Approximating Graph Pattern Queries Using Views

Citation for published version:
Li, J, Cao, Y & Liu, X 2016, Approximating Graph Pattern Queries Using Views. in Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. CIKM '16, ACM, New York, NY, USA, pp. 449-458, 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, United States, 24/10/16. DOI: 10.1145/2983323.2983766

Digital Object Identifier (DOI):
10.1145/2983323.2983766

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 25th ACM International on Conference on Information and Knowledge Management

Approximating Graph Pattern Queries Using Views

Jia Li[1] Yang Cao[2] Xudong Liu[1]
[1]Beihang University [2]University of Edinburgh
{lijia1108, liuxd}@buaa.edu.cn, yang.cao@ed.ac.uk

ABSTRACT

This paper studies approximation of graph pattern queries using views. Given a pattern query Q and a set $\mathcal{V}$ of views, we propose to find a pair of queries Q_u and Q_l, referred to as the *upper* and *lower* approximations of Q *w.r.t.* $\mathcal{V}$, such that (a) for any data graph G, answers to (part of) Q in G are contained in $Q_u(G)$ and contain $Q_l(G)$; and (b) both Q_l and Q_u can be answered by using views in $\mathcal{V}$. We consider pattern queries based on both graph simulation and subgraph isomorphism. We study fundamental problems about approximation using views. Given Q and $\mathcal{V}$, (1) we study whether there exist upper and lower approximations of Q *w.r.t.* $\mathcal{V}$. (2) How to find approximations that are closest to Q *w.r.t.* $\mathcal{V}$ if exist? (3) How to answer upper and lower approximations using views in $\mathcal{V}$? We give characterizations of the problems, study their complexity and approximation-hardness, and develop algorithms with provable bounds. Using real-life datasets, we verify the effectiveness and efficiency of approximating simulation and subgraph queries using views.

1. INTRODUCTION

Answering relational queries using views has been extensively studied for decades (see [9] for a survey). The idea has recently been extended to graph pattern queries such as graph simulation [8] and SPARQL [13], and has proven effective. A pattern query Q can be answered using a set $\mathcal{V}$ of pattern views $\mathsf{V}_1, \ldots, \mathsf{V}_n$ if, for any data graph G, the matches $Q(G)$ to Q in G can be computed using nodes and edges in the view answers $\mathsf{V}_1(G), \ldots, \mathsf{V}_n(G)$. Here a view V_i is simply a pattern query whose match result in G is materialized. However, although desirable, in many cases queries cannot be exactly answered as such, when there are only limited views.

Not all is lost. Even when Q cannot be exactly answered using $\mathcal{V}$, Q may still be upper and lower "bounded" via *approximations* that can be answered using $\mathcal{V}$. More specifically, there may exist two patterns Q_u and Q_l for Q, such that

(1) both Q_u and Q_l can be answered using $\mathcal{V}$;

(2) Q is contained in Q_u and contains Q_l, *i.e.,* for *all* data graphs G, $Q_l(G) \subseteq Q(G)$ and $Q(G) \subseteq Q_u(G)$, where $\subseteq$ denotes the inclusion of match results.

That is, $Q(G)$ can be upper and lower bounded by $Q_u(G)$ and $Q_l(G)$ for any G. We call Q_u and Q_l *upper* and *lower approximations* of Q *w.r.t.* $\mathcal{V}$. That is, Q_u and Q_l together give us an approximation of Q using views in $\mathcal{V}$.

Example 1: Consider a recommendation network G taken from [16] and shown in Fig. 1. A node denotes an entity labeled with expertise, *e.g.,* project manager (PM), software developer (SD), software tester (ST), user interface designer (UD) and business analyst (BA); an edge *e.g.,* (PM, SD) indicates that SD worked well under the supervision of PM in previous projects. A human resource manager wants to set up a team to develop a new software product. The requirement is specified by a pattern query Q_1, also shown in Fig. 1 (in the dashed rectangular). It aims to find a group of PM, SD, UD and ST, that (1) both SD and UD worked well under PM, and (2) there exists a ST worked well under SD and UD. A graph pattern matching process is conducted for Q_1 on G, based on *graph simulation* [10], to find matched nodes in G to Q_1.

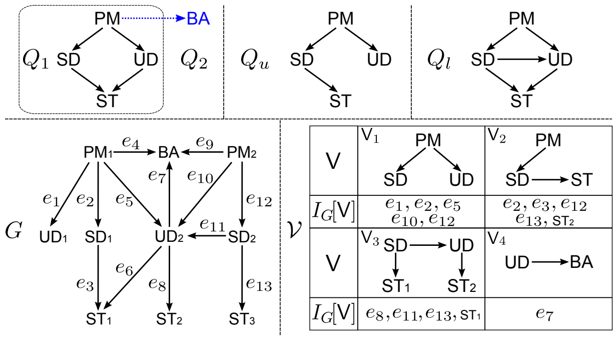


Figure 1: Querying recommendation network

Suppose that a set of views $\mathcal{V} = \{\mathsf{V}_1, \mathsf{V}_2, \mathsf{V}_3, \mathsf{V}_4\}$ is defined and their answers in G are cached, as shown in Fig. 1. Observe the following. (1) Q_1 cannot be answered using the views only, as there exist no views that can be combined to get the match result of Q_1. (2) However, there exist two pattern graphs Q_u and Q_l can be answered using view answers $I_G[\mathsf{V}_1]$, $I_G[\mathsf{V}_2]$ and $I_G[\mathsf{V}_1]$, $I_G[\mathsf{V}_3]$, respectively, yielding match result $Q_u(G) = \{\mathsf{PM_i}, \mathsf{UD_j}, \mathsf{SD_k}, \mathsf{ST_h}\}$ $(i, j, k \in [1, 2], h \in [1, 3])$ and $Q_l(G) = \{\mathsf{PM_2}, \mathsf{UD_2}, \mathsf{SD_2}, \mathsf{ST_h}\}(h \in [1, 3])$. (3) One can verify that $Q_1(G) = \{\mathsf{PM_i}, \mathsf{UD_2}, \mathsf{SD_k}, \mathsf{ST_h}\}$ $(i, k \in [1, 2], h \in [1, 3])$. Therefore, $Q_l(G) \subseteq Q_1(G) \subseteq Q_u(G)$. These indicate that, although Q cannot be exactly answered using views, it can be approximately answered with the views.

Consider another pattern query Q_2 also shown in Fig. 1. A business analyst (BA) is in demand for the project. This is reflected in Q_2 by a new node labeled with BA (colored in blue) with an edge from PM to BA. One can verify that Q_2 cannot be answered using the views. Worse still, there exist no pattern graphs that can upper and lower bound Q_2 as Q_u and Q_l for Q_1. However, the answers to subgraphs of Q_2, *e.g.,* Q_1, can be approximated by Q_u and Q_l using views. □

To make full use of this idea, several fundamental problems call for a full treatment. How to formalize upper and lower approximations *w.r.t.* views so that we can cover both "complete" approximation (*e.g.,* Q_1 above) and "subgraph" approximation (*e.g.,* Q_2) using views? Given a query Q and views $\mathcal{V}$, does Q have an upper (resp. lower) approximation

Q_u (resp. Q_l) *w.r.t.* $\mathcal{V}$? How to find accurate approximations for Q *w.r.t.* $\mathcal{V}$, *i.e.,* Q_u and Q_l that are closest to Q?

Contributions. This paper tackles these questions.

(1) We propose *upper* and *lower approximation* of graph pattern queries *w.r.t. a set* $\mathcal{V}$ *of views* (Section 2). We consider both simulation and subgraph patterns, based on graph simulation [8, 10] and subgraph isomorphism [18], respectively.

(2) We study fundamental problems for upper and lower approximations for simulation queries (Section 3). We develop characterizations and investigate their complexity.

(3) We develop algorithms with provable guarantees to find approximations closest to Q *w.r.t.* $\mathcal{V}$ (Sections 4 and 5).

(4) We extend the study to subgraph queries (Section 6). We characterize upper and lower approximations of subgraph queries *w.r.t.* views. We also study subgraph query answering using views to build a query-driven approximation framework for both simulation and subgraph queries.

(5) On real-life datasets, we experimentally verify the effectiveness and efficiency of the framework (Section 7).

Related work. We categorize previous work as follows.

Query answering using views. There has been recent work on answering graph queries using views over graph data, such as simulation queries [8] and SPARQL queries [12, 13] over graph and RDF data, respectively. This work extends the prior work in the following. (1) We consider approximating graph queries using views *i.e.,* find approximate answers using views instead of exact answers as in [8, 12, 13]. (2) We also investigate subgraph isomorphism query answering using views, which has not been studied before.

Query approximation. Closer to us is approximate query answering, which can be classified as (a) data-driven approximation which builds data synopses first on which the queries are then evaluated [6,11], (b) query-driven approximation [3,5,7], which uses "cheaper" queries Q' instead of the original queries Q and computes answers to Q' as approximate answers to Q, and (c) heuristic approximation which computes approximate answers via compromised search [15, 17, 20].

This work differs from theirs in the following. (1) We study graph pattern matching queries based on graph simulation and subgraph isomorphism, instead of relational queries [5–7, 11] or graph primitives [15]. (2) Unlike data-driven and heuristic approximation [6, 11, 15, 17, 20], we consider query-driven approximation, which approximates queries in the absence of data graphs. (3) We focus on approximate queries that can be answered using views, which is not considered in previous work on query-approximation [5, 7].

2. QUERY-DRIVEN APPROXIMATION

In this section, we review basic notions and formulate upper and lower approximations *w.r.t.* views.

Graphs. A *data graph* G is a triple (V, E, l), where (1) V is a finite set of nodes; (2) $E \subseteq V \times V$ is a set of edges, in which (v, v') denotes the edge from v to v'; (3) $l()$ is a function such that for each node v in V, $l(v)$ is a label in a label set Σ.

We write G as (V, E) when it is clear from the context. The *size* of G, denoted by $|G|$, is defined to be the total number of nodes and edges in G, *i.e.,* $|G| = |V| + |E|$. For a graph G, we denote by V_G and E_G the nodes and edges of G, respectively.

Subgraphs. Graph $H(V_s, E_s, l_H)$ is a *subgraph* of data graph $G(V, E, l)$, denoted as $G_s[V_s, E_s]$, if (1) for each node $u \in V_s$, $u \in V$ and $l_H(u) = l(u)$, and (2) for each edge $(u, u') \in E_s$, $(u, u') \in E$. That is, subgraph $G_s[V_s, E_s]$ only contains a subset of nodes and edges of graph G.

A subgraph H of G is an *induced subgraph* if for any nodes v and v' in H, (v, v') is an edge in H if it is an edge in G.

Pattern queries. A pattern query Q is a directed connected graph (V_Q, E_Q, l_Q), where V_Q, E_Q and l_Q are analogous to their counterparts in data graphs. We simply write Q as (V_Q, E_Q) when it is clear from the context.

We consider two semantics of graph pattern matching.

Simulation queries. Graph G matches pattern Q via *graph simulation* [10], denoted by $Q \prec G$, if there exists a binary match relation $R \subseteq V_Q \times V$ such that (a) for each $(u, v) \in R$, $l_Q(u) = l(v)$; (b) for each node u in V_Q, there exists a node v in V such that (i) $(u, v) \in R$, and (ii) for any edge (u, u') in Q, there exists an edge (v, v') in G such that $(u', v') \in R$.

For any G that matches Q, there exists a *unique maximum* match relation via graph simulation [10], denoted by R_M.

Simulation queries are widely used in social community analysis and social marketing [4].

Subgraph queries. Graph G matches pattern Q via *subgraph isomorphism* [18], denoted by $Q \lhd G$, if there exists a subgraph G_s of G that is isomorphic to Q, *i.e.,* there exists a bijection h from V_Q to V_s such that (a) $(u, u') \in E_Q$ if and only if $(h(u), h(u')) \in E_s$; and (b) for each $u \in V_Q$, $l_Q(u) = l(h(u))$.

Match results and images. For simulation queries, the *match result* $Q(G)$ to Q in G is a subset of nodes of G, such that a node $v \in Q(G)$ if and only if v is in R_M. The *image* of Q in G, denoted by $I_G[Q]$, is the subgraph $G_s[V_s, E_s]$ of G, where (1) $V_s = Q(G)$, and (2) an edge $(v, v') \in E_s$ if and only if there exists an edge (u, u') in Q with $(u, v) \in R_M$ and $(u', v') \in R_M$. Intuitively, the image $I_G[Q]$ is a subgraph of G with nodes from $Q(G)$ and associated matched edges in G.

Similarly, for subgraph queries, the *match result* $Q(G)$ to Q in G is a subset of nodes in G that are in some subgraph G_s of G isomorphic to Q. The *image* $I_G[Q]$ of Q in G is a subgraph of G consisting of all those nodes and edges in G that are in some subgraph G_s of G isomorphic to Q.

A pattern query Q_1 is *equivalent* to pattern query Q_2 if for any data graph G, $Q_1(G) = Q_2(G)$.

Views. A view query (or simply view), denoted by V, is a pattern query whose images in data graphs are cached. We also consider both simulation views and subgraph views. For a set $\mathcal{V}$ of views, we denote by $\|\mathcal{V}\|$ and $|\mathcal{V}|$ the cardinality of $\mathcal{V}$ and the total size of views in $\mathcal{V}$, respectively.

Note that, for both simulation and subgraph queries Q, their images are subgraphs of the data graph G, even though there may exist exponentially many isomorphisms from Q to G. This ensures that the cached view images are of total size bounded by $\|\mathcal{V}\||G|$ instead of an exponent in $|G|$ or $\|\mathcal{V}\|$.

Query answering using views ([8]). Consider a set $\mathcal{V}$ of views $\mathsf{V}_1, \ldots, \mathsf{V}_n$. Following [8], a query Q is *answerable using views in* $\mathcal{V}$ if there exists another query A such that for *all* data graphs G, (1) A only refers views in $\mathcal{V}$ and their images $I_G[\mathcal{V}] = \{I_G[\mathsf{V}_1], \ldots, I_G[\mathsf{V}_n]\}$ in G, and (2) A is equivalent to Q, *i.e.,* the match result to A in G is the same to $Q(G)$.

Intuitively, when a pattern query Q is answerable using views in $\mathcal{V}$, for any data graph G, $Q(G)$ can be identified by accessing the images $I_G[\mathcal{V}]$ of views in G only.

Example 2: Consider pattern query Q_2 and view set $\mathcal{V}$ in

Fig. 1. The match image $I_G[\mathsf{V}_i]$ ($i \in [1,4]$) of each view V_i in $\mathcal{V}$ is given in Fig. 1, denoted by edges and the isolated nodes. Observe that Q_u and Q_l are answerable using $\mathcal{V}$. Indeed, $Q_u(G)$ can be answered by combing view images $I_G[\mathsf{V}_1]$ and $I_G[\mathsf{V}_2]$, and $Q_l(G)$ can be answered via $I_G[\mathsf{V}_1]$ and $I_G[\mathsf{V}_3]$. □

Partial and complete query containment. Pattern Q *is partially upper contained in* pattern Q_u, denoted by $Q \sqsubseteq_{\mathcal{U}} Q_u$, if there exists an induced subgraph Q_s of Q such that for any graph G, $Q_s(G) \subseteq Q_u(G)$. Similarly, Q *partially lower contains* pattern Q_l, denoted by $Q_l \sqsubseteq_{\mathcal{L}} Q$, if there exists an induced subgraph Q_s of Q such that for any graph G, $Q_l(G) \subseteq Q_s(G)$. In particular, when Q_s is Q above, Q is called *completely upper contained in* Q_u, denoted by $Q \sqsubseteq^c_{\mathcal{U}} Q_u$; and *completely lower contains* Q_l, denoted by $Q_l \sqsubseteq^c_{\mathcal{L}} Q$.

Example 3: Consider pattern queries Q_1, Q_2, Q_u, Q_l and view set $\mathcal{V}$ in Fig. 1 of Example 1. One can verify that $Q_1 \sqsubseteq^c_{\mathcal{U}} Q_u$, $Q_2 \sqsubseteq_{\mathcal{U}} Q_u$, $Q_l \sqsubseteq^c_{\mathcal{L}} Q_1$ and $Q_l \sqsubseteq_{\mathcal{L}} Q_2$.

In particular, consider data graph G of Fig. 1 and Q_1 as an induced subgraph of Q_2. Observe the following. $Q_1(G) = \{\mathsf{PM_i}, \mathsf{UD_2}, \mathsf{SD_k}, \mathsf{ST_h}\}(i, k \in [1,2], h \in [1,3])$. The match results of Q_u and Q_l are $Q_u(G) = \{\mathsf{PM_i}, \mathsf{UD_j}, \mathsf{SD_k}, \mathsf{ST_h}\}$ $(i, j, k \in [1,2], h \in [1,3])$ and $Q_l(G) = \{\mathsf{PM_2}, \mathsf{UD_2}, \mathsf{SD_2}, \mathsf{ST_h}\}$ $(h \in [1,3])$. Therefore, $Q_l(G) \subseteq Q_1(G) \subseteq Q_u(G)$. □

We are ready to define upper and lower approximations.

Upper and lower approximations. A graph pattern Q_u is an *upper approximation* of Q *w.r.t.* $\mathcal{V}$, if (1) $Q \sqsubseteq_{\mathcal{U}} Q_u$ and (2) Q_u is answerable using $\mathcal{V}$. Similarly, a graph pattern Q_l is a *lower approximation* of Q *w.r.t.* $\mathcal{V}$ if (1) $Q_l \sqsubseteq_{\mathcal{L}} Q$ and (2) Q_l is answerable using $\mathcal{V}$. We consider non-empty approximations, *i.e.,* patterns with at least one edge.

In particular, if $Q \sqsubseteq^c_{\mathcal{U}} Q_u$ (resp. $Q_l \sqsubseteq^c_{\mathcal{L}} Q$) in condition (1), *i.e.,* Q (resp. Q_l) is completely upper (resp. lower) contained in Q_u (resp. Q), then we call Q_u (resp. Q_l) a *complete* upper (resp. lower) approximation of Q *w.r.t.* $\mathcal{V}$.

Example 4: Continue Example 3. We know that $Q_2 \sqsubseteq_{\mathcal{U}} Q_u$ and $Q_l \sqsubseteq_{\mathcal{L}} Q_2$. Furthermore, from Example 2 we also know that Q_u and Q_l are answerable using $\mathcal{V}$. Thus, Q_u (resp. Q_l) is an upper (resp. lower) approximation of Q *w.r.t.* $\mathcal{V}$,

In particular, Q_u (resp. Q_l) is a complete upper (resp. lower) approximation of Q_1 *w.r.t.* $\mathcal{V}$. Indeed, $Q_1 \sqsubseteq^c_{\mathcal{U}} Q_u$, $Q_l \sqsubseteq^c_{\mathcal{L}} Q_1$, and both Q_u and Q_l are answerable using $\mathcal{V}$. □

3. FUNDAMENTAL ANALYSES

In this section, we first study some properties of simulation query containment (Section 3.1). We then study fundamental problems for upper and lower approximations and investigate their complexity and approximability (Section 3.2). We will extend the study to subgraph queries in Section 6.

3.1 Characterizations

To study approximation, it is essential to characterize simulation query containment and answering using views.

Query containment. We first characterize containment of simulation queries below.

Theorem 1: *Consider simulation queries Q, Q_u and Q_l,*

(1) $Q \sqsubseteq_{\mathcal{U}} Q_u$ if and only if (iff) $Q_u \prec Q$;

(2) $Q \sqsubseteq^c_{\mathcal{U}} Q_u$ iff $Q_u \prec Q$ and $V_Q = V_{I_Q[Q_u]}$;

(3) $Q_l \sqsubseteq_{\mathcal{L}} Q$ iff there exists an induced subgraph Q_s of Q such that $Q_s \prec Q_l$ and $V_{Q_l} = V_{I_{Q_l}[Q_s]}$; and

(4) $Q_l \sqsubseteq^c_{\mathcal{L}} Q$ iff $Q \prec Q_l$ and $V_{Q_l} = V_{I_{Q_l}[Q]}$;

Here $I_{Q'}[Q'']$ is the image of Q'' in Q'. □

Intuitively, the characterization boils down the containment testing of simulation queries to simulation testing between the queries. We will use this later in the paper.

Query answering using views. A characterization for simulation query answering using views is given in [8]. To make the paper self-contained, we cite it as follows (rephrased). We will extend it to subgraph queries in Section 6.

Lemma 2 [8]: *For a view set $\mathcal{V}$ and a simulation query Q, Q can be answered using $\mathcal{V}$ iff $E_Q = \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_Q[\mathsf{V}]}$.* □

Example 5: Consider pattern query Q_l and view set $\mathcal{V}$ in Fig. 1. One can verify that Q_l can be answered using $\mathcal{V}$, as $\bigcup_{\mathsf{V}_i\in\mathcal{V}(i\in[1,4])} E_{I_{Q_l}[\mathsf{V}_i]} = \{(\mathsf{PM}, \mathsf{SD}), (\mathsf{PM}, \mathsf{UD})\} \cup \{(\mathsf{PM}, \mathsf{SD}), (\mathsf{SD}, \mathsf{ST})\} \cup \{(\mathsf{SD}, \mathsf{UD}), (\mathsf{SD}, \mathsf{ST}), (\mathsf{UD}, \mathsf{ST})\} \cup \emptyset = E_{Q_l}$. □

Based on the characterization, an algorithm for answering simulation queries using views is also given in [8].

3.2 Fundamental Problems and Complexity

We next study fundamental problems related to upper and lower approximations. For each of them, we also study special cases where approximations are required to be complete.

Existence of approximation. Upper and lower approximations do not always exist, as shown in Example 1. We first investigate the existence of (complete) approximation.

Existence of upper approximation. The first problem, denoted by EUA (resp. EUA^c), is to ask, for any pattern Q and set $\mathcal{V}$ of views, whether there exists an upper approximation Q_u (resp. complete upper approximation Q^c_u) for Q using $\mathcal{V}$.

Theorem 3: *For a simulation query Q and set $\mathcal{V}$ of views,*

(a) there exists an upper approximation for Q using $\mathcal{V}$ iff there exists $\mathsf{V} \in \mathcal{V}$ such that the match result $\mathsf{V}(Q) \neq \emptyset$;

(b) there exists a complete upper approximation for Q using $\mathcal{V}$ iff $V_Q = \bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_Q[\mathsf{V}]}$; and

(c) EUA and EUA^c are quadratic time in $|Q|$ and $|\mathcal{V}|$. □

We will constructively prove Theorem 3(c) in Section 4, by giving quadratic time algorithms for EUA and EUA^c. The proof of Theorem 3(a,b) is deferred to the full version.

Example 6: Consider pattern query Q_1 and view set $\mathcal{V}$ in Fig. 1. By Theorem 3(a), there exists an upper approximation for Q_1 *w.r.t.* $\mathcal{V}$ since $\mathsf{V}_1(Q_1) \neq \emptyset$ for view V_1 in $\mathcal{V}$. Furthermore, by Theorem 3(b), one can verify that there exists a complete upper approximation for Q_1 *w.r.t.* $\mathcal{V}$, since $\bigcup_{\mathsf{V}_i\in\mathcal{V}(i\in[1,4])} V_{I_{Q_1}[\mathsf{V}_i]} = \{\mathsf{PM}, \mathsf{SD}, \mathsf{UD}, \mathsf{ST}\} = V_{Q_1}$. □

Existence of lower approximation. Similarly, we study the existence of lower (resp. complete lower) approximation for simulation query Q using $\mathcal{V}$, denoted by ELA (resp. ELA^c).

Theorem 4: *For any simulation query Q and set $\mathcal{V}$ of views,*

(a) there exists a complete lower approximation Q^c_l of Q using $\mathcal{V}$ iff $E_Q \subseteq \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{\widehat{Q}}[\mathsf{V}]}$;

(b) ELA^c is in $O(|\mathcal{V}||Q|^2)$ time; and in contrast,

(c) ELA is NP-complete.

Here $\widehat{Q}$ is the complete graph of Q. □

Unlike upper approximation, the existence of generic lower approximation is much harder than complete approximation.

We will give a constructive proof of Theorem 4(b) in Section 5, by giving a PTIME algorithm for ELA^c. The proofs for Theorem 4(a,c) is deferred to the full version.

Example 7: Consider pattern query Q_1 and view set $\mathcal{V}$ in Fig. 1. By Theorem 4(a), there exists a complete lower approximation for Q_1 *w.r.t.* $\mathcal{V}$. Indeed, by computing simulation for views $\mathsf{V}_i \in \mathcal{V}$ ($i \in [1,4]$) on the complete graph $\widehat{Q_1}$ of Q_1, $\bigcup_{\mathsf{V}_i \in \mathcal{V}(i \in [1,4])} E_{I_{\widehat{Q_1}}[\mathsf{V}_i]} = \{(\mathsf{PM}, \mathsf{SD}), (\mathsf{PM}, \mathsf{UD}), (\mathsf{SD}, \mathsf{UD}), (\mathsf{SD}, \mathsf{ST}), (\mathsf{UD}, \mathsf{ST})\} \supseteq E_{Q_1}$. □

Closest approximation. There may exist multiple approximations for a simulation pattern query Q *w.r.t.* views $\mathcal{V}$. We naturally want to compute the one that is closest to Q, *i.e.,* the *closest* approximation. Below we first define the notion of closeness to measure the quality of approximations. We then study closest upper and lower approximations.

Closeness. Consider two patterns Q_1 and Q_2, we define the *closeness* of Q_1 and Q_2, denoted by $\mathsf{clo}(Q_1, Q_2)$, as the number of edges in Q_1 and Q_2 that are not in the edge-induced maximum common subgraph of Q_1 and Q_2. Intuitively, the smaller $\mathsf{clo}(Q_1, Q_2)$ is, the closer Q_1 and Q_2 are.

Closest upper approximation. The *closest (resp. complete closest) upper approximation* problem, denoted by CUA (resp. CUA^c), is to find, given a simulation query Q and views $\mathcal{V}$, the upper (resp. complete upper) approximation Q_u (resp. Q_u^c) of Q that is closest to Q, *i.e.,* for any other (resp. complete) upper approximation Q'_u (resp. Q'^c_u), $\mathsf{clo}(Q_u, Q) \le \mathsf{clo}(Q'_u, Q)$ (resp. $\mathsf{clo}(Q^c_u, Q) \le \mathsf{clo}(Q'^c_u, Q)$). We refer to Q_u (resp. Q_u^c) as the closest (resp. complete) approximation to Q *w.r.t.* $\mathcal{V}$.

Although the closeness measure clo is NP-hard due to the hardness of computing maximum common subgraphs, the analysis of closest upper approximation is efficient.

Theorem 5: *Both problems* CUA *and* CUA^c *are quadratic time in* $|Q|$ *and* $|\mathcal{V}|$. □

We will give a constructive proof of Theorem 5 in Section 4, by giving quadratic time algorithms for CUA and CUA^c.

Closest lower approximation. Similarly, the *closest (resp. complete closest) lower approximation* problem, denoted by CLA (resp. CLA^c), is to compute, given a simulation query Q and views $\mathcal{V}$, the lower (resp. complete lower) approximation Q_l (resp. Q_l^c) of Q that is closest to Q, if exists.

Unlike upper approximation, the analysis of closest lower approximation is much harder. Denote by DCLA (resp. DCLA^c) the decision problem of CLA (resp. CLA^c), which is to decide, given Q, $\mathcal{V}$ and a natural number k, whether there exists a lower (resp. complete lower) approximation Q_l (resp. Q_l^c) such that $\mathsf{clo}(Q_l, Q) \le k$ (resp. $\mathsf{clo}(Q_l^c, Q) \le k$). Denote by OCLA (resp. OCLA^c) the optimization problem of CLA (resp. CLA^c), which is to compute the minimum closeness $\mathsf{clo}(Q_l, Q)$ (resp. $\mathsf{clo}(Q_l^c, Q)$) for all lower (resp. complete lower) approximations Q_l (resp. Q_l^c) of Q using $\mathcal{V}$. Then we have the following.

Theorem 6: *(1) Both* DCLA *and* DCLA^c *are* NP*-complete. (2) Both* OCLA *and* OCLA^c *are not in* APX. □

Despite the hardness, we will develop approximation algorithms with guarantees for CLA and CLA^c in Section 5.

Algorithm CUAsim^c

Input: Simulation query Q, set $\mathcal{V}$ of simulation views.
Output: A complete upper approximation of Q *w.r.t.* $\mathcal{V}$ if exists.

1. **for each** V in $\mathcal{V}$ **do** compute the image $I_Q[\mathsf{V}]$ of V in Q;
2. $V_u := \bigcup_{\mathsf{V} \in \mathcal{V}} V_{I_Q[\mathsf{V}]}$; $E_u := \bigcup_{\mathsf{V} \in \mathcal{V}} E_{I_Q[\mathsf{V}]}$; Let Q_u be (V_u, E_u);
3. **if** $V_u = V_Q$ **then return** Q_u;
4. **return** "no"; */* no complete upper approximation for Q */*

Figure 2: Algorithm CUAsim^c

4. UPPER APPROXIMATION

In this section, we develop algorithms for computing closest upper approximations, as a constructive proof of Theorem 5.

The algorithms are based on a *small model property* of upper approximation. Consider a simulation query Q and a set $\mathcal{V}$ of simulation views. We have the following property.

Lemma 7: *For any upper (resp. complete upper) approximation Q_u (resp. Q_u^c) of Q* w.r.t. *$\mathcal{V}$, there exists a subgraph Q_s (resp. Q_s^c with $V_{Q_s^c} = V_Q$) of Q equivalent to Q_u, such that*

- *Q_s (resp. Q_s^c) is also an upper (resp. complete upper) approximation of Q* w.r.t. *$\mathcal{V}$; and*
- *Q_s (resp. Q_s^c) is closer to Q than Q_u,* i.e., $\mathsf{clo}(Q_s, Q) \le \mathsf{clo}(Q_u, Q)$ *(resp.* $\mathsf{clo}(Q_s^c, Q) \le \mathsf{clo}(Q_u^c, Q)$*).* □

This ensures that, for any upper approximation Q_u of Q *w.r.t.* $\mathcal{V}$, if Q_u is not a subgraph of Q, then there must exist another upper approximation Q_s of Q *w.r.t.* $\mathcal{V}$ that is a subgraph of Q, such that (i) Q_u and Q_s are equivalent, *i.e.,* $Q_u(G) = Q_s(G)$ for any G; and (ii) Q_s is at least as close to Q as Q_u is *w.r.t.* the closeness. Thus, we only need to focus on subgraphs of Q when computing upper approximations of Q, instead of the infinitely many upper approximations.

Based on Lemma 7, we develop exact algorithms for CUA and CUA^c, as a proof of Theorem 5. We start with CUA^c.

4.1 On Complete Upper Approximation

The algorithm for CUA^c, denoted by CUAsim^c, is shown in Fig. 2. It takes as input a simulation query Q and a set $\mathcal{V}$ of simulation views, and returns the closest complete upper approximation of Q using $\mathcal{V}$ if it exists.

More specifically, algorithm CUAsim^c first computes the images $I_Q[\mathsf{V}]$ of all views V of $\mathcal{V}$ in Q (line 1), and then constructs a subgraph Q_u of Q with nodes and edges from the images (line 2). It returns Q_u if it covers all nodes in Q (line 3); and "No" otherwise (line 4), *i.e.,* there exists no complete upper approximation for Q *w.r.t.* $\mathcal{V}$.

Example 8: Consider pattern query Q_1 and view set $\mathcal{V}$ in Fig. 1. CUAsim^c first computes the images $I_{Q_1}[\mathsf{V}_i]$ ($i \in [1,4]$) of all views V_i of $\mathcal{V}$ in Q_1 (line 1), and then constructs a subgraph Q_u of Q_1 with $V_u = \{\mathsf{PM}, \mathsf{SD}, \mathsf{UD}, \mathsf{ST}\}$, and $E_u = \{(\mathsf{PM}, \mathsf{SD}), (\mathsf{PM}, \mathsf{UD}), (\mathsf{SD}, \mathsf{ST})\}$ (line 2). It finally checks that $V_u = V_{Q_1}$. Thus CUAsim^c returns Q_u as the closest complete upper approximation, as shown in Fig. 1 (line 3). □

Correctness & Complexity. The algorithm is in $O((|V_Q| + |E_Q|)|\mathcal{V}| + |V_Q|^2)$ time, where $|\mathcal{V}| = \sum_{\mathsf{V} \in \mathcal{V}}(|V_\mathsf{V}| + |E_\mathsf{V}|)$. Indeed, observe that line 1 of CUAsim^c takes $O((|V_Q|+|E_Q|)|\mathcal{V}|)$ time and the checking in line 3 takes at most $O(|V_Q|^2)$ time.

To see algorithm CUAsim^c is correct, observe the following. (1) By Lemma 7, the closest upper approximation of Q *w.r.t.* $\mathcal{V}$, if exists, must be a subgraph of Q. (2) Any subgraph Q_s

Algorithm CLAsim^c

Input: Simulation query Q, set $\mathcal{V}$ of views.
Output: A complete lower approximation of Q *w.r.t.* $\mathcal{V}$ if exists.

1. **for each** V in $\mathcal{V}$ **do**
2. compute image $I_{\widehat{Q}}[\mathsf{V}]$ of V in $\widehat{Q}$;
3. $\mathsf{neg}(\mathsf{V}) := E_{I_{\widehat{Q}}[\mathsf{V}]} \setminus E_Q$; $\mathsf{pos}(\mathsf{V}) := E_{I_{\widehat{Q}}[\mathsf{V}]} \cap E_Q$;
4. $U := \emptyset$;
5. **while** $E_Q \not\subseteq \bigcup_{\mathsf{V}\in U} E_{I_{\widehat{Q}}[\mathsf{V}]}$ **and** $\bigcup_{\mathsf{V}\in\mathcal{V}} \mathsf{pos}(\mathsf{V}) \not\subseteq \bigcup_{\mathsf{V}'\in U} E_{I_{\widehat{Q}}[\mathsf{V}']}$ **do**
6. find V in $\mathcal{V}\setminus U$ *minimizing* $\rho(\mathsf{V}) = \frac{|\mathsf{neg}(\mathsf{V})|}{|\mathsf{pos}(\mathsf{V})\setminus\bigcup_{\mathsf{V}'\in U} E_{I_{\widehat{Q}}[\mathsf{V}']}|}$;
7. $U := U \cup \{\mathsf{V}\}$;
8. $V_l := \bigcup_{\mathsf{V}\in U} V_{I_{\widehat{Q}}[\mathsf{V}]}$; $E_l := \bigcup_{\mathsf{V}\in U} E_{I_{\widehat{Q}}[\mathsf{V}]}$; Let Q_l be (V_l, E_l);
9. **if** $E_Q \subseteq E_{Q_l}$ **then return** Q_l;
10. **return** "no"; */* no complete lower approximation for Q */*

Figure 3: Algorithm CLAsim^c

of Q that can be answered using $\mathcal{V}$ contains only nodes and edges from Q_u computed by CUAsim^c(line 2). (3) Therefore, Q_u is the closest upper approximation of Q *w.r.t.* $\mathcal{V}$. (4) Algorithm CUAsim^c returns Q_u if and only if Q_u is a complete upper approximation of Q *w.r.t.* $\mathcal{V}$.

4.2 Extending to Generic Approximation

We next present algorithm CUAsim that extends CUAsim^c for problem CUA as follows. First, CUAsim computes Q_u in the same way as CUAsim^c does. Instead of checking whether all nodes in Q are covered by Q_u, it directly returns Q_u if Q_u is not empty; and returns "no" otherwise.

Example 9: Recall pattern query Q_2 and view set $\mathcal{V}$ in Fig. 1. Algorithm CUAsim constructs Q_u as shown in Fig. 1 in the same way as in Example 8, and directly returns it as the closest upper approximation of Q_2. □

Correctness & Complexity. Following the analysis of algorithm CUAsim^c, one can verify that algorithm CUAsim determines whether there exists, and if so finds the closest upper approximation of Q in $O((|V_Q| + |E_Q|)|\mathcal{V}|)$ time.

These together complete the proof of Theorems 3 and 5.

5. LOWER APPROXIMATION

In this section, we develop approximation algorithms with guarantees for CLA and CLA^c, to handle the hardness of computing closest lower approximations as shown in Theorem 6.

The algorithms are based on a *small model property* as follows. Consider simulation query Q and set $\mathcal{V}$ of simulation views. Let $\widehat{Q}$ be the complete graph of Q.

Lemma 8: *For any lower (resp. complete lower) approximation Q_l (resp. Q_l^c) of Q* w.r.t. *$\mathcal{V}$, there exists a subgraph Q_s (resp. Q_s^c with $V_{Q_s^c} = V_Q$) of $\widehat{Q}$ equivalent to Q_l, such that*

- *Q_s (resp. Q_s^c) is also a lower (resp. complete lower) approximation of Q* w.r.t. *$\mathcal{V}$; and*
- *Q_s (resp. Q_s^c) is closer to Q than Q_l,* i.e., $\mathsf{clo}(Q_s, Q) \leq \mathsf{clo}(Q_l, Q)$ *(resp. $\mathsf{clo}(Q_s^c, Q) \leq \mathsf{clo}(Q_l^c, Q)$).* □

Lemma 8 tells us that we only need to consider subgraphs of the *complete* graph $\widehat{Q}$ of Q when computing lower approximations of Q *w.r.t.* $\mathcal{V}$. Based on this, below we develop approximation algorithms for CLA and CLA^c. We start with complete lower approximations for CLA^c.

5.1 On Complete Lower Approximation

The algorithm for CLA^c, denoted by CLAsim^c and shown in Fig. 3, computes a complete lower approximation of Q *w.r.t.* $\mathcal{V}$ if exists, and returns "no" otherwise. It works in three steps. (a) It first computes the images of views in $\mathcal{V}$ in the complete graph $\widehat{Q}$ of Q. (b) It then derives a subgraph Q_l of $\widehat{Q}$ from the images. (c) Finally it checks whether Q_l covers all edges of Q and returns Q_l if so.

More specifically, for each view V in $\mathcal{V}$, it computes the image $I_{\widehat{Q}}[\mathsf{V}]$ of V in the complete graph $\widehat{Q}$ of Q, with two additional sets $\mathsf{neg}(\mathsf{V})$ and $\mathsf{pos}(\mathsf{V})$ (lines 1-3). Here $\mathsf{neg}(\mathsf{V})$ contains edges in the image $I_{\widehat{Q}}[\mathsf{V}]$ that are not edges of Q, and $\mathsf{pos}(\mathsf{V})$ contains edges that are both in $I_{\widehat{Q}}[\mathsf{V}]$ and Q.

It then iteratively identifies relevant views in $\mathcal{V}$ (lines 4-7). In each iteration, it selects V in $\mathcal{V}$ with minimum $\rho(\mathsf{V}) =$

$$\frac{|\mathsf{neg}(\mathsf{V})|}{|\mathsf{pos}(\mathsf{V}) \setminus \bigcup_{\mathsf{V}'\in U} E_{I_{\widehat{Q}}[\mathsf{V}']}|}$$

among all views in $\mathcal{V}$ that are not selected yet (lines 6-7). The iteration terminates when either all edges in Q are covered by images of selected views in $\widehat{Q}$ or no more new edges in Q can be covered by selecting the remaining views (line 5). Intuitively, $\rho(\mathsf{V})$ denotes the "average number" of edges in $S(\mathsf{V})$ per edges in Q that are newly covered by V, where $S(\mathsf{V})$ is the set of edges in $\widehat{Q}$ that are covered by V but are not in Q. Algorithm CLAsim^c chooses V with minimum $\rho(\mathsf{V})$ to get more edges in Q covered while covering as few edges that are not in Q as possible. With Lemma 8, we will show later that this leads to a lower approximation close to Q.

Finally, CLAsim^c builds a graph Q_l that consists of nodes and edges in the images of the selected views in $\widehat{Q}$ (line 8). It returns Q_l as the closest complete lower approximation if Q_l covers all edges in Q, and returns "no" otherwise (lines 9-10).

Example 10: Recall pattern query Q_1 and view set $\mathcal{V}$ in Fig. 1. For each view V_i ($i \in [1, 4]$) in $\mathcal{V}$, algorithm CLAsim^c computes the image of V_i in the complete graph $\widehat{Q}_1$ of Q_1 (lines 1-3). It then enters the iteration process. In first iteration, CLAsim^c selects V_1 with minimum $\rho(\mathsf{V}_1) = 0$ and adds it into U, while $\rho(\mathsf{V}_2) = 0$, $\rho(\mathsf{V}_3) = \frac{1}{2}$ and $\rho(\mathsf{V}_4) = +\infty$. In the second iteration, it selects V_2 with minimum $\rho(\mathsf{V}_2) = 0$ while $\rho(\mathsf{V}_3) = \frac{1}{2}$ and $\rho(\mathsf{V}_4) = +\infty$. In the third iteration, it selects V_3 with minimum $\rho(\mathsf{V}_2) = 1$ while $\rho(\mathsf{V}_4) = +\infty$, and stops the process as the termination condition $E_{Q_1} \subseteq \bigcup_{\mathsf{V}_i\in U(i\in[1,3])} E_{I_{\widehat{Q}_1[\mathsf{V}_i]}}$ holds (lines 4-7). Finally, CLAsim^c builds a graph Q_l, as shown in Fig. 1, and returns Q_l as the closest complete lower approximation as Q_l covers all edges in Q_1 (lines 8-10). □

Although problem CLA^c does not admit any approximation algorithm with constant approximation ratios as shown in Theorem 6(2), we show that algorithm CLAsim^c has the following provable performance guarantee. For each edge e in $E_{\widehat{Q}} \setminus E_Q$, let $\mathsf{occ}(e)$ be $|\{\mathsf{V} \mid e \in E_{I_{\widehat{Q}}[\mathsf{V}]}, \mathsf{V} \in \mathcal{V}\}|$, *i.e.,* the number of views in $\mathcal{V}$ whose images in $\widehat{Q}$ contain e.

Theorem 9: *Algorithm* CLAsim^c *is a* $\mathsf{max}_{e\in E_{\widehat{Q}}\setminus E_Q}\mathsf{occ}(e) \cdot \ln(\mathsf{max}_{\mathsf{V}\in\mathcal{V}}|E_{I_{\widehat{Q}}[\mathsf{V}]} \cap E_Q|)$*-approximation algorithm that always returns a complete lower approximation of Q* w.r.t. $\mathcal{V}$ *in $O(|\mathcal{V}||Q|^2)$-time whenever there exists one.* □

By Theorem 1 and Lemma 2, one can verify that CLAsim^c returns a lower approximation of Q *w.r.t.* $\mathcal{V}$ in $O(|\mathcal{V}||Q|^2)$-

Algorithm CLAsim

Input: Simulation query Q, set $\mathcal{V}$ of views.
Output: A lower approximation of Q *w.r.t.* $\mathcal{V}$ if exists.

1. **for each** V in $\mathcal{V}$ **do** compute image $I_{\widehat{Q}}[\mathsf{V}]$ of V in $\widehat{Q}$ of Q;
2. $V_l := \bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_{\widehat{Q}}[\mathsf{V}]}$; $E_l := \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{\widehat{Q}}[\mathsf{V}]}$; Let Q_l be (V_l, E_l);
3. **while** there exist u, u' in V_{Q_l} with $(u,u') \in E_Q \setminus E_{Q_l}$ **do**
   /* Q_l is not an induced subgraph of Q yet */
4.   **for each** V in $\mathcal{V}$ with u, u' in $V_{I_{\widehat{Q}}[\mathsf{V}]}$ **do**
5.   remove the view image $I_{\widehat{Q}}[\mathsf{V}]$ from Q_l;
6.   **if** both u and u' remain in V_{Q_l} of Q_l **then**
7. $u_0 := \arg\min_{v\in\{u,u'\}} f(v)$;
8.   remove from Q_l all view images $I_{\widehat{Q}}[\mathsf{V}]$ that contain u_0;
9. **if** $E_{Q_l} \neq \emptyset$ **then return** Q_l;
10. **return** "no"; /* *no lower approximation for Q* */

Figure 4: Algorithm CLAsim

time whenever there exists one (detailed proof is deferred to the full version). This also gives a proof of Theorem 4. Below we focus on the approximation ratio of CLAsim^c.

Let Q_{OPT} be the closest complete lower approximation of Q *w.r.t.* $\mathcal{V}$. By Lemma 2, there exists a subset U_{OPT} of views in $\mathcal{V}$ such that Q_{OPT} is composed by images of views in U_{OPT}, *i.e.*, $E_{Q_{\mathsf{OPT}}} = \bigcup_{\mathsf{V}\in U_{\mathsf{OPT}}} E_{I_{\widehat{Q}}[\mathsf{V}]}$. Observe that $\mathsf{clo}(Q_l, Q)$ $= |\bigcup_{\mathsf{V}\in U} \mathsf{neg}(\mathsf{V})|$ and $\mathsf{clo}(Q_{\mathsf{OPT}}, Q) = |\bigcup_{\mathsf{V}\in U_{\mathsf{OPT}}} \mathsf{neg}(\mathsf{V})|$. Furthermore, for any U' of views,

$$\begin{aligned}|\textstyle\bigcup_{\mathsf{V}\in U'} \mathsf{neg}(\mathsf{V})| &\leq \textstyle\sum_{\mathsf{V}\in U'} |\mathsf{neg}(\mathsf{V})| = \sum_{e\in\bigcup_{\mathsf{V}\in U'}\mathsf{neg}(\mathsf{V})} \mathsf{occ}(e)\\ &\leq \textstyle\max_{e\in E_{\widehat{Q}}\setminus E_Q} \mathsf{occ}(e)\cdot|\bigcup_{\mathsf{V}\in U'} \mathsf{neg}(\mathsf{V})|.\end{aligned}$$

Thus, $\dfrac{\mathsf{clo}(Q_l,Q)}{\mathsf{clo}(Q_{\mathsf{OPT}},Q)} \leq \dfrac{\sum_{\mathsf{V}\in U}|\mathsf{neg}(\mathsf{V})|}{|\bigcup_{\mathsf{V}\in U_{OPT}}\mathsf{neg}(\mathsf{V})|}$

$$\leq \frac{\max_{e\in E_{\widehat{Q}}\setminus E_Q}\mathsf{occ}(e)\cdot\sum_{\mathsf{V}\in U}|\mathsf{neg}(\mathsf{V})|}{\sum_{\mathsf{V}\in U_{\mathsf{OPT}}}|\mathsf{neg}(\mathsf{V})|}$$

Observe that the **while** loop encodes an approximation-preserving reduction from computing U with minimizing $\sum_{\mathsf{V}\in U}\mathsf{neg}(\mathsf{V})$ to the *minimum weighted set cover* problem [19], by treating E_Q as the universe, $\mathsf{pos}(\mathsf{V})$ for views V in $\mathcal{V}$ as the collection of sets, and $\mathsf{neg}(\mathsf{V})$ as the weight of each set $\mathsf{pos}(\mathsf{V})$. Moreover, it simulates a B-approximation for the later problem [19], where $B = \ln(\max_{\mathsf{V}\in\mathcal{V}}|E_{I_{\widehat{Q}}[\mathsf{V}]}\cap E_Q|)$. Thus, $\dfrac{\sum_{\mathsf{V}\in U}|\mathsf{neg}(\mathsf{V})|}{\sum_{\mathsf{V}\in U_{\mathsf{OPT}}}|\mathsf{neg}(\mathsf{V})|} \leq \dfrac{B\cdot\sum_{\mathsf{V}\in U_*}|\mathsf{neg}(\mathsf{V})|}{\sum_{\mathsf{V}\in U_{\mathsf{OPT}}}|\mathsf{neg}(\mathsf{V})|} \leq B$, where U_* is the optimal solution to the reduced set cover problem.

Therefore, $\dfrac{\mathsf{clo}(Q_l,Q)}{\mathsf{clo}(Q_{\mathsf{OPT}},Q)} \leq \max_{e\in E_{\widehat{Q}}\setminus E_Q}\mathsf{occ}(e)\cdot B$. That is, algorithm CLAsim^c is a $\max_{e\in E_{\widehat{Q}}\setminus E_Q}\mathsf{occ}(e)\cdot\ln(\max_{\mathsf{V}\in\mathcal{V}}$ $|E_{I_{\widehat{Q}}[\mathsf{V}]}\cap E_Q|)$-approximation for problem CLA^c.

5.2 On Generic Lower Approximation

We next develop an algorithm for problem CLA, to find generic lower approximations. From Theorem 4(c), we know that it is even NP-hard to decide whether there exists a lower approximation of a given query Q and set $\mathcal{V}$ of views, not to mention the closest one. In light of this and Theorem 6, we develop an efficient heuristic algorithm for CLA.

The algorithm, denoted by CLAsim, is shown in Fig. 4. It first computes the images of views in the complete graph $\widehat{Q}$ of Q and combines them into Q_l (lines 1-2). It then checks whether Q_l is an induced subgraph of Q and iteratively removes images of views from Q_l to make it an induced subgraph if not (lines 3-8). It finally returns Q_l as the lower approximation of Q *w.r.t.* $\mathcal{V}$ if Q_l is not empty (lines 9-10).

More specifically, algorithm CLAsim reduces Q_l as follows. In each iteration, it checks whether Q_l is already an induced subgraph of Q, by checking whether there exists an edge (u,u') in Q such that $u, u' \in V_{Q_l}$ but (u,u') is not in E_{Q_l} (line 3). If such an edge (u,u') exists, it identifies all views V in $\mathcal{V}$ whose images in $\widehat{Q}$ of Q contain u and u', and removes $I_{\widehat{Q}}[\mathsf{V}]$ from Q_l if so (lines 4-5). After that, it checks whether both u and u' remain in Q_l (line 6), and identifies the one that has smaller value of function $f(v) = |N(v)| + |H(v)|$, where

$$N(v) = \{(u,u') \,|\, u,u' \in V_{Q_l(v)}, (u,u') \in E_Q \setminus E_{Q_l(v)}\},$$

$$H(v) = \{(u,u') \,|(u,u') \in E_Q \cap (E_{Q_l} \setminus E_{Q_l(v)})\},$$

in which $Q_l(v)$ is the pattern graph $(\bigcup_{\mathsf{V}\in\mathcal{V}\setminus S(v)} V_{I_{\widehat{Q}}[\mathsf{V}]}$, $\bigcup_{\mathsf{V}\in\mathcal{V}\setminus S(v)} E_{I_{\widehat{Q}}[\mathsf{V}]})$, where $S(v) = \{\mathsf{V} \,|\, v \in V_{I_{\widehat{Q}}[\mathsf{V}]}, \mathsf{V}\in\mathcal{V}\}$.

Intuitively, $N(v)$ contains the *new* "bad edges" in Q_l after removing all view images in Q_l associated to v, and $H(v)$ contains the "good edges" in Q_l that are removed due to the removal of images related to v. The smaller $f(v)$ is, the less impact on $\mathsf{clo}(Q_l, Q)$ when removing v from Q_l via associated view images containing v. Therefore, algorithm CLAsim removes the one from $\{u,u'\}$ with smaller $f(v)$ value from Q_l, by dropping all relevant view images (line 8).

Example 11: Consider pattern query Q_2 and view set $\mathcal{V}$ in Fig. 1. For each view V_i ($i \in [1,4]$) in $\mathcal{V}$, CLAsim computes the image of V_i in the complete graph $\widehat{Q}_2$ of Q_2 and combines them into Q_l, using the images of all the four views (lines 1-2). CLAsim then checks and finds that Q_l is not an induced subgraph of Q_2, as there exists an edge (PM, BA) belonging to Q_2 but not in Q_l, while nodes PM and BA in Q_l (line 3). In order to make Q_l an induced subgraph of Q_2, CLAsim enters the iteration process to remove "bad" images from Q_l. In the process, CLAsim first removes images that contain PM and BA, and turns out no view needs to be removed at this point (lines 4-5). It then removes images contain either PM or BA. It finds that $f(\mathsf{BA}) = 0$, less than $f(\mathsf{PM}) = 2$ ($N(\mathsf{PM}) = \emptyset$ and $H(\mathsf{PM}) = \{(\mathsf{PM}, \mathsf{SD}), (\mathsf{PM}, \mathsf{UD})\}$). Thus it removes all images containing BA from Q_l, *i.e.*, $I_{\widehat{Q_2}}[\mathsf{V}_4]$ composed of one edge (UD, BA) (lines 6-8). The remaining Q_l is an induced subgraph of Q_2 with non-empty edge set, as shown in Fig. 1. CLAsim returns Q_l as the closest lower approximation of Q_2 (line 9). □

Correctness & Complexity. The correctness of CLAsim is guaranteed by Theorem 1 and Lemma 2. It can be implemented in $O(|\mathcal{V}||Q|^2)$ time with an inverted index from nodes in Q to images of views in $\mathcal{V}$.

6. EXTENDING TO SUBGRAPH QUERIES

In this section, we extend the study of upper and lower approximations to subgraph queries. We first characterize and revisit fundamental problems for approximating subgraph queries (Sections 6.1 and 6.2). We then develop algorithms for approximating and answering subgraph queries using views (Section 6.3). We finally present a query-driven approximation framework based on the algorithms (Section 6.4).

6.1 Characterizations

We develop characterizations for subgraph query containment *w.r.t.* views and subgraph query answering using views.

Note that graph query answering using views has only been studied for simulation queries [10] and subgraph queries have not been investigated before.

Query containment. Subgraph query containment can be characterized along the same lines as simulation queries.

Theorem 10: *Consider subgraph queries Q, Q_u and Q_l,*

(1) $Q \sqsubseteq_{\mathcal{U}} Q_u$ *iff* $Q_u \lhd Q$;

(2) $Q \sqsubseteq^c_{\mathcal{U}} Q_u$ *iff* $Q_u \lhd Q$ *and* $V_Q = V_{I_Q[Q_u]}$;

(3) $Q_l \sqsubseteq_{\mathcal{L}} Q$ *iff there exists an induced subgraph* Q_s *of* Q *such that* $Q_s \lhd Q_l$ *and* $V_{Q_l} = V_{I_{Q_l}[Q_s]}$; *and*

(4) $Q_l \sqsubseteq^c_{\mathcal{L}} Q$ *iff* $Q \lhd Q_l$ *and* $V_{Q_l} = V_{I_{Q_l}[Q]}$; □

Subgraph query answering using views. Similarly, we characterize subgraph query answering using views based on the notion of images as follows.

Theorem 11: *Consider subgraph query Q and views $\mathcal{V}$,*

(1) Q *can be answered using* $\mathcal{V}$ *iff* $E_Q = \bigcup_{\mathsf{V} \in \mathcal{V}} E_{I_Q[\mathsf{V}]}$; *and*

(2) it is NP*-complete to decide whether* Q *can be answered using* $\mathcal{V}$. □

6.2 Fundamental Problems and Complexity

Below we investigate the existence and closeness of upper and lower approximations for subgraph queries.

Existence of approximation. We first revisit problems EUA, EUA^c, ELA and ELA^c for subgraph queries.

Theorem 12: *For a subgraph query Q and set $\mathcal{V}$ of views,*

(1) there exists an upper approximation for Q *using* $\mathcal{V}$ *iff there exists* $\mathsf{V} \in \mathcal{V}$ *such that the match result* $\mathsf{V}(Q) \neq \emptyset$;

(2) there exists a complete upper approximation for Q *using* $\mathcal{V}$ *iff* $V_Q = \bigcup_{\mathsf{V} \in \mathcal{V}} V_{I_Q[\mathsf{V}]}$;

(3) there exists a complete lower approximation of Q *using* $\mathcal{V}$ *iff* $E_Q \subseteq \bigcup_{\mathsf{V} \in \mathcal{V}} E_{I_{\hat{Q}}[\mathsf{V}]}$; *and*

(4) problems EUA, EUA^c, ELA *and* ELA^c *(decision version) are all* NP*-complete.* □

In contrast to simulation queries, it is already NP-hard to decide whether a subgraph query Q has a lower or upper approximation *w.r.t.* $\mathcal{V}$. This is because it is NP-hard to decide whether there exists an isomorphism from a view V to Q, which is essential in deciding the existence of approximations.

Note that the complexity of ELA does not go up beyond NP, though ELA is already NP-hard for simulation queries and it involves isomorphism checking, which is NP-hard, when comes to subgraph queries. Therefore, unlike simulation queries, the existence of complete and generic lower approximations have the same complexity for subgraph queries.

Closest approximation. Using the same closeness measure clo as for simulation queries in Section 3, we revisit problems CUA, CUA^c, CLA and CLA^c for computing closest approximations of subgraph queries *w.r.t.* subgraph views $\mathcal{V}$.

Theorem 13: *For a subgraph query Q and set $\mathcal{V}$ of views,*

(1) pattern graph $Q_u(\bigcup_{\mathsf{V} \in \mathcal{V}} V_{I_Q[\mathsf{V}]}, \bigcup_{\mathsf{V} \in \mathcal{V}} E_{I_Q[\mathsf{V}]})$ *is the closest upper approximation of* Q *using* $\mathcal{V}$;

(2) if $\bigcup_{\mathsf{V} \in \mathcal{V}} V_{I_Q[\mathsf{V}]} = V_Q$, $Q^c_u(\bigcup_{\mathsf{V} \in \mathcal{V}} V_{I_Q[\mathsf{V}]}, \bigcup_{\mathsf{V} \in \mathcal{V}} E_{I_Q[\mathsf{V}]})$ *is the closest complete upper approximation of* Q; *and*

(3) problems CUA, CUA^c, CLA *and* CLA^c *(decision version) are all* NP*-complete.* □

Algorithm QAViso

Input: Subgraph query Q, set $\mathcal{V}$ of subgraph views.
Output: A query plan $\mathcal{P}$ to Q if Q can be answered using $\mathcal{V}$.

1. $\mathcal{P} := []$; $S := \emptyset$;
2. **for each** V in $\mathcal{V}$ **do if** $\mathsf{V}(Q) \neq \emptyset$ **then** $S := S \cup \{\mathsf{V}\}$;
3. **if** $\bigcup_{\mathsf{V} \in S} E_{I_Q[\mathsf{V}]} \neq E_Q$ **then return** "no";
   /* assume $S = \{\mathsf{V}_1, \ldots, \mathsf{V}_m\}$ */
4. initialize $\mathcal{P}$ to $[T_1 = I_G[\mathsf{V}_1]]$; $V := \emptyset$;
5. **for** i in $[1, m-1]$ **do**
6. append $T_{i+1} = T_i \bowtie I_G[\mathsf{V}_{i+1}]$ to $\mathcal{P}$;
7. add common nodes in T_i and $I_G[\mathsf{V}_{i+1}]$ to V;
8. append $T_{m+1} = \sigma(T_m, V, d_Q)$ to $\mathcal{P}$;
9. append $T_{m+2} = \mathsf{mat}(Q, T_{m+1})$ to $\mathcal{P}$; **return** $\mathcal{P}$;

Figure 5: Algorithm QAViso

Similar to ELA, CLA and CLA^c for subgraph queries have the same complexity as for simulation queries, though subgraph isomorphism is used, which is already NP-hard.

6.3 Algorithms

We next study algorithms for approximating subgraph queries with subgraph views.

Computing closest upper and lower approximations. We develop algorithms for computing the closest upper and lower approximations of subgraph queries *w.r.t.* subgraph views, complete or generic, denoted by CUAiso^c, CUAiso, CLAiso^{c}and CLAiso, respectively, by minorly revising their counterparts for simulation queries in Sections 4 and 5. The only change is that we simply use subgraph isomorphism instead of graph simulation when computing view images. The correctness of these algorithms are guaranteed by the characterizations in Sections 6.1 and 6.2, and small model properties analogous to Lemma 7 and Lemma 8 for simulation queries.

Answering subgraph queries using views. Based on Theorem 11, we also develop an algorithm for subgraph query answering using views, denoted by QAViso, to support the use of upper and lower approximations of subgraph queries. Algorithm QAViso takes as input a subgraph query Q and a set $\mathcal{V}$ of views and returns a query plan $\mathcal{P}$ for Q using $\mathcal{V}$ if Q can be answered using $\mathcal{V}$. For any data graph G, $\mathcal{P}(G)$ finds $Q(G)$ in G by accessing images $I_G[\mathsf{V}]$ for views V in $\mathcal{V}$ only.

The plan $\mathcal{P}$ is a sequence of operations $T_1 = \delta_1, \ldots, T_n = \delta_n$, such that for any G, $T_n(G) = Q(G)$, and moreover, each δ_i is one of the following that only accesses the view images.

<u>*(a) Join* $G_1 \bowtie G_2$</u>, where G_1 and G_2 are two graphs. $G_1 \bowtie G_2$ returns all connected components of the graph $(V_{G_1} \cup V_{G_2}, E_{G_1} \cup E_{G_2})$ that contain nodes from both G_1 and G_2.

<u>*(b) Filter* $\sigma(G, V, r)$</u>, where G is a graph, V is a subset of nodes in G and r is an integer. It returns the subgraph $G_{V,r}$ of G that is induced by nodes of G within distance no larger than r to some node in V.

<u>*(c) Match* $\mathsf{mat}(Q, G)$</u>, where Q is a pattern graph and G is a data graph. It computes the match result $Q(G)$ to Q in G.

Algorithm QAViso generates a plan $\mathcal{P}$ for Q and $\mathcal{V}$ as follows. It first checks whether Q can be answered by using views in $\mathcal{V}$ and identifies relevant views if so, based on Theorem 11 (lines 2-3). It then generates a plan $\mathcal{P}$ of three parts. (i) The first part joins all images of relevant views together (lines 4-7). (ii) The second part filters a subgraph of the joined graph in (i) consisting of nodes that are within distance d_Q to those nodes involved in the join, where d_Q is the diameter of Q (line 8). (iii) The final part returns the match to Q in the filtered subgraph in (ii) (line 9).

Remark. When identifying relevant views, view selection optimization can also be employed in QAViso, along the same lines as for answering simulation queries using views [8], to select those relevant views with minimum cost.

6.4 Putting things together

Algorithms developed above and in Sections 4 and 5, together with the one for answering simulation queries with simulation views in [8], give us a framework of *query-driven approximation using views.*

(a) Given a pattern query Q and views $\mathcal{V}$, we first check whether Q can be answered using $\mathcal{V}$, via QAViso and [8].

(b) If so, we then generate query plans that exactly answer Q using cached answered to views in $\mathcal{V}$ only, by algorithm QAViso (resp. [8]) if Q is a subgraph (resp. simulation) query.

(c) Otherwise, we check whether there exist upper and lower approximations of Q *w.r.t.* $\mathcal{V}$ and find the closest approximations Q_u and Q_l if exist, via algorithms CUAsim^c, CUAsim, CLAsim^c, CLAsim, CUAiso^c, CUAiso, CLAiso^c and CLAiso.

(d) We generate query plans that answer Q_u and Q_l by using $\mathcal{V}$ only, which are guaranteed to exist, via QAViso and [8].

7. EXPERIMENTAL STUDY

Using real-life data, we conducted two sets of experiments to evaluate the effectiveness and the efficiency of our framework of query-driven approximation using views.

Experimental setting. We first give the settings we used.

Real-life graphs. We used two real-life data graphs.

(1) Knowledge graph (DBpedia) was taken from DBpedia 201504 [1] with 4.43M nodes, 8.43M edges and 735 labels.

(2) YouTube graph (YouTube) is a video network taken from YouTube with 2.03M video nodes, 12.22M video-video directed edges and 398 labels [2]. An edge from videos x to y indicates that if one watch x, then he is very likely to watch y.

Queries. We implemented a generator for graph patterns. It is controlled by three parameters: the number #n of nodes, the number #e of edges, and label f_v from an alphabet Σ of labels in the corresponding real-life graphs. We use $(|V_Q|, |E_Q|)$ to denote the size of a pattern query $Q(V_Q, E_Q)$. We generated 100 patterns in total by varying #n from 3 to 11 and #e from 5 to 13.

Views. We generated a set of 60 views for each data graph following [8,14]. The views were designed of sizes (2,1), (3,2), (4,3) and (4,4), and we varied the structure for views of the same size. (a) For DBpedia, each view image has 72K nodes and edges in average, and view images in total take 32.58% of the physical memory of the entire DBpedia dataset. (b) For YouTube, each image takes 80K nodes and edges in average, and all images take 34.29% of the memory of YouTube.

Algorithms. We implemented the following algorithms, all in C++: (1) our algorithms CUAsim^c, CUAsim, CLAsim^c, CLAsim and CUAiso^c, CUAiso, CLAiso^c, CLAiso for finding closest complete and generic upper and lower approximations of simulation queries and subgraph queries; (2) our algorithm QAViso for answering subgraph approximations using views; (3) QAVsim for answering simulation approximations using views, taken from [8]; (4) conventional algorithms gSim [10] and VF2 (using C++ Boost Graph Library) for answering simulation and subgraph queries directly, respectively.

% of views used	25%	50%	75%	100%
DBpedia:%-ap: sim(sub)	65(53)%	74(64)%	85(78)%	95(88)%
YouTube:%-ap: sim(sub)	72(61)%	80(69)%	88(81)%	98(90)%
DBpedia:%-ap^c: sim(sub)	53(41)%	64(52)%	75(68)%	88(82)%
YouTube:%-ap^c: sim(sub)	58(51)%	70(64)%	83(77)%	92(85)%
DBpedia:%-answerable	0(0)%	3(0)%	8(3)%	12(8)%
YouTube:%-answerable	0(0)%	5(1)%	10(4)%	17(10)%

Table 1: Percentages of queries approximable using $\mathcal{V}$

All the experiments were run on a machine with Intel Core(TM) Duo 3.00GHz CPU and 16GB of memory. Each test was repeated 10 times, and the average is reported here.

Experimental results. We next report our findings.

Exp-1: Effectiveness. We evaluated the effectiveness of upper and lower approximation of pattern queries using views. By default we use all queries and views.

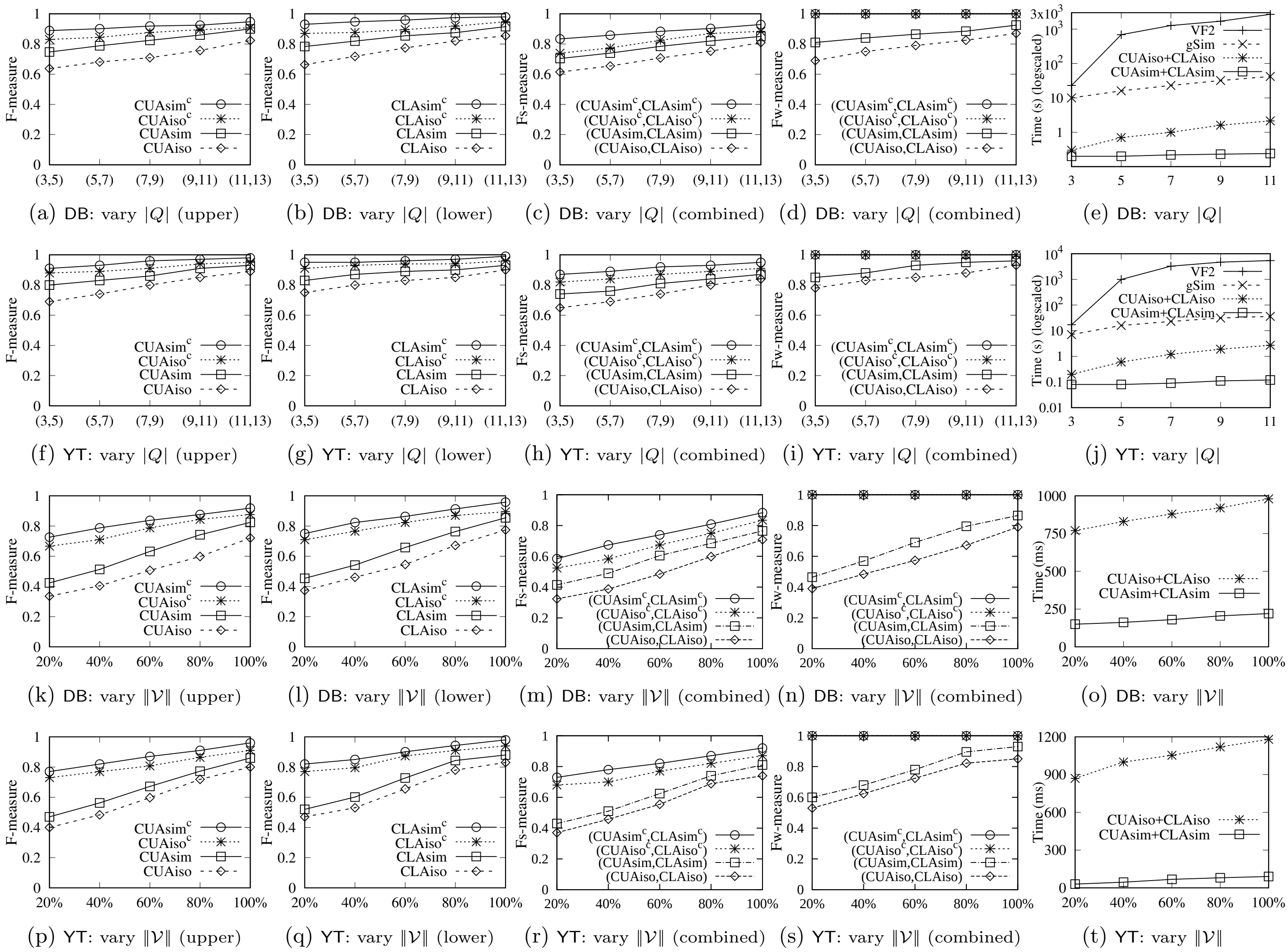
(1) Percentage of queries approximable using views. Varying the percentage of the total views used, we tested the percentage of queries that (a) have upper or lower (resp. complete upper or lower) approximations, denoted by %-ap (resp. %-ap^c) and (b) can be answered using views, denoted by %-answerable. The results are reported in Table 1.

Observe the following. (1) The majority of queries have approximations with available views, among which a large portion are complete. Indeed, 85% of simulation queries on DBpedia and 81% of subgraph queries on YouTube have approximations when only 75% of the views are used, respectively. The percentages are 75% and 77% when complete approximations are considered. (2) Simulation queries are more likely to have (complete) approximations than subgraph queries. (3) Very few patterns can be answered using views. Indeed, even using all the views, only 12% and 17% of simulation queries on DBpedia and YouTube can be answered using the views, respectively. The percentages are even lower for subgraph queries. This further verifies the need for studying approximations using views.

(2) Accuracy of approximation using views. We evaluated the average accuracy of the closest upper and lower approximations by the accuracy of their answers in the data graphs. We use three measures: F-measure $F(Q, Q', G)$ for measuring the accuracy of an upper or lower approximation Q' *w.r.t.* Q in G; and strong F-measure $F_s(Q, Q_u, Q_l, G)$ and weak F-measure $F_w(Q, Q_u, Q_l, G)$ for measuring the accuracy of a pair (Q_u, Q_l) of upper and lower approximations *w.r.t.* Q in G. More specifically, for any two sets S and S', let $\mathsf{prec}(S', S) = \frac{|S' \cap S|}{|S'|}$ and $\mathsf{recall}(S', S) = \frac{|S' \cap S|}{|S|}$. Then we define

- $F(Q, Q', G) = \dfrac{2\mathsf{prec}(Q'(G), Q(G)) \cdot \mathsf{recall}(Q'(G), Q(G))}{\mathsf{prec}(Q'(G), Q(G)) + \mathsf{recall}(Q'(G), Q(G))}$
- $F_s(Q, Q_u, Q_l, G) = \dfrac{2\mathsf{prec}(Q_u(G), Q(G)) \cdot \mathsf{recall}(Q_l(G), Q(G))}{\mathsf{prec}(Q_u(G), Q(G)) + \mathsf{recall}(Q_l(G), Q(G))}$
- $F_w(Q, Q_u, Q_l, G) = \dfrac{2\mathsf{prec}(Q_l(G), Q(G)) \cdot \mathsf{recall}(Q_u(G), Q(G))}{\mathsf{prec}(Q_l(G), Q(G)) + \mathsf{recall}(Q_u(G), Q(G))}$

Intuitively, $F(Q, Q', G)$ is the conventional F-measure. The strong F-measure $F_s(Q, Q_u, Q_l, g)$ is a variant of F-measure by using $Q_u(G)$ for prec and $Q_l(G)$ for recall, while $F_w(Q, Q_u, Q_l, G)$ uses $Q_l(G)$ for prec and $Q_u(G)$ for recall. Q_l is likely to have higher prec and Q_u tends to have higher recall. Therefore, the weak F-measure measures the accuracy



(a) DB: vary $|Q|$ (upper) (b) DB: vary $|Q|$ (lower) (c) DB: vary $|Q|$ (combined) (d) DB: vary $|Q|$ (combined) (e) DB: vary $|Q|$

(f) YT: vary $|Q|$ (upper) (g) YT: vary $|Q|$ (lower) (h) YT: vary $|Q|$ (combined) (i) YT: vary $|Q|$ (combined) (j) YT: vary $|Q|$

(k) DB: vary $\|\mathcal{V}\|$ (upper) (l) DB: vary $\|\mathcal{V}\|$ (lower) (m) DB: vary $\|\mathcal{V}\|$ (combined) (n) DB: vary $\|\mathcal{V}\|$ (combined) (o) DB: vary $\|\mathcal{V}\|$

(p) YT: vary $\|\mathcal{V}\|$ (upper) (q) YT: vary $\|\mathcal{V}\|$ (lower) (r) YT: vary $\|\mathcal{V}\|$ (combined) (s) YT: vary $\|\mathcal{V}\|$ (combined) (t) YT: vary $\|\mathcal{V}\|$

Figure 6: Effectiveness of upper and lower approximation using views (DB = DBpedia; YT = YouTube)

of approximating Q with the "best" of Q_u and Q_l combined. In contrast, the strong F-measure assesses the "worst" of Q_u and Q_l together. In particular, when Q_u and Q_l are complete upper and lower approximations of Q, $\mathsf{prec}(Q_l(G), Q(G)) = \mathsf{recall}(Q_u(G), Q(G)) = F_w(Q, Q_u, Q_l, G) = 1$.

(a) Impact of Q. Varying the size $|Q|$ of Q from (3, 5) to (11, 13), we evaluated the impact of pattern queries Q on the accuracy of upper and lower approximations Q_u and Q_l, measured by $F(Q, Q_u, G)$, $F(Q, Q_l, G)$, $F_w(Q, Q_u, Q_l, G)$ and $F_s(Q, Q_u, Q_l, G)$ on DBpedia and YouTube. The results are shown in Figures 6(a), 6(b), 6(c) and 6(d) for DBpedia and Figures 6(f), 6(g), 6(h) and 6(i) for YouTube.

Observe the following. (1) The closest upper and lower approximations Q_u and Q_l computed by our algorithms are capable to approximate Q well. The F-measure of both Q_u and Q_l is consistently above 0.7 and 0.6 on DBpedia for simulation and subgraph queries, respectively. (2) Complete upper and lower approximations are much more accurate than generic approximations. For example, for subgraph queries, when only complete upper and lower approximations are considered, the F-measure of upper approximations is 0.89 when $|Q|$ is (11,13) while it is 0.82 for generic approximations. (3) Combining the best of Q_u and Q_l together can give us much higher accuracy for approximating Q. Indeed, the weak F-measure of Q_u and Q_l is consistently higher than the F-measure of both Q_u and Q_l taken alone, on both DBpedia and YouTube for both simulation and subgraph queries. For example, for simulation queries, the weak F-measure is 0.93 when $|Q|$ is (11,13) on DBpedia, while the F-measure of both Q_u and Q_l is lower than 0.9. (4) Even when the worst of Q_u and Q_l are taken together, *i.e.,* the prec of Q_u and the recall of Q_l, they are still capable of approximating Q well with views. For example, the strong F-measure is above 0.7 when $|Q|$ is (3, 5) on DBpedia for simulation queries. (5) The accuracy of upper and lower approximations is not sensitive to the size of Q for both simulation and subgraph queries.

(b) Impact of $\mathcal{V}$. Varying the percentage of views used from 20% to 100% and fixed the size $|Q|$ of Q to be (7,9), we evaluated the impact of views $\mathcal{V}$ on the accuracy of upper and lower approximations Q_u and Q_l, measured by $F(Q, Q_u, G)$, $F(Q, Q_l, G)$, $F_w(Q, Q_u, Q_l, G)$ and $F_s(Q, Q_u, Q_l, G)$ on DBpedia and YouTube. The results are

shown in Figures 6(k), 6(l), 6(m) and 6(n) for DBpedia and Figures 6(p), 6(q), 6(r) and 6(s) for YouTube.

The results tell us the following. (1) More views lead to higher accuracy of upper and lower approximations Q_u and Q_l in all cases. For example, for simulation queries on YouTube, the F-measure of Q_l is above 0.52 when 20% of the views are used, and it increases to 0.88 when 100% of the views are used. Indeed, Q_u and Q_l are more close to Q with more views, which lead to higher accuracy in terms of their answers in the data graph. (2) The weak F-measure of Q_u and Q_l combined is higher than both the F-measure of Q_u and of Q_l, while the strong F-measure is the lowest. This is consistent with the case of varying $|Q|$ above.

(3) Speedup of approximation using views. We further evaluated the benefit of using upper and lower approximations in terms of the speedups. We compared the time for evaluating upper and lower approximations Q_u and Q_l for Q together (using QAVsim and QAViso) against the time for evaluating Q directly (using gSim and VF2). To eliminate the noises from small approximations, we calculated the average evaluation time over approximations that have accuracy (by F-measure) above 0.6. The results are given in Figures 6(e), 6(j), 6(o) and 6(t), and tell us the following. (1) Upper and lower approximations are much more efficient to answer than directly evaluate Q on data graphs. Indeed, when all the views are used, for subgraph (simulation) queries, the total evaluation time of both Q_u and Q_l is 90 to 2000 (80 to 300) times faster than that of Q with VF2 (gSim) on YouTube. Similar for subgraph queries. (2) More views lead to better query plans for answering upper and lower approximation using views, as QAVsim contains an optimization procedure to select views of larger size to answer queries.

Exp-2: Efficiency. We also evaluated the efficiency of our algorithms. We found that they all took at most 2.7s for all queries on both data graphs with all the views.

Summary. From the experiments we find the following. (1) The closest upper and lower approximations are practical and effective in approximating graph pattern queries using views. About 65% (resp. 53%) simulation (resp. subgraph) queries have upper and lower approximations using a small number of views. (2) The approach is effective for both simulation and subgraph queries. Upper and lower approximations achieve accuracy (F_w) above 0.79 and 0.86 and scale with million graphs within 0.24s and 2.7s, for simulation and subgraph queries, respectively, while it takes 42s and 5382s to evaluate the queries directly. (3) Our algorithms are efficient: they take no more than 2.7s in all cases.

8. CONCLUSION

We have studied approximating simulation and subgraph queries using views. We have proposed a notion of upper and lower approximation of pattern queries *w.r.t.* a set of views. We have studied their properties and characterizations. We have also identified eight fundamental problems for approximation using views, and investigated the complexity and approximation-hardness. Based on the characterizations, we have developed efficient exact and approximation algorithms with provable bounds for computing the closest upper and lower approximations, complete or not. We have also developed characterizations and algorithms for subgraph query answering using views, an open question in [8]. These together give us a practical framework of query-driven approximation using views. Our experimental results have verified the effective and efficiency of our techniques and the framework. These results extend the use of views from exact query answering to query approximation.

The study of query approximation using views is still in its infancy. One issue is to study optimal upper and lower approximations when views are associated with costs. Another issue is to study optimal pair of upper and lower approximations covering the same part of the queries. The third topic is to extend the idea to, *e.g.,* relational queries. The fourth direction is to study approximations for dynamic data graphs.

9. REFERENCES

[1] DBpedia. *http://wiki.dbpedia.org/Downloads2015-04*.

[2] YouTube. *http://netsg.cs.sfu.ca/youtubedata/*.

[3] P. Barceló, L. Libkin, and M. Romero. Efficient approximations of conjunctive queries. *SICOMP*, 43(3):1085–1130, 2014.

[4] J. Brynielsson, J. Högberg, L. Kaati, C. Martenson, and P. Svenson. Detecting social positions using simulation. In *ASONAM*, 2010.

[5] S. Chaudhuri and P. G. Kolaitis. Can datalog be approximated? *JCSS*, 55(2):355–369, 1997.

[6] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *FTDB*, 2012.

[7] W. Fan, F. Geerts, Y. Cao, T. Deng, and P. Lu. Querying big data by accessing small data. In *PODS*, 2015.

[8] W. Fan, X. Wang, and Y. Wu. Answering graph pattern queries using views. In *ICDE*, 2014.

[9] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[10] M. R. Henzinger, T. Henzinger, and P. Kopke. Computing simulations on finite and infinite graphs. In *FOCS*, 1995.

[11] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, 1998.

[12] K. Karanasos. *View-based techniques for the efficient management of web data*. PhD thesis, 2012.

[13] W. Le, S. Duan, A. Kementsietsidis, F. Li, and M. Wang. Rewriting queries on SPARQL views. In *WWW*, 2011.

[14] J. Leskovec, A. Singh, and J. Kleinberg. Patterns of influence in a recommendation network. In *PAKDD*, 2006.

[15] Z. Shang and J. X. Yu. Auto-approximation of graph computing. *PVLDB*, 2014.

[16] L. Terveen and D. W. McDonald. Social matching: A framework and research agenda. *TOCHI*, 12(3):401–434, 2005.

[17] Y. Tian and J. M. Patel. Tale: A tool for approximate large graph matching. In *ICDE*, 2008.

[18] J. R. Ullmann. An algorithm for subgraph isomorphism. *JACM*, 23(1):31–42, 1976.

[19] N. E. Young. Greedy set-cover algorithms. In *Encyclopedia of Algorithms*. 2008.

[20] S. Zhang, J. Yang, and W. Jin. SAPPER: subgraph indexing and approximate matching in large graphs. *PVLDB*, 2010.