Relaxing Graph Pattern Matching With Explanations

Jia Li
SKLSDE Lab, Beihang University
lijia1108@buaa.edu.cn

Yang Cao
University of Edinburgh
yang.cao@ed.ac.uk

Shuai Ma
SKLSDE Lab, Beihang University
mashuai@buaa.edu.cn

ABSTRACT

Traditional graph pattern matching is based on subgraph isomorphism, which is often too restrictive to identify meaningful matches. To handle this, taxonomy subgraph isomorphism has been proposed to relax the label constraints in the matching. Nonetheless, there are many cases that cannot be covered. In this study, we first formalize taxonomy simulation, a natural matching semantics combing graph simulation with taxonomy, and propose its pattern relaxation to enrich graph pattern matching results with taxonomy information. We also design topological ranking and diversified topological ranking for top-k relaxations. We then study the top-k pattern relaxation problems, by providing their static analyses, and developing algorithms and optimization for finding and evaluating top-k pattern relaxations. We further propose a notion of explanations for answers to the relaxations and develop algorithms to compute explanations. These together give us a framework for enriching the results of graph pattern matching. Using real-life datasets, we experimentally verify that our framework and techniques are effective and efficient for identifying meaningful matches in practice.

KEYWORDS

Taxonomy simulation; pattern relaxation; query result explanation

1 INTRODUCTION

Graph pattern matching is being widely used in social network analysis, among other things. It is to find subgraphs in a large data graph that satisfy both the label and topological matching constraints carried by a pattern graph. However, traditional graph pattern matching is based on subgraph isomorphism [7], which requires identical label and topological matching and is often too restrictive to find matches in, *e.g.,* social search.

To handle this, taxonomy assisted subgraph isomorphism [5] has been proposed to capture more matches by relaxing label constraints, which makes use of a taxonomy of the labels such that a pattern node with label l' is allowed to match a data node with label l when l is a descendant of l' in the taxonomy. Nonetheless, not all meaningful matches can be covered by this, as shown below.

Example 1: Consider a real-life example taken from [21] and shown in Fig. 1. The data graph G_1 depicts a social travel network. A node denotes an entity labeled types such as river and restaurant; an edge indicates a relation between two entities, *e.g.,* newspaper$_1$

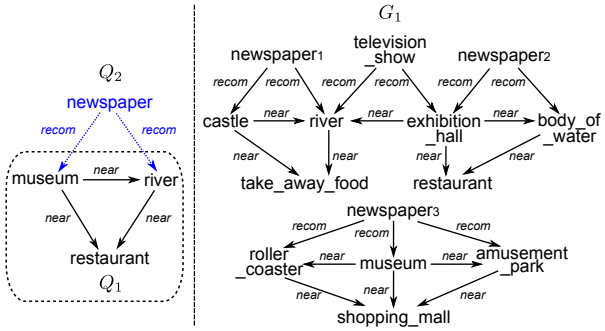


Figure 1: Querying knowledge network

recommends river (recom), and exhibition_hall is close to river (near). A tourist wants to find a travel plan that she could (1) visit a museum, (2) sightsee on a river close to the museum, and (3) dine at a restaurant close to the two places. This query can be specified by pattern graph Q_1 (in the dashed rectangular). Under subgraph isomorphism, one can verify that there is no match to Q_1 in G_1.

This remains the case even we adopt taxonomy assisted subgraph isomorphism. Consider a taxonomy graph T_1 from DBpedia [1] and shown in Fig. 2. It tells us that (a) exhibition_hall *is a* museum, while theater is not, and (b) take_away_food *is a* restaurant.

Nonetheless, due to strict topological matching constraints, under taxonomy subgraph isomorphism with T_1, the match result to Q_1 remains empty. However, in the presence of T_1, one can see that the subgraph consisting of river, exhibition_hall, take_away_food and restaurant is a sensible match to Q_1 in G_1. □

To tackle this, a natural idea is to further relax the matching constraints of taxonomy subgraph isomorphism, so that both label and structural matching semantics can be relaxed for graph pattern matching. One immediate approach is to, along the same lines as taxonomy subgraph isomorphism, combine taxonomy with graph simulation [11, 14, 18], which has recently been used to relax the topological matching constraints of subgraph isomorphism.

Unlike subgraph isomorphism which requires a *bijective* mapping function from pattern nodes to data nodes, graph simulation [11, 14, 18] is defined by a binary *relation* that preserves the child relationship. We refer to the "simulation version" of taxonomy subgraph isomorphism as *taxonomy simulation*, *i.e.,* observe the hierarchical "downward" *is-a* relationship between labels when computing simulation relations between pattern and data graphs. One can verify that taxonomy simulation can identify the sensible match in Example 1, and can find all matches that can be found by taxonomy subgraph isomorphism by the matching semantics.

Nonetheless, it is still not bullet-proof and can come short to capture matches in real-life, as shown by an experimental study below.

Using data and taxonomy graphs from DBpedia [1] and YAGO [2], we have conducted an experiment on the percentage of pattern graphs that have non-empty match results. We randomly generated

CIKM'17, November 6–10, 2017, Singapore.

DOI: 10.1145/3132847.3132992

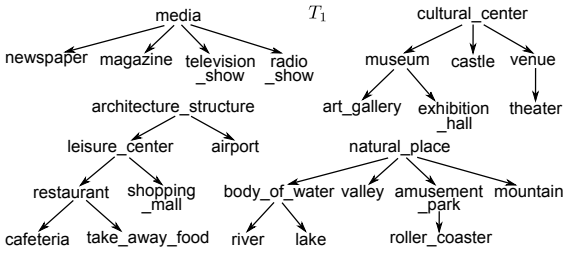


Figure 2: Taxonomy graph

pattern graphs of size ranging from 2 to 10 by drawing random labels from the data graphs, and queried DBpedia and YAGO with them via taxonomy simulation. The results are shown below.

$\|V_Q\|$	2	4	6	8	10
DBpedia	90%	18%	0%	0%	0%
YAGO	54%	2%	0%	0%	0%

Only 18% (resp. 2%) of the patterns with 4 nodes can find matches in DBpedia (resp. YAGO) via taxonomy simulation; no patterns with 6 or more nodes can identify matches. The percentages are even much lower with (taxonomy) subgraph isomorphism or graph simulation. However, by examining patterns and data graphs, we found there are indeed many sensible matches to those empty patterns.

Example 2: Recall Q_1 and G_1 in Example 1. Consider pattern Q_2 also in Fig. 1 that extends Q_1 with a new node with label newspaper (in blue), to further restrict that river and museum have to be recommended by a newspaper (blue dashed edges). Using taxonomy simulation, no match can be found for Q_2 in G_1. However, both newspaper and television_show are media as shown in the taxonomy graph T_1 in Fig. 2. Hence, a sensible match to Q_2 in G_1 is the one in Example 1, together with data node television_show. □

This example suggests that we make further use of taxonomy in an "upward" direction, to relax taxonomy simulation patterns to capture more sensible matches, *e.g.,* observing that newspaper and television_show are media when matching Q_2 in G_1.

Contributions. This paper studies how to enable graph pattern matching to capture more sensible matches in real-life complex data graphs, by relaxing taxonomy simulation queries.

(1) We formalize taxonomy simulation and propose its relaxation to enrich graph pattern matching results with a taxonomy in both the "downward" and "upward" directions (Section 2). We also design topological ranking and diversified topological ranking for top-k relaxations. The functions combine taxonomy simulation semantics, taxonomy graph, and the correlation between pattern and data graphs together when ranking relaxations (Section 3).

(2) We study the problem of computing top-k pattern relaxations *w.r.t.* the ranking functions (Section 4). We show that the problem is (a) in PTIME when the topological ranking function is used; and (b) NP-complete and APX-hard for approximation when diversification is also considered. For (a), we develop a PTIME exact algorithm by employing the Lawler's procedure [16] for top-k combinatorial optimization problems. For (b), we reduce the problem to the maximum dispersion problem, which enables us to use existing efficient algorithms for the latter to compute top-k diversified relaxations.

(3) We give an evaluation algorithm for answering top-k relaxed patterns while maximizing the sharing of computation (Section 5). It is built upon hierarchical connections among the pattern relaxations and bounded decremental taxonomy simulation algorithm.

(4) We study the minimum explanation problem to explain why a match to a relaxed pattern is returned, in terms of the essential part of the relaxation that captures the match (Section 6). We show that the problem is (a) in PTIME when only the topological ranking function is concerned, and (b) becomes NP-complete in general. We give a linear time optimal algorithm for (a) and a parameterized algorithm for (b), which returns explanations with accuracy parameterized by its time complexity.

(5) Using real-life graphs, we experimentally verify the effectiveness and efficiency of the techniques (Section 7). We find the following. (a) The top-k relaxations are effective: they find 11.9 times more answers, among which 74% are verified sensible. (b) The average evaluation time of relaxed patterns is 1.8 times faster than conventional evaluation method when $k = 15$, and the gap grows with larger k. (c) It can explain relaxations with accuracy above 85% without access to G, and achieve 99% accuracy with 2 data accesses.

Related work. We categorize related work as follows.

Graph pattern matching. Traditional matching is by subgraph isomorphism, which is NP-complete [7] and found often too restrictive to capture sensible matches [11]. To loosen the restriction, one direction is to relax matching semantics of isomorphism by adopting graph simulation based pattern matching [11, 13, 14, 18]. The other direction is to loosen the identical label matching with an ontology/taxonomy [5, 25, 26], Here we combine the two and use taxonomy assisted simulation as the base semantics of our framework.

Pattern relaxation. There has been work on pattern relaxation to generate relaxed queries over XML [3], RDF [9, 10, 15] and graph data [23]. They are based on (a) structure rewriting [3, 10, 23] or (b) predicate relaxation of *e.g.,* SPARQL pattern triples [9, 15]. Following (b), we use taxonomy to relax taxonomy simulation patterns on property graphs while retaining user specified pattern structures.

Our work differs from these work in the following. (1) *Targeted queries.* We study relaxation of taxonomy simulation based on graph simulation that is defined by a recursively computed relation over the general schemaless graph model, while existing works are for SPARQL and its variants over RDF [9, 10, 15] and subgraph isomorphism over graphs [23]. Their "implicit" pattern matching is defined via homomorphism or isomorphism, and is more restrictive due to the use of functions for valuations. (2) *Ranking relaxations.* The essence of query relaxation is the ranking of relaxations. However, ranking relaxations highly depends on the semantics of the queries to be relaxed. As a result, existing approaches for generating and ranking relaxations do not work well with taxonomy simulation due to its recursive nature and relation based structural matching. (3) *Additional feature.* We also study the explanation for answers to relaxed queries, which has not been addressed by existing works on query relaxation.

Multi-query optimization. There has been work on multi-query optimization for graph pattern (*e.g.,* [20]) and SPARQL (*e.g.,* [17]) relevant to the evaluation of relaxed patterns. They typically work by decomposing queries and using shared computation on common sub-queries. We adopt the general method for evaluating top-k relaxed patterns but differ from prior work in techniques. Indeed,

our approach is PTIME while [17, 20] are NP-hard. Moreover, we adopt bounded decremental algorithm for sharing computation.

Explanation. Related to the relaxation explanation are also query resilience [12], phenomenon explanation [22], why-not queries [4] and provenance [6], for relational queries. Different from theirs, we study explanations that are defined by relaxations for graph pattern queries, instead of relational query answering or provenance.

2 RELAXING TAXONOMY SIMULATION

In this section, we formalize taxonomy simulation (Section 2.1) and propose relaxations for taxonomy simulation patterns (Section 2.2).

2.1 Taxonomy Simulation

A *labeled directed graph* G is a triple (V, E, f), where V and E are sets of nodes and edges, respectively; and f is a total labeling function such that for each node $v \in V$ (resp. $e \in E$), $f(v)$ (resp. $f(e)$) is a label from an alphabet Σ_V (resp. Σ_E). The size $|G|$ of G is $|V| + |E|$.

Data graphs and *pattern graphs* are both labeled directed graphs, denoted by $G(V, E, f)$ and $Q(V_Q, E_Q, f_Q)$, respectively. Intuitively, node labels carry the description of entities, *e.g.,* place, job. Edge labels specify the relationships between respective entities.

Graph simulation. Data graph G *matches* pattern Q via *graph simulation*, denoted by $Q \trianglelefteq G$, if there exists a *left-total* binary match *relation* $R \subseteq V_Q \times V$ in G for Q such that (1) for each $(u, v) \in R$, u and v have the same label, *i.e.,* $f_Q(u) = f(v)$; and (2) for each edge $e = (u, u') \in E_Q$, there exists an edge $e' = (v, v') \in E$ such that $(u', v') \in R$ and $f_Q(e) = f(e')$, *i.e., e* and e' have the same label.

Intuitively, graph simulation preserves the label match and the child relationships between pattern and data graphs. It relaxes the topological matching constraints of subgraph isomorphism.

As shown in Example 1, there are *is-a* like category relationships between labels, which can be captured by a taxonomy.

Taxonomy graphs. A *taxonomy graph* is a labeled rooted forest, defined as $T(V_T, E_T, f_T)$, in which (1) an edge from node u to v represents an *is-a* relationship; and (2) f_T is an injective labeling function that maps nodes of V_T to distinct labels in the label set Σ_V.

Intuitively, T defines a specialization-generation hierarchy for labels in the data graphs. The distance from node u to u' in T, denoted by $\text{dist}_T(u, u')$, is the number of edges in the shortest path from u to u' if u is an ancestor of u' in T; and is $+\infty$ otherwise. We denote by $\text{desc}_T(u)$ the set of descendants of u (including u) in T. We also write $\text{dist}_T(u, u')$ and $\text{desc}_T(u)$ as $\text{dist}_T(f_T(u), f_T(u'))$ and $\text{desc}_T(f_T(u))$, respectively, when labels are concerned.

Taxonomy simulation. Given data graph $G(V, E, f)$, pattern $Q(V_Q, E_Q, f_Q)$ and taxonomy T, G *matches* Q *w.r.t.* T via *taxonomy simulation*, denoted by $Q \trianglelefteq_T G$, if there is a *left-total* binary match *relation* $R^T \subseteq V_Q \times V$ in G for Q such that (1) for each $(u, v) \in R^T$, $f(v) \in \text{desc}_T(f_Q(u))$; and (2) for each edge $e = (u, u') \in E_Q$, there exists an edge $e' = (v, v') \in E$ such that $(u', v') \in R$ and $f_Q(e) = f(e')$.

That is, taxonomy simulation observes the is-a relation among node labels when computing match relations of graph simulation.

When $Q \trianglelefteq_T G$, one can readily verify there exists a *unique maximum* match relation R_M^T in G for Q *w.r.t.* T. Given Q, G and T, the answers to Q in G *w.r.t.* T via taxonomy simulation, denoted by $Q(G)$, is the unique maximum match relation R_M^T in G for Q *w.r.t.* T.

Example 3: Consider Q_1 and G_1 in Fig. 1, and T_1 in Fig. 2. Note that $Q_1 \not\trianglelefteq G_1$, *i.e.,* G_1 does not match Q_1 via graph simulation. However, $Q_1 \trianglelefteq_T G_1$. Indeed, the maximum match relation in G_1 for Q_1 *w.r.t.* T_1 maps museum, river and restaurant of Q_1 to {exhibition_hall}, {river} and {take_away_food, restaurant} in G_1, respectively. □

Algorithm TSim. Given Q, G and T, we extend the $O(|Q||G|)$-time graph simulation algorithm [14] to compute taxonomy simulation in $O(|Q||G||T|)$-time such that we check label containment (condition (1) of taxonomy simulation definition) in $O(|T|)$ by traversing T instead of identical label checking. We also optimize it to $O(|Q||G|)$-time (denoted by TSim; omitted), with a bit string pre-computed offline in $O(|V_T||T|)$ that encodes descendant labels of each taxonomy node, so that the label containment is checked in $O(1)$.

2.2 Relaxations for Taxonomy Simulation

We next propose relaxations for taxonomy simulation patterns and a relaxation framework for graph pattern matching based on it.

Label relaxation. A *label relaxation* δ *w.r.t.* a taxonomy T is of form $l \to l'$ such that l' is an ancestor label of l in T.

Pattern relaxation. Consider pattern graph Q, a taxonomy graph T and positive integer μ. A *μ-bounded pattern relaxation* Δ for Q *w.r.t.* T is a set of label relaxations *w.r.t.* T such that, (1) for each $l \to l'$ in Δ, l is a label in Q and $\text{dist}_T(l', l) \leq \mu$; and (2) for any two label relaxations $l_1 \to l_1'$ and $l_2 \to l_2'$ in Δ, $l_1 \neq l_2$. When it is clear from the context, we simply call Δ a pattern relaxation for Q *w.r.t.* T.

Let $Q \oplus \Delta$ denote the pattern derived from Q by replacing each occurrence of l with l' in Q for each $l \to l'$ in Δ. We refer to $Q \oplus \Delta$ as the *relaxed pattern* of Q *w.r.t.* Δ.

Intuitively, μ is to bound the distance of label relaxations in Δ, so that changes to pattern graphs by Δ can be controlled.

Example 4: Recall Q_2 and T_1 from Example 2. When $\mu = 2$, $\Delta = \{\delta = \text{newspaper} \to \text{media}\}$ is a 2-bounded pattern relaxation for Q_2 *w.r.t.* T_1. The relaxed pattern $Q_2 \oplus \Delta$ is derived from Q_2 by replacing newspaper with media in Fig. 1. □

A relaxation framework. We next propose a relaxation framework for graph pattern matching, built upon taxonomy simulation. Given pattern Q, data graph G and taxonomy T, it works as follows.

(1) It generates and ranks relaxations for Q *w.r.t.* T, including Q.

(2) It evaluates top ranked relaxations for Q in G.

(3) It finally explains relaxations to the end user by showing why matches are found and returned by the relaxations.

In the sequel, we discuss step (1) in Sections 3 and 4, step (2) in Section 5, and step (3) in Section 6.

3 RANKING PATTERN RELAXATIONS

For a pattern Q, a data graph G, a taxonomy T and a positive integer μ, there are up to exponentially many μ-bounded pattern relaxations for Q *w.r.t.* T. This suggests that we define functions to rank the relaxations and compute the top-k relaxations accordingly.

Below we design two ranking functions: *topological ranking* to measure pattern relaxations in terms of their relaxation distances *w.r.t.* Q, with G and T taken into account together(Section 3.1); and *diversified-topological ranking* which combines the topological

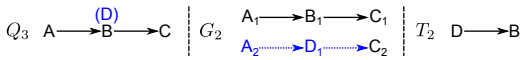


Figure 3: Relaxation ratio

function with a diversification function (Section 3.2). Based on the functions, we introduce two top-k pattern relaxation problems.

3.1 Topological Ranking

In real world, one typically wants to capture more match results while ensuring the results are sensible. Motivated by this, we define a bi-criteria relaxation ranking function, referred to as the *topological ranking*, with two components: (a) the relaxation ratio to measure the quality and accuracy of relaxed patterns in terms of their distance to the original pattern and (b) the information ratio to estimate the effectiveness of the relaxations in terms of their ability to capture matches. We present it below.

(a) Relaxation ratio. The *relaxation ratio* of label relaxation $\delta = l \to l'$ *w.r.t.* taxonomy T for $Q(V_Q, E_Q, f_Q)$, denoted by $\gamma_Q(\delta)$, is defined as

$$\sum_{u \in V_Q, f_Q(u)=l} \mathsf{rank}_Q(u) \cdot \rho(\mathsf{dist}_T(l', l)),$$

where $\mathsf{rank}_Q(u)$ denotes the number of nodes u' in Q that can reach u via a directed path from u' to u, and $\rho(x)$ is a monotonically increasing function that normalizes the weight of $\mathsf{dist}_T(l', l)$. Common choices for $\rho(x)$ are $\rho(x) = x$ and e^x, as frequently used in measuring social positions [24]. We use $\rho(x) = e^x$ by default.

Intuitively, larger $\mathsf{dist}_T(l', l)$ gives a higher chance for node u to find matches via taxonomy simulation. This effect is amplified by $\mathsf{rank}_Q(u)$ which observes the *recursive* nature of taxonomy simulation. Indeed, by taxonomy simulation, relaxation on u of Q will possibly introduce more matches to nodes in Q that can reach u (ancestors), but not descendants of u.

Example 5: Consider pattern Q_3, data graph G_2 and taxonomy T_2 in Fig. 3. The match result for Q_3 in G_2 via taxonomy simulation contains nodes A_1, B_1, C_1 and C_2. A pattern relaxation $\Delta = \{B \to D\}$ on node B in Q_3 turns data node D_1 to a match of node D in $Q_3 \oplus \Delta$, and further its parent data node A_2 to a match of pattern node A. However, the relaxation does not change the match status of the child node C_2 of D_1 in G_2. Indeed, $\mathsf{rank}_{Q_3}(C) = 3 > \mathsf{rank}_{Q_3}(B) = 2 > \mathsf{rank}_{Q_3}(A) = 1$. □

Observe that, the smaller $\gamma_Q(\delta)$ is, the closer is the relaxed pattern *w.r.t.* δ to the original pattern Q.

(b) Information ratio. Consider pattern $Q(V_Q, E_Q, f_Q)$ and data graph $G(V, E, f)$. The *information ratio* $\mathcal{I}_{(Q,G)}(\delta)$ of label relaxation $\delta = l \to l'$ *on* Q *in* G is defined as $|\mathsf{cand}_{(G,T)}(l)|/|\mathsf{cand}_{(G,T)}(l')|$, where $\mathsf{cand}_{(G,T)}(l)$ is the set of nodes v in G with $f(v) \in \mathsf{desc}_T(l)$.

Intuitively, $\mathcal{I}_{(Q,G)}(\delta)$ captures the impacts of δ on Q by observing the candidate match information from G. A smaller $\mathcal{I}_{(Q,G)}(\delta)$ gives a higher potential to introduce more matches.

Topological ranking function. Given a pattern graph Q, a data graph G, a taxonomy graph T, and a pattern relaxation Δ for Q *w.r.t.* T, the *topological ranking function* of Δ on Q for G *w.r.t.* T, denoted by $\Gamma(Q, \Delta)$, is defined as $\sum_{\delta \in \Delta} \gamma_Q(\delta) \cdot \mathcal{I}_{(Q,G)}(\delta)$.

Intuitively, $\Gamma(Q, \Delta)$ is a bi-criteria function specialized for G that measures the quality of the relaxed pattern $Q \oplus \Delta$ in terms of their closeness to Q and their ability to capture matches. By minimizing $\Gamma(Q, \Delta)$, we can on one hand maximize the accuracy of matches to $Q \oplus \Delta$ in G via the relaxation ratio, and on the other hand maximize the ability to capture matches in G via the information ratio.

This motivates us to study the *top-k pattern relaxation problem* (kPR), formulated as follows.

*Top-k pattern relaxation problem (*kPR*).* Let $\mathcal{U}_\mu(Q, T)$ be the set of all μ-bounded pattern relaxations for pattern Q *w.r.t.* taxonomy T. Given Q, G, T, and integers μ and k, problem kPR is to find a k-set $S \subseteq \mathcal{U}_\mu(Q, T)$, such that

$$S = \operatorname*{arg\,min}_{S' \subseteq \mathcal{U}_\mu(Q,T), |S'|=k} \sum_{\Delta \in S'} \Gamma(Q, \Delta).$$

That is, kPR is to identify a set of k μ-bounded pattern relaxations with the minimum total topological ranking.

Example 6: Consider Q_2 and G_1 in Fig. 1, and T_1 in Fig. 2. Suppose that there are 7 isolated nodes in G_1 labeled magazine, radio_show, venue, theater, valley, mountain and airport (not shown in Fig. 1). Assume $k = 2$ and $\mu = 2$. The label relaxations and part of possible pattern relaxations are listed in the table below.

Pattern relaxations	$\Gamma(Q_2, \Delta_i)$
$\Delta_1 = \{\delta_1 =$ newspaper $\to$ media $\}$	1.359
$\Delta_2 = \{\delta_2 =$ museum $\to$ cultural_center $\}$	2.175
$\Delta_3 = \{\delta_3 =$ river $\to$ natural_place $\}$	3.695
$\Delta_4 = \{\delta_4 =$ river $\to$ body_of_water $\}$	4.077
$\Delta_5 = \{\delta_5 =$ restaurant $\to$ leisure_center $\}$	7.249
$\Delta_6 = \{\delta_6 =$ restaurant $\to$ architecture.. $\}$	14.778
...	...

Observe the following. (a) One can verify that S=$\{\Delta_1, \Delta_2\}$ is the top-2 pattern relaxations for kPR. (b) $\Gamma(Q_2, \Delta_3)$ is smaller than $\Gamma(Q_2, \Delta_4)$, although $\mathsf{dist}_T(\text{natural\_place}, \text{river}) = 2$ is larger than $\mathsf{dist}_T(\text{body\_of\_water}, \text{river}) = 1$. This is because the information ratio ranks relaxations with more potential matches in G_1 higher, despite relaxation ratio favors relaxations with smaller distance. □

3.2 Diversified Topological Ranking

It is desirable that the relaxed patterns are not only close to original patterns, but also diverse to provide more information in the match results. Hence, we introduce diversified topological ranking.

Diversification. To characterize the diversity of a set of pattern relaxations, we define a distance function to measure the "dissimilarity" of two pattern relaxations. For any two pattern relaxations Δ_1 and Δ_2 for Q, we define the *similarity distance* between Δ_1 and Δ_2, denoted by $\theta_Q(\Delta_1, \Delta_2)$, to be

$$\frac{|L(Q \oplus \Delta_1) \cap L(Q \oplus \Delta_2)|}{|L(Q \oplus \Delta_1) \cup L(Q \oplus \Delta_2)|},$$

where $L(Q)$ denotes the set of labels in pattern graph Q. That is, the similarity distance measures the overlap of labels in the relaxed patterns of the relaxations.

Example 7: Recall Example 6. We have the following: (a) $\theta_{Q_2}(\Delta_1, \Delta_2) = \frac{2}{6}$; and (b) $\theta_{Q_2}(\Delta_7, \Delta_8) = 0$, in which Δ_7=$\{\delta_1, \delta_2\}$ and Δ_8=$\{\delta_3, \delta_5\}$; that is, there are no overlapped labels between $Q_2 \oplus \Delta_7$ and $Q_2 \oplus \Delta_8$. Thus, Δ_7 and Δ_8 are most dissimilar to each other, among others. One can verify that the connected component in the bottom of G_1 is in the match result of $Q_2 \oplus \Delta_8$. The result is a bit inconsistent with the target of Q_2, turning a scenery trip to an entertainment trip, as topological ranking is not considered here yet. □

We next combine topological ranking and diversification.

Diversified topological ranking function. Consider a set S of k pattern relaxations $\Delta_1, \ldots \Delta_k$ for pattern graph Q *w.r.t.* taxonomy T. We define the *diversified topological ranking function* $F(Q, S)$ over the pattern relaxation set S as

$$\lambda \cdot (k-1) \sum_{\Delta_i \in S} \widehat{\Gamma}(Q, \Delta_i) + 2 \cdot (1-\lambda) \sum_{\Delta_i \in S, \Delta_j \in S, i<j} \theta_Q(\Delta_i, \Delta_j),$$

where $\lambda \in [0, 1]$ is a user specified parameter, $\widehat{\Gamma}(Q, \Delta_i) = \frac{\Gamma(Q, \Delta_i)}{|V_Q| \cdot |L(Q)| \cdot e^{\mu}}$ is a normalized topological function, and $|L(Q)|$ is the number of labels appeared in Q. We scale up the topological ranking component with a factor of $(k-1)$ since there are $\frac{k \cdot (k-1)}{2}$ numbers in the diversification component, as opposed to k numbers in the topological ranking.

Diversified top-k pattern relaxation problem ($\mathsf{kPR}_{\mathsf{DF}}$*).* We next introduce the *diversified top-k pattern relaxation problem*, denoted by $\mathsf{kPR}_{\mathsf{DF}}$, along the same lines as kPR. More specifically, given Q, G, T, and two integers μ and k, $\mathsf{kPR}_{\mathsf{DF}}$ is to find a k-set $S \subseteq \mathcal{U}_\mu(Q, T)$ (recall $\mathcal{U}_\mu(Q, T)$ from the kPR problem), such that

$$S = \underset{S' \subseteq \mathcal{U}_\mu(Q,T), |S'|=k}{\arg\min} F(Q, S').$$

That is, $\mathsf{kPR}_{\mathsf{DF}}$ aims to find a set S of k μ-bounded pattern relaxations for Q *w.r.t.* T that minimizes $F(Q, S)$.

Example 8: Recall Example 6. One can verify that (a) when $\lambda = 1$, *i.e.,* when only topological function is considered, a top-2 set is $\{\Delta_1, \Delta_2\}$; and (b) when $\lambda = 0$, *i.e.,* when only diversification function is considered, a top-2 set is $\{\Delta_7 = \{\delta_1, \delta_2\}, \Delta_8 = \{\delta_3, \delta_5\}\}$. Moreover, (c) when $0.823 < \lambda < 0.924$, $\{\Delta_3, \Delta_7\}$ is the best set; (d) when $\lambda \leq 0.823$, $\{\Delta_7, \Delta_8\}$ is the best; and (e) when $\lambda \geq 0.924$, $\{\Delta_1, \Delta_2\}$ is the best. □

4 FINDING TOP-K RELAXATIONS

We next develop algorithms for the two top-k pattern relaxation problems, which, together with Section 3, provide the foundation to step (1) of the framework in Section 2.2. We focus on problem kPR in Section 4.1 first, and move on to $\mathsf{kPR}_{\mathsf{DF}}$ in Section 4.2.

4.1 Finding Top-k Relaxations for kPR

We first present the main result for kPR. We assume that the numbers $|\mathsf{cand}_{(G,T)}(l)|$ for labels l in G have already been precomputed. Note that this can be done by a $O(|V_T||V|)$-time offline computing.

Theorem 1: *There exists an algorithm that computes the top-k* μ*-bounded pattern relaxations for* Q w.r.t. T *in* G *within* $O(\mu k |V_Q||Q|)$*-time, independent of* $|G|$ *and* $|T|$. □

As a proof, we next give such an algorithm. It utilizes the *Lawler's procedure* [16]. Lawler's procedure has the following *property*: for an optimization problem that can be formulated in integer programming with n 0-1 variables, if the optimal (top-1) solution can be found in $c(n)$ time, then the top-k solutions can be found in $O(k \cdot c(n)) + B$ time, where B is the total time for branching the space of feasible solutions into subspaces.

Algorithm relTF. The algorithm, denoted by relTF, is shown in Fig. 4. It uses (a) a list L_{TR} to store the top-k pattern relaxations identified so far; and (b) a priority queue Q to cache candidate top-k pattern relaxations, which will also be used to divide and branch the search space. After initialization (line 1), it generates the collection $\mathcal{L}_1$ of candidate label relaxations for all labels in Q with relaxed distance bounded by μ (lines 2-3). Here $\mathcal{L}_1$ includes, for each label ℓ_i in Q, a list L_i of label relaxations $\ell_i \to \ell'$ ($0 \leq \mathsf{dist}_T(\ell', \ell_i) \leq \mu$). It then finds top-1 pattern relaxation Δ_1 within $\mathcal{L}_1$ via procedure topRel, which is then pushed into Q together with $\mathcal{L}_1$ (line 4).

Input: Q, G, T, two positive integers μ and k.
Output: A list L_{TR} of top-k pattern relaxations.
1. $L_{\mathsf{TR}} := [\,]$; $Q := nil$; $K := 1$;
2. **for each** label ℓ_i in Q **do** /* assume Q has labels $\ell_1, \ldots, \ell_m$ */
3. generate a list L_i of label relaxations for ℓ_i bounded by μ in T;
4. $\mathcal{L}_1 := (L_1, \ldots, L_m)$; $\Delta_1 := \mathsf{topRel}(Q, G, \mathcal{L}_1)$; $Q.\mathsf{push}(\langle \Delta_1, \mathcal{L}_1 \rangle)$;
5. **while** $Q \neq \emptyset$ **do**
6. **if** $K = k + 1$ **then break**;
7. $\langle \Delta_K, \mathcal{L}_K \rangle := Q.\mathsf{pop}()$; /*$\Delta_K$ *has minimum* $\Gamma(Q, \Delta_K)$ *in* Q*/
8. append Δ_K to L_{TR};
9. sub-collections $\mathcal{L}^{s_1}, \ldots, \mathcal{L}^{s_m} := \mathsf{LawlerBranch}(\Delta_K, \mathcal{L}_K)$;
10. **for** i in $[1, m]$ **do** $\Delta_i := \mathsf{topRel}(Q, G, \mathcal{L}^{s_i})$; $Q.\mathsf{push}(\langle \Delta_i, \mathcal{L}^{s_i} \rangle)$;
11. $K := K + 1$;
12. **return** L_{TR};

Procedure topRel(Q, G, $\mathcal{L}$)

Input: Q, G, a collection $\mathcal{L}$ of candidate label relaxations.
Output: The best pattern relaxation Δ within $\mathcal{L}$ *w.r.t.* $\Gamma(Q, \Delta)$.
1. **for each** i in $[1, m]$ **do** $\delta^i_{\min} := \arg\min_{\delta \in L_i} \gamma_Q(\delta) \cdot \mathcal{I}_{(Q,G)}(\delta)$;
2. $\Delta := \{\delta^1_{min}, \delta^2_{min}, \ldots, \delta^m_{min}\}$; **return** Δ;

Figure 4: Algorithm relTF

Procedure topRel. Given Q, G and a collection $\mathcal{L}$ of candidate label relaxations, procedure topRel is also shown in Fig. 4. It generates the top-1 pattern relaxation by selecting label relaxations δ with minimum $\gamma_Q(\delta) \cdot \mathcal{I}_{(Q,G)}(\delta)$ for each label in Q.

Correctness & Complexity. The correctness of relTF is ensured by the property of Lawler's procedure. It is in $O(\mu k |V_Q||E_Q|)$ time. Indeed, topRel is in $O(\mu |V_Q||E_Q|)$ and the time for branching candidate label relaxations is $O(k|V_Q|^2)$. By the property of Lawler's procedure, relTF is in $O(k \cdot \mu |V_Q||E_Q|) + O(k|V_Q|^2) = O(\mu k |V_Q||Q|)$.

4.2 Top-k Diversified Relaxations for $\mathsf{kPR}_{\mathsf{DF}}$

We next study the problem $\mathsf{kPR}_{\mathsf{DF}}$. While kPR can be solved in PTIME, $\mathsf{kPR}_{\mathsf{DF}}$ is intractable.

Theorem 2: *(1) The decision problem of* $\mathsf{kPR}_{\mathsf{DF}}$ *is* NP*-complete. (2) The optimization problem of* $\mathsf{kPR}_{\mathsf{DF}}$ *is* APX*-hard to approximate.* □

Despite of the hardness and inapproximability, we develop an algorithm for kPR_{DF}, denoted by relDF, by reducing kPR_{DF} to the *maximum dispersion problem* (maxDP), to utilize algorithms for the later. Here problem maxDP is to find a subgraph G'_c induced by a k-node set V_k from a weighted complete graph G_c, with the maximum sum of (positive) edge weights. It is a well-studied maximization problem, with a number of efficient exact, approximation and heuristic algorithms already developed [8].

Algorithm relDF. We next present relDF by giving the reduction from kPR_{DF} to maxDP. The subtlety is that kPR_{DF} is a minimization problem while maxDP is maximization. Nonetheless, we guarantee that optimal solutions to maxDP give us optimal answers to kPR_{DF}.

Given Q, G, T, μ and k, algorithm relDF constructs G_c of maxDP as follows. (1) Each μ-bounded pattern relaxation Δ for Q is encoded by a node u_Δ in G_c. (2) For any two nodes $u_{\Delta_1}, u_{\Delta_2}$ in G_c, the weight $w(e)$ for edge $e = (u_{\Delta_1}, u_{\Delta_2})$ is

$$M - \lambda \sum_{i\in\{1,2\}} \widehat{\Gamma}(Q, \Delta_i) - 2(1-\lambda)\theta_Q(\Delta_1, \Delta_2),$$

where $M = 2\lambda \cdot \max_{\Delta\in U} \widehat{\Gamma}(Q, \Delta) + 2(1-\lambda)$, in which U is the set of all μ-bounded pattern relaxations for Q *w.r.t.* T. Note that G_c is an instance of maxDP since $w(e) > 0$.

It is easy to see that a k-node set V_k encodes the set of k pattern relaxations for Q. Thus, algorithm relDF simply returns k pattern relaxations encoded by V_k for maxDP.

Proposition 3: *If V_k is the optimal solution to G_c of* maxDP*, then the set of k pattern relaxations encoded by nodes in V_k is the optimal solution to Q, G, T, μ and k of* kPR_{DF}. □

Proposition 3 ensures the following: relDF *always returns the top-k pattern relaxations for* kPR_{DF} *as long as the reduced* maxDP *returns exact answers.* To see this holds, observe the following. Let S_k be the set of relaxations encoded by nodes in V_k. The sum W_k of edge weights in the subgraph induced by V_k is $\sum_{u,v\in S_k, u\neq v} w(u,v) =$

$$\frac{k(k-1)}{2} \cdot M - \lambda(k-1)\sum_{\Delta\in S_k} \widehat{\Gamma}(Q, \Delta) - 2(1-\lambda) \sum_{\Delta_i,\Delta_j\in S_k, i<j} \theta_Q(\Delta_i, \Delta_j).$$

Thus, $W_k = \frac{k(k-1)}{2} \cdot M - F(Q, S_k)$. Since V_k is the optimal solution to G_c of maxDP, thus W_k is the maximum among all such k-node sets in G_c. Therefore, $F(Q, S_k)$ is the minimum among all k-sets of pattern relaxations for Q of kPR_{DF}.

Remark. The rationale behind the reduction is that μ and $L(Q)$ are typically small, so that it is affordable to compute all the μ-bounded pattern relaxations for Q beforehand. Indeed, the maximum height of the taxonomy trees (*i.e.,* maximum value for μ) in DBpedia taxonomy is only 6 and the average height (*i.e.,* average μ value) is 2.29 (see Section 7 for details).

5 ANSWERING PATTERN RELAXATIONS

In this section, we further study the evaluation of the top-k relaxed patterns produced by algorithms in Section 4, providing foundation to step (2) of the framework in Section 2.2.

Given Q, G, T and k pattern relaxations $\Delta_1, \ldots, \Delta_k$, we aim to compute answers to the relaxed patterns $Q \oplus \Delta_1, \ldots, Q \oplus \Delta_k$ in G *w.r.t.* T. A naive solution is to evaluate the k relaxed patterns one by one. However, observe that they share the same structure and interrelated labels. Inspired by this, we develop an algorithm

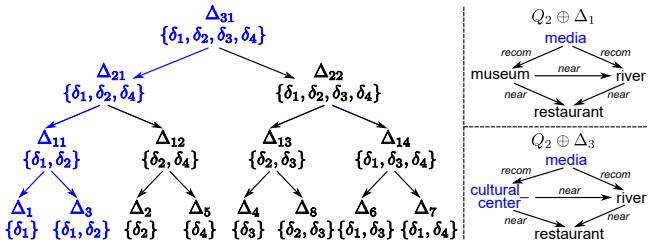


Figure 5: Minimum pairing tree

that maximally utilizes computation sharing among the relaxed patterns, based on the semantics of taxonomy simulation.

Algorithm evalPR. The algorithm, denoted by evalPR, works in two steps: (1) constructing the *minimum pairing tree* $\mathcal{T}$, a structure to organize the evaluation of relaxed patterns while maximizing shared computation; and (2) bounded decremental taxonomy simulation that carries out the computation *boundedly* over $\mathcal{T}$.

(1) Minimum pairing tree construction. The *minimum pairing tree* $\mathcal{T}$ of the k pattern relaxations $\Delta_1, \ldots, \Delta_k$ for Q is a hierarchical organization of the relaxations as follows. (a) Every node of $\mathcal{T}$ is a pattern relaxation. (b) $\mathcal{T}$ has k leaves (at level 0), each corresponding to one of the k pattern relaxations $\Delta_1, \ldots, \Delta_k$ for Q. (c) Nodes at level $i+1$ are the *minimum paring* of pattern relaxations at level i ($i \in [0, \lceil \log k \rceil - 1]$). Here a *pairing* of pattern relaxations $\Delta_1, \ldots, \Delta_n$ is a set P of $\lceil \frac{n}{2} \rceil$ pattern relaxations $\Delta'_1, \ldots, \Delta'_{\lceil \frac{n}{2} \rceil}$ such that (i) each $\Delta'_j (j \in [1, \lceil \frac{n}{2} \rceil])$ at level $i+1$ is a union of two pattern relaxations Δ_p and Δ_q ($p, q \in [1, n]$) at level i, where Δ'_j combines all label relaxations in Δ_p and Δ_q together such that each label l is relaxed to the further label l' if two label relaxations are attached to it; (ii) Δ'_j and $\Delta'_{j'}$ correspond to different pairs of relaxations if $j \neq j'$.

A pairing set P of a set S of n pattern relaxations is *minimum* for pattern graph Q *w.r.t.* taxonomy T if

$$\sum_{\Delta'\in P} \sum_{\Delta_i\in\Delta'; i=1,2} \sum_{u\in V_Q} |\text{cand}_{(G,T)}(f_{Q\oplus\Delta'}(u)) \setminus \text{cand}_{(G,T)}(f_{Q\oplus\Delta_i}(u)|$$

is the minimum among all pairings of relaxations in S. Here $\Delta_i \in \Delta' (i = 1, 2)$ denotes that Δ' combines Δ_1 and Δ_2 and $\text{cand}_{(G,T)}(l)$ is the set of nodes v in G such that $f(v) \in \text{desc}_T(l)$.

Intuitively, the minimum pairing set P of S groups relaxations in S into pairs such that, by first evaluating relaxed patterns *w.r.t.* relaxations in P as shared computation, and then "recovering" answers to relaxed patterns *w.r.t.* S, the total computation can be minimized. The minimum pairing tree organizes such minimization recursively.

Here $\mathcal{T}$ can be built in $O(k^{2.5} \log k)$ time, by invoking $\lceil \log k \rceil$ times of the *maximum weighted matching* algorithm [19].

Example 9: Recall label relaxations $\delta_1, \delta_2, \ldots, \delta_6$ in the table in Example 6. When $k = 8$ and $\mu = 2$, one can verify that $\Delta_1, \Delta_2, \ldots, \Delta_8$ are the top-8 pattern relaxations *w.r.t.* the topological ranking function, as shown in the leaves in Fig. 5. Algorithm evalPR firstly constructs the minimum pairing tree as shown in Fig. 5. Nodes at level $i+1$ are the *minimum paring* of pattern relaxations at level i ($i \in [0, 2]$), *e.g.,* Δ_{11},Δ_{12},...,Δ_{14} at level 1 is the minimum pairing of Δ_1,Δ_2,...,Δ_8 at level 0, and Δ_{11} is the union of Δ_1 and Δ_3. □

(2) Bounded decremental evaluation. After constructing $\mathcal{T}$ of $\Delta_1, \ldots, \Delta_k$, evalPR then evaluates relaxed patterns of Q *w.r.t.* relaxations

in $\mathcal{T}$, from root to leaves. For each node u, it evaluates the relaxed pattern *w.r.t.* relaxation in u by *reusing* answers to the relaxed pattern in the parent node pre(u) of u in $\mathcal{T}$, via BDeval (see below). It finally returns answers to relaxed patterns in leaves of $\mathcal{T}$.

Here procedure BDeval *decrementally* computes $(Q \oplus \Delta_l)(G)$ and $(Q \oplus \Delta_r)(G)$ from $(Q \oplus \Delta')(G)$ if Δ' combines Δ_l and Δ_r in $\mathcal{T}$, by iteratively removing matches in $(Q \oplus \Delta')(G)$ that are found not in $(Q \oplus \Delta_{l(r)})(G)$, similar to the incremental graph simulation algorithm [11]. Following [11], BDeval is also *bounded* in that its cost is determined by the changes in input and output, *i.e.,* $|(Q \oplus \Delta')(G) \setminus (Q \oplus \Delta_{l(r)})(G)|$ and the size of an *affected area* in G that any algorithm has to access, and is *not directly dependent of* $|G|$.

Example 10: Continue Example 9. Algorithm evalPR then carries out the decremental evaluation over $\mathcal{T}$ in Fig. 5. Take $Q_2 \oplus \Delta_1$ and $Q_2 \oplus \Delta_3$ for example. evalPR evaluates them by starting from root Δ_{31}, to Δ_{21}, and then to Δ_{11} sequentially. Firstly, (a) evalPR finds that the connected component C_1 containing newspaper$_1$ in G_1 is the match result to $Q_2 \oplus \Delta_{31}$, and (b) further verifies that C_1 is also the answer to $Q_2 \oplus \Delta_{21}$. Then, (c) evalPR removes nodes body_of_water and newspaper$_2$ from C_1, yielding the match result C_2 to $Q_2 \oplus \Delta_{11}$. Finally, (d) evalPR removes nodes castle and newspaper$_1$ from C_2 as the match result to $Q_2 \oplus \Delta_1$, and (e) also finds that the answer C_2 in (c) is exactly the match result to $Q_2 \oplus \Delta_3$. □

6 EXPLAINING RELAXATIONS

In Section 5, we have studied the evaluation of top-k relaxed patterns. A natural follow-up question is to ask, why certain nodes in G are returned after relaxation? Below we answer this question, by studying the *match-relaxation explanation problem*. This gives the foundation to step (3) of the framework in Section 2.2.

Minimum explanation. Given pattern graph Q, data graph G, taxonomy T, pattern relaxation Δ, and a node v of G that is in the match result $(Q \oplus \Delta)(G)$ to the relaxed pattern $Q \oplus \Delta$ in G, an *explanation* for v *w.r.t.* Δ, denoted by $\mathcal{E}_\Delta(v)$, is a subset of Δ such that v is in $(Q \oplus \mathcal{E}_\Delta(v))(G)$. Intuitively, $\mathcal{E}_\Delta(v)$ explains why v is in the match result to $Q \oplus \Delta$ in G using label relaxations in Δ. In particular, when v is already in the match result to Q in G, $\emptyset$ is an explanation for v *w.r.t.* Δ.

A *minimum explanation* $\mathcal{E}^m_\Delta(v)$ for v *w.r.t.* Δ is an explanation of minimum cardinality among all such explanations. Intuitively, $\mathcal{E}^m_\Delta(v)$ is the minimum part of Δ that is essential for relaxing Q so that v can be captured by $Q \oplus \Delta$.

Example 11: Recall Q_2, G_1 and $\Delta_3 = \{\delta_1, \delta_2\}$ from Example 10. The match result to the relaxed pattern $Q_2 \oplus \Delta_3$ in G_1 is {newspaper$_1$, television_show, castle, river, exhibition_hall, take_away_food, restaurant }. One can verify that (1) the minimum explanation $\mathcal{E}^m_{\Delta_3}$(exhibition_hall) for exhibition_hall *w.r.t.* Δ_3 is $\{\delta_1\}$, a subset of Δ_3; and (2) the minimum explanation for castle *w.r.t.* Δ_3 is also a subset of Δ_3, *i.e.,* $\mathcal{E}^m_{\Delta_3}$(castle) = $\{\delta_2\}$. □

Match-relaxation explanation problem. Given a pattern Q, data graph G, taxonomy T, k pattern relaxations $\Delta_1, \ldots, \Delta_k$ for Q *w.r.t.* T and their match results $(Q \oplus \Delta_1)(G), \ldots, (Q \oplus \Delta_k)(G)$, integer $i \in [1, k]$, and node v in $(Q \oplus \Delta_i)(G)$, the *match-relaxation explanation problem* (MRE) is to compute the minimum explanation $\mathcal{E}^m_{\Delta_i}(v)$ for v *w.r.t.* Δ_i.

The study of MRE allows the user to ask for the minimum explanation about why a particular node in G is in the match results to the relaxed patterns, after computing the top-k relaxations (Section 4) and their answers in G (Section 5). Nonetheless, it is intractable.

Theorem 4: *The decision problem of* MRE *is* NP*-complete.* □

Despite of the complexity, we below develop practical algorithms for MRE. In particular, we study two instances of MRE, denoted by MRE_{TF} and MRE_{DF}, respectively, to explain top-k relaxations computed by algorithm relTF and relDF in Section 4.

(1) Instance MRE_{TF}. When we consider top-k topological relaxations, MRE becomes tractable. Indeed, the minimum explanation can be found in linear time for MRE_{TF}, based on the property below.

Proposition 5: *For any pattern Q, data graph G, taxonomy T and top-k pattern relaxations $\Delta_1, \ldots, \Delta_k$ identified by algorithm* relTF*, for any $i \in [1, k]$ and any node v in $(Q \oplus \Delta_i)(G)$, there must exist $j \in [1, k]$ such that Δ_j is the minimum explanation for v* w.r.t. Δ_i. □

Proposition 5 tells us that the minimum explanation for v *w.r.t.* the top-k topologically ranked relaxations must also be one of those k relaxations. It gives us a *linear* time algorithm, denoted by expTF, for MRE_{TF} even *without access to G or T*: do a linear scan of $(Q \oplus \Delta_1)(G), \ldots, (Q \oplus \Delta_k)(G)$, and return Δ_j with minimum cardinality such that v is in $(Q \oplus \Delta_j)(G)$. Note that algorithm expTF is *optimal* since it only parses the input in linear time.

(2) Instance MRE_{DF}. We next study MRE_{DF} to explain the top-k diversified pattern relaxations identified by relDF.

Due to the diversification component in the ranking function for problem kPR_{DF}, Proposition 5 does not hold anymore, and any possible top-k pattern relaxations can be returned by relDF.

In spite of the hardness of MRE, we provide a *parameterized* algorithm, denoted by expDF for MRE in general case, and thus also works for MRE_{DF}. It returns an explanation $\mathcal{E}_{\Delta_i}(v)$ for v *w.r.t.* relaxation Δ_i for Q with a *parameter* M, indicating the maximum times of taxonomy simulation evaluation it can invoke to compute $\mathcal{E}_{\Delta_i}(v)$. The parameter M balances the quality of the answer, *i.e.,* the size of $\mathcal{E}_{\Delta_i}(v)$, and the time complexity of the algorithm: a larger M gives a smaller $\mathcal{E}_{\Delta_i}(v)$ with longer time.

Algorithm expDF(M) works in two steps as follows.

(1) It first finds relaxation Δ_j ($j \in [1, k]$) with minimum cardinality satisfying (a) $\Delta_j \subseteq \Delta_i$ and (b) $v \in (Q \oplus \Delta_j)(G)$.

(2) It then tests label relaxations $\delta = l \rightarrow l'$ in Δ_j in a descending order by $\text{cand}_{(G,T)}(l')$. Each time it checks whether $v \in (Q \oplus (\Delta_j \setminus \{\delta\}))(G)$ (via TSim in Section 2 if $(Q \oplus (\Delta_j \setminus \{\delta\}))(G)$ is not computed by evalPR in Section 5), and removes δ from Δ_j if so. It returns Δ_j after checking all remaining label relaxations in Δ_j, or it has invoked TSim M times.

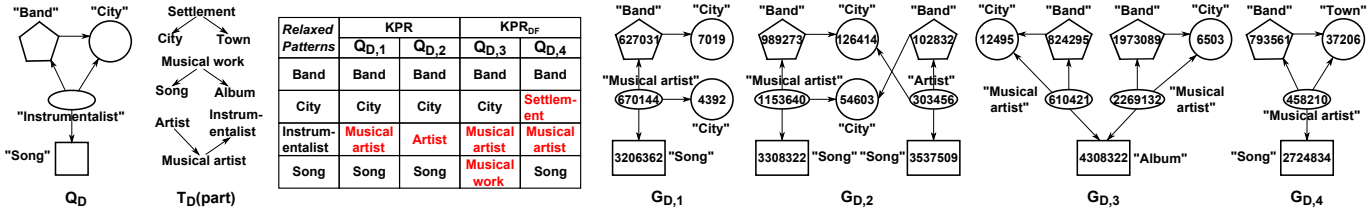
7 EXPERIMENTAL STUDY

Using real-life data, we verify the effectiveness and efficiency of the framework for relaxing graph pattern matching with explanation.

Experimental setting. We use the following settings.

Data and taxonomy graphs. We used two real-life graphs.

(1) DBpedia was taken from DBpedia 201504 [1]. It consists of (i) an RDF data graph with 4.32M nodes and 8.43M edges, and (ii) a



Relaxed Patterns	KPR		kPR_{DF}	
	$Q_{D,1}$	$Q_{D,2}$	$Q_{D,3}$	$Q_{D,4}$
Band	Band	Band	Band	Band
City	City	City	City	Settlement
Instrumentalist	Musical artist	Artist	Musical artist	Musical artist
Song	Song	Song	Musical work	Song

Q_D T_D(part) $G_{D,1}$ $G_{D,2}$ $G_{D,3}$ $G_{D,4}$

Figure 6: Real-life taxonomy simulation relaxation and matches on DBpedia

built-in taxonomy graph with 735 concepts (nodes) for data graph labels, where edges indicate an *is-a* relation. The taxonomy graph is a rooted forest with average height 2.29 (maximum height 6).

(2) YAGO [2] consists of (i) a data graph with 5.13M nodes and 5.39M edges; and (ii) a built-in taxonomy graph (forest) with 6488 concepts (nodes), with average height 3.27 (maximum height 13).

Pattern generator. We implemented a generator for producing random pattern graphs $Q(V_Q, E_Q, f_Q)$, controlled by 3 parameters: $|V_Q|$ varying from 2 to 10, $|E_Q| = \lfloor \alpha |V_Q| \rfloor$, and the number $\lfloor \beta |V_Q| \rfloor$ of labels, which are from the same label set with data graphs.

Implementation. We implemented the following in C++: (a) our algorithms TSim, relTF, relDF, evalPR, expTF and expDF; (b) our algorithms evalTF and evalDF for using evalPR to answer top-k relaxed patterns for kPR and kPR_{DF}, respectively; (c) algorithms TSimTF and TSimDF for using TSim k times to answer top-k relaxed patterns for kPR and kPR_{DF}, respectively; (d) algorithms gsim [14], VF2 [7] and TISO [5] for answering graph simulation, subgraph isomorphism and taxonomy isomorphism queries, respectively; (e) algorithm relC for generating top-k relaxations with a function that ranks relaxations by the total relaxation distance on the taxonomy between the relaxed and original pattern labels, as used in *e.g.,* [15, 25]; and (f) algorithm TSimC for answering the k relaxed patterns found by relC by invoking TSim k times.

We used a machine powered by an Intel Core(TM) Duo 3.00GHz CPU with 16GB of memory. Each experiment was run 5 times and the average is reported here.

Experimental results. In all the experiments reported below, we fixed $\alpha = 1.2$, and set $\beta = 1$ by default when generating patterns. We fixed $\lambda = 0.5$, and set $|V_Q| = 6$, $k = 15$ and $\mu = 3$ by default. We did not report match results for (taxonomy) subgraph isomorphism as they return even fewer answers than taxonomy simulation.

(1) Case study. We first exemplify the framework with a case study.

Consider a real-life pattern graph Q_D, shown in Fig. 6, to find all "city" items in DBpedia such that (a) the city is a hometown of an "instrumentalist", (b) the "instrumentalist" has founded a "band" in the "city", and (c) the "instrumentalist" has released a "song". In DBpedia graph nodes are entities with unique ids and labels which indicate their domains. They only match the nodes of Q_D with the same geometry shapes, *e.g.,* circles, ellipses, squares and pentagons.

By evaluating Q_D against DBpedia, we found that there is no match result to Q_D on DBpedia via taxonomy simulation (not to mention taxonomy isomorphism, subgraph isomorphism and graph simulation). We next examine its relaxations. Using taxonomy T_D in Fig. 6, we computed the top-2 relaxed patterns of Q_D and their match results in G, also shown in Fig. 6. Here $Q_{D,1}$ and $Q_{D,2}$ are top-2 relaxed patterns for kPR, and $Q_{D,3}$ and $Q_{D,4}$ are top-2 relaxed patterns for kPR_{DF}. Part of match results to the four relaxed patterns are $G_{D,1}$, $G_{D,2}$, $G_{D,3}$ and $G_{D,4}$, respectively. One can see that they all are sensible matches and can capture the intention of pattern Q_D. However, they cannot be found when using Q_D directly.

We ascribe the situation to the inaccurate specification of patterns from users versus the numerous labels residing in knowledge graphs, *e.g.,* 735 built-in labels for DBpedia.

(2) Match results of taxonomy simulation. We evaluated the match results of taxonomy simulation, to elaborate the need of pattern relaxation for taxonomy simulation.

Quantity. To justify the need for studying pattern relaxation, we tested the capability of common matching semantics in capturing matches. We used the pattern generator to generate random patterns and compared the average number of matches found by TSim and gsim. Varying $|V_Q|$ of Q from 2 to 10, we report the results on DBpedia and YAGO in Figures 7(a) and 7(b). We found that both TSim and gsim find very few matches for medium-sized patterns on DBpedia and YAGO, *e.g.,* when $|V_Q| \geq 6$, they both cannot identify any matches, although TSim can identify more matches than gsim on small patterns, *e.g.,*TSim found 1.4 times mores matches than gsim did when $|V_Q|$ is 4 on DBpedia. This confirms the need for studying pattern relaxation.

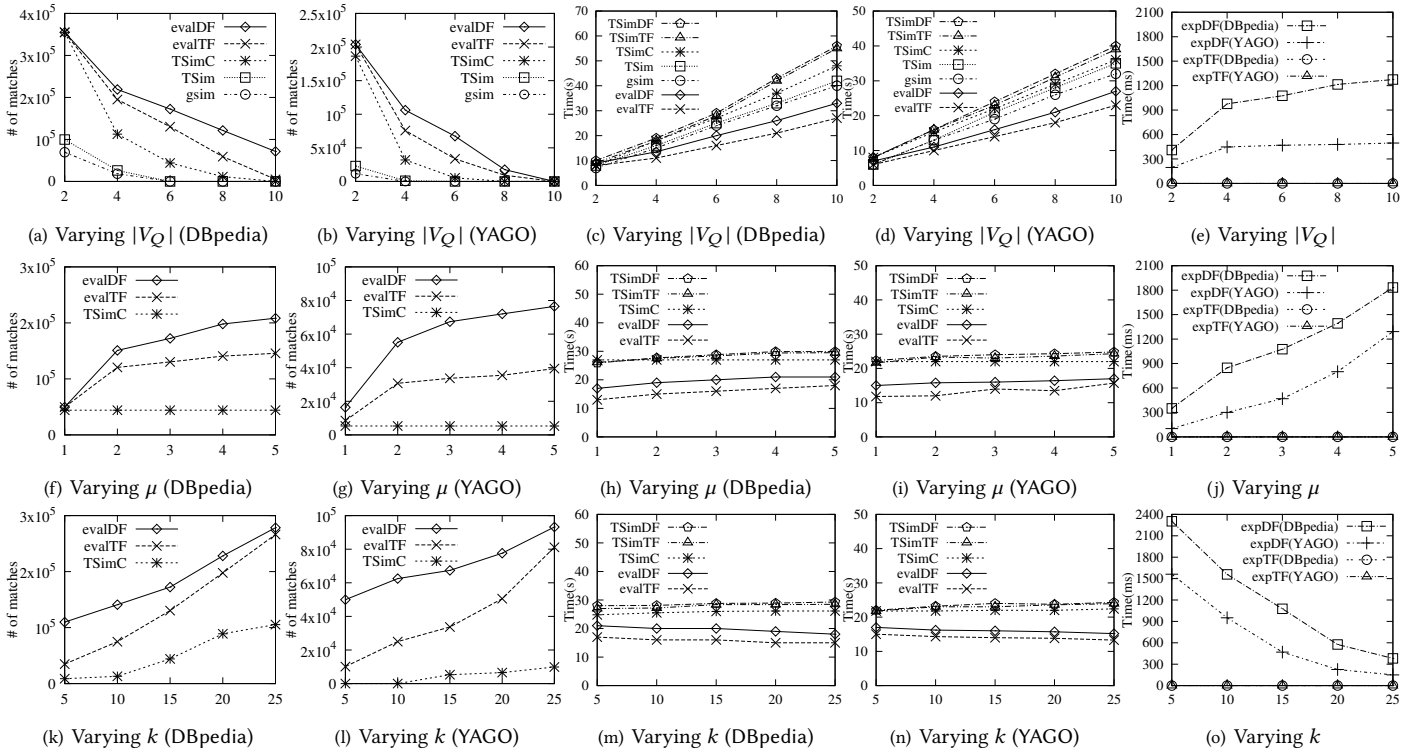
Quality. To further justify the focus of relaxation on taxonomy simulation queries, we also evaluated the quality of taxonomy simulation matches. We designed 30 patterns using randomly sampled subgraphs from the two real-life graphs. For each pattern Q, we maintained a collection of nodes with clear search purpose in the context of Q, *e.g.,* places or books satisfying certain conditions, which we used as the golden standard for checking the validity of their matches. We define the accuracy of a match relation S to Q in G as

$$\text{acc}(S, Q, G) = \sum_{(u,v) \in S} \text{valid}(u, v)/|S|,$$

where $\text{valid}(u, v)$ is 1 if data node v in G is an entity satisfying the conditions maintained for u in Q (we also used WordNet and Wiki to observe synonymous entities); and is 0 otherwise. Note that $\text{acc}(S, Q, G)$ is in $[0, 1]$ and is 1 if S contains accurate answers only. To make it possible for manually inspection, we restrict that $|S| \leq 50$: if the match relation $Q(G)$ to Q in G is larger than 50, we use a sample $S \subseteq Q(G)$ such that $|S| = 50$; otherwise we use $S = Q(G)$.

Using the accuracy measure, we inspected the match relations returned by TSim to the 30 patterns and found that their average accuracy is 0.98 and 0.94 on DBpedia and YAGO, respectively. This justifies the focus on taxonomy simulation for pattern relaxation.

(3) Effectiveness of pattern relaxation. We next evaluated the effectiveness of pattern relaxation.



(a) Varying $|V_Q|$ (DBpedia) (b) Varying $|V_Q|$ (YAGO) (c) Varying $|V_Q|$ (DBpedia) (d) Varying $|V_Q|$ (YAGO) (e) Varying $|V_Q|$

(f) Varying μ (DBpedia) (g) Varying μ (YAGO) (h) Varying μ (DBpedia) (i) Varying μ (YAGO) (j) Varying μ

(k) Varying k (DBpedia) (l) Varying k (YAGO) (m) Varying k (DBpedia) (n) Varying k (YAGO) (o) Varying k

Figure 7: Effectiveness and efficiency of taxonomy simulation relaxation and explanation

Quality. We first evaluated the quality of relaxed patterns. Using the same setting as Exp-(2), we inspected match relations to the top-5 relaxed patterns of the 30 patterns, and found that the average accuracy on DBpedia and YAGO is 0.81 and 0.75, respectively, when topological ranking function is used, and is 0.72 and 0.68 with the diversified topological ranking function.

Quantity. We then tested the effectiveness by comparing average number of matches found by relaxed patterns returned by relTF, relDF and relC, respectively, via evalTF, evalDF and TSimC.

(a) Impact of $|V_Q|$. Varying the number $|V_Q|$ of nodes in Q from 2 to 10, we report the results in Figures 7(a) and 7(b) for DBpedia and YAGO, respectively. Observe the following. (i) The number of matches found by evalTF, evalDF and TSimC increases dramatically compared to gsim on all cases, and is 10.7, 11.9 and 6.1 times larger than gsim when $|V_Q| = 4$ on DBpedia, respectively. (ii) In all cases, evalTF and evalDF consistently find more matches than TSimC. This verifies the effectiveness of information ratio in topological ranking function, which enables the use of information of data graphs in ranking. (iii) evalDF identifies more matches than evalTF. This is because the diversification function tends to generate more relaxed ancestor labels on taxonomy graphs, in order to optimize both topological ranking and diversification.

(b) Impact of μ. To evaluate the impact of parameter μ, we varied μ from 1 to 5 and tested the average number of matches. The results, as shown in Figures 7(f) and 7(g), tell us the following. (i) Even when μ is small, both evalTF and evalDF are able to find sensible matches on the two datasets, and the quantities are larger than that of TSimC, *e.g.,* they found 8525 and 16351 matches when $\mu = 1$ on YAGO, respectively, compared to 5320 by TSimC. (ii) Both evalTF and evalDF make more use of larger μ as they can take into account more information on the data graphs and taxonomy, *e.g.,* evalTF and evalDF found 39478 and 76502 matches when $\mu = 5$ on YAGO, while TSimC still found 5320 matches.

(c) Impact of k. Similar to (b), we evaluated the impact of k by varying it from 5 to 25. The results are in Figures 7(k) and 7(l). We find that both evalTF and evalDF can find sensible matches even when k is small. For example, evalTF identified 10100 matches when $k = 5$ on YAGO, and even more for evalDF, while TSimC found no match.

(4) Effectiveness of explanation. To evaluate the effectiveness of relaxation explanation, we randomly selected 100 matched nodes to the relaxed patterns found by relTF and relDF, respectively, and used algorithms expTF and expDF (with parameter M varying from 0 to 4) to compute minimum explanations. We report the *accuracy* of the explanations, which is defined to be the percentage of explanations that are real minimum explanations. Since the accuracy for expTF is always 1, we only report results for expDF in Table 1.

Table 1: Effectiveness of explanations

M	0	1	2	3	4
DBpedia	85%	93%	99%	100%	100%
YAGO	88%	97%	100%	100%	100%

We find that the explanations computed by expDF are capable to explain the relaxations well. The accuracy is consistently above 85% and 88% on DBpedia and YAGO, respectively, even when $M = 0$; and is above 99% when M is above 2 on both datasets.

(5) Efficiency of pattern relaxation and evaluation. We tested the efficiency of our algorithms for relaxation and evaluation. Since all top-k pattern relaxation algorithms relTF, relDF and relC terminate within 2s on all cases. We only report the efficiency of evaluation. The offline pre-process takes 0.02s (resp. 2.97s) to encode labels into bit strings, and 7.38s (resp. 6.21s) to compute the numbers $|\mathsf{cand}_{(G,T)}(l)|$ for labels l on DBpedia (resp. YAGO).

We compared the evaluation time of evalTF, evalDF, TSimTF, TSimDF and TSimC, using the same setting as Exp-(3). We report the findings in Figures 7(c), 7(d), 7(h), 7(i), 7(m) and 7(n), which tell us the following. (i) evalTF and evalDF improved TSimTF and TSimDF significantly, *e.g.,* evalTF is 1.8 times faster than TSimTF when $k = 15$, $|V_Q| = 6$ and $\mu = 3$ on DBpedia. The improvement increases when $|V_Q|$ or k grows larger. (ii) Algorithms TSimTF and TSimDF are a bit slower than TSimC. This is because TSimTF and TSimDF encode the dataset information into the ranking functions, such that they both find many more meaningful matches than TSimC, while TSimC always returns empty results even with medium size patterns, *e.g.,* queries Q on YAGO with $|V_Q| = 6$.

(6) Efficiency of explanation. Using the same setting as in Exp-(4) with $M = 4$ by default, we evaluated the efficiency of our explanation algorithms expTF and expDF on DBpedia and YAGO. We report the results in Figures 7(e), 7(j) and 7(o).

Observe the following. (i) Both expTF and expDF are efficient in finding minimum explanations for relaxations, *e.g.,* they took less than 1ms and 2.4s in all cases, respectively. (ii) The running time of expDF increases when $|V_Q|$ and μ increase, as expected. It decreases when k increases. Indeed, when k is large, more pattern relaxations are generated, such that the chance to invoke TSim by expDF is low.

We also find that smaller λ leads diverse relaxations with more match results, *e.g.,* evalDF ($\lambda = 0.5$) finds more matches than evalTF ($\lambda = 1$) in all cases. This is because evalDF tends to generate more relaxed ancestor labels for diversification of relaxed patterns. Accordingly, evalDF is also a bit slower than evalTF.

Summary. From the experiments we find the following.

(1) Taxonomy simulation returns 1.4 times more sensible matches than graph simulation for Q with 4 nodes on DBpedia, within almost equal time. But it still cannot identify matches for larger patterns, *e.g.,* more than 5 nodes.

(2) Based on our top-k ranking functions and algorithms, our relaxation framework is effective and efficient in relaxing patterns and explaining results: (a) It enables us to identify more sensible matches by relaxation, *e.g.,* 11.9 times more for Q with 4 nodes on DBpedia, among which 74% are verified sensible. (b) Its average evaluation time of the top-k relaxed patterns is 1.8 times faster than direct evaluation, and the gap grows bigger with larger k. (c) It can explain matches to the relaxed patterns accurately and efficiently, *e.g.,* it explains relaxations with accuracy above 85% even without access to the data graphs, and achieves 99% accuracy when two data accesses are allowed on both DBpedia and YAGO, in 2.4s.

8 CONCLUSION

We have proposed a framework for relaxing graph pattern matching queries. We have formalized taxonomy simulation by combining taxonomy with graph simulation, and proposed a notion of relaxation for it. We have designed ranking functions for relaxations of taxonomy simulation patterns. We have also developed practical algorithms to compute and answer top-k relaxed patterns, and to explain matches captured by the relaxations. Finally, we have experimentally verified the effectiveness and efficiency of the techniques.

There are several issues of interests for future research. We are to study relaxing and explaining other graph queries, *e.g.,* taxonomy subgraph isomorphism, and to integrate the relaxation framework into existing graph-based systems.

Acknowledgments. This work is supported in part by NSFC U1636210, 973 Program 2014CB340300, NSFC 61421003 & 61322207, Special Funds of Beijing Municipal Science & Technology Commission, Beijing Advanced Innovation Center for Big Data and Brain Computing, and MSRA Collaborative Research Program.

REFERENCES

[1] 2015. DBpedia. http://wiki.dbpedia.org/Downloads2015-04. (2015).
[2] 2016. YAGO. http://www.mpi-inf.mpg.de/yago. (2016).
[3] Sihem Amer-Yahia, Nick Koudas, Amlélie Marian, Divesh Srivastava, and David Toman. 2005. Structure and Content Scoring for XML. In *PVLDB*.
[4] Nicole Bidoit, Melanie Herschel, and Aikaterini Tzompanaki. 2015. Efficient Computation of Polynomial Explanations of Why-Not Questions. In *CIKM*.
[5] Ali Cakmak and Gultekin Ozsoyoglu. 2008. Taxonomy-superimposed graph mining. In *EDBT*.
[6] James Cheney, Laura Chiticariu, and Wang-Chiew Tan. 2009. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases* 1, 4 (2009).
[7] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 10 (2004).
[8] Andrzej Czygrinow. 2000. Maximum dispersion problem in dense graphs. *Operations Research Letters* 27, 5 (2000), 223–227.
[9] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. 2009. Language-model-based Ranking for Queries on RDF-Graphs. In *CIKM*.
[10] Shady Elbassuoni, Maya Ramanath, and Gerhard Weikum. 2011. Query Relaxation for Entity-Relationship Search. In *ESWC*. 62–76.
[11] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, Yinghui Wu, and Yunpeng Wu. 2010. Graph pattern matching: From intractability to polynomial time. *PVLDB* 3, 1 (2010), 1161–1172.
[12] Cibele Freire, Wolfgang Gatterbauer, Neil Immerman, and Alexandra Meliou. 2015. The complexity of resilience and responsibility for self-join-free conjunctive queries. *PVLDB* (2015).
[13] Jianliang Gao, Ping Liu, Xuedan Kang, Lixia Zhang, and Jianxin Wang. 2016. PRS: Parallel Relaxation Simulation for Massive Graphs. *Comput. J.* 59, 6 (2016).
[14] M. R. Henzinger, T. Henzinger, and P. Kopke. 1995. Computing simulations on finite and infinite graphs. In *FOCS*.
[15] Hai Huang, Chengfei Liu, and Xiaofang Zhou. 2008. Computing Relaxed Answers on RDF Databases. In *WISE*. 163–175.
[16] Eugene L. Lawler. 1972. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science* 18, 7 (1972).
[17] Wangchao Le, Anastasios Kementsietsidis, Songyun Duan, and Feifei Li. 2012. Scalable Multi-Query Optimization for SPARQL. In *ICDE*.
[18] Shuai Ma, Yang Cao, Wenfei Fan, Jinpeng Huai, and Tianyu Wo. 2014. Strong simulation: Capturing topology in graph pattern matching. *TODS* 39, 1 (2014).
[19] Silvio Micali and Vijay V. Vazirani. 1980. An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs. In *FOCS*. 17–27.
[20] Xuguang Ren and Junhu Wang. 2016. Multi-Query Optimization for Subgraph Isomorphism Search. *PVLDB* 10, 3 (2016).
[21] Netta Aizenbud Reshef, Artem Barger, Yael Dubinsky, Ido Guy, and Shiri Kremer Davidson. 2012. Bon Voyage: Social Travel Planning in the Enterprise. In *CSCW*.
[22] Sudeepa Roy and Dan Suciu. 2014. A formal approach to finding explanations for database queries. In *SIGMOD*.
[23] Elena Vasilyeva, Maik Thiele, Adrian Mocan, and Wolfgang Lehner. 2015. Relaxation of subgraph queries delivering empty results. In *SSDBM*.
[24] Stanley Wasserman and Katherine Faust. 1994. *Social network analysis: Methods and applications*. Cambridge university press.
[25] Yinghui Wu, Shengqi Yang, and Xifeng Yan. 2013. Ontology-based subgraph querying. In *ICDE*.
[26] Shengqi Yang, Yinghui Wu, Huan Sun, and Xifeng Yan. 2014. Schemaless and Structureless Graph Querying. In *VLDB*.