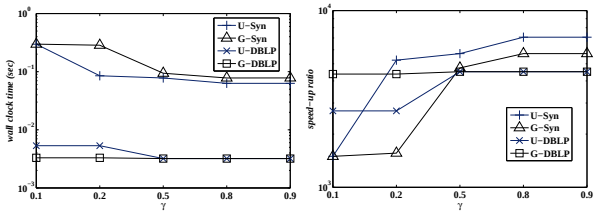
5.3 PS^{2}J Procedure

Our PS^{2}J processing procedure traverses the two M-trees constructed on two probabilistic set databases in parallel. For any pair of two nodes or probabilistic sets that we encounter, we will apply our aforementioned Jaccard distance pruning, aggregate pruning, or probability upper bound pruning to filter out false alarms. If a node pair cannot be pruned, we will further check their child nodes; if an object pair cannot be pruned, we will add this pair to a candidate set, PS^2J_cand. Finally, we refine candidate pairs in PS^2J_cand and return the actual PS^{2}J answers. The pseudo code of PS^{2}J processing and its detailed descriptions can be found in Appendix H.

6. EXPERIMENTAL STUDY

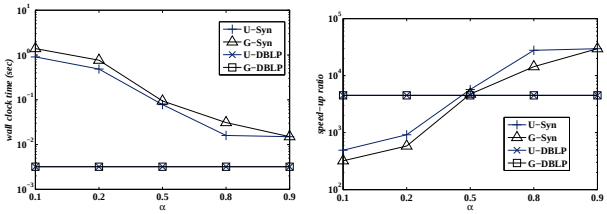
In this section, we evaluate the efficiency and effectiveness of our proposed PS^{2}J processing approaches on both set and element levels over real and synthetic data sets. Synthetic data sets include $U\text{-}Syn$ and $G\text{-}Syn$ whose set elements are within [1, 100], following *Uniform* and *Gaussian distribution* (with the mean 50 and variance 20), respectively. For the set-level model, $[\lambda_{min}, \lambda_{max}]$ is the range of the number of set instances per probabilistic set, and $[\sigma_{min}, \sigma_{max}]$ is the range of the number of elements in each set instance. For the element-level model, $[u_{min}, u_{max}]$ is the range of the number of instances for each probabilistic elements, and θ is the percentage of element positions in a set that are probabilistic. We also test real data set, $DBLP$, which contains around $20K$ titles of papers extracted from DBLP [2]. We parse the tokens in titles and generate probabilistic set instances/elements following *Uniform* or *Gaussian distribution*, resulting in two data sets $U\text{-}DBLP$ and $G\text{-}DBLP$, respectively. We index the above mentioned probabilistic sets with M-trees[3] [10], where the page size is $4K$. The detailed descriptions of data sets can be found in Appendix I.

Evaluation measures. To report the performance of PS^{2}J processing, in the sequel, we test two measures the *wall clock time* and *speed-up ratio*. In particular, the wall clock time is the total time cost that executes the PS^{2}J procedure in Figure 11, including both filtering and refinement cost. Moreover, to our best knowledge, no prior work has studied the set similarity join problem in probabilistic set databases. Thus, the only available method is the *nested loop join* (denoted as NLJ) as mentioned in Section 2.2. That is, for each probabilistic set $r_i \in R^P$, we access those sets $s_j \in S^P$ that have common elements with r_i (via inverted index). The speed-up ratio is defined as the wall clock time of NLJ divided by that of PS^2J. In particular, for NLJ over set-level probabilistic sets, we apply the state-of-the-art approach, $ppjoin^+$ [23], to filter out false alarms of pairs of (certain) set instances in probabilistic sets. For element-level probabilistic sets, however, since $ppjoin^+$ (and other works like [7, 3] as well) requires sorting tokens in the (certain) set according to a global ordering, such a sorting cannot be achieved in our problem with condensed probabilistic elements (except for materializing all possible set instances, which is however not space-efficient). Thus, for element-level probabilistic sets, we will online materialize them, and directly check the predicate in Definition 2.3. Since the total time cost of NLJ is rather high (especially for the element level), we take 100 random sample sets from R^P, join them with S^P, obtaining the total joining time, J_time, and estimate the wall clock time of NLJ by $(J\_time \cdot |R^P|/100)$, where $|R^P|$ is the number of uncertain sets in R^P. All our subsequent experiments are conducted on a PC with Core(TM)2 Duo 3GHz CPU with 3G memory.



(a) wall clock time (b) speed-up ratio

Figure 2: Set-level PS^{2}J performance vs. γ.



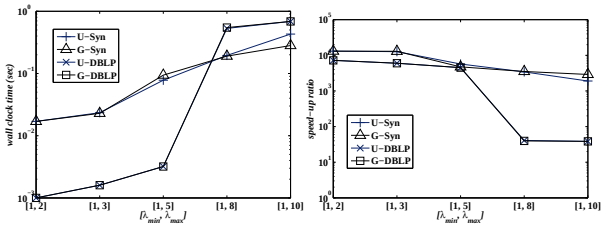
(a) wall clock time (b) speed-up ratio

Figure 3: Set-level PS^{2}J performance vs. α.

6.1 PS^{2}J over Set-Level Probabilistic Data

In this subsection, we present the experimental results of PS^{2}J on set-level probabilistic data. Each time we vary the value of one parameter, while setting others to their default values (i.e., $\gamma = 0.5$, $\alpha = 0.5$, $[\lambda_{min}, \lambda_{max}] = [1, 5]$, $[\sigma_{min}, \sigma_{max}] = [1, 5]$, and $N = 50K$). Detailed settings can be found in Appendix J.

PS^{2}J performance vs. similarity threshold γ. Figure 2 illustrates the PS^{2}J processing performance over 4 real and synthetic data, $U\text{-}Syn$, $G\text{-}Syn$, $U\text{-}DBLP$, and $G\text{-}DBLP$. From figures, when the similarity threshold γ increases (other parameters are set to their

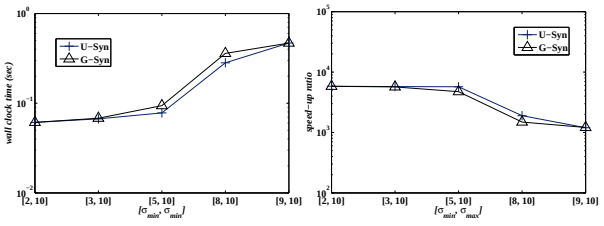


(a) wall clock time (b) speed-up ratio

Figure 4: Set-level PS^{2}J performance vs. $[\lambda_{min}, \lambda_{max}]$.

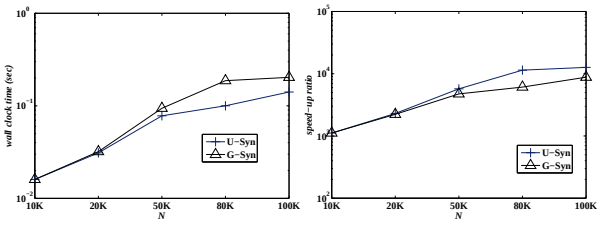
[2] *http://dblp.uni-trier.de/xml/.*

[3] Source code is available at *http://www-db.deis.unibo.it/Mtree/.*



(a) wall clock time (b) speed-up ratio

Figure 5: Set-level PS^{2}J performance vs. $[\sigma_{min}, \sigma_{max}]$.



(a) wall clock time (b) speed-up ratio

Figure 6: Set-level PS^{2}J performance vs. N.

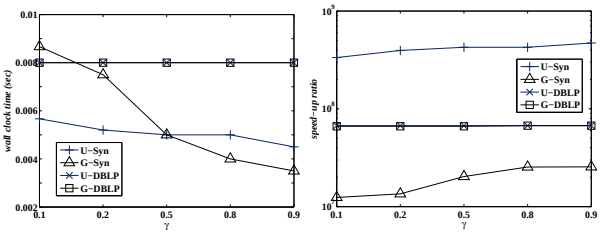
default values), the wall clock time for all the 4 data sets decreases. This is because with large γ value, the condition of Jaccard distance pruning in Eq. (9) can filter out more false alarms, and probability upper bound pruning in Eq. (13) can achieve tighter (smaller) upper bound. As a result, fewer candidates need to be retrieved and refined, which incurs lower time cost. Note that, due to different data sizes between U-(G-)$DBLP$ and U-(G-)Syn, in this and subsequent experiments, the trends of curves for $DBLP$ data are always more smooth than that for Syn. Figure 7(b) shows the speed-up ratio of our PS^2J approach, compared with NLJ via $ppjoin^+$ filtering. PS^2J performs better than NLJ by about 3-4 orders of magnitude, which indicates good performance of our approach.

PS^{2}J performance vs. probabilistic threshold α. Figure 3 varies the probabilistic threshold α from 0.1 to 0.9, where other parameters are set to default values. Similar to previous results, when α increases, the wall clock time of PS^2J decreases. This is because for large α, the probability upper bound pruning can filter out more candidate pairs, and thus the cost of search/refinment decreases. PS^2J outperforms NLJ by about 2-4 orders of magnitude.

PS^{2}J performance vs. range of the instance number in a probabilistic set $[\lambda_{min}, \lambda_{max}]$. Figure 4 evaluates the effect of the number of instances in a probabilistic set (i.e., λ) on PS^{2}J performance, where the range $[\lambda_{min}, \lambda_{max}]$ of λ varies from $[1, 2]$ to $[1, 10]$ and other parameters are set to default values. In figures, the wall clock time increases with wider λ range (or higher λ value on average). This is because more set instances in probabilistic sets would incur higher retrieval and refinement costs. Moreover, similar to previous results, PS^2J has better performance than NLJ. To further evaluate the robustness of our approach, we also evaluate the effects of other parameters on synthetic data U-Syn and G-Syn below.

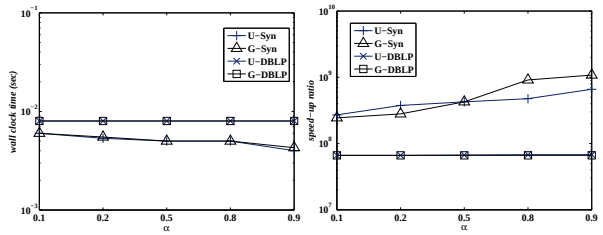
PS^{2}J performance vs. range of the number of elements in set instances $[\sigma_{min}, \sigma_{max}]$. Figure 5 varies the range $[\sigma_{min}, \sigma_{max}]$ of set instance size from $[2, 10]$ to $[9, 10]$, where other parameters are set to default values. For wider range of (or larger) set sizes, more wall clock time is needed for retrieval and refined, which is confirmed in Figure 5(a). Furthermore, PS^2J performs better than NLJ by about 3-4 orders of magnitude.

PS^{2}J performance vs. data size N. Figure 6 tests the scalability of our PS^2J approach by varying the total number of probabilistic sets in each database (i.e., N) from $10K$ to $100K$, where other parameters are set to default values. From figures, we can see that when N becomes larger, the wall clock time of PS^2J also



(a) wall clock time (b) speed-up ratio

Figure 7: Element-level PS^{2}J performance vs. γ.



(a) wall clock time (b) speed-up ratio

Figure 8: Element-level PS^{2}J performance vs. α.

increases due to the filtering and refinement with more candidate pairs. Nonetheless, compared with NLJ, the speed-up ratio of our PS^2J approach increases with the increasing data size, which indicates the good scalability of our approach against data size.

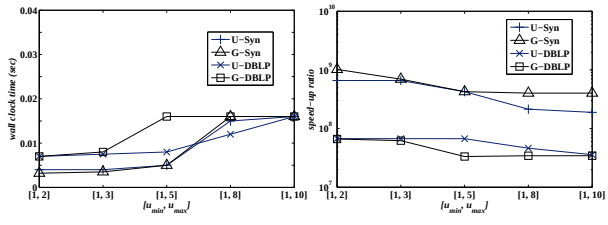
6.2 PS^{2}J over Element-Level Probabilistic Data

Next, we evaluate the PS^{2}J performance on element-level probabilistic set databases. As mentioned earlier in this section, we compare our PS^2J approach with NLJ in which we directly compute the PS^{2}J probability in the join predicate (i.e., Eq. (3)) via Eq. (5).

PS^{2}J performance vs. similarity threshold γ and probabilistic threshold α. Figure 7 presents the experimental results of PS^{2}J processing under element-level probabilistic set level, for different γ values. In Figure 7(a), the wall clock time of all the data sets are small (below $0.01s$), and slight decreases with the increasing γ, which can be reflected by the increasing speed-up ratio shown in Figure 7(b). While NLJ needs to materialize instances for all probabilistic sets, our PS^2J approach only needs to refine the obtained candidate pairs, which thus incurs lower cost by about 7-9 orders of magnitude. Figure 8 shows the results with different α from 0.1 to 0.9, with similar trends to that of γ.

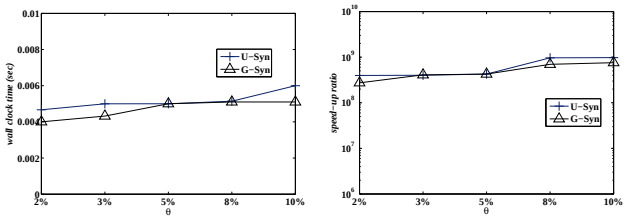
PS^{2}J performance vs. range of the instance number for each element position $[u_{min}, u_{max}]$. Figure 9 evaluates the PS^{2}J performance with different ranges of the number of instances for probabilistic elements. From figures, we can see that the wall clock time slightly increases with wider range of instance numbers (i.e., more expected instances). This is because higher cost is required for refining candidate pairs with more element instances. Nevertheless, the time cost is low (i.e., below $0.02s$), and can perform better than NLJ by 8-9 orders of magnitude.

PS^{2}J performance vs. percentage of probabilistic elements in a probabilistic set θ. Figure 10 illustrates the results for different



(a) wall clock time (b) speed-up ratio

Figure 9: Element-level PS^{2}J performance vs. $[u_{min}, u_{max}]$.



(a) wall clock time (b) speed-up ratio

Figure 10: Element-level PS^{2}J performance vs. θ.

percentages, θ, of probabilistic elements in a probabilistic set from 2% to 10%. The wall clock time increases with larger θ due to the higher refinement cost; meanwhile, the speed-up ratio also increases, which indicates the scalability of our approach against θ, compared with NLJ.

In addition, similar to the set-level case, the PS^{2}J performance of our approach also shows good scalability against large data size.

7. RELATED WORK

The existing works [21, 7, 2, 3, 23] on set similarity join usually focus on the join over certain sets, where each set (including its elements) are assumed to be precisely known. The join operator is conducted between two certain set databases, and aims to retrieve those pairs of sets that are similar to each other under some set similarity function (e.g., Jaccard similarity, Cosine similarity, and overlap similarity). In order to efficiently perform the threshold-based set similarity join, many pruning techniques have been proposed, including the signature-based filtering [2], prefix filtering [7, 3], and positional/suffix filtering [23]. In these works, either synopses or data pre-processing techniques are designed specific for certain set data, which cannot be directly or efficiently applied to our PS^{2}J problem for probabilistic sets. For example, the signature proposed in [2] summarizes precise elements in each set, which can employ the pigeon hole principle to facilitate pruning during the join processing. However, it is not trivial to directly use signatures to characterize probabilistic sets associated with probabilities, and moreover help the pruning under possible worlds semantics. Further, for prefix filtering [7, 3] or positional/suffix filtering [23], a global ordering of elements is required for sorting each (certain) set, which is however not applicable to our PS^{2}J problem over element-level probabilistic sets (in Section 6, we tested the filtering methods in [23] for the join over set-level probabilistic sets, whose performance is inferior to our approach). In addition, Jacox and Samet [14] studied the join on certain data in metric spaces, whereas our work focuses on the join on uncertain data under possible worlds semantics.

Due to the existence of data uncertainty in many real applications such as sensor networks [19], efficient and effective manipulation of probabilistic data has recently been extensively studied [11], and many systems such as MystiQ [5], Orion [8], TRIO [4], MayBMS [1], MCDB [15], and BayesStore [22] have been proposed. To the best of our knowledge, no prior work has studied the set similarity join problem over either set- or element-level probabilistic set databases. There are some existing works on join over uncertain databases such as [9, 17, 18]. However, the underlying uncertain database is assumed to contain numerical data (instead of set data) with distance functions such as L_2-norm (rather than set similarity measure). Thus, their proposed techniques cannot be directly used in our PS^{2}J problem. Recently, Jestes et al. [16] studied the probabilistic string similarity join with the expected Edit distance over all possible worlds, where string- and character-level uncertainties are considered. To help the pruning, a notion of probabilistic q-grams is proposed. In contrast, our PS^{2}J problem considers the join over probabilistic sets (rather than strings) and under a different measure, Jaccard distance (not expected Edit distance), which thus cannot borrow the proposed techniques in our PS^{2}J scenario.

8. CONCLUSIONS

Inspired by the importance of joining noisy set data in emerging applications such as data integration and near duplicate detection, in this paper, we propose two models for probabilistic sets on set and element levels of uncertainty. We propose a novel problem of joining two probabilistic set databases, namely *probabilistic set similarity join* (PS^{2}J), under these two models. To facilitate efficient processing, we design effective filtering techniques to reduce the PS^{2}J search space. We have demonstrated the PS^{2}J performance of our propped approaches through extensive experiments.

Acknowledgments

Funding for this work was provided by Hong Kong RGC GRF Grant No. 611608 and NSFC Grant No. 60933011 and 60933012.

9. REFERENCES

[1] L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing incomplete information with probabilistic world-set decompositions. In *ICDE*, 2007.
[2] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *VLDB*, 2006.
[3] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, 2007.
[4] O. Benjelloun, A. Das Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *VLDB*, 2006.
[5] J. Boulos, N. N. Dalvi, B. Mandhani, S. Mathur, C. Ré, and D. Suciu. Mystiq: a system for finding more answers by using probabilities. In *SIGMOD*, 2005.
[6] A. Broder. On the resemblance and containment of documents. In *SEQUENCES*, 1997.
[7] S. Chaudhuri, V. G., and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, 2006.
[8] R. Cheng, S. Singh, and S. Prabhakar. U-DBMS: A database system for managing constantly-evolving data. In *VLDB*, 2005.
[9] R. Cheng, S. Singh, S. Prabhakar, R. Shah, J. S. Vitter, and Y. Xia. Efficient join processing over uncertain data. In *CIKM*, 2006.
[10] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, 1997.
[11] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4), 2007.
[12] X. L. Dong, A. Halevy, and C. Yu. Data integration with uncertainty. *The VLDB Journal*, 18(2), 2009.
[13] R. Gupta and S. Sarawagi. Creating probabilistic databases from information extraction models. In *VLDB*, 2006.
[14] E. H. Jacox and H. Samet. Metric space similarity joins. *TODS*, 33(2), 2008.
[15] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. J. Haas. Mcdb: a monte carlo approach to managing uncertain data. In *SIGMOD*, 2008.
[16] J. Jestes, F. Li, Z. Yan, and K. Yi. Probabilistic string similarity joins. In *SIGMOD*, 2010.
[17] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. Probabilistic similarity join on uncertain data. In *DASFAA*, 2006.
[18] V. Ljosa and A. K. Singh. Top-k spatial joins of probabilistic objects. In *ICDE*, 2008.
[19] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X.-Y. Li, and G. Dai. Canopy closure estimates with greenorbs: Sustainable sensing in the forest. In *ACM Sensys*, 2009. http://greenorbs.org.
[20] T. C. Redman. The impact of poor data quality on the typical enterprise. *Commun. ACM*, 41(2), 1998.
[21] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In *SIGMOD*, 2004.
[22] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. Hellerstein. Bayestore: Managing large, uncertain data repositories with probabilistic graphical models. In *VLDB*, 2008.
[23] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, 2008.

Appendix

A. Notations

Table 3 summarizes the commonly used symbols in this paper.

Symbol	Description
R^P (or S^P)	a probabilistic set database
$pw(R^P)$ (or $pw(S^P)$)	a possible world of probabilistic set database R^P (or S^P)
r_i (or s_j)	a probabilistic set in R^P (or S^P)
r_{ik}, r' (or s_{jk}, s')	a set instance in the set-level probabilistic set r_i (or s_j)
$r_{ik}.p$ (or $s_{jk}.p$)	the existence probability of set instance r_{ik} (or s_{jk}) on the set level
$r_i[k]$ (or $s_j[k]$)	a probabilistic element in the element-level probabilistic set r_i (or s_j)
$r_i^{u_{ik}}[k]$ (or $s_j^{v_{jk}}[k]$)	a possible value of probabilistic element $r_i[k]$ (or $s_j[k]$) on the element level
$r_i^{u_{ik}}[k].p$ (or $s_j^{v_{jk}}[k].p$)	the existence probability of value $r_i^{u_{ik}}[k]$ (or $s_j^{v_{jk}}[k]$) on the element level

Table 3: Symbols and descriptions.

B. Proof of Lemma 3.1

Proof. In Eq. (5), when either probabilistic set r_i or s_j does not have set instance r' or s' appearing in possible worlds, it holds that $J(r', s') = 0$, and we always have $\chi(sim(r', s') \geq \gamma) = 0$. Thus, we can simplify the formula by condensing those possible worlds that both contain set instances r' and s' of r_i and s_j, respectively. Therefore, we can rewrite Eq. (5) as:

$$\begin{aligned} & Pr\{sim(r_i, s_j) \geq \gamma\} \\ = & \sum_{\forall r' \in r_i} \sum_{\forall s' \in s_j} r'.p \cdot s'.p \sum_{\forall pw^{SL}(R^P - \{r_i\})} \sum_{\forall pw^{SL}(S^P - \{s_j\})} \\ & Pr\{pw^{SL}(R^P - \{r_i\})\} \cdot Pr\{pw^{SL}(S^P - \{s_j\})\} \\ & \cdot \chi(sim(r', s') \geq \gamma) \\ = & \sum_{\forall r' \in r_i} \sum_{\forall s' \in s_j} r'.p \cdot s'.p \cdot 1 \cdot \chi(sim(r', s') \geq \gamma) \end{aligned}$$

which is exactly Eq. (7).

Hence, Eq. (7) holds, which completes the proof. □

C. Proof of Lemma 3.2

Proof. As mentioned in Section 2.2, the element-level PS^{2}J problem is equivalent to the set-level one by materializing all the instances of probabilistic sets in the databases. Therefore, the probability computation on the element level in Eq. (8) is equivalent to Eq. (7) on the set level, by expanding $r'.p$ and $s'.p$ to their element levels. Hence, similar to the proof of Lemma 3.1, Eq. (8) condenses the possible worlds of probabilistic set databases that contain both materialized set instances r' and s'. □

D. Proof of Lemma 4.1

Proof. It is sufficient to show that for any instance sets $r' \in r_i$ and $s' \in s_j$, it holds that $J\_dist(r', s') > 1 - \gamma$ (since it is equivalent to $J(r', s') < \gamma$, that is, $Pr\{J(r', s') \geq \gamma\} = 0 < \alpha$).

According to the definition of $L(\cdot, \cdot)$, we have

$$J\_dist(r', piv_{r_i}) \leq L(r_i, piv_{r_i})$$

and

$$J\_dist(s', piv_{s_j}) \leq L(s_j, piv_{s_j}),$$

for any $r' \in r_i$ and $s' \in s_j$.

Since Jaccard distance function $J\_dist(\cdot, \cdot)$ follows the triangle inequality, by inequality transition, we obtain:

$$\begin{aligned} & J\_dist(r', s') \\ \geq & J\_dist(piv_{r_i}, s') - J\_dist(r', piv_{r_i}) \\ \geq & J\_dist(piv_{r_i}, s') - L(r', piv_{r_i}) \\ \geq & J\_dist(piv_{r_i}, piv_{s_j}) - J\_dist(s', piv_{s_j}) - L(r', piv_{r_i}) \\ \geq & J\_dist(piv_{r_i}, piv_{s_j}) - L(s', piv_{s_j}) - L(r', piv_{r_i}) \end{aligned}$$

From the lemma assumption given by Eq. (9) and the inequality transition, we have:

$$J\_dist(r', s') > 1 - \gamma.$$

Hence, pair (r', s') cannot be one of the PS^{2}J results, and thus can be safely pruned. □

E. Proof of Lemma 4.2

Proof. According to the definition of $UB\_P(r_i, s_j)$ and Eq. (10), we have the following inequality transition:

$$Pr\{sim(r_i, s_j) \geq \gamma\} \leq UB\_P(r_i, s_j) < \alpha.$$

Thus, it violates the PS^{2}J condition given in Eq. (3) (i.e., $Pr\{sim(r_i, s_j) \geq \gamma\} \geq \alpha$). As a result, based on Definition 2.3, the pair (r_i, s_j) can be safely pruned. Hence, the lemma holds. □

F. Proof of Lemma 4.3

Proof. As illustrated in Figure 1, the probability upper bound is given by summing up the (multiplied) existence probabilities of set instances $r_{ik} \in r_i$ and $s_{jn} \in s_j$, when their set sizes fall into the shaded region in the 2D space. Thus, $min\_len(|r_{ik}|)$ and $max\_len(|r_{ik}|)$ are exactly lower and upper bound for the shaded region along vertical axis, when the horizontal coordinate equals to $|r_{ik}|$. As a result, the upper bound probability can be given by summing up the existence probability of set instance r_{ik} times the probability that $|s_{jn}|$ is within $[min\_len(|r_{ik}|), max\_len(|r_{ik}|)$ (i.e., the difference of cumulative probability $CPV_{s_j}[\lceil max\_len(|r_{ik}|)\rceil] - CPV_{s_j}[\lfloor min\_len(|r_{ik}|)\rfloor]$. Hence, Eq. (14) is equivalent to the $UB\_P(r_i, s_j)$ definition in Eq. (13), which completes the proof. □

G. Proof of Lemma 4.4

Proof. As illustrated in Figure 1, the probability upper bound is given by summing up the (multiplied) existence probabilities of set instances $r_{ik} \in r_i$ and $s_{jn} \in s_j$, when their set sizes fall into the shaded region in the 2D space. Thus, $min\_len(w)$ and $max\_len(w)$ are exactly lower and upper bound for the shaded region along vertical axis, when the horizontal coordinate equals to w. As a result, the upper bound probability can be given by summing up the probability that set instance r_{ik} has size w (i.e., $F(r_i, l_i, w)$) times the probability that $|s_{jn}|$ is within $[min\_len(w), max\_len(w)$ (i.e., the difference of cumulative probability $CPV_{s_j}[\lceil max\_len(w)\rceil] - CPV_{s_j}[\lfloor min\_len(w)\rfloor]$. Hence, Eq. (15) is equivalent to the $UB\_P(r_i, s_j)$ definition in Eq. (13), which completes the proof. □

H. Descriptions of PS^{2}J Processing Procedure

Figure 11 presents the details of our PS^{2}J processing procedure, namely PS^2_Processing. Specifically, the procedure PS^2_Processing utilizes a *minimum heap* $\mathcal{H}$ to traverse the two M-tree indexes in parallel, that is, $\mathcal{I}_R$ and $\mathcal{I}_S$, constructed over probabilistic set databases R^P and S^P, respectively. Each heap entry has the form (e_R, e_S, key)

(line 2), where e_R and e_S are nodes from $\mathcal{I}_R$ and $\mathcal{I}_S$, respectively, and key is defined as the LHS of Eq. (9) (i.e., a lower bound of the Jaccard distance between any set instances under nodes e_R and e_S).

Initially, we insert one entry containing the roots of both indexes into heap $\mathcal{H}$ (line 3). Each time an entry (e_R, e_S, key) with the minimum key is popped out from the heap (intuitively, small key indicates high Jaccard similarity; lines 4-5). In case e_R and e_S are both leaf nodes, for each set pair (r_i, s_j) under them, we check whether or not it can be pruned by our proposed 2 pruning techniques (i.e., in Lemmas 4.1 and 4.2) on the set- or element-level. If the answer is no, then we add this pair to the PS^{2}J candidate set PS^2J_cand (lines 7-10). Similarly, when e_R and e_S are not both leaf nodes, we expand the children of non-leaf nodes, and obtain node pairs (e_1, e_2). If (e_1, e_2) cannot be pruned by our proposed node-level pruning techniques (i.e., Jaccard distance pruning, aggregate pruning, or probability upper bound pruning), we need to insert this pair back into the heap $\mathcal{H}$ for further filtering (lines 11-14). When heap $\mathcal{H}$ is empty (line 4) or all remaining entries in $\mathcal{H}$ can be pruned by threshold $(1-\gamma)$ (line 5), the loop above terminates. We refine the remaining candidate pairs in the candidate set PS^2J_cand and return the final PS^2J results (lines 15-16).

Procedure PS^{2}J_Processing {
  Input: two probabilistic set databases R^P and S^P, with their corresponding M-trees $\mathcal{I}_R$ and $\mathcal{I}_S$, respectively, a similarity threshold $\gamma \in (0, 1]$, and a probabilistic threshold $\alpha \in (0, 1]$
  Output: the PS^{2}J results in the form (r_i, s_j) satisfying Eq. (3)
  (1) $PS^2J\_cand = \emptyset$;
  (2) initialize an empty min-heap $\mathcal{H}$ accepting entries (e_R, e_S, key)
  (3) insert $(root(\mathcal{I}_R), root(\mathcal{I}_S))$ into heap $\mathcal{H}$
  (4) while heap $\mathcal{H}$ is not empty
  (5) (e_R, e_S, key) = de-heap($\mathcal{H}$)
  (6) if $key > 1-\gamma$, then terminate the loop;
  (7) if e_R and e_S are both leaf nodes
  (8) for any set pair (r_i, s_j) such that $r_i \in e_R$ and $s_j \in e_R$
  (9) if (r_i, s_j) cannot be pruned by Jaccard distance pruning or probability upper bound pruning // Lemmas 4.1 and 4.2, respectively
  (10) add (r_i, s_j) to PS^2J_cand
  (11) else
  (12) for any pair (e_1, e_2) such that $e_1 \in e_R$ and $e_2 \in e_R$
  (13) if (e_1, e_2) cannot be pruned by Jaccard distance pruning, aggregate pruning, or probability upper bound pruning // Lemma 5.1, Eq. (19), and Lemma 5.2, respectively
  (14) insert (e_1, e_2, key) into heap $\mathcal{H}$
  (15) refine pairs (r_i, s_j) in PS^2J_cand via Eq. (14) or (15)
  (16) return the refined PS^{2}J results in PS^2J_cand
}

Figure 11: Procedure of probabilistic set similarity join.

I. Descriptions of Experimental Data Sets

For synthetic data, we generate l_i ($\in [\lambda_{min}, \lambda_{max}]$) set instances for each set-level probabilistic set $r_i \in R^P$ (or $s_j \in S^P$) as follows. For each set instance r' of r_i, we first randomly produce its set size $|r'| = \sigma \in [\sigma_{min}, \sigma_{max}]$, and then, for each (the k-th) element position, we generate a random element $r'[k] \in [1, 100]$, following either *Uniform* or *Gaussian distribution* (with the mean 50 and variance 20). We also associated each set instance $r' \in r_i$ with its existence probability $r'.p \in (0, 1]$ such that $(\sum_{\forall r' \in r_i} r'.p) \in [p_{min}, p_{max}]$ (we set $[p_{min}, p_{max}] = [0.9, 1]$ as default value range in our experiments). On the other hand, for the element-level probabilistic sets, we first synthetically generate a pivot set $piv_{r'}$ of a random size $|r'| = \sigma \in [\sigma_{min}, \sigma_{max}]$ (using the above mentioned method), and then for each of its randomly selected $(\theta \cdot |r'|)$ element positions $r'[k]$, we synthetically produce u ($\in [u_{min}, u_{max}]$) possible probabilistic element numbers, $r'^u[k]$, with *Uniform* or *Gaussian distribution*, where $\sum_{\forall u} r'^u[k] \in [p_{min}, p_{max}]$. For brevity, we denote the synthetic data with uniform and Gaussian element number distributes as $U\text{-}Syn$ and $G\text{-}Syn$, respectively. In the sequel, we report the results over two data pairs, $U\text{-}Syn \sim U\text{-}Syn$ and $G\text{-}Syn \sim G\text{-}Syn$ (for short $U\text{-}Syn$ and $G\text{-}Syn$, respectively), and omit similar results for other data combinations due to space limit. We also use the real data set, $DBLP$, which contains around $20K$ titles of papers extracted from DBLP (*http://dblp.uni-trier.de/xml/*).. We parse tokens (words) of these paper titles, and remove some frequent but meaningless tokens such as "of", "a", "the", "for", "and", "in", "on", "with", "an", "to", and so on. As a result, we can obtain sets of about 5-10 tokens for each paper title. For each paper title (probabilistic set r_i), we let its corresponding token set be the pivot set r', based on which we generate other set instances by altering $(\theta \cdot |r'|)$ elements of r' for set-level probabilistic set, or probabilistic element numbers $r'^u[k]$ for $\theta \cdot |r'|$ randomly selected element positions $r'[k]$ for element-level probabilistic sets. The resulting data sets are denoted as $U\text{-}DBLP$ and $G\text{-}DBLP$, respectively. We divide each data set into two parts of equal size, and use them as two joining data sets for testing PS^{2}J performance. For those data with other parameter settings (e.g., distributions of set elements, mean/variance of Gaussian distributions, or join combinations of data sets), the experimental results are similar and thus omitted.

J. Experimental Settings

Table 4 depicts the experimental settings in our experiments, where the values in bold font indicate *default values*. For each set of experiments, we will vary the value of one parameter, while setting others to their default values.

parameters		values
γ	set- / element-level	0.1, 0.2, **0.5**, 0.8, 0.9
α	set- / element-level	0.1, 0.2, **0.5**, 0.8, 0.9
$[\lambda_{min}, \lambda_{max}]$	set-level	[1, 2], [1, 3], **[1, 5]**, [1, 8], [1, 10]
$[u_{min}, u_{max}]$	element-level	[1, 2], [1, 3], **[1, 5]**, [1, 8], [1, 10]
$[\sigma_{min}, \sigma_{max}]$	set- / element-level	[2, 10], [3, 10], **[5, 10]**, [8, 10], [9, 10]
θ	element-level	2%, 3%, **5%**, 8%, 10%
N	set- / element-level	10K, 20K, **50K**, 80K, 100K

Table 4: The Parameter Settings