**Noname manuscript No.**
(will be inserted by the editor)

# A Comparison of Statistical Relational Learning and Graph Neural Networks for Aggregate Graph Queries

**Varun Embar*** · **Sriram Srinivasan*** · **Lise Getoor**

**Abstract** Statistical relational learning (SRL) and graph neural networks (GNNs) are two powerful approaches for learning and inference over graphs. Typically, they are evaluated in terms of simple metrics such as accuracy over individual node labels. Complex *aggregate graph queries* (AGQ) involving multiple nodes, edges, and labels are common in the graph mining community and are used to estimate important network properties such as social cohesion and influence. While graph mining algorithms support AGQs, they typically do not take into account uncertainty, or when they do, make simplifying assumptions and do not build full probabilistic models. In this paper, we examine the performance of SRL and GNNs on AGQs over graphs with partially observed node labels. We show that, not surprisingly, inferring the unobserved node labels as a first step and then evaluating the queries on the fully observed graph can lead to sub-optimal estimates, and that a better approach is to compute these queries as an expectation under the joint distribution. We propose a sampling framework to tractably compute the expected values of AGQs. Motivated by the analysis of subgroup cohesion in social networks, we propose a suite of AQGs that estimate the community structure in graphs. In our empirical evaluation, we show that by estimating these queries as an expectation, SRL-based approaches yield up to a 50-fold reduction in average error when compared to existing GNN-based approaches.

* Equal contribution

V. Embar
Dept. of Computer Science and Engineering
University of California, Santa Cruz, USA
E-mail: vembar@ucsc.edu

S. Srinivasan
Dept. of Computer Science and Engineering
University of California, Santa Cruz, USA
E-mail: ssriniv9@ucsc.edu

L. Getoor
Dept. of Computer Science and Engineering
University of California, Santa Cruz, USA
E-mail: getoor@soe.ucsc.edu

## 1 Introduction

Large realworld graphs in domains such as social media (e.g., friendship and follower graphs), computational biology (e.g., protein interaction networks), and IoT (e.g., sensor networks) often have missing information that needs to be inferred. Making use of the graph, or relational structure, can help immensely in accurately inferring missing values [34, 22]. Statistical relational learning (SRL) [11, 29] and graph neural networks (GNNs) [13, 14, 18, 36, 27] are two powerful machine learning approaches for inferring the missing node labels. These approaches have been shown to be quite effective; however, current literature has largely focused on maximizing *locally decomposable* metrics such as node label accuracy over individual nodes.

Unfortunately, good performance on these locally decomposable metrics does not necessarily translate to accurate estimation of global graph properties. Properties such as node centrality are important in the analysis of graph phenomena such as influence maximization and resilience to attacks, and involve all the nodes and edges in the graph. Global graph properties can be computed using complex graph queries. While many such graph properties have been proposed [33, 38, 6, 30], along with efficient algorithms to estimate them [35, 19, 39, 28, 10], the task of estimating these queries when there is missing information, such as node labels, has not received much attention. In such graphs, we need to combine the tasks of estimating the queries with the inference of missing information such as node labels. These complex queries generally involve many nodes and edges and require joint reasoning over multiple node labels to compute them.

In this work, we introduce the notion of *aggregate graph queries* (AGQs), and argue that researchers should focus more attention on accurately estimating these richer queries. In order to support this, we introduce a suite of useful AGQs[1] that measure subgroup cohesion in graphs [38]. We study the effectiveness of SRL and GNN-based approaches in computing AGQs on graphs with missing node labels. For approaches that infer the *best* possible values for the missing node labels, we propose a *point estimate approach*, where we first infer the missing values, and then compute the query. For approaches that infer the joint distribution over all the missing node labels, we propose an *expectation-based approach* that estimates the query as an expectation over the joint distribution. Further, to compute the expectation tractably using Monte Carlo approximation, we propose a novel sampling approach for probabilistic soft logic (PSL), one of the SRL approaches that we study.

We include a theoretical analysis that shows that the point estimate approach leads to sub-optimal estimates even for simple graphs with just two nodes. We also provide an extensive empirical analysis showing the extent to which this happens over richer queries over realworld data. Further, we analyze the effect of training data size on the performance of these approaches.

The contributions of our paper include:

- We introduce a suite of practical AGQs that measures the key graph property of subgroup cohesion and study the effectiveness of SRL and GNNs in estimating them.
- We show that first inferring the missing values and then estimating the AGQ leads to poor performance.
- We propose a novel Metropolis-within-Gibbs sampling framework, *MIG*, for PSL that is faster than existing SRL samplers by a factor of up to three.
- Through experiments on three benchmark datasets, we show that computing aggregate properties as an expectation outperforms point estimate approaches up to a factor of 50.

[1] We will make the data and code public on acceptance of the paper.

– The runtime experiments show that the proposed MIG approach for PSL is up to 3 times faster than other SRL sampling approaches.

## 2 Background

In this section, we briefly review several important statistical relational learning and graph neural network based approaches.

### 2.1 Statistical Relational Learning

Statistical relational learning(SRL) or statistical relational learning and artificial intelligence (StarAI) methods combine probabilistic reasoning with knowledge representations that capture the structure in the domain [11, 29]. SRL frameworks typically define a declarative probabilistic model or theory consisting of weighted first-order logic rules. The rules can encode probabilistic information about the attributes and labels of nodes, and the existence of edges between nodes. Intuitively, the weight of a rule indicates how likely it is that the rule is true in the world. The higher the weight, the higher is the probability of rule being true.

SRL approaches can be broadly classified into proof-theoretic or model-theoretic approaches based on the inference technique used [9]. In proof-theoretic approaches, a sequence of logical reasoning steps or a proof is generated and this is used to define a probability distribution. Probabilistic logic programs [8] and Stochastic Logic Programs [20] are some popular proof-theoretic approaches. In a model-theoretic approach, the model is used to generate a graphical model or a ground weighted logical theory through a process called *grounding*. Inference is then performed on the ground model. Probabilistic Soft Logic [2], Markov Logic Networks [31] and Bayesian logic programs [17] are some popular model-theoretic based approaches.

#### *2.1.1 Markov Logic Networks*

Markov Logic Networks (MLN) [31, 23, 37] are a notable model-theoretic SRL framework. A MLN induces an undirected graphical model using the set of logical rules by a process known as *grounding*. In grounding, the variables in the rules are replaced with values from the data. The atoms in the rules, where the variables are replaced with the values, are called ground atoms and are modeled as Boolean random variables(RVs) in the undirected graph. The ground rules represent cliques in the graph. Based on the data, some RVs are observed ($X$) and some are unobserved ($Y$). The probability distribution represented by the graphical model over the unobserved random variables $Y$ is given by:

$$P(Y|X;w) = \frac{1}{Z} exp\left(\sum_{i=1}^{N} w_i f_i(X,Y)\right) \tag{1}$$

where $f_i(X,Y)$ is the potential defined using Boolean satisfiability, $w_i$ is the weight, $N$ is the number of ground formulas and $Z$ is the normalization constant. $f_i(X,Y)$ takes the value 1 if the ground formula is satisfied, and 0 otherwise.

#### 2.1.2 Probabilistic Soft Logic

Probabilistic Soft Logic (PSL) [2] is another recently introduced SRL framework. Similar to MLNs, PSL induces an undirected graphical model using the set of logical rules. Unlike MLNs, the ground atoms in PSL are continuous and defined over the range $[0, 1]$. For the potential functions, PSL uses a continuous relaxation of Boolean logic, which results in hinge functions instead of Boolean satisfiability. The probability distribution represented by the graphical model over the unobserved random variables $Y$ is given by:

$$P(Y|X; w) = \frac{1}{Z} exp\left(-\sum_{i=1}^{N} w_i \phi_i(X, Y)\right) \tag{2}$$

where $\phi_i(X, Y)$ is the potential defined using Lukasiewicz logic, $w_i$ is the weight, $N$ is the number of ground formulas and $Z$ is the normalization constant. The potential function $\phi_i(X, Y)$ takes the form of a hinge and makes the MAP inference in PSL convex.

### 2.2 Graph Neural Networks

GNNs build on top of neural networks to learn non-linear representation for each node in a graph. These node representations are learned by encoding information about the local graph structure [18, 36], edge labels [32], adjacent node labels [27, 24] and external domain knowledge [40, 26, 15]. GNNs can be broadly classified into non-probabilistic and probabilistic approaches based on whether they explicitly model the joint distribution.

Non-probabilistic approaches learn a non-linear representation for each node in a graph using a neural network and use them to classify nodes independently. These approaches do not explicitly model the joint probability distribution. Graph Convolutional Networks (GCNs)[18] , Relational GCN[32], Graph Attention Networks (GATs) [36] are some popular GNN approaches belonging to this category.

Recently, several probabilistic approaches have been proposed that learn a joint distribution over the unobserved node labels in a graph. The distribution is parameterized using a graph neural network. GMNN [27], ExpressGNN [40], pGAT [15], pLogicNet [26] and Column Networks [24] are some popular probabilistic approaches. To make the inference tractable, approaches such as Qu et al. [27], Qu and Tang [26], and Zhang et al. [40] use variational expectation maximization [21]. In these approaches the joint distribution is approximated with a mean-field variational distribution that is more tractable for inference. Pham et al. [24] employ an approximate, multi-step, iterative method similar to *stacked learning*, where the intermediate marginal probabilities for a node are used as relational features in the next step.

#### 2.2.1 Graph Convolutional Networks

Graph Convolutional Network (GCN) [18] is a popular non-probabilistic GNN approach. GCNs iteratively update the representation of each node by combining each node's representation with its neighbors' representation. The propagation rule to update the hidden representation of a node is given by:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}H^{(l)}W^{(l)}\right) \tag{3}$$

where $H^{(l)}$ denotes the representation at layer $l$, $\tilde{D}$ represents the degree matrix, $\tilde{A}$ represents the adjacency matrix with self-loop, $W$ represents the weights, and $\sigma$ denotes an activation function, such as the ReLU. The final representations are fed into a linear softmax layer classifier for label prediction.

#### *2.2.2 Graph Attention Networks*

Graph Attention Networks (GATs) [36] are similar to GCNs and use self-attention while combining the representation of each node with its neighbors. This allows the model to assign different weights to each of its neighbors' representations. The propagation rule for GAT is given by:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}} \alpha_{ij} h_i^{(l)} W\right) \tag{4}$$

where $h_i^{(l)}$ is the representation of node $i$ at layer $l$, $W$ is the weight matrix, $\mathcal{N}$ is the set of neighbors and $\alpha$ is the attention weights.

#### *2.2.3 Graph Markov Neural Networks*

Graph Markov Neural Networks (GMNNs) [27] is a recently introduced probabilistic approach. GMNNs build on graph neural networks such as GCNs or GATs by adding a second neural network to capture the latent dependencies in the inferred data. The pair of neural networks are trained using a variational EM algorithm. In the E-step, the object representations are learned by the first neural network. In the M-step, the latent dependencies are learned by the other neural network.

## 3 Problem Definition

Consider a graph $G = (V, \mathcal{E})$, where $V$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node $i \in V$ is associated with a set of attributes denoted by $\mathbf{a}_i$ and a label denoted by $c_i \in \{1, \ldots, K\}$. All nodes and edges of the graph are observed and the node labels are partially observed. The set of observed node labels is denoted by $C_o$, unobserved node labels by $C_u$, and $C_o \cup C_u = C$. As an example, consider a computer science citation graph.

*Example 1* In a computer science citation graph $G_c$, the nodes $V_c$ represent computer science documents and the edges $E_c$ represent citation links between these documents. The documents in the graph can belong to several categories such as AI, Systems, Compilers and Databases. The document category is represented as a node labels $C_c$. The contents of the document $i$ such as the tokens in the abstract are represent by the node attributes $a_i$. The documents with observed categories correspond to $C_o$. Documents with categories that need to be inferred are correspond to $C_u$.

**Definition 1 (Graph queries)** A graph query $GQ$ is a Boolean expression over nodes, edges and node labels.

The most common form of graph queries are those that define a subgraph pattern. A graph query $GQ$, when evaluated on a graph $G$ with node labels $C$, returns a set of subgraphs that satisfy the Boolean expression and is denoted by $GQ(G, C)$. We refer to graph queries that involve a single node or an edge as *simple graph queries*, and queries that involve multiple nodes and/or edges as *complex graph queries*.

*Example 2* For the citation graph in Example 1, we might want to infer how dense the citation links are within the categories. The $GQ$ that returns the set of all citation links between documents that belong to the same category is given by:

$$GQ_1 = \{\forall_{(i,j)\in V_c \times V_c}(e_{ij} \wedge c_i = c_j)\}$$

The Boolean expression is true when a pair of documents have a citation link between them and also belong to the same category.

**Definition 2 (Aggregate graph queries)** Aggregate graph queries (AGQs) are a class of graph queries that compute an aggregate function on the set of subgraphs that match the Boolean expression, i.e., an AGQ $Q(G, C) = Agg(GQ(G, C))$ where $Agg$ is an aggregate function.

For example, $Count$ is an aggregate function that returns the number of subgraphs in the set. AGQs can be considered as a mapping from the graph $G$ and the node labels $C$ to a real number, i.e., $Q : (G, C) \rightarrow \mathbb{R}$.

*Example 3* For the citation graph in Example 1, one way to summarize the density of citations within the categories is to count the number of such citations. The aggregate graph query representing the number of citation links between documents that belong to the same category is given by:

$$Q_1 = Count_{(i,j)}(\{\forall_{(i,j)\in V_c \times V_c}(e_{ij} \wedge c_i = c_j)\})$$

**Definition 3 (Aggregate graph query estimation)** Given a graph $G$, the observed and unobserved node labels $C_o$, $C_u$, and an aggregate graph query $Q$, the task of *aggregate graph query estimation* is to compute the value of $Q(G, C_o, C_u)$.

*Example 4* For the citation graph in Example 1, the aggregate graph query in Example 3 cannot be computed directly due the missing document categories in $C_u$. We need to first infer the category labels before computing the AGQ.

## 4 Aggregate Graph Queries

In this section we motivate and introduce several complex AGQs that are useful in analyzing the community structure (also called cohesive subgroups) in graphs. Analyzing the community structure of a graph is necessary to understand the social forces operating in a network and is widely used in social sciences, particularly in social psychology and sociology [38]. One of the approaches to quantitatively measure this is the nodal degree approach that computes various statistics regarding the membership of a node and its adjacent nodes to various communities.

We define five different AGQs that can be used to quantitatively measure the community structure of a graph. These queries compute statistics of the entire graph using node and edge frequencies, between nodes that belong to the same category, across different categories and also relative frequency between and across categories. We also include an AGQ that measures the accuracy of the predicted node labels to show that AGQs can also capture the traditional locally decomposable metrics. The queries are also of varying *complexity*, where the complexity is the number of nodes jointly involved in the query. Query Q0 involves a

single node and queries Q1 and Q2 involve two nodes. Queries Q3 to Q5 are more complex and involve all the neighbors of a node. Q1 and Q2 are based on edge frequencies and Q3 to Q5 are based on node label frequency. We illustrate these queries using the citation graph introduced in Example 1.

**[Q0]: Accuracy:** This query measures the number of documents with the correct categories assigned to them. It is a locally decomposable query and is given by:

$$Q0 = Count_{c_i}(c_i = c_i^*)$$

where $c_i^*$ is the ground truth label.

**[Q1]: Edge Cohesion:** This query measures the number of citation links between documents $i, j$ that belong to the same category. It is given by:

$$Q1 = Count_{e_{ij}}(e_{ij} \in \mathcal{E} \wedge c_i = c_j)$$

A citation graph with a small number of large, tight-knit categories tends to have a large number of citations between documents of the same categories.

**[Q2]: Edge Separation:** This query measures the number of citation links between documents $i, j$ that belong to the different category. It is given by:

$$Q2 = Count_{e_{ij}}(e_{ij} \in \mathcal{E} \wedge c_i \neq c_j)$$

A citation graph with large number of small communities tends to have a large number of citations between documents across different categories.

**[Q3]: Diversity of Influence:** This query measures the number of documents $i$ in the graph that are connected to at least half of the different document categories. It is given by:

$$Q3 = Count_i\Big(Count_j\big(\{c_j \mid \forall_{j=1}^{n}\big(e_{ij} \wedge c_i \neq c_j\big)\}\big) \geq \frac{K}{2}\Big)$$

The inner $Count$ computes the number of distinct document categories that a document $i$ is cited by and the outer $Count$ computes the number of documents that are connected to at least half of the document categories. This query is computes the number of $k$-core nodes in the graph where $k$ is set to half of the number of categories.

**[Q4]: Exterior Documents:** This query measures the number of documents $i$ that have more than half of its neighbors belonging to categories other than the documents category, i.e.,

$$Q4 = Count_i\Big(\Big(Count_j\big(\{c_j \mid e_{ij} \in \mathcal{E} \wedge c_i \neq c_j\}\big)\Big) > \frac{Count_j(\{c_j | e_{ij} \in \mathcal{E}\})}{2}\Big)$$

The inner counts compute the number of adjacent documents with different labels and adjacent documents respectively, and the outer count computes the number of such documents in the graph. This AGQ helps measure the monophily in the graph as given by [5].

**[Q5]: Interior Documents:** This query measures the number of documents $i$ that have more that half of its neighbors belonging to the same category as the document. It is given by:

$$Q5 = Count_i\Big(\Big(Count_j\big(\{c_j \mid e_{ij} \in \mathcal{E} \wedge c_i = c_j\}\big)\Big) > \frac{Count_j(\{c_j | e_{ij} \in \mathcal{E}\})}{2}\Big)$$

Similar to the previous query, the inner counts compute the number of adjacent documents with the same label and adjacent documents respectively and the outer count computes the number of such documents in the graphs.

## 5 Estimating Aggregate Graph Queries

In this section, we first introduce the point-estimate approach to estimate the AGQs. For models that explicitly learn the joint distribution, we also propose an expectation-based approach.

### 5.1 Point Estimation Approach

One approach for aggregate graph query estimation is to impute the *locally best* possible value for the unobserved node labels $C_u$ and then compute the AGQ. Here, we first learn a model by minimizing a locally decomposable objective function, such as the likelihood of node labels or a loss function defined over the labels, using the graph $G$, node attributes $\mathbf{a}_i$ and observed node labels $C_o$, and impute values for the unobserved node labels $C_u$ using the learned model. We refer to this approach as a *point estimate approach*. The point estimate approach is formally defined as follows:

**Definition 4 (Point estimate approach)** Given an aggregate graph query estimation task, the point estimate approach estimates $Q$ by first imputing the values for $C_u$ (denoted by $\hat{C}_u$) and then computes a value for $Q$, i.e., estimate $\hat{Q} = Q(G, C_o, \hat{C}_u)$.

Non-probabilistic GNN approaches such as GCNs and GATs model the marginal distribution for each unobserved node label and impute labels using the mode of the distribution. SRL approaches such as PSL and MLNs, and probabilistic GNN approaches such GMNNs model the joint distribution over all unobserved node labels and impute node labels using the mode or the mean of the joint distribution.

### 5.2 Expectation-Based Approach

Another approach for aggregate graph query estimation is to define a joint probability distribution over the unobserved node labels and take the expectation of the aggregate graph query $Q$ over the joint distribution. We refer to this approach as the *expectation-based approach*. Since the range of the aggregate graph query $Q$ is $\mathbb{R}$, the expectation is well-defined. The expectation-based approach is formally defined as follows:

**Definition 5 (Expectation-based approach)** Given an aggregate graph query estimation task, the expectation-based approach estimates $Q$ as an expectation over the joint distribution of the unobserved node labels $C_u$, i.e., estimate $\hat{Q} = E_{p(\hat{C}_u|G,C_o)}[Q(G, C_o, \hat{C}_u)]$.

AGQs can be computed as an expectation using approaches that explicitly model and perform inference on the joint distribution over the unobserved node labels. Non-probabilistic GNNs such as GCN and GAT do not model the joint distribution and cannot be used to compute the expected value. SRL approaches such as PSL and MLN and probabilistic GNNs such as GMNNs and ExpressGNNs model the joint distribution explicitly. However, computing the expectation analytically for these approaches is challenging due the intractability

of the integration in the expectation. The expectation can be approximated using Monte Carlo methods by sampling from the distribution.

To make the inference tractable, approaches such as GMNN and ExpressGNN replace the joint distribution with a mean-field variational distribution. The mean-field approximation breaks dependencies between the node labels in the joint distribution. As an example, Pham et al. [24] use an iterative approach to estimate the joint distribution. The final layer of the GNN estimates the marginal node label probabilities using the labels of its neighbors from the previous iteration. Sampling from each node's marginal distribution independently or from a mean-field distribution results in samples with limited dependence between adjacent node labels. This makes computing expectation of the AGQs using Monte Carlo approximation challenging for these approaches.

## 6 Analysis of the Estimation Approaches

In the previous section, we proposed two approaches to estimate the AGQs. In this section, we analyze the two approaches by estimating the value of the AGQ introduced in Example 3 on a graph consisting of two nodes. We use stochastic block models (SBMs) [16, 4, 1] as a generative model for the graph. SBMs are a popular class of generative models used extensively in statistics, physics, and network analysis. SBMs take as input the number of nodes $n$, a $K$ dimensional vector ($\gamma$), where $\gamma_k > 0$ and $\sum_{k=1}^{K} \gamma_k = 1$, representing the fraction of nodes that belong to category $k$, and a $K \times K$ symmetric matrix ($\Pi$) whose elements $\Pi_{k_1 k_2}$ represent the probability of edge between two nodes belonging to categories $k_1, k_2$. We assume that at least one of the $\Pi_{k_1 k_2}$ where $k_1 \neq k_2$ is non-zero, i.e., there is a non-zero probability of observing an edge across nodes belonging to different categories.

The SBM generative process for a graph $G = (V, \mathcal{E})$ with node labels $C$ is:

$$c_i \sim Multinomial(\gamma) \quad \forall i \in V$$
$$e_{ij} \sim Bernoulli(\Pi_{c_i c_j}) \quad \forall i, j \in V \times V$$

Consider a graph $G$ with two nodes $i, j$ connected by an edge $e_{ij}$. The joint distribution for the node labels $c_i, c_j$, under the SBM, is given by:

$$p(c_i, c_j | e_{ij}) = \frac{p(c_i)p(c_j)p(e_{ij}|c_i, c_j)}{p(e_{ij})} \tag{5}$$

We now show that even for the simple aggregate graph query introduced earlier that counts the number of adjacent nodes belonging to the same category, the point estimate approach leads to large errors.

**Theorem 1** *For a graph $G$ generated using SBM with two nodes $i, j$ and an edge between them, the point estimate approach cannot minimizes the expected mean squared error for the AGQ $Q = Count_{(i,j)}(\{\forall_{(i,j)\in V\times V}(e_{ij} \wedge c_i = c_j)\})$*

*Proof* The expected MSE for $Q$ is given by $E[(Q - \hat{Q})^2]$. We know that, expected MSE is minimized when $\hat{Q} = E[Q]$, i.e.,

$$argmin_{\hat{Q}} E[(Q - \hat{Q})^2] = E[Q] \tag{6}$$

Since the query $Q$ takes the value 1 when both nodes $i, j$ have the same label and 0 otherwise, the expected value for the query $Q$, $E[Q]$, is equal to the probability of $i, j$ having the same node label. Thus $E[Q]$ is given by:

$$E[Q] = \sum_{k \in C} \gamma_k^2 \Pi_{kk} \tag{7}$$

Since $\sum_{k_1 \in C} \sum_{k_2 \in C} \gamma_{k_1} \gamma_{k_2} \Pi_{k_1 k_2} = 1$ and at least one of the terms $\gamma_{k_1} \gamma_{k_2} \Pi_{k_1 k_2} \neq 0$ when $i \neq j$, $\sum_{k \in C} \gamma_k^2 \Pi_{kk}$ lies strictly between 0 and 1. Thus $0 < E[Q] < 1$.

The point estimate approach imputes labels for the nodes $i, j$ and estimates $\hat{Q}$ to be 1 if the imputed values $i, j$ belong to the same category and 0 otherwise. Since the point estimate approach estimates $\hat{Q}$ to be either 0 or 1, no point-estimate approach can minimize the expected MSE.

The above theorem shows that even for simple queries, the point estimate approach leads to sub-optimal estimation. We show in the empirical evaluation that this is true also for more complex queries on larger graphs. Further, from Equation 6, we know that an optimal estimate can be obtained using an expectation-based approach which directly computes the expectation of AGQs under the joint distribution.

## 7 Expectation-Based Approach for PSL

In the previous section, we showed that point estimate approaches do not obtain optimal estimates. Better estimates of AGQs can be obtained by computing the expectation of AGQs over the joint distribution. Computing the expectation analytically for SRL approaches may not always be possible due the intractability of the integration in the expectation. One way to overcome this problem is to use Monte Carlo methods to approximate the expectation by sampling from the distribution. The expectation can be approximated as follows:

$$Q(G, C_u, C_o) \approx \frac{1}{S} \sum_{j=1}^{S} Q(G, C_o, C_{u(j)}) \tag{8}$$

where $S$ is the number of samples and $C_{u(j)}$ are samples drawn from the distribution $p(C_u|G, C_o)$.

Gibbs sampling [12] is a type of MCMC sampling approach that generates samples from the joint distribution by iteratively sampling from the conditional distribution of each RV. For MLNs, where conditional distributions follow a binomial distribution, approaches such as MC-SAT have been proposed [25] that combine MCMC and satisfiability.

In PSL the unobserved node labels $C_u$ are modeled as unobserved RVs $Y_{0:m}$ where $m$ is the number of nodes with unobserved labels. The conditional distribution for a RV $y_i \in Y$ conditioned on all other variables $X$, $Y_{-i}$ is given by:

$$p(y_i|X, Y_{-i}) \propto exp\{-\sum_{r=1}^{N_i} w_r \phi_r(y_i, X, Y_{-i})\} \tag{9}$$

where $N_i$ is the number of groundings in which variable $y_i$ participates. The above distribution neither corresponds to a standard named distribution nor has a form amenable to techniques such as inversion sampling. Hence, it is non-trivial to generate samples from the conditional distributions of PSL.

To address this challenge, unlike a previous hit-and-run based sampling approach [3], we propose a simple but effective approach for samping from the joint distribution. We overcome the challenge of sampling from the conditional by incorporating a single step of a Metropolis algorithm within the Gibbs sampler (also called Metropolis-within-Gibbs [12]). The algorithm for our proposed approach (MIG sampler) is given in Algorithm 1. For each RV $y_i$, we first sample a new value $y_i'$ from a uniform distribution $Unif(0,1)$ and compute the acceptance ratio $\alpha$ given by:

$$\alpha = \frac{exp\{-\sum_{r=1}^{N_i} w_r \phi_r(Y_{0:i-1}, y_i', Y_{i+1:m}, X)\}}{exp\{-\sum_{r=1}^{N_i} w_r \phi_r(Y_{0:i-1}, y_i, Y_{i+1:m}, X)\}} \tag{10}$$

We then accept the new value $y_i'$, as a sample from the conditional with a probability proportional to $\alpha$. We ignore the first $b$ samples as burn-in. Further, for faster convergence we start the sampling from the MAP state of PSL.

**Algorithm 1** MIG Sampler for PSL

**Input:** Unobserved RVs $Y$, Observed RVs $X$, $N$ ground rules, # of iterations T, burn-in period $b$.
**Output:** Set of samples $\mathcal{S}$
$Y^{(0)} \leftarrow argmax_Y p(Y|X)$ // Initialize $Y^{(0)}$ to MAP state
**for** $t$ from 1 to $T$ **do**
  **for** $i \in 1$ to $m$ **do**
    $y' \sim Unif(0,1)$
    $\alpha = \frac{exp\{-\sum_{r=1}^{N_i} w_r \phi_r(Y_{1:i-1}^{(t)}, y', Y_{i+1:m}^{(t-1)}, X)\}}{exp\{-\sum_{r=1}^{N_i} w_r \phi_r(Y_{1:i-1}^{(t)}, y_i^{(t-1)}, Y_{i+1:m}^{(t-1)}, X)\}}$
    **if** $Unif(0,1) < \alpha$ **then**
      $Y_i^{(t)} = y'$ //accept with probability $\alpha$
    **else**
      $Y_i^{(t)} = Y_i^{(t-1)}$
    **end if**
  **end for**
  **if** $t > b$ **then**
    $\mathcal{S} = \mathcal{S} \cup Y^{(t)}$ // Consider samples after burn-in
  **end if**
**end for**
Return $\mathcal{S}$

## 8 Empirical Evaluation

In this section we analyze the performance of SRL and GNN-based approaches on AGQs. We answer the following research questions:

- RQ1: How does the performance of expectation-based approaches compare with point estimate approaches?
- RQ2: How does the performance vary with the amount of labeled data?
- RQ3: What is the trade-off in performance between estimating aggregate graph queries and locally decomposable evaluation metrics such as accuracy?
- RQ4: What is the runtime performance of these approaches?

### 8.1 Experimental Setup and Datasets

| Dataset | #Categories | #Nodes | #Edges | #Attributes |
|---|---|---|---|---|
| Cora | 7 | 2708 | 5429 | 1433 |
| Pubmed | 3 | 19717 | 44338 | 500 |
| Citeseer | 6 | 3327 | 4732 | 3703 |

Table 1: Statistics for the three datasets: Cora, Pubmed and Citeseer.

We consider three benchmark citation datasets for node classification: Cora, Pubmed and Citeseer [34]. The nodes correspond to documents, the edges correspond to citations, the attributes correspond to words in the document, and the categories correspond to areas of research. The statistics for these datasets are given in Table 1. We assume all the attributes $a_i$ and citations $\mathcal{E}$ are observed, while the categories $C$ are only partially observed. We generate five folds consisting 500 nodes for training, 100 nodes for validation (600 observed node labels) and use the remaining as test nodes. All approaches are given access to observed node labels during training and metrics are evaluated on the test data.

**SRL approaches**: For both MLNs and PSL, we extend the model defined in Bach et al. [2] to incorporate node attributes. We use a bag-of-words representation for the node attributes. We train a logistic regression model(LR) to predict the node labels using the bag-of-words vectors. For each node, we consider the category with the highest probability as the LR prediction. Since LR does not need early stopping, we use all the observed node labels to train the model. We set the L2 regularizer weight to 0.001.

The model contains the following rules:

$$w_1 : \textsc{HasCat}(\texttt{A}, \texttt{Cat}) \wedge \textsc{Link}(\texttt{A}, \texttt{B}) \rightarrow \textsc{HasCat}(\texttt{B}, \texttt{Cat})$$
$$w_2 : \textsc{LR}(\texttt{A}, \texttt{Cat}) \rightarrow \textsc{HasCat}(\texttt{A}, \texttt{Cat})$$

The predicate $\textsc{HasCat}(\texttt{A}, \texttt{Cat})$ is true if document $\texttt{A}$ belongs to category $\texttt{Cat}$ and predicate $\textsc{Link}(\texttt{A}, \texttt{B})$ is true if documents $A$ and $B$ have an citation link between them. The model incorporates the logistic regression predictions using the predicate $\textsc{LR}(\texttt{A}, \texttt{Cat})$, which is true if LR predicts category $Cat$ for document $A$. For MLNs, we include a functional constraint that prevents a document from having multiple categories set to true. For PSL, we include a highly weighted rule that states that the truth values across all categories must sum to 1 for a node. We learn the rule weights using MC-SAT for MLN and maximum likelihood estimation for PSL using training and validation data.

The different SRL-based approaches that we consider are:

- **LR**: We compute the AGQs using the predictions of logistic regression trained on the node attributes. This is a point estimate approach.
- **MLN-MAP**: This is a point estimate approach that computes the mode of the joint distribution defined by the MLN model. We use the MaxWalkSAT algorithm implemented in the Tuffy framework [23].
- **MLN-SAM**: This is an expectation-based approach that estimates the AGQ as an expectation over the distribution defined by the MLN model. We generate 1000 samples using the MC-SAT algorithm, discard the first 500 samples as burn-in samples and randomly choose 100 samples from the 500 (to ensure minimal correlation) and use Monte Carlo approximation to compute AGQs.

- **PSL-MAP**: This is a point estimate approach that computes the mode of the distribution defined by the PSL model. We use the ADMM algorithm implemented in the PSL framework [2].
- **PSL-SAM**: This is an expectation-based approach that estimates the AGQs as an expectation over the distribution defined by the PSL model. Similar to MLN-SAM, we generate 1000 samples are generated using the proposed *MIG-sampler* introduced in Algorithm 1, discard the first 500 samples as burn-in samples and randomly choose 100 samples from the 500 (to ensure minimal correlation) and use Monte Carlo approximation to compute AGQs.

**GNN based approaches:** These are point estimate approaches that use the node representations to infer node labels. These models are trained using the training and validation data where the validation data is used to perform early-stopping. We used the code provided by the authors of the respective papers. For all three approaches we performed hyperparameter tuning and found that hyperparameters provided by authors performed best. The different GNN-based approaches we consider are:

- **GCN**: This approach uses the representation computed using a graph convolutional network [18].
- **GAT**: This approach uses the representation computed using a graph attention network [36].
- **GMNN**: This approach uses the representation computed using a Markov neural network introduced recently [27].

**Metric:** In Subsection 8.2 and Subsection 8.3, we evaluate the performance on the AGQs (Q0 to Q5) using the relative query error ($QE$) and in Subsection 8.4 we evaluate the categorical accuracy (Acc) and homophily error. $QE$ is computed using: $QE = \frac{|\hat{Q}-Q|}{Q}$ where $Q$ is the true value of the query and $\hat{Q}$ is the predicted value. We evaluate the overall performance of each method by computing the average $QE$ over all queries denoted by $AQE$. For homophily error we use the homophily measure $H$ defined in Dandekar et al. [7] and compute error similar to $QE$ by computing the absolute difference w.r.t. true $H$ computed using the true labels. Homophily measure $H$ is given by $H = \frac{|e \in S|}{|e \in NS|} = \frac{Q1}{Q2}$ where $S$ and $NS$ as sets of edges such that the nodes have the same category and not the same category, respectively. All reported metrics are averaged across five folds.

## 8.2 Performance on AGQs

In this section we answer RQ1 by computing the $QE$ for the AGQs proposed in Section 4. The $QE$ and $AQE$ for all datasets are shown in Table 2. We observe that PSL-SAM has the lowest or the second lowest error across most of the non-decomposable queries $(Q1 - Q5)$. GNNs perfrom better on accuracy $(Q0)$ which is a locally decomposable query. In Citeseer, although LR performs worse on locally decomposable AGQs such a $Q0$, it performs better on other AGQs. This is due to the sparse nature of the graph, where non-collective approaches perform better. Among collective approaches, PSL-SAM outperforms all other approaches. GNNs have a high query errors for non-decomposable AGQs. This is consistent with our theoretical analysis.

Among the queries, we observe that Q1 and Q5 have lower error compared to the other queries for all the methods. Both Q1 and Q5 estimate node pairs that are adjacent and have the same category. These are easier to estimate as these nodes typically lie at the center of

(a) Query error for Cora

| Methods | Q0 | Q1 | Q2 | Q3 | Q4 | Q5 | $AQE$ |
|---|---|---|---|---|---|---|---|
| GCN | 0.143 | 0.0756 | 0.323 | 0.281 | 0.768 | 0.363 | 0.325 |
| GAT | 0.159 | 0.076 | 0.326 | 0.281 | 0.729 | 0.361 | 0.322 |
| GMNN | **0.142** | 0.081 | 0.348 | 0.254 | 0.754 | 0.367 | 0.324 |
| LR | 0.324 | 0.320 | 1.371 | 1.854 | 0.993 | 0.401 | 0.709 |
| MLN-MAP | 0.188 | **0.011** | 0.110 | 0.136 | 0.529 | 0.268 | 0.207 |
| PSL-MAP | 0.162 | 0.027 | 0.116 | 0.063 | 0.060 | 0.034 | 0.077 |
| MLN-SAM | 0.170 | 0.021 | 0.092 | 0.068 | 0.074 | 0.035 | 0.076 |
| PSL-SAM | 0.170 | 0.015 | **0.066** | **0.005** | **0.040** | **0.022** | **0.053** |

(b) Query error for Pubmed

| Methods | Q0 | Q1 | Q2 | Q3 | Q4 | Q5 | $AQE$ |
|---|---|---|---|---|---|---|---|
| GCN | **0.152** | 0.129 | 0.524 | 2.732 | 0.737 | 0.126 | 0.733 |
| GAT | 0.168 | 0.144 | 0.583 | 2.364 | 0.764 | 0.132 | 0.692 |
| GMNN | 0.157 | 0.134 | 0.545 | 2.581 | 0.743 | 0.127 | 0.714 |
| LR | 0.219 | 0.126 | 0.513 | 6.342 | 0.712 | 0.120 | 1.33 |
| MLN-MAP | 0.205 | 0.075 | 0.362 | 3.613 | 0.435 | 0.080 | 0.795 |
| PSL-MAP | 0.170 | 0.016 | 0.064 | 4.259 | **0.007** | **0.001** | 0.752 |
| MLN-SAM | 0.223 | 0.037 | 0.070 | 0.057 | 0.051 | 0.007 | 0.073 |
| PSL-SAM | 0.171 | **0.009** | **0.038** | **0.011** | 0.022 | 0.003 | **0.042** |

(c) Query error for Citeseer

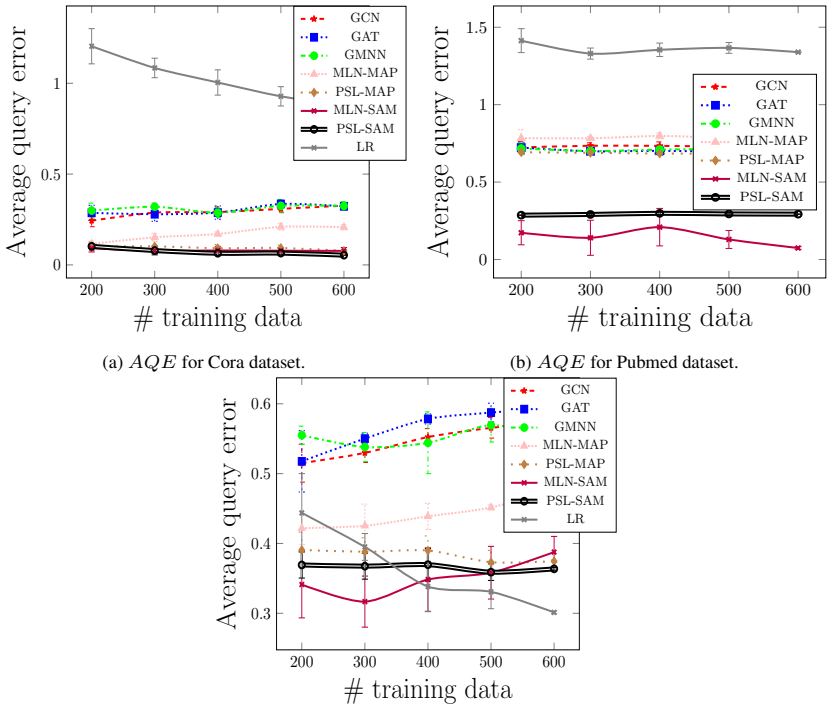
| Methods | Q0 | Q1 | Q2 | Q3 | Q4 | Q5 | $AQE$ |
|---|---|---|---|---|---|---|---|
| GCN | **0.263** | 0.241 | 0.736 | 0.876 | 0.907 | 0.429 | 0.575 |
| GAT | 0.272 | 0.254 | 0.775 | 0.888 | 0.939 | 0.439 | 0.594 |
| GMNN | 0.268 | 0.251 | 0.766 | 0.867 | 0.905 | 0.412 | 0.578 |
| LR | 0.327 | **0.134** | **0.408** | **0.378** | **0.382** | **0.176** | **0.300** |
| MLN-MAP | 0.292 | 0.192 | 0.595 | 0.625 | 0.789 | 0.369 | 0.477 |
| PSL-MAP | 0.283 | 0.151 | 0.460 | 0.600 | 0.516 | 0.237 | 0.374 |
| MLN-SAM | 0.297 | 0.161 | 0.506 | 0.641 | 0.485 | 0.217 | 0.384 |
| PSL-SAM | 0.286 | 0.143 | 0.435 | 0.586 | 0.509 | 0.231 | 0.365 |

Table 2: Query error obtained for all queries on the three datasets and the average query error ($AQE$) across queries. The lowest error is indicated in bold and the second lowest error is underlined.

the category clusters. Since all the approaches propagate the similarity between the node neighbors, the models have a lower error on these queries. Queries Q2, Q3, and Q4 estimate nodes that have neighbors with different categories. These are nodes that lie in the boundary of the category clusters and are harder to infer. GNN-based approaches have very large errors for these queries, resulting in overall poor performance.

### 8.3 Effect of Training Data

To address RQ2, we create five variants of the datasets by varying the amount of training data available for each method from 200 to 600 with increments of 100. Fig. 1 shows the performance of different methods on AGQs as we increase the number of training examples. We report the mean and the standard deviation of $AQE$ across the five folds. We observe that on all three datasets expectation-based approaches have the lowest error. The average query error for logistic regression decreases sharply in the Citeseer data as we increase the




(a) $AQE$ for Cora dataset.

(b) $AQE$ for Pubmed dataset.




(c) $AQE$ for Citeseer dataset.

Fig. 1: Effect of training data on average query error ($AQE$) on all methods and datasets.

size of the training data. We also observe that expectation-based approaches are more robust to the amount of training data when compared to point-estimate approaches.

### 8.4 Trade-off between Estimating AGQs and Locally Decomposable Metrics

To answer RQ3, we compute the accuracy of the predicted node labels which is locally decomposable. Accuracy involves correctly estimating the node labels of each node individually. Estimating AGQs, on the other hand, requires correctly estimating the node labels for several adjacent nodes.

In Fig. 2, we plot the accuracy of the predicted node labels for all three datasets with different amount of training data. We observe that GNNs have a higher accuracy compared to SRL approaches. This is due to the sparsity of node attributes in these datasets which leads to inferior predictions by the logistic regression classifier. GNNs overcome this sparsity by aggregate features of the neighboring nodes. However, this implicitly assumes that a node's neighbors have the same label. While this is true for most nodes, it is not always true. As a result, GNNs tend to perform poorly on AGQs which involve correctly estimating multiple node labels that may belong to different categories.

The error in the homophily between the estimated node labels and the true labels in shown in Fig. 2. We observe that GNN-based approaches have a large error when compared to SRL approaches. Further, we observed that by artificially modifying weight of first rule

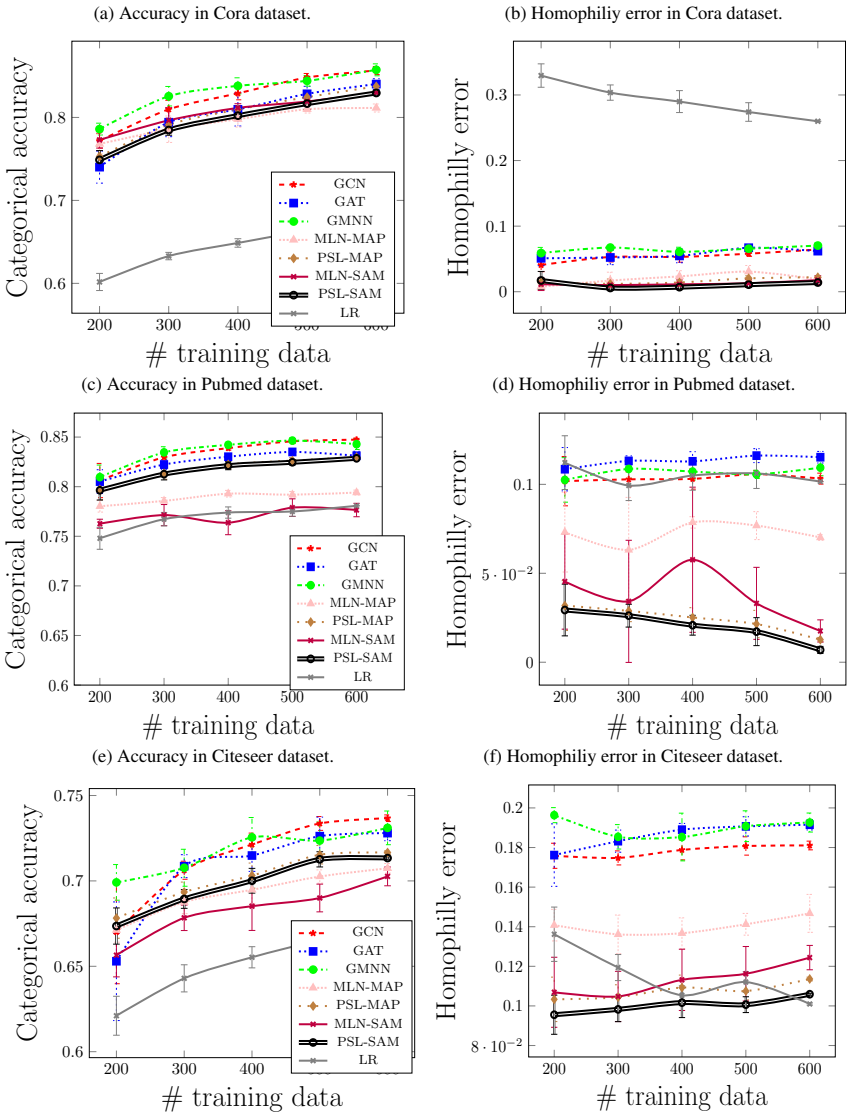



Fig. 2: Figures shows the accuracy and homophily error of different approaches on all three datasets as the number of training data increases.

in PSL that propogates the node labels across the citation edges, the accuracy could be improved at the cost of poor AGQ estimates. This shows that there is a trade-off between locally decomposable metrics such as accuracy and AGQs. While GNN-based approaches are good at estimating locally decomposable metrics they perform poorly when estimating AGQs. SRL-based approaches due to their flexibility in modeling can be altered to perform well on either of the two metrics.

### 8.5 Runtime Comparisons

| Methods | Cora Time (sec) | Pubmed Time (sec) | Citeseer Time (sec) |
|---|---|---|---|
| GCN | 24 | 59 | 29 |
| GAT | 142 | 138 | 122 |
| GMNN | 30 | 17 | 8 |
| LR | 2 | 5 | 2 |
| PSL-MAP | 14 | 124 | 37 |
| PSL-MEAN | 105 | 638 | 124 |
| MLN-MEAN | 270 | 1947 | 166 |
| MLN-MAP | 65 | 368 | 36 |
| PSL-SAMPLES | 105 | 638 | 124 |
| MLN-SAMPLES | 270 | 1947 | 166 |

Table 3: Table showing runtimes for each of the approaches on the three datasets.

To answer RQ4, we recorded the runtimes of the different approaches and is given in Table 3. As expected, we observed that point estimate approaches are significantly faster compared to expectation-based approaches. This is not surprising as point estimates are computed using efficient optimization approaches. Among the GCN, GAT, GMNN, PSL-MAP and MLN-MAP, we observe that GMNN takes the least amount of time in Pubmed and Citeseer dataset and PSL-MAP takes the least amount of time in Cora dataset. Among PSL-SAM and MLN-SAM, we observe that our proposed $MIG$ sampler for PSL is faster than MLN-SAM by a factor of two for Cora and three for Pubmed.

## 9 Conclusion and Future Work

In this paper, we motivate the practical need for aggregate graph queries (AGQs), and show that existing approaches which optimize for locally decomposable metrics such as accuracy neither perform well theoretically nor empirically. In order to compute the expectation under the joint distribution, we introduce a novel sampling approach, MIG, for PSL that is both effective and efficient. We perform an extensive evaluation of SRL and GNN approaches for answering AGQs. Through our experiments we show that SRL methods can get up to 50 times less error compared to GNNs and that our proposed MIG sampler is up to three times faster than other SRL sampling approaches. An interesting future direction is to combine GNN approaches with SRL models that can learn node representations and also infer a joint distribution over the unobserved data. Extending this analysis for networks with missing edges and nodes is another interesting line of future work.

## References

1. Emmanuel Abbe. Community detection and stochastic block models: Recent developments. *JMLR*, 18:1–86, 2018.
2. Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *JMLR*, 18:1–67, 2017.

3. Matthias Broecheler and Lise Getoor. Computing marginal distributions over continuous markov networks for statistical relational learning. In *NeuRIPS*, 2010.
4. T N Bui, S Chaudhuri, F T Leighton, and M Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7:171–191, 1987.
5. Alex Chin, Yatong Chen, Kristen M. Altenburger, and Johan Ugander. Decoupled smoothing on graphs. In *WWW*, 2019.
6. Diane J Cook and Lawrence B Holder. *Mining graph data*. John Wiley & Sons Inc., 2006.
7. Pranav Dandekar, Ashish Goel, and David Lee. Biased assimilation, homophily, and the dynamics of polarization. In *WINE*, 2012.
8. Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100:5–47, 2015.
9. Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. *IJCAI*, 2020.
10. Cody Dunne and Ben Shneiderman. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. In *CHI*, 2013.
11. Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.
12. Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC, 1995.
13. Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.
14. Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeuRIPS*, 2017.
15. L Vivek Harsha Vardhan, Guo Jia, and Stanley Kok. Probabilistic logic graph attention networks for reasoning. In *WWW Companion*, 2020.
16. Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Soc. Netw.*, 5:109 – 137, 1983.
17. Kristian Kersting and Luc De Raedt. Bayesian logic programming: Theory and tool. *, An introduction to Statistical Relational Learning*, 2007.
18. Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
19. Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *CSUR*, 51:62–96, 2018.
20. Stephen Muggleton et al. Stochastic logic programs. *Advances in inductive logic programming*, 32:254–264, 1996.
21. Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.
22. Jennifer Neville and David Jensen. Iterative classification in relational data. In *AAAI Workshop on Learning Statistical Models from Relational Data*, 2002.
23. Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *VLDB*, 4:373–384, 2011.
24. Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Column networks for collective classification. In *AAAI*, 2017.
25. Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, 2006.
26. Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In *NeuRIPS*, 2019.

27. Meng Qu, Yoshua Bengio, and Jian Tang. Gmnn: Graph markov neural networks. In *ICML*, 2019.
28. Qiang Qu, Siyuan Liu, Christian S Jensen, Feida Zhu, and Christos Faloutsos. Interestingness-driven diffusion process summarization in dynamic networks. In *ECML*, 2014.
29. Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical relational artificial intelligence: Logic, probability, and computation*. Morgan & Claypool Publishers, 2016.
30. Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.
31. Matthew Richardson and Pedro Domingos. Markov logic networks. *ML*, 62:107–136, 2006.
32. Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.
33. John Scott. Social network analysis. *Sociology*, 22:109–127, 1988.
34. Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29:93–93, 2008.
35. Lei Shi, Hanghang Tong, Jie Tang, and Chuang Lin. Vegas: Visual influence graph summarization on citation networks. *TKDE*, 27:3417–3431, 2015.
36. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.
37. D Venugopal, S Sarkhel, and V Gogate. Magician: Scalable inference and learning in markov logic using approximate symmetries. Technical report, UofM, Memphis, 2016.
38. Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.
39. Ye Wu, Zhinong Zhong, Wei Xiong, and Ning Jing. Graph summarization for attributed graphs. In *ISEEE*, 2014.
40. Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. 2020.