Probabilistic Graph Summarization

by

Nasrin Hassanlou

B.Sc., Sharif University of Technology, 2009

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

University of Victoria

Probabilistic Graph Summarization

by

Nasrin Hassanlou

B.Sc., Sharif University of Technology, 2009

Supervisory Committee

Dr. Alex. Thomo, Supervisor

(Department of Computer Science)

Dr. Venkatesh Srinivasan, Departmental Member

(Department of Same As Candidate)

Supervisory Committee

Dr. Alex. Thomo, Supervisor

(Department of Computer Science)

Dr. Venkatesh Srinivasan, Departmental Member

(Department of Same As Candidate)

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 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.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem and Motivation

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 dissertation, 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 [5, 6]. 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 [5, 6], and we address the problem of summarizing such probabilistic graphs.

Based on the notion of "possible worlds" for probabilistic databases [1–3, 7], a probabilistic graph G defines a set of regular graphs called possible instances. 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. [9,12,16]). 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.

1.2 Our Contributions

The contributions of this dissertation are as follow:

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 syn-

thetic 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.

1.3 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.

1.4 Agenda

The rest of the dissertation is organized as follows.

Chapter 2 formally defines graph summarization, probabilistic graphs, and the summarization of probabilistic graphs. It also gives a detailed account of the notion of possible worlds, which is central to this thesis.

Chapter 3 presents the characterization theorems needed to compute the expected values of aggregations for probabilistic graph summarization.

Chapter 4 describes our algorithm to implement the proposed probabilistic graph summarization method using pure relational technology.

Chapter 5 explains the design of our algorithm on an extended Epinions dataset and presents the experimental results.

Chapter 6 analyzes our algorithm in terms of the efficiency and the effectiveness of summarization.

Chapter 7 presents the SQL statements needed for the implementation of our algorithm.

Chapter 8 contains a restatement of the method and results of the dissertation. It also enumerates avenues for future work and further development of the concept.

Note. This work has been submitted to the *18th International Conference on Database Systems for Advanced Applications (DASFAA'13).*

Chapter 2

Probabilistic Graph Summarization

In this chapter we define the problem precisely. In order to do that, we first define a graph model and we will use this model to represent probabilistic graphs as well. In a graph data structure there are two basic elements:

1. Nodes which are representations of real objects in a dataset (e.g. users in social networks, proteins in protein interaction networks, or resources in RDF databases);

2. Edges which are representations of relationships between objects (or nodes);

More generally graphs can also include node attributes, such as “age”, “affiliation”, “subject”, “protein type”, etc. We consider this more general graph model.

2.1 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 [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*

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

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

functions (w_1, w_2)*, where*

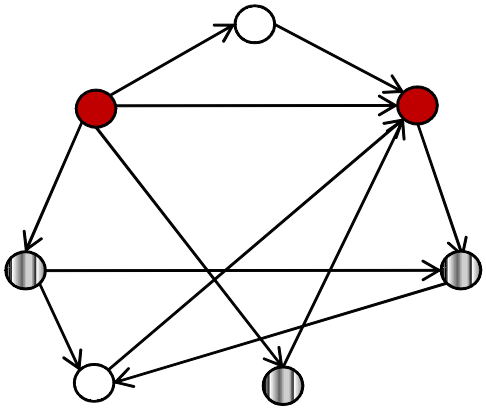
$$
\begin{aligned}
w_1 &: \mathcal{V}_{\mathcal{A}} \longrightarrow \mathbb{N} \\
w_2 &: \mathcal{E}_{\mathcal{A}} \longrightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{N} \\
w_1(g) &= |g| \\
w_2(g', g'') &= (x, y, z), \ \textit{where} \\
x &= |\{v' \in g' : \exists v'' \in g'', \ s.t. \ (v', v'')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

Fig. 2.1.(a) shows a graph containing seven nodes. Consider the color of the nodes to be the grouping attribute. Fig. 2.1.(b) shows the $\mathcal{A}$-graph summarization of the graph in Fig. 2.1.(a) with the corresponding values of the w_1 and w_2 measures.

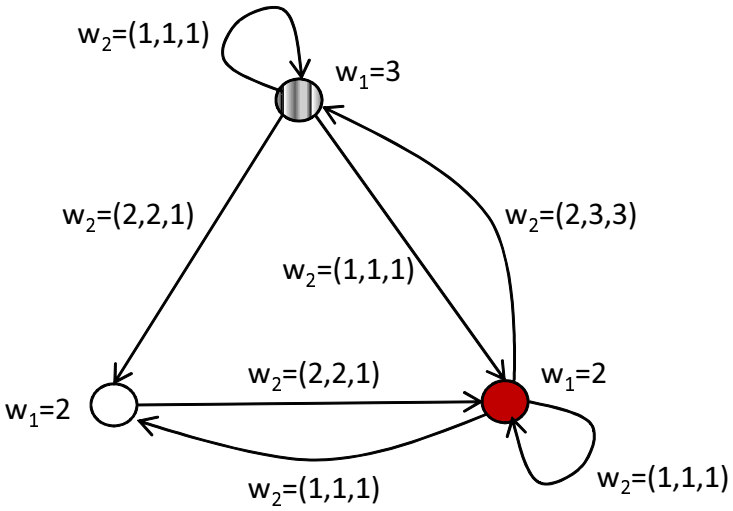
2.2 Probabilistic Graphs

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 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



(a)



(b)

Figure 2.1: (a) A Graph G. (b) The summary graph of G.

does not. The existence probability of each PI is computed as

$$p(PI) = \prod_{e \in E(PI)} p(e) \cdot \prod_{e \notin E(PI)} (1 - p(e)) \tag{2.1}$$

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 a confidence (probability) of 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.

In Figure 2.2, we give a probabilistic graph and the possible instances that have a non-zero probability.

2.3 Summarization of Probabilistic Graphs

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.

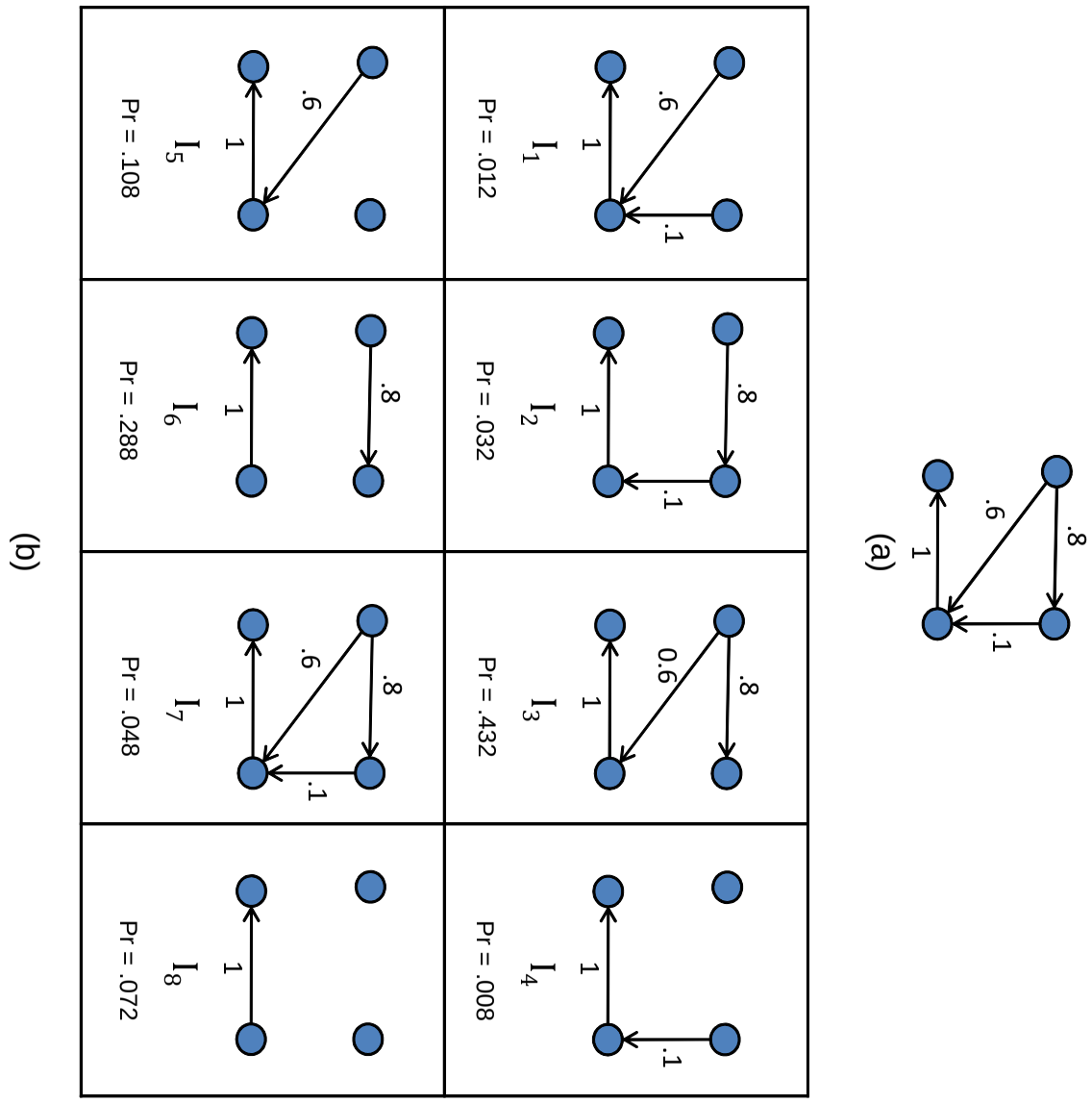


Figure 2.2: (a) A Probabilistic graph. (b) The corresponding possible instances with a non-zero probability.

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 next chapter, 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.

Chapter 3

Characterization Theorems

In this chapter, we give characterization theorems for $E[X]$, $E[Y]$, and $E[Z]$. These theorems serve as the basis for constructing our algorithm in Chapter 4. Let us start with $E[X]$ ($E[Z]$ is similar).

3.1 Characterizing $E[X]$

For this, we first need the following proposition.

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

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

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

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

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

$$E[X] = \sum_{W_{PI} \ni v_1} p(PI) + \cdots + \sum_{W_{PI} \ni v_{|W|}} p(PI). \tag{3.1}$$

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

$$\sum_{W_{PI} \ni v_j} p(PI) + \sum_{W_{PI} \not\ni v_j} p(PI) = 1. \tag{3.2}$$

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 (3.2) 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 (2.1), we can rewrite Equation (3.2) 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

$$\begin{aligned} \sum_{W_{PI} \ni v_j} p(PI(G)) &= \sum_{PI \in \mathcal{PI}(G')} p(PI(G')) \cdot \sum_{S \in 2^{E_{v_j}}, S \neq \emptyset} \left(\prod_{e \in S} p(e) \cdot \prod_{e \in S^c} (1 - p(e)) \right) \\ &= 1 - \prod_{e \in E_{v_j}} (1 - p(e)) \end{aligned} \tag{3.3}$$

and using Equations (3.1) and (3.3) 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. □

3.2 Characterizing $E[Y]$

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

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

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.

□

Note 1. *These characterization theorems can also be proved using Indicator Random Variables*[1] *since any node(edge) can either exist or not.*

[1] http://www.statlect.com/indica1.htm

Chapter 4

Our 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 2.3.

The following algorithm uses the *Nodes* and *Edges* tables illustrated in Table 4.1 and Table 4.2 and returns the w_2 measure in the *Summary* table depicted in Table 4.3. All the steps of our algorithm can be expressed in SQL. Here we give only the plain

language description of the steps and we will provide the SQL statements later in Chapter 7 .

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.

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

Table 4.1: Nodes Information

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

Table 4.2: Edges Information

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}

Table 4.3: Table *Summary*

Chapter 5

Experiments

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.

5.1 Dataset

The real dataset we use for the evaluation is a trust network dataset from *Epinions*[1]. 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 thesis we use the *Extended Epinions* dataset. 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

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

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.

The Extended Epinions dataset is comprised of three files:

File 1 Trust/distrust information

File 2 Article Author information

File 3 Article Ratings information

Each line in the trust/distrust file contains two different user identifiers, the value of trust rate between them, which is 1 (trust) or -1 (distrust), and the date in which the trust value is made. Using this file we create the *Edges* table described in Section 4.

The second file consists of the information of the authorships. Each line of this file shows an article Id, a user identifier as the author of the article, and the type or the subject of the article. Using this file we build the *Nodes* table in which *subjectId* is the only attribute.

Field	Type
userId	int
subjectId	int

Table 5.1: Table *Nodes* (User information)

Field	Type
userId1	int
userId2	int
prob	double

Table 5.2: Table *Edges* (Trust information)

The third file is the rating information of different objects based on user's ideas. Since in our experiment we need a network of users and trust/distrust relationships between them, we do not involve the ratings file in this implementation.

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. The schemas of the *Nodes* and *Edges* tables created from the Epinions dataset are shown in Table 5.1 and Table 5.2, respectively.

5.2 Implementation

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* (Table. 5.3) 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.

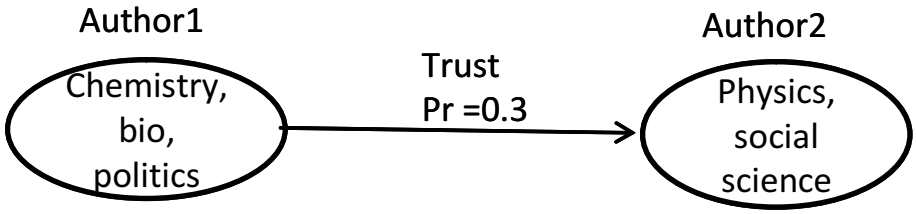
1. We can consider each distinct *userId-subjectId* pair in *Nodes* table as a node in the graph. In such a graph, we also need to consider the trust between the nodes having identical *userId*s. With the assumption that each user trusts completely on his/herself, we connect all the nodes having the 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. 5.1 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 exper-

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

Table 5.3: Table *Joint*



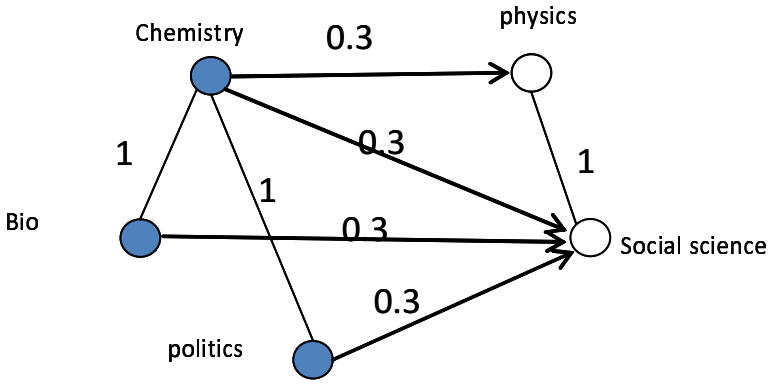


Figure 5.1: Graph generation strategy.

iments 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.

Chapter 6

Evaluation, Analysis and Comparisons

6.1 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.

6.1.1 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.

6.1.2 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 5.2, 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. The experimental results on both graphs are illustrated in Fig. 6.1.

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%

Figure 6.1: The experimental results on the Epinions dataset.

6.1.3 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. 6.2 and Fig. 6.3.

Fig. 6.2 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. Fig. 6.3 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.

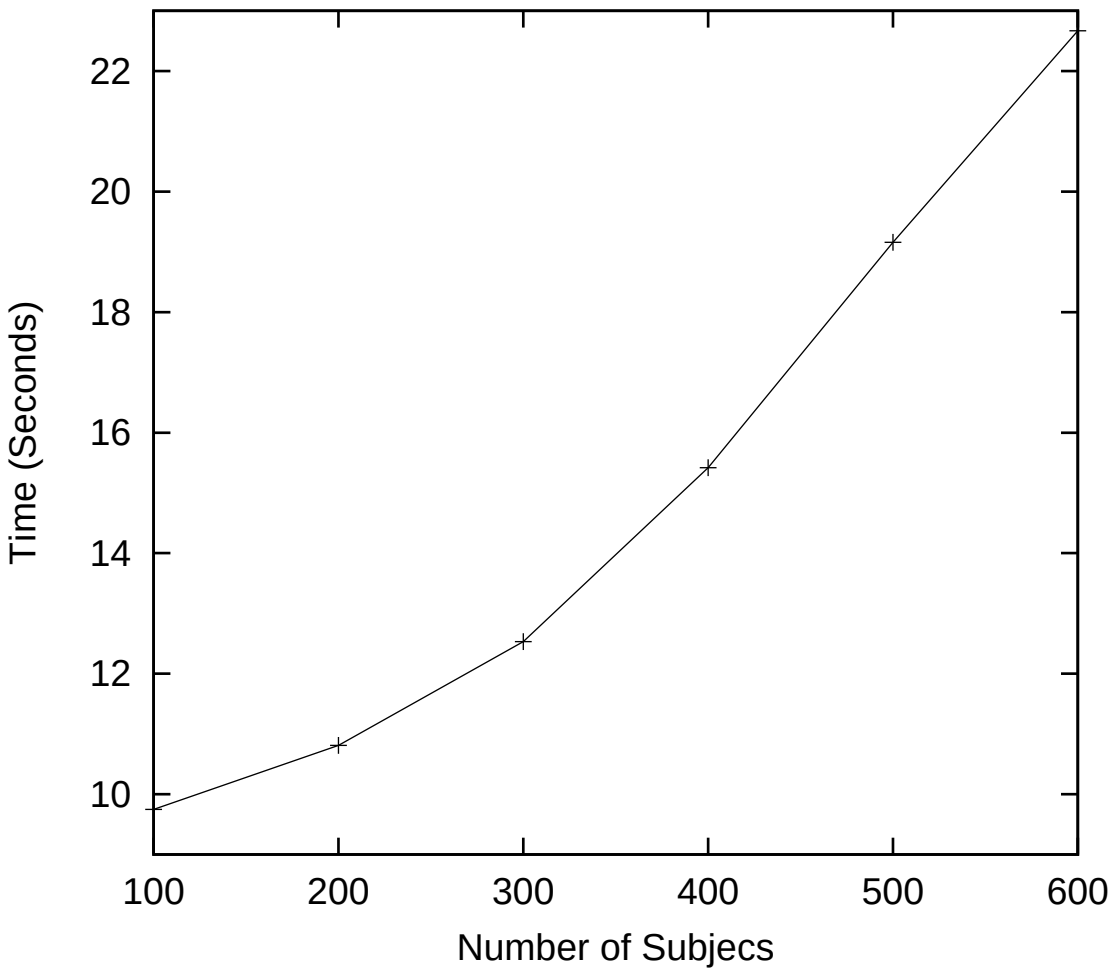


Figure 6.2: Execution time vs. number of subjects

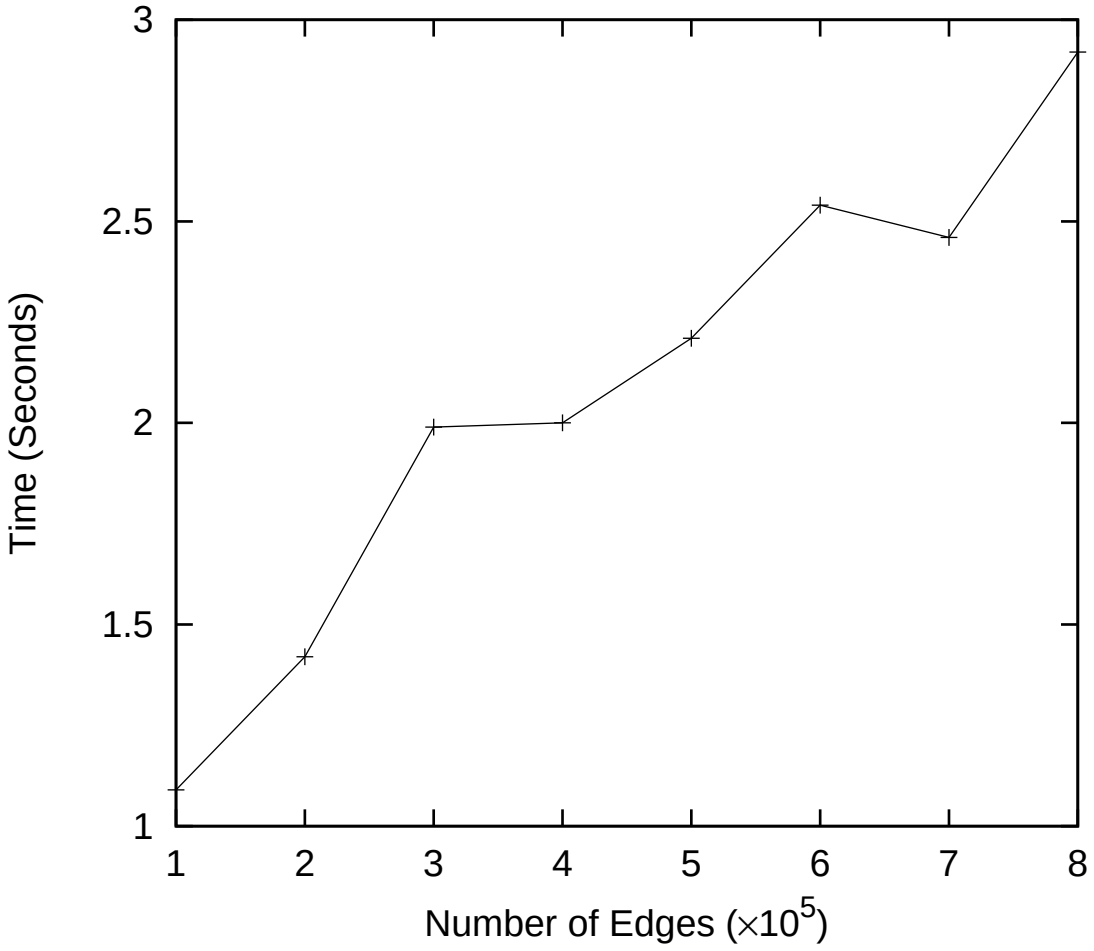


Figure 6.3: Execution time vs. graph size (number of edges).

6.1.4 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. 6.4 and Fig. 6.5 show the compression degree as functions of the number of groups (subjects) and graph sizes, respectively. These figures verify that our algorithm is effective in building a very concise representation of large probabilistic graphs while maintaining 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%.

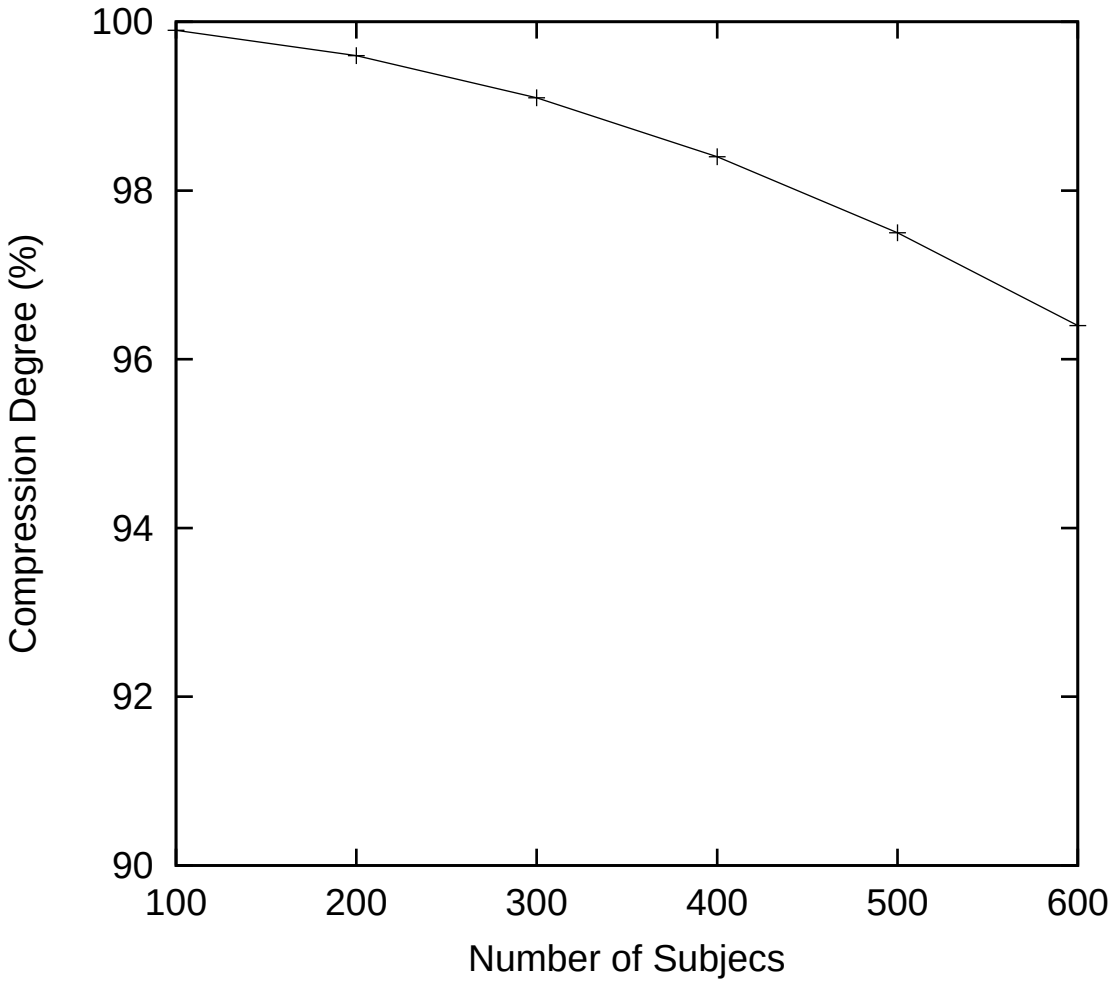


Figure 6.4: Compression degree vs. number of subjects.

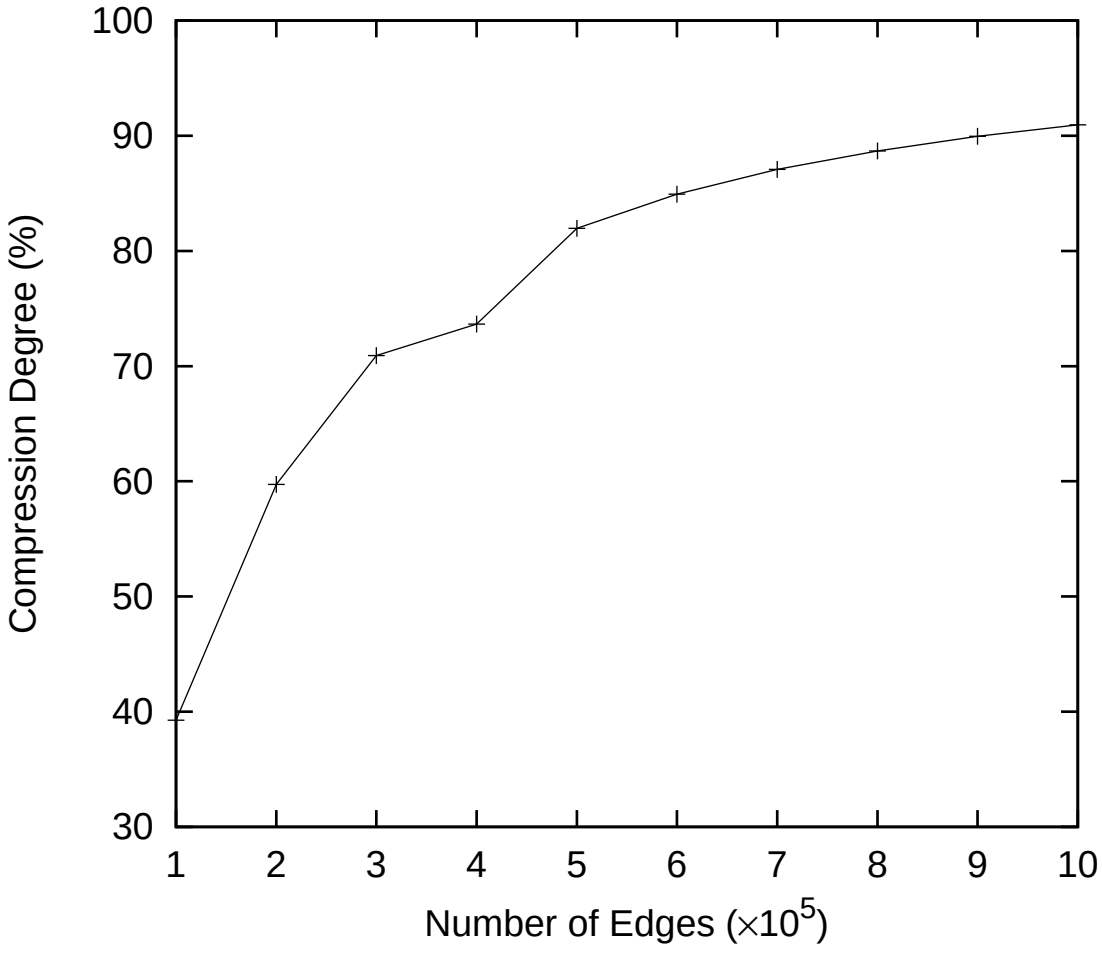


Figure 6.5: Compression degree vs. graph size (number of edges).

Chapter 7

SQL Queries

In this chapter we provide the SQL statements which we have used in order to implement the steps in Algorithm 1.

Q1:
```
CREATE TABLE Nodes (
    userId     int,
    subjectId  int
);
```

Q2:
```
CREATE TABLE Edges(
    userId1    int,
    userId2    int,
    prob       double
);
```

Q3:
```
CREATE TABLE Jointbl(
```

```
  userId1    int,
  subjectId1 int,
  userId2    int,
  prob       double
);
```

The *Jointbl* table is a temporary table which is created by $Q3$ and filled by $Q4$ with the result of joining the *Edges* and *Nodes* table on $userId1$.

Q4:
```
INSERT INTO Jointbl
  SELECT   userId1,
           subjectId1,
           userId2,
           prob
  FROM     Edges, Nodes
  WHERE    Edges.userId1 = Nodes.userId;
```

Q5:
```
CREATE TABLE Joint(
  userId1    int,
  subjectId1 int,
  userId2    int,
  subjectId2 int,
  prob       double
);
```

Table *Joint* is created by $Q4$ and filled by $Q6$ with the result of joining tables *Jointbl* and *Nodes* on $userId2$.

Q6:
```
INSERT INTO Joint
  SELECT   userId1,
           subjectId1,
           userId2,
           subjectId,
           prob
  FROM     Jointbl, Nodes
  WHERE    Jointbl.userId2 = Nodes.userId;
```

Q7:
```
INSERT INTO Joint
  SELECT   A1.userId,
           A1.subjectId,
           A2.userId,
           A2.subjectId,
           1.0
  FROM     Nodes A1, Nodes A2
  WHERE    A1.userId = A2.userId AND
           A1.subjectId <> A2.subjectId;
```

$Q7$ adds to the *Joint* table the edges which are between the nodes with identical $userId$s.

Q8:
```
CREATE TABLE K1(
  userId     int,
  subjectId1 int,
  subjectId2 int,
  product    double
);
```

Q9:
```
CREATE TABLE K2(
  subjectId1 int,
  userId2    int,
  subjectId2 int,
  product    double
);
```

Tables $K1$ and $K2$ are temporary tables that we will use to compute $E[X]$ and $E[Z]$.

Q10:
```
INSERT INTO K1(
  SELECT   userId1,
           subjectId1,
           subjectId2,
           EXP(SUM(LOG(1-prob))
  FROM     Joint
  GROUP BY userId1, subjectId1, subjectId2
```

```
);
```

Since MYSQL does not support PRODUCT function, in order to compute $E[X]$ and $E[Z]$ we use the SUM and LOG functions as in $Q10$ and $Q11$. Function

$$EXP(SUM(LOG(1 - prob)))$$

in the above $SELECT$ statement computes the term $\prod_e (1 - p(e))$ in the equation stated in Theorem 1.

Q11:
```
INSERT INTO K2(
  SELECT   subjectId1,
           userId2,
           subjectId2,
           EXP(SUM(LOG(1-prob))
  FROM     Joint
  GROUP BY subjectId1, userId2, subjectId2
);
```

Q12:
```
CREATE TABLE Summary(
  subjectId1  int,
  subjectId2  int,
  E_X         double
);
```

Q13:
```
INSERT INTO Summary
  SELECT   subjectId1,
           subjectId2,
           SUM(1-product)
  FROM     K1
  GROUP BY subjectId1, subjectId2;
```

$Q13$ performs the last step of the computation of $E[X]$ values and inserts them into the result table *Summary*.

Q14:
```
ALTER TABLE Summary(
  ADD COLUMN E_Z    double
);
```

Q15:
```
UPDATE Summary
  SET E_Z = (SELECT   SUM(1-product)
             FROM     K2
             WHERE    K2.subjectId1 =
                             Summary.subejctId1
                      AND
                      K2.subjectId2 =
                             Summary.subjectId2
             GROUP BY subjectId1, subjectId2
```

```
);
```

$Q15$ performs the last step of the computation of $E[Z]$ values and inserts them into the new column in *Summary* table.

Q16:
```
ALTER TABLE Summary(
  ADD COLUMN E_Y    double
);
```

Q17:
```
UPDATE Summary
  SET E_Y = (SELECT   SUM(prob)
             FROM     Joint
             WHERE    Joint.subjectId1 =
                             Summary.subejctId1
                      AND
                      Joint.subjectId2 =
                             Summary.subjectId2
             GROUP BY subjectId1, subjectId2
            );
```

$Q17$ computes $E[Z]$ values and inserts them into the new column in *Summary* table.

Chapter 8

Conclusions

8.1 Summary

This dissertation 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 high degree of compression.

8.2 Future Work

In this work we assume graphs with deterministic nodes. The problem of summarizing graphs with uncertain nodes and edges is an open problem to solve. Also we considered only one type of edges in a graph. But probabilistic graphs also can contain more than one type of edges. Of course these more general graph summarizations could be straightforward using our proposed method and algorithm, however, they still can be studied more as well.

Bibliography

[1] Serge Abiteboul and Gösta Grahne. Update semantics for incomplete databases. In *VLDB*, pages 1–12, 1985.

[2] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1):158–187, 1991.

[3] Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[4] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, and Andreas Züfle. Probabilistic frequent itemset mining in uncertain databases. In *KDD*, pages 119–128, 2009.

[5] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. Limiting the spread of misinformation in social networks. In *WWW*, pages 665–674, 2011.

[6] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208, 2009.

[7] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[8] Edward H. Frank. Shortest paths in probabilistic graphs. volume 17, pages 583–599, 1969.

[9] Jun Gao, Ruoming Jin, Jiashuai Zhou, Jeffrey Xu Yu, Xiao Jiang, and Tengjiao Wang. Relational approach for shortest path discovery over large graphs. *CoRR*, abs/1201.0232, 2012.

[10] Joseph J. Pfeiffer III and Jennifer Neville. Methods to determine node centrality and clustering in graphs with uncertain structure. In *ICWSM*, 2011.

[11] George Kollios, Michalis Potamias, and Evimaria Terzi. Clustering large probabilistic graphs. IEEE TKDE, 2010.

[12] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. Eracer: a database approach for statistical inference and data cleaning. In *SIGMOD Conference*, pages 75–86, 2010.

[13] Raghotham Murthy, Robert Ikeda, and Jennifer Widom. Making aggregation work in uncertain and probabilistic databases. *IEEE Trans. Knowl. Data Eng.*, 23(8):1261–1273, 2011.

[14] Jian Pei, Ming Hua, Yufei Tao, and Xuemin Lin. Query answering techniques on uncertain and probabilistic data: tutorial summary. In *SIGMOD Conference*, pages 1357–1364, 2008.

[15] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios. k-nearest neighbors in uncertain graphs. *PVLDB*, 3(1):997–1008, 2010.

[16] Sriganesh Srihari, Shruti Chandrashekar, and Srinivasan Parthasarathy. A framework for sql-based mining of large graphs on relational databases. In *PAKDD (2)*, pages 160–167, 2010.

[17] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. Efficient aggregation for graph summarization. In *SIGMOD Conference*, pages 567–580, 2008.

[18] Hannu Toivonen, Fang Zhou, Aleksi Hartikainen, and Atte Hinkka. Compression of weighted graphs. In *KDD*, pages 965–973, 2011.

[19] Ning Zhang, Yuanyuan Tian, and Jignesh M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.

[20] Qin Zhang, Feifei Li, and Ke Yi. Finding frequent items in probabilistic data. In *SIGMOD Conference*, pages 819–832, 2008.

[21] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. Graph cube: on warehousing and olap multidimensional networks. In *SIGMOD Conference*, pages 853–864, 2011.

[22] Zhaonian Zou, Hong Gao, and Jianzhong Li. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In *KDD*, pages 633–642, 2010.

[23] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Mining frequent subgraph patterns from uncertain graph data. *IEEE Trans. Knowl. Data Eng.*, 22(9):1203–1218, 2010.