Probabilistic Graph Summarization

Nasrin Hassanlou, Maryam Shoaran, and Alex Thomo

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

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 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 scalability using an extended Epinions dataset as well as synthetic datasets. The experimental results show that our algorithm produces compressed summary graphs in reasonable time.

1 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.

As graphs of millions of nodes and their relationships are ubiquitous now, e.g. Facebook, Twitter, Weibo, or DBpedia, there is a pressing need to summarize graphs in order to have a representation that can be consumed by human analysts. In this paper, we consider a graph summarization notion in which nodes are grouped based on node attributes and groups are connected by edges representing inter-group connectedness.

Being able to group graph nodes and edges is only the first step in understanding real graphs. Another challenge is the uncertainty or impreciseness of edges, which represent the connectedness or influence of nodes to each other. *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 [4,5]. Uncertainty can also be a result of data collection processes, machine-learning methods employed in preprocessing, and privacy-preserving processes. Our focus in this work is on graphs where edges (relationships) have existence or influence probabilities as in [4,5], and we address the problem of summarizing such probabilistic graphs.

Based on the notion of "possible worlds" for probabilistic databases [1–3, 6], a probabilistic graph G defines a set of regular graphs called possible instances.

J. Wang et al. (Eds.): WAIM 2013, LNCS 7923, pp. 545–556, 2013.
© Springer-Verlag Berlin Heidelberg 2013

Assigned to each possible instance there is an existence probability. While the theoretical framework of possible worlds is useful to define what we want, e.g. the mean group connectedness over the possible instances, the number of possible instances is exponential in the size of the original graph, thus rendering approaches that materialize the possible instances very infeasible in practice. Therefore, we present a method that, while based on the possible worlds semantics, does not create any possible instance at all of the original graph. More specifically, we give characterization theorems to compute expected values of the aggregations included in the summary graph using the edge probabilities only.

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. [8, 11, 13]). Experimentally we evaluate our algorithm by implementing it on an Epinions dataset and show that our presented approach is scalable and efficiently computes aggregates on large datasets. In summary, 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 an experimental evaluation on a real life dataset and synthetic datasets. Our experiments show the scalability of our algorithm in producing summary graphs in reasonable time.

Organization. We review related work in Section 2. In Section 3 we define our method for summarizing regular graphs. In Sections 4 and 5 we define probabilistic graphs and introduce our probabilistic graph summarization method. The theorems and proofs for our probabilistic method are also presented in Section 5. In Section 6 we propose an algorithm to implement our method. In Section 7 we explain the implementation of our algorithm on an Epinions dataset and analyze the efficiency and scalability of our method. Section 8 concludes the paper.

2 Related Work

Summarization of regular (non-probabilistic) graphs has been studied with respect to different aspects (cf. [14–16]). Various problems have been studied on

probabilistic graphs (cf. [7, 9, 10, 12, 17]). However, to the best of our knowledge we are the first to address the problem of summarization of uncertain data graphs.

Grouping the nodes of a graph based on a set of attributes is one of the most common techniques to summarize graphs. Tian et al. [14,15] introduce a framework to interactively produce such summary graphs. In [16], Zhao et al. introduce graph cubes which are graph summaries created using node grouping operations based on selected attributes.

3 Graph Summarization

We denote a graph database as $G = (V, E)$, where V is the set of nodes, and $E \subseteq V \times V$ is the set of edges connecting the nodes.

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 [15].

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*

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

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

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

Fig. 1.(a) shows a 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.

4 Probabilistic Graphs

A probabilistic graph is $G = (V, E)$ (as above), however, associated with each edge e there is a probability $p(e)$ expressing the confidence on the existence of

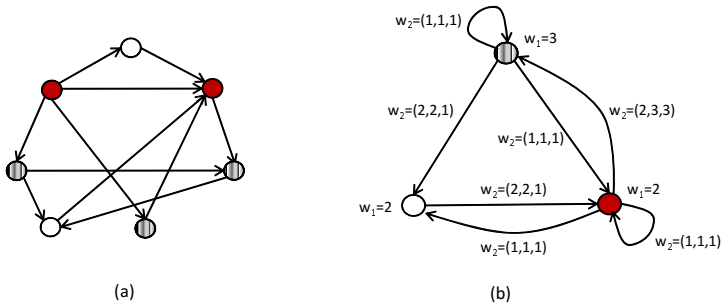

Fig. 1. (a) A Graph G. (b) The summary graph of G.

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.

5 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 the non-probabilistic case.

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 can be different, and hence, the *exact* values of x, y, and z can 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 the expected value of random variables. For example, for the expected value of x we have $E[X] = \sum_{\mathcal{PI}} x_i \cdot p(PI_i)$, 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.

Proposition 1. *For any subgraph G' of a probabilistic graph G we have* $\sum_{PI \in \mathcal{PI}(G')} p(PI) = 1$.

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

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

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

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

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

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

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 $\sum_{PI \in \mathcal{PI}(G')} p(PI) = 1$ (Proposition 1), we have that

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

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

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

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. We have that

$$E[Y] = \sum_{PI_i \in \mathcal{PI}(\mathcal{G})} y_i \cdot p(PI_i) = \underbrace{p(PI_1) + \ldots + p(PI_1)}_{y_1 \ times} + \ldots + \underbrace{p(PI_m) + \ldots + p(PI_m)}_{y_m \ times}$$

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. □

6 Algorithm

In this section we present our algorithm to build the summary graph of a probabilistic graph. We assume that the probabilistic graph is stored in database tables. The first primary table is the *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 the $E[X]$, $E[Y]$, and $E[Z]$ elements of the w_2 measure for group pairs can be done by using the theorems and formulas provided in Section 5.

nId	A_1	...	A_d
1	a_{11}	...	a_{1d}
2	a_{21}	...	a_{2d}
⋮			
n	a_{n1}	...	a_{nd}

nId1	**nId2**	**prob**
1	2	p_{12}
2	1	p_{21}
⋮		
i	j	p_{ij}

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}
⋮				
g_i	g_j	x_{ij}	y_{ij}	z_{ij}

Fig. 2. Table *Nodes* (left), Table *Edges* (middle), Table *Summary*(right)

The following algorithm uses the *Nodes* and *Edges* tables illustrated in Fig. 2 (two left tables) and returns the w_2 measure in the *Summary* table depicted in Fig 2(right table). All the steps of our algorithm can be expressed in SQL. Due to space constraint we only give the plain language description of the steps here and refer the reader to the full version[1] of the paper for the SQL statements.

[1] `http://webhome.cs.uvic.ca/~maryam/probgraphsum.pdf`

Algorithm 1

Input:

1. Table *Nodes* containing the nodes and their attribute values.
2. Table *Edges* containing the 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 the 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*.
6. To compute element $E[Z]$ in the w_2 measure, group records in *K2* based on *gId1* and *gId2* and sum of $(1 - product)$ as the aggregation function and update table *Summary*.
7. To compute element $E[Y]$ in the 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.

7 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.

7.1 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 categories or subjects and express trust to each other.

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.

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

The ratings in this dataset are about reviews, also called articles. That is, the ratings represent how much a user rates a given article written by another user. This dataset contains about:

- 132,000 users,
- 841,000 statements (trusts and distrusts),
- 85,000 users received at least one statement,
- 1,500,000 articles.

In this dataset, we are interested in finding the strength of the connections between users grouped by the subject of the articles they have written. 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 random numbers between 0 and 1 for each record.

7.2 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).

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* 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 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. 3 depicts this strategy to build the desired graph from the available dataset.
2. 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 as well.

We have followed both strategies for our evaluation. We performed all the experiments on a machine with Linux server, 12 GB memory, and 3.4 GHz CPU. All steps 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.

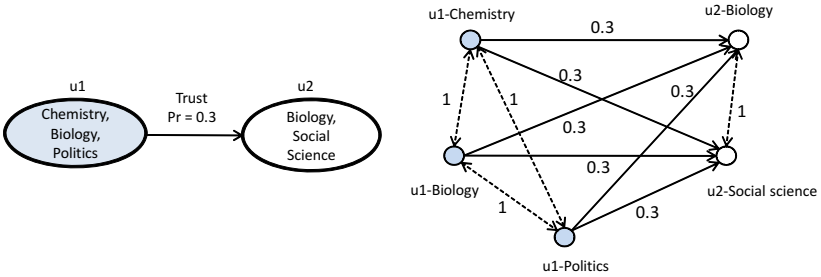

Fig. 3. Graph generation strategy

7.3 Analysis

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

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

Considering the memory space, our algorithm can keep the statistics variables for 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.

7.5 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 resulting from restricting the number of subjects to 85, which is almost 0.1% of the whole number of different subjects. This is a reasonable number of groups for a human analyst to easily use in order to understand the interconnectedness of his/her subjects of interest in terms of user trust relationships. The execution time was only 113 seconds, and the produced summary graph contained 85 different groups and 1,691 edges. The second graph was generated using the second strategy illustrated in Section 7.2, which resulted in a graph containing 11,224 different subjects in total. The execution time was only 3.86 seconds, and the produced summary graph contained 11,224 different groups and 35,259 edges.

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. 4. The experimental results on the Epinions dataset

7.6 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. 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. 5.

The left figure in Fig. 5 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. 5 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.

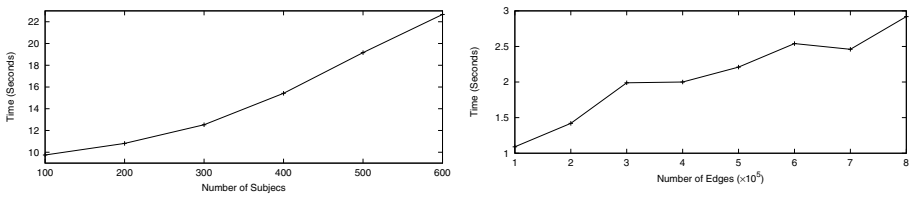

Fig. 5. Left: Execution time vs. number of subjects, Right: Execution time vs. graph size (number of edges)

8 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.

References

1. Abiteboul, S., Grahne, G.: Update semantics for incomplete databases. In: VLDB, pp. 1–12 (1985)
2. Abiteboul, S., Kanellakis, P.C., Grahne, G.: On the representation and querying of sets of possible worlds. Theor. Comput. Sci. 78(1), 158–187 (1991)
3. Benjelloun, O., Sarma, A.D., Halevy, A.Y., Widom, J.: Uldbs: Databases with uncertainty and lineage. In: VLDB, pp. 953–964 (2006)
4. Budak, C., Agrawal, D., Abbadi, A.E.: Limiting the spread of misinformation in social networks. In: WWW, pp. 665–674 (2011)
5. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: KDD, pp. 199–208 (2009)
6. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. VLDB J. 16(4), 523–544 (2007)
7. Frank, E.H.: Shortest paths in probabilistic graphs 17, 583–599 (1969)
8. Gao, J., Jin, R., Zhou, J., Yu, J.X., Jiang, X., Wang, T.: Relational approach for shortest path discovery over large graphs. CoRR, abs/1201.0232 (2012)
9. Pfeiffer III, J.J., Neville, J.: Methods to determine node centrality and clustering in graphs with uncertain structure. In: ICWSM (2011)
10. Kollios, G., Potamias, M., Terzi, E.: Clustering large probabilistic graphs. In: IEEE TKDE (2010)
11. Mayfield, C., Neville, J., Prabhakar, S.: Eracer: a database approach for statistical inference and data cleaning. In: SIGMOD Conference, pp. 75–86 (2010)
12. Potamias, M., Bonchi, F., Gionis, A., Kollios, G.: k-nearest neighbors in uncertain graphs. PVLDB 3(1), 997–1008 (2010)
13. Srihari, S., Chandrashekar, S., Parthasarathy, S.: A framework for SQL-based mining of large graphs on relational databases. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part II. LNCS, vol. 6119, pp. 160–167. Springer, Heidelberg (2010)
14. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: SIGMOD Conference, pp. 567–580 (2008)
15. Zhang, N., Tian, Y., Patel, J.M.: Discovery-driven graph summarization. In: ICDE, pp. 880–891 (2010)
16. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and olap multidimensional networks. In: SIGMOD Conference, pp. 853–864 (2011)
17. Zou, Z., Gao, H., Li, J.: Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In: KDD, pp. 633–642 (2010)