Probabilistic Graph Summarization

Nasrin Hassanlou, Maryam Shoaran, and Alex Thomo

University of Victoria, Victoria, Canada
{hassanlou,maryam,thomo}@cs.uvic.ca

1 Abstract

We study group-summarization of probabilistic graphs that naturally arise in social networks, semistructured data, and other applications. Our proposed framework groups the nodes and the edges of the graph based on a user selected set of node attributes. We present methods to compute useful graph aggregates without the need to create all of the possible graph-instances of the original probabilistic graph. Also, we present an algorithm for graph summarization based on pure relational (SQL) technology. We analyze our algorithm and practically evaluate its efficiency using an extended Epinions dataset as well as synthetic datasets. The experimental results show the scalability of our algorithm and its efficiency in producing highly compressed summary graphs in reasonable time.

2 Introduction

Graphs are very popular in modeling social networks, protein interactions, web and communication networks, and semistructured data. Nodes in such graphs represent objects or users and edges depict relationships between them. Also, there is often a set of characterizing attributes assigned to each node, such as age, location, function, etc.

Probabilistic graphs are commonly used to model networks with uncertainties on the relationships between nodes. An important application of probabilistic graphs is in social networks, where the users' influence is modeled as probabilities on the edges [5, 6]. Uncertainty can also be a result of data collection processes, machine-learning methods employed in preprocessing, and privacy-preserving processes [11]. Our focus in this work is on graphs where edges (relationships) have existence or influence probabilities as in [5], and we address the problem of summarizing such probabilistic graphs.

Summarization is important because the real life graphs are very large, and thus, there is a pressing need to describe them by focusing on groups of nodes and their relationships, rather than individual nodes. We consider a graph summarization notion in which nodes are grouped based on node attributes.

Based on the common notion of *possible worlds* (possible instances) of probabilistic databases [1–3, 7], a probabilistic graph G defines a set of regular graphs called *possible instances* (PI). Assigned to each possible instance, there is an existence probability. Although our method uses the possible worlds semantics, it does not create all of the possible instances of the original graph. Note that, the number of possible instances is exponential in the number of probabilistic edges in the graph. We give characterization theorems to compute expected values of the aggregations included in the summary graph using the edge probabilities.

The massive size of graph data, such as social networks, requires devising effective management methods that employ disk operations and do not necessarily need to load the entire graph in the memory. We present a summarization algorithm that is SQL-based and employs relational operations to create the summary graph. Notably, using relational technology for solving graph problems has been shown to satisfactorily support other graph problems as well (cf. [9,12,16]).

Experimentally, we evaluate our algorithm by implementing it on the Epinions dataset and show that our presented approach is scalable and efficiently computes aggregates on large datasets. More specifically, our contributions are:

1. We present a framework for group-based summarization of probabilistic graphs. Our summarization produces useful expected values for the strength of inter-group connectedness.
2. We give characterization theorems for the aggregates of our graph summarization. Some of our results involve sophisticated probabilistic reasoning.
3. We present an algorithm to compute the aggregates of our graph summarization that can be implemented completely using relational operators in an RDBMS. This is a desirable advantage as relational databases are a sound and mature technology that has been proven to scale for very large data.
4. We conduct a detailed experimental evaluation on a real life dataset and synthetic datasets. Our experiments show the scalability of our algorithm as well as the effectiveness of graph summarization regarding the compressibility/understanding of the original graph.

Organization. We review related work in Section 3. In Section 4 we define our method for summarizing regular graphs. In Sections 5 and 6 we define probabilistic graphs and introduce our probabilistic graph summarization method. The theorems and proofs for our probabilistic method are also presented in Section 6. In Section 7 we propose an algorithm to implement our method. In Section 8 we explain the implementation of our algorithm on Extended Epinions dataset and analyze the efficiency and the effectiveness of our method. Section 9 concludes the paper.

3 Related Work

Summarization of regular (non-probabilistic) graphs has been studied with respect to different aspects (cf. [17, 19, 21]). Various problems have been studied on probabilistic graphs (cf. [8, 10, 11, 15, 22]). However, to the best of our knowledge we are the first to address the problem of summarization of uncertain data graphs.

Graph mining, uncertain data mining, and probabilistic database management issues have motivated several studies in the database and data mining research communities.

In the general graph mining area, statistical methods have been introduced in order to present graph specifications (cf. [4, 20, 22, 23]). Graph compression methods are used to understand the main structure of the underlying large graphs [18]. Graph summarization can be considered as lossy graph compression, however, whereas the summarization compresses the graphs based on a user selected set of attributes, graph compression usually compresses the graphs based only on the original graph structure.

Graph partitioning algorithms also help discovering dense subgraphs. Finding frequent patterns is another graph mining technique for understanding large graphs [23]. However, the nature of understandings we obtain from summarization and frequent pattern mining are different.

The management of databases consisting of incomplete uncertain data could be challenging enough as well. The management task includes, for example, query answering techniques [7, 14] and aggregations [13]. The possible world semantics is one of the most useful notions which has been used in many uncertain database mining studies [1, 2]. We take advantage of this concept to summarize probabilistic graphs.

4 Graph Summarization

Let $\Delta = \{r, s, \ldots\}$ be a finite alphabet of labels. We denote a graph database as $G = (V, E, \Delta)$, where V is the set of nodes, and $E \subseteq V \times \Delta \times V$ is the set of Δ-labeled edges connecting the nodes. With E_r, where $r \in \Delta$, we denote the subset of edges labeled by r.

Furthermore, there is a set of attributes $A_1, A_2, \cdots, A_d$ associated with the nodes. Attributes can be nominal or numerical. Numerical attributes can be discretized as in [19].

We represent the attribute values for a node $v \in V$ as a d-tuple $(a_1, a_2, \cdots, a_d)$, where a_i, for $i \in [1, d]$, is the value of A_i for v.

Let $\mathcal{A}$ be a subset of node attributes. Using $\mathcal{A}$ we group the nodes of G in the usual GROUP BY way and obtain a set $\mathcal{V}_{\mathcal{A}}$ of node groups. Now we have

Definition 1. *The* $\mathcal{A}$-grouping graph *is* $\mathcal{G}_{\mathcal{A}} = (\mathcal{V}_{\mathcal{A}}, \mathcal{E}_{\mathcal{A}})$ *where*

$$\mathcal{E}_{\mathcal{A}} = \{(g', r, g'') : g', g'' \in \mathcal{V}_{\mathcal{A}} \textit{ and } \exists v' \in g' \textit{ and } \exists v'' \in g'' \\ \textit{such that } (v', r, v'') \in E_r\}.$$

Definition 2. *The* $\mathcal{A}$-graph summarization (A-GS) *is a node-edge weighting pair of functions* (w_1, w_2), *where*

$$
\begin{aligned}
w_1 &: \mathcal{V}_{\mathcal{A}} \longrightarrow \mathbb{N} \\
w_2 &: \mathcal{E}_{\mathcal{A}} \longrightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{N} \\
w_1(g) &= |g| \\
w_2(g', r, g'') &= (x, y, z), \;\; \textit{where} \\
x &= |\{v' \in g' : \exists v'' \in g'', \;\; s.t. \; (v', v'') \in E_r\}| \\
z &= |\{v'' \in g'' : \exists v' \in g', \;\; s.t. \; (v', v'') \in E_r\}| \\
y &= |\{(v', v'') : v' \in g', v'' \in g'', (v', v'') \in E_r\}|.
\end{aligned}
$$

Fig. 1.(a) shows a simple graph containing seven nodes. Consider the color of the nodes to be the grouping attribute. Fig. 1.(b) shows the $\mathcal{A}$-graph summarization of the graph in Fig. 1.(a) with the corresponding values of the w_1 and w_2 measures.

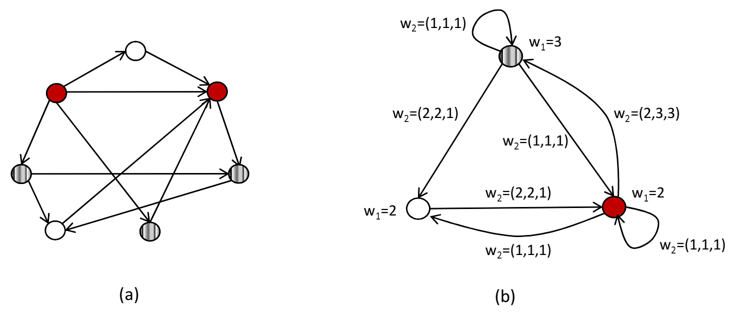


Fig. 1. (a) A Graph G. (b) The summary graph of G.

5 Probabilistic Graphs

A probabilistic graph is $G = (V, E, \varDelta)$ (as above), however, associated with each edge e there is a probability $p(e)$ expressing the confidence on the existence of e. A probabilistic graph defines a set of *possible instances* (PIs). We denote the set of all possible instances of a probabilistic graph G as $\mathcal{PI}(G)$ or $\mathcal{PI}$ if G is clear from the context.

A possible instance (PI) of G is denoted as $PI_i(G)$ or simply PI_i. Each PI is a regular graph derived from the probabilistic graph where each edge either exists or does not. The existence probability of each PI is computed as

$$
p(PI) = \prod_{e \in E(PI)} p(e) \cdot \prod_{e \notin E(PI)} (1 - p(e)) \tag{1}
$$

, where $E(PI)$ is the set of the edges existent in the possible instance PI. Each edge e in the probabilistic graph G appears in a subset of the PIs. For a given edge e the probabilities of PIs containing e sum up to the confidence value (probability) of e, which is denoted as $p(e)$. That is, we have $p(e) = \sum_{E(PI)\ni e} p(PI)$. If e has confidence 1, then it appears in all of the PIs.

Since the number of PIs doubles with each additional probabilistic edge in the input graph, the result of queries on these graphs is exponential in the size of the input graph.

6 Probabilistic Graph Summarization

We define summarization of probabilistic graphs in a similar way as in Definition 2. However, having probabilistic edges between nodes in a graph results in probabilistic edges between groups in the summary graph. Thus, instead of the *exact* value of w_2 the *expected* value should be computed for each of its elements x, y, and z. Note that, the exact value of w_1 is computable as in non-probabilistic case, since in our data model no probabilities are assigned to nodes or node attributes.

For an easier illustration, we assume a simple graph where all the edges have identical labels (i.e., $|\Delta| = 1$). Let g and g' be two groups of nodes in the summary graph $\mathcal{G}_{\mathcal{A}}$. In each PI the set of edges that connect the nodes of these two groups are different, and hence, the *exact* values of x, y, and z differ in the summary graph corresponding to each PI. The expected values for x, y, and z in $\mathcal{G}_{\mathcal{A}}$ can be computed using the basic formula for expected value of random variables. For example, for the expected value of x we have $E[X] = \sum_{\mathcal{PI}} x_i.p(PI_i)$, where X is the random variable representing the x measure. Note that, using this formula directly requires building all the possible instances of the original graph.

In the following we present (and prove) equations that compute $E[X]$, $E[Y]$, and $E[Z]$ by using only the probability of edges in G with no need to create all PIs and compute their corresponding w_2's.

Lemma 1. *For any subgraph G' of a probabilistic graph G we have*

$$\sum_{PI\in\mathcal{PI}(G')} p(PI) = 1.$$

Proof. We prove the lemma for the case when only one of the edges of G is missing in G'. The proof can then be easily extended to complex cases.

The set of possible instances of G can be divided into two disjoint sets. One set is the set of PIs where e exists, and the other includes PIs where e does not exist. Let $\mathcal{PI}_e(G)$ and $\mathcal{PI}_{-e}(G)$ be these two sets of PIs, respectively. We have

$$\sum_{PI\in\mathcal{PI}(G)} p(PI) = \sum_{PI\in\mathcal{PI}_e(G)} p(PI) + \sum_{PI\in\mathcal{PI}_{-e}(G)} p(PI).$$

We can write the above equation as

$$
\begin{aligned}
1 = \sum_{PI \in \mathcal{PI}(G)} p(PI) = p(e) \cdot & \sum_{PI \in \mathcal{PI}(G')} p(PI) \\
& + (1 - p(e)) \cdot \sum_{PI \in \mathcal{PI}(G')} p(PI) \\
& = \sum_{PI \in \mathcal{PI}(G')} p(PI)
\end{aligned}
$$

and this concludes the proof. □

Theorem 1. *Let g and g' be two groups in a probabilistic summary graph G, and let $E_{v_j} = \{e_1, \ldots, e_{n_j}\}$ be the set of edges connecting a node $v_j \in g$ to the nodes of g'. We have that*

$$E[X(g, g')] = E[X] = \sum_{v_j \in g} \left(1 - \prod_{e \in E_{v_j}} (1 - p(e)) \right).$$

Proof. Let $W = \{v_1, \ldots, v_{|W|}\}$ be the set of nodes in group g which are connected to the nodes of group g' in G, and let $W_{PI} \subseteq W$ be the set of nodes in group g which are connected to the nodes of group g' in the possible instance PI of G. Also, let $m = |\mathcal{PI}(G)|$. We have that

$$
\begin{aligned}
E[X] &= \sum_{PI_i \in \mathcal{PI}(G)} x_i \cdot p(PI_i) \\
&= \underbrace{p(PI_1) + p(PI_1) + \ldots}_{x_1 \ times} \\
& \cdots \\
&+ \underbrace{p(PI_m) + p(PI_m) + \ldots}_{x_m \ times}
\end{aligned}
$$

where x_i is the number of nodes in g that are connected to some nodes of g' in the instance PI_i. That is, $x_i = |W_{PI_i}|$.

We can organize this equation in a different way. Note that for each node v_j, the term $p(PI_i)$ appears once in the right hand summation if $v_j \in W_{PI_i}$. Therefore, we can rewrite the equation as

$$E[X] = \sum_{W_{PI} \ni v_1} p(PI) + \cdots + \sum_{W_{PI} \ni v_{|W|}} p(PI). \qquad (2)$$

Now we compute the value of each term above. From equality $\sum_{PI \in \mathcal{PI}} p(PI) = 1$ we have that

$$\sum_{W_{PI} \ni v_j} p(PI) + \sum_{W_{PI} \not\ni v_j} p(PI) = 1. \qquad (3)$$

As defined, $E_{v_j} = \{e_1, \ldots, e_{n_j}\}$ is the set of edges incident to v_j which connect v_j to some nodes in g'. The first sum in (3) includes possible instances where at least one of the edges in E_{v_j} exists. The second sum includes possible instances where none of the edges in E_{v_j} exists.

Now, suppose G' is a probabilistic graph constructed from G by removing all the edges in E_{v_j}. That is, the probability of existence of those edges is zero in G'. Since each possible instance of G can be constructed from G' and based on (1), we can rewrite Equation (3) as

$$\sum_{PI \in \mathcal{PI}(G')} p(PI(G')) \cdot \sum_{S \in 2^{E_{v_j}}, S \neq \emptyset} \left(\prod_{e \in S} p(e) \cdot \prod_{e \in S^c} (1 - p(e)) \right) + \sum_{PI \in \mathcal{PI}(G')} p(PI(G')) \cdot \prod_{e \in E_{v_j}} (1 - p(e)) = 1$$

where $\mathcal{PI}(G')$ is the set of all possible instances of graph G', and S is a set in the power set of E_{v_j}. Since $\sum_{PI \in \mathcal{PI}(G')} p(PI) = 1$ (Lemma 1), we have that

$$\sum_{W_{PI} \ni v_j} p(PI(G)) = \sum_{PI \in \mathcal{PI}(G')} p(PI(G')) \cdot \sum_{S \in 2^{E_{v_j}}, S \neq \emptyset} \left(\prod_{e \in S} p(e) \cdot \prod_{e \in S^c} (1 - p(e)) \right) = 1 - \prod_{e \in E_{v_j}} (1 - p(e)) \quad (4)$$

and using Equations (2) and (4) we have

$$\begin{aligned} E[X] &= \sum_{W_{PI} \ni v_1} p(PI) + \cdots + \sum_{W_{PI} \ni v_{|W|}} p(PI) \\ &= \sum_{v_j \in W} \left(1 - \prod_{e \in E_{v_j}} (1 - p(e)) \right). \end{aligned}$$

This proves the theorem. □

For the expected value of y we present the following theorem.

Theorem 2. *In the summary graph, the expected value for y, $E[Y]$, is the sum of the probabilities of the edges going from one group to the other.*

Proof. Let $m = |\mathcal{PI}(G)|$ and let $S = \{e_1, \ldots, e_{|S|}\}$ be the set of all probabilistic edges (with non-zero probability) that connect the nodes of two given groups in a probabilistic summary graph. Let also $E(PI_i)$ be the set of edges in an instance PI_i.

Based on the definition of expected value of a random variable, we have that

$$\begin{aligned} E[Y] &= \sum_{PI_i \in \mathcal{PI}(\mathcal{G})} y_i.p(PI_i) \\ &= \underbrace{p(PI_1) + p(PI_1) + \dots}_{y_1\ times} + \cdots \\ &+ \underbrace{p(PI_m) + p(PI_m) + \dots}_{y_m\ times} \end{aligned}$$

where y_i is the number of edges in S that exist in PI_i. Now, we can organize this equation in a different way. Note that for each edge $e_j \in S$, if $e_j \in E(PI_i)$, the term $p(PI_i)$ appears once in the right hand summation. Therefore, we can rewrite the equation as

$$E[Y] = \sum_{E(PI_i) \ni e_1} p(PI_i) + \cdots + \sum_{E(PI_i) \ni e_{|S|}} p(PI_i).$$

On the other hand, for each edge e we have that

$$p(e) = \sum_{E(PI_i) \ni e} p(PI_i).$$

Thus,

$$E[Y] = p(e_1) + \cdots + p(e_{|S|}) = \sum_{e \in S} p(e),$$

and this proves the theorem. □

7 Algorithm

In this section we present our algorithm to build the summary graph of a probabilistic graph. We assume that the probabilistic graph is represented using some database tables. The first primary table is *Nodes* table which consists of all the nodes in the graph and their attribute values. The second is the *Edges* table which stores all the node connections (edges) in the graph. We assume that each edge has an existence probability which is stored in the same table as a separate column.

The algorithm starts by grouping the nodes based on the desired attributes. Grouping can start by sorting nodes according to their values on the selected attributes. Then, computing $E[X]$, $E[Y]$, and $E[Z]$ elements of the w_2 measure for group pairs can be done by using the theorems and formulas provided in Section 6.

The following algorithm uses the *Nodes* and *Edges* tables illustrated in Fig. 2 and returns the w_2 measure in the *Summary* table depicted in Fig 3. All the steps of our algorithm can be expressed in SQL. Due to space constraint we only

nId	A_1	...	A_d
1	a_{11}	...	a_{1d}
2	a_{21}	...	a_{2d}
$\vdots$			
n	a_{n1}	...	a_{nd}

nId1	**nId2**	**prob**
1	2	p_{12}
2	1	p_{21}
$\vdots$		
i	j	p_{ij}

Fig. 2. Tables *Nodes* and *Edges*

gId1	**gId2**	**E[X]**	**E[Y]**	**E[Z]**
g_1	g_2	x_{12}	y_{12}	z_{12}
g_2	g_1	x_{21}	y_{21}	z_{21}
$\vdots$				
g_i	g_j	x_{ij}	y_{ij}	z_{ij}

Fig. 3. Table *Summary*

give the plain language description of the steps here and refer the reader to the full version[1] of the paper.

Algorithm 1

Input:
1. Table *Nodes* including records of nodes and their attribute values.
2. Table *Edges* containing records of edges with their existence probabilities.
3. Grouping attribute set $\mathcal{A}$, which is a subset of node attributes.

Output: Table *Summary* consisting of all possible pairs of groups and their expected measures $E[X]$, $E[Y]$, and $E[Z]$.

Method:
1. Assign a group identifier, *gId*, to each node in the *Nodes* table based on the user selected attributes.
2. Update table *Edges* and add two new columns called *gId1* and *gId2*. Then, for each record insert the corresponding group Ids of node 1 (*nId1*) and node 2 (*nId2*) into *gId1* and *gId2*, respectively.
3. Group records in *Edges* based on *nId1*, *gId1*, and *gId2* using the product of $(1 - prob)$ as the aggregation function, then, insert the result into a temporary table called *K1* with the aggregate field as *product.*
4. Group records in *Edges* based on *nId2*, *gId1*, and *gID2* using the product of $(1 - prob)$ as the aggregation function, then, insert the result into a temporary table called *K2* with the aggregate field as *product.*
5. To compute element $E[X]$ in w_2 measure, group records in *K1* based on *gId1* and *gId2* using sum of $(1 - product)$ as the aggregation function and store the result in table *Summary.*

[1] http://webhome.cs.uvic.ca/∼maryam/probgraphsum05.pdf.

6. To compute element $E[Z]$ in w_2 measure, group records in *K2* based on *gId1* and *gId2* and sum of $(1 - \textit{product})$ as the aggregation function and update table *Summary*.
7. To compute element $E[Y]$ in w_2 measure, sum up *prob* values from table *Edges* by grouping records based on *gId1* and *gId2* and update table *Summary*.
8. Return the *Summary* table.

8 Evaluation

In this section we describe the implementation of our algorithm on a real dataset and evaluate its efficiency. We then analyze the scalability of our algorithm by implementing it on synthetic data.

8.1 Dataset

The real dataset we use for the evaluation is a trust network dataset from *Epinions*[2]. Epinions is a website in which users write reviews for different products of different subjects and express trust to each other. This can be competitive for users because they can get paid based on how valuable their reviews are. In order to see which reviews are better, a trust system is used.

Two different versions of the Epinions dataset are available in the Trustlet website (www.trustlet.org). In this paper we use the *Extended Epinions* dataset. The ratings in this dataset are about the reviews, also called articles. That is, the ratings represent how much a user rates a given textual article written by another user. This dataset contains:

- About 132,000 users
- 841,372 statements (trusts and distrusts)
- Around 85,000 users received at least one statement
- 1,560,144 articles

In this dataset, we are interested in finding the strength of the connections between subject groups. Using the *users* information and the *statements* we created tables *Nodes* and *Edges*, respectively. In order to have edge existence probabilities, we added the field *prob* in the *Edges* table and filled it with a random number between 0 and 1 for each record. The schemas of the *Nodes* and *Edges* tables created from the Epinions dataset are shown in Fig. 4.

8.2 Implementation of Algorithm 1

Since the *Nodes* table created from the Epinions dataset contains only one attribute, *SubjectId*, we use it as the grouping attribute and group Id will be the *SubjectId* (see Step 1 of Algorithm 1).

[2] http://www.trustlet.org/wiki/Epinions.

Field	**Type**
userId	int
subjectId	int

Field	**Type**
userId1	int
userId2	int
prob	double

Fig. 4. Tables *Nodes* and *Edges* (User and Trust information)

To assign the *subjectId*s to the nodes in the *Edges* table (Step 2 of Algorithm 1), we join tables *Nodes* and *Edges* twice, once on *userId1* and the second time on *userId2*. The result table called *Joint* (Fig. 5) represents all the valid edges in the trust graph. After these joins we end up with much more records in the *Joint* table than table *Edges*. The reason is that in the Epinions dataset a user/author may have articles in different subjects. Before joining the tables, we can follow two different strategies.

1. We can consider just one subject for each user and remove the other records for that user from the *Nodes* table. In this approach, there will be one node for each user in the graph. Applying this strategy we built a graph consisting of 130,068 nodes each corresponding to a record in *Nodes* table, and 785,286 edges corresponding to the records in the *Joint* table. The number of distinct subjects (groups) was 11,224. This graph is large enough and can be useful for evaluating our algorithm.
2. We can consider each distinct *userId*-*subjectId* pair in *Nodes* table as a node in the graph. In such a graph, we also need to consider the trust between the nodes having identical *userId*s. With the assumption that each user trusts completely on his/herself, we connect all the nodes having the same *userId* to each other with the probability of 1 and add the corresponding records in the *Edges* table. The result graph is very large with billions of nodes and edges. Fig. 6 depicts this strategy to build the desired graph from the available dataset.

We have followed the second strategy for our evaluation. We performed all the experiments on a machine with Linux server, 12 GB memory, and 3.4 GHz CPU. All methods have been implemented as SQL queries. We executed our queries on MySQL version 5.5.24. In the following section we analyze the results of our experiments on graphs with different sizes.

8.3 Analysis

In this section we analyze the complexity, the effectiveness, and the efficiency of our algorithm based on the experimental results obtained in the previous section.

8.4 Complexity of the Algorithm

In the first step, a sorting or hashing can be performed to group by the nodes based on their attribute values (the value of *subjectId*). The rest of the algorithm

Field	Type
userId1	int
gId1	int
userId2	int
gId2	int
prob	double

Fig. 5. Table *Joint*

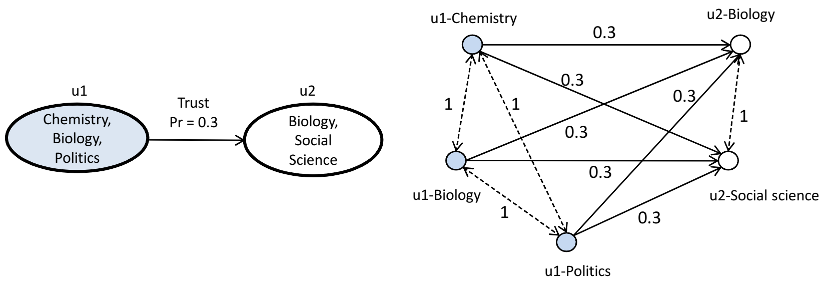


Fig. 6. Graph generation strategy.

can be completed by scanning the edges in two passes to compute the $E[X]$, $E[Y]$ and $E[Z]$ values.

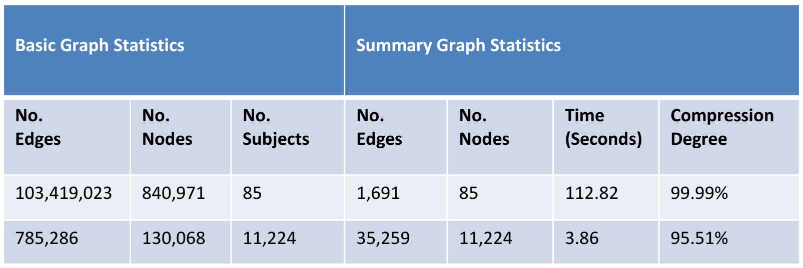
Considering memory space, our algorithm can keep the information of all the groups in the memory. If there is not enough memory, only the information about the groups for which the expected values are requested are kept in the memory. The algorithm can even run in a memory of size equal to the space needed to store statistics variables for only a pair of groups. This is because the algorithm can work with just two groups at a time and compute the expected values of the statistics. However, in this case we would need one pass for each pair of groups.

8.5 Efficiency of the Algorithm

We ran the algorithm on two graphs with different sizes created from the Epinions dataset. The first graph had 840,971 nodes and 103,419,023 edges. The algorithm finished in 113 seconds and created the summary graph containing 85 different nodes (which is exactly the number of different subjects in the base graph) and 1,691 edges. The experimental results on both graphs are illustrated in Fig. 7.

8.6 Scalability Experiments

In order to analyze the scalability of the algorithm we took advantage of synthetic graphs created based on the trust network structure of the Epinions data.



Basic Graph Statistics			Summary Graph Statistics			
No. Edges	No. Nodes	No. Subjects	No. Edges	No. Nodes	Time (Seconds)	Compression Degree
103,419,023	840,971	85	1,691	85	112.82	99.99%
785,286	130,068	11,224	35,259	11,224	3.86	95.51%

Fig. 7. The experimental results on the Epinions dataset.

We generated random graphs of different sizes and different number of groups. Each time we simply assigned random group identifiers to each node of the original graph. The experimental results on the datasets having different number of subjects or different graph sizes are shown in Fig. 8.

The left figure in Fig. 8 illustrates the execution time of the summarization algorithm (in seconds) as a function of the number of different groups (subjects) in a graph having 10,000,000 edges. The figure shows that when the graph size is constant, depending on how we group the nodes and how many different groups we get, the execution time can change. The result shows that as the number of different groups increases, the execution time would increase as well in an almost linear manner. Therefore, we can handle the summarization of graphs with large number of groups in reasonable time.

The right figure in Fig. 8 shows the execution time of the algorithm on some graphs of different sizes. In this experiment we group the nodes into exactly 300 different categories each time. The result shows that in the case of constant number of groups, the execution time increases almost linearly based on the graph size. This result shows the scalability of our algorithm.

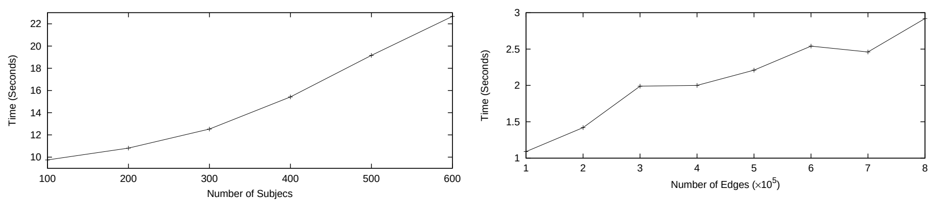


Fig. 8. Left: Execution time vs. number of subjects, Right: Execution time vs. graph size (number of edges).

8.7 Effectiveness

The experiments on large datasets show that our method is practical. To show the effectiveness of our algorithm we define a measure called *compression degree* which is the percentage of 1 minus the fraction of the number of edges in the summary graph to the number of edges in the original graph. We use this metric in order to demonstrate how powerful our proposed method is in compressing very large graphs.

Fig. 9 shows the compression degree as functions of the number of groups (subjects) and graph sizes. These figures verify that our algorithm is effective in building a very concise representation of large probabilistic graphs while it maintains the statistics of the original graphs. The compression degree assessment shows that when the number of different groups is constant, the compression degree increases as the number of edges increases. In the case of a constant graph size, when the number of groups increases, the degree of compression is still high (more than 90%) in spite of some overall decrease. For instance, when the number of different groups is 1000, the compression degree is 90%.

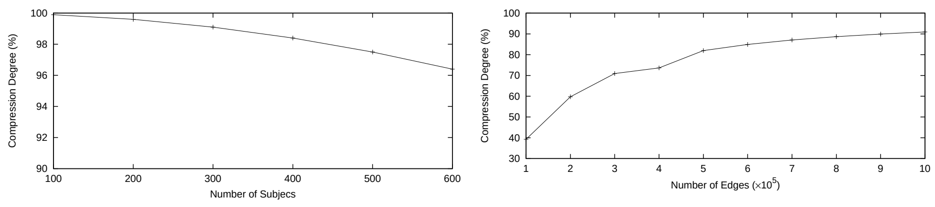


Fig. 9. Left: Compression degree vs. number of subjects, Right: Compression degree vs. graph size (number of edges).

9 Conclusions

This paper addressed the problem of summarizing probabilistic graphs using a relational database approach. We focused on a useful summarization method which groups the nodes based on a subset of attributes. In the summary graph we considered aggregates which reveal significant information about the groups and the connections between them. We gave theorems to compute these aggregates without the need to compute all possible data graphs from the original probabilistic graph. We also presented an algorithm, which uses pure SQL queries to build the summary graph. We evaluated the proposed algorithm on Epinions data and some synthetic datasets. The evaluation shows that our algorithm is practically scalable to large graphs and effectively summarizes large graphs with a very high degree of compression.

References

1. S. Abiteboul and G. Grahne. Update semantics for incomplete databases. In *VLDB*, pages 1–12, 1985.
2. S. Abiteboul, P. C. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1):158–187, 1991.
3. O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.
4. T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Züfle. Probabilistic frequent itemset mining in uncertain databases. In *KDD*, pages 119–128, 2009.
5. C. Budak, D. Agrawal, and A. E. Abbadi. Limiting the spread of misinformation in social networks. In *WWW*, pages 665–674, 2011.
6. W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208, 2009.
7. N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.
8. E. H. Frank. Shortest paths in probabilistic graphs. volume 17, pages 583–599, 1969.
9. J. Gao, R. Jin, J. Zhou, J. X. Yu, X. Jiang, and T. Wang. Relational approach for shortest path discovery over large graphs. *CoRR*, abs/1201.0232, 2012.
10. J. J. P. III and J. Neville. Methods to determine node centrality and clustering in graphs with uncertain structure. In *ICWSM*, 2011.
11. G. Kollios, M. Potamias, and E. Terzi. Clustering large probabilistic graphs. IEEE TKDE, 2010.
12. C. Mayfield, J. Neville, and S. Prabhakar. Eracer: a database approach for statistical inference and data cleaning. In *SIGMOD Conference*, pages 75–86, 2010.
13. R. Murthy, R. Ikeda, and J. Widom. Making aggregation work in uncertain and probabilistic databases. *IEEE Trans. Knowl. Data Eng.*, 23(8):1261–1273, 2011.
14. J. Pei, M. Hua, Y. Tao, and X. Lin. Query answering techniques on uncertain and probabilistic data: tutorial summary. In *SIGMOD Conference*, pages 1357–1364, 2008.
15. M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. k-nearest neighbors in uncertain graphs. *PVLDB*, 3(1):997–1008, 2010.
16. S. Srihari, S. Chandrashekar, and S. Parthasarathy. A framework for sql-based mining of large graphs on relational databases. In *PAKDD (2)*, pages 160–167, 2010.
17. Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD Conference*, pages 567–580, 2008.
18. H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka. Compression of weighted graphs. In *KDD*, pages 965–973, 2011.
19. N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.
20. Q. Zhang, F. Li, and K. Yi. Finding frequent items in probabilistic data. In *SIGMOD Conference*, pages 819–832, 2008.
21. P. Zhao, X. Li, D. Xin, and J. Han. Graph cube: on warehousing and olap multidimensional networks. In *SIGMOD Conference*, pages 853–864, 2011.
22. Z. Zou, H. Gao, and J. Li. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In *KDD*, pages 633–642, 2010.
23. Z. Zou, J. Li, H. Gao, and S. Zhang. Mining frequent subgraph patterns from uncertain graph data. *IEEE Trans. Knowl. Data Eng.*, 22(9):1203–1218, 2010.