Relaxing Graph Pattern Matching with Explanations

Jia Li[1]
[1]*Beihang University*
lijia1108@buaa.edu.cn

Yang Cao[1,2]
[2]*University of Edinburgh*
yang.cao@ed.ac.uk

***Abstract*—Traditional graph pattern matching is based on subgraph isomorphism, which is found too restrictive to identify meaningful matches in, *e.g.,* social search. To handle this, taxonomy assisted subgraph isomorphism has been proposed to relax the label constraints in the matching. Nonetheless, there are many cases cannot be covered. To this end, we first propose taxonomy simulation, a matching semantics that relaxes both label and topology constraints in the matching. We then introduce a notion of pattern relaxation for taxonomy simulation, to further enrich the results of graph pattern matching by utilizing the taxonomy information. We develop ranking functions and algorithms to rank top-k relaxations of taxonomy simulation queries. We also study the evaluation of top-k relaxations and develop algorithms and optimization techniques. Finally, we 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. We experimentally verify that the framework and the techniques are effective in identifying meaningful matches in practice.**

I. Introduction

Graph pattern matching is being widely used in social network analysis, among other things. However, traditional graph pattern matching is based on subgraph isomorphism, 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 [9], [29] has been proposed to capture more matches by relaxing label constraints, which makes use of a taxonomy of the label hierarchy in a "downward" manner, 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.

Example 1: Consider a real-life example taken from [25]. The data graph G_1 shown in Fig. 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$_1$ 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 which is close to the museum, and (3) dine at a restaurant that is close to the two places. This query can be specified by a pattern graph Q_1 (in the dashed rectangular), also shown in Fig. 1. It is easy to see that conventional pattern matching via subgraph isomorphism cannot identify any match for Q_1 in G_1. This remains the case even we adopt taxonomy assited subgraph isomorphism with additional taxonomy information.

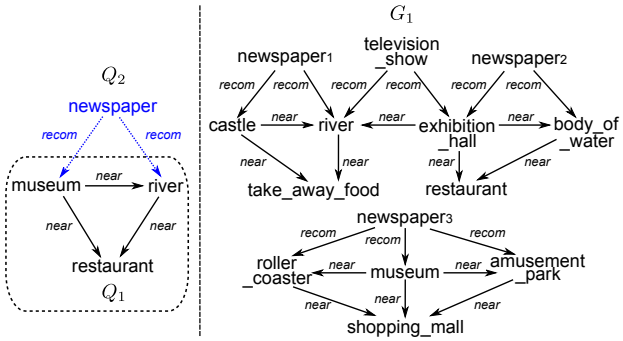


Fig. 1. Querying knowledge network

Consider a taxonomy graph T_1 taken 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 the strict topological matching constraints, when taxonomy subgraph isomorphism is used with T_1, the match result to Q_1 in G_1 remains to be empty. However, in the presence of T_1, one can readily 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 topological constraints of taxonomy subgraph isomorphism, so that both label and structural matching semantics can be relaxed for graph pattern matching. To this end, we propose *taxonomy simulation*, to combine taxonomy with graph simulation based matching [13], [16] which has been used to relax the topological constraints of subgraph isomorphism.

We will show that taxonomy simulation observes the hierarchical *is-a* relationship between labels when computing match relations for graph simulation queries and is able to identify the sensible match in Example 1.

Nonetheless, taxonomy simulation is still not bullet-proof. It still comes short to capture matches in real-life graphs.

Using data and taxonomy graphs from DBpedia [1] and YAGO [4], we have conducted an experiment on the percentage of pattern graphs that have non-empty match results. We randomly generated 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 reported in the table below.

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

We found that only 18% and 2% of the queries can find matches

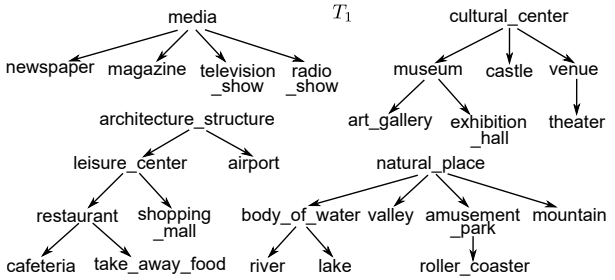


Fig. 2. Taxonomy graph

in DBpedia and YAGO using taxonomy simulation, and none of pattern graphs with 6 or more nodes can identify matches. The percentages are even much lower if we use (taxonomy) subgraph isomorphism or graph simulation. However, by examining the pattern and data graphs, we found there are indeed many sensible matches to those empty queries.

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). We will show later that, using taxonomy simulation, no match can be found for Q_2 in G_1. However, from the taxonomy graph T_1 in Fig. 2, we know that both newspaper and television_show are media. Therefore, 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" manner, to capture more sensible matches together with the "downward" label enhancement and topological relaxation of taxonomy simulation. To do this, several questions have to be settled. (1) How to define and answer taxonomy simulation queries? (2) How to further relax patterns to capture matches that cannot be found by taxonomy simulation? (3) How to rank and evaluate the relaxations? (4) How do relaxations capture answers? That is, can we explain to the user that why certain answers are returned to them after relaxations?

Contributions. This paper answers these questions and proposes a framework of pattern relaxation with explanations.

(1) We propose taxonomy simulation, to further relax taxonomy subgraph isomorphism with softer structural constraints from graph simulation, in addition to the "downward" use of taxonomy hierarchy. We develop a cubic-time algorithm for answering taxonomy simulation patterns, based on an encoding scheme that identifies candidate matches without accessing edges of the taxonomy graph (Section II).

(2) We propose pattern relaxation for taxonomy simulation queries, which further enables the "upward" use of taxonomy information in graph pattern matching (Section III). We design two functions to rank top-k relaxations of taxonomy pattern queries, namely, (a) *topological ranking function* to rank relaxations based on their relaxation ratio that measures their accuracy via the topological distance to the original pattern, and information ratio estimating the ability to capture potential matches. and (b) *diversified topological ranking function* that extends topological ranking with diversification, aiming both to optimize the topological ranking of relaxations and to distribute the relaxation impact on the pattern graph as diverse as possible.

(3) We study the problem of computing top-k pattern relaxations *w.r.t.* the two ranking functions (Section IV). 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 $O(\mu k|V_Q|^2)$-time exact algorithm, where μ is constant and $|V_Q|$ is the number of nodes in Q. For (b), we reduce the problem to the well-studied maximum dispersion problem, which enables us to use existing algorithms for the latter problem to compute the top-k diversified relaxations.

(4) We give an algorithm for answering top-k relaxed patterns (Section V). It uses (a) a structure that hierarchically organizes the evaluation of the k relaxed patterns to maximize shared computation; and (b) bounded decremental evaluation of taxonomy simulation for recovering answers from shared computation to each relaxed pattern, to minimize total computation.

(5) We propose *minimum explanation* to explain why a match is returned, in terms of the essential part of the relaxation that captures the match, and study the minimum explanation problem *w.r.t.* the two ranking functions (Section VI). We show that the problem is (a) in PTIME with the topological ranking function, and (b) becomes NP-complete when diversification is considered. 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.

(6) Using real-life graphs, we experimentally verify the effectiveness and efficiency of the framework of relaxing taxonomy simulation queries with explanations (Section VII). We find the following. (a) Taxonomy simulation can find 1.5 times more sensible matches than simulation for patterns with 4 nodes. (b) The ranking functions and algorithms are effective: they find 11.9 times more answers, among which 74% are verified sensible. (c) The average evaluation time of relaxed patterns is 1.53 times faster than conventional batch evaluation when $k = 5$, and the gap grows with larger k. (d) Our algorithms can explain relaxations with accuracy above 85% without access to the data graphs, and achieve 99% accuracy with 2 data accesses.

All the proofs can be found in the full version [2].

Related work. We categorize related work as follows.

Taxonomy assisted graph queries. The taxonomy and ontology information has been used for semantic web [11], [15], keyword search [27], [32], and graph search [9], [18], [29], [32]. They typically either interpret queries with hierarchical taxonomies [9], merge the hierarchies into data graphs [18], or use them during the search for matches [11], [29], [32]. Here we combine taxonomy information with graph pattern matching based on graph simulation [13], [16], so that both the structure of graph patterns and the labels can be relaxed in the query semantics of taxonomy simulation. Instead, [9], [18], [29], [32] consider subgraph queries defined by variants of subgraph isomorphism, which carry more restrictive structural constraints.

Query relaxation. Closer to our work is query relaxation for queries over relational data [20], [23], XML [5], RDF [12], [30]

and graph data [19], [31]. Our work differs from the previous work in the following. (1) We study relaxation of graph pattern matching via taxonomy simulation on graph data, while [5], [20], [23] focus on queries over relational and XML data. (2) We propose to extend the "downward" use of taxonomy in query answering with the "upward" direction via graph pattern relaxations. Instead, while [12], [19], [30], [31] are about RDF and graph query relaxation, they are carried out via structural relaxation without the use of external information. (3) We rank relaxations by designing functions that observe (i) the recursive nature of taxonomy simulation, (ii) the informativity of potential matches to the relaxed patterns and (iii) diversification of the relaxations, which are not considered by previous work.

Explanation. Related to the relaxation explanation are also phenomenon explanation [26], query resilience [14], why-not queries [7] and provenance [10]. Different from theirs, our work studies the explanation of relaxations of graph pattern matching, instead of general query results or provenance.

II. Taxonomy Simulation

In this section, we first introduce the notion of taxonomy simulation (Section II-A). We then present an algorithm for answering taxonomy simulation queries (Section II-B).

A. Taxonomy Simulation

We start with some basic notations.

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 the graph is defined as $|V| + |E|$.

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, community. The edge labels specify the relationships between respective entities.

Graph simulation. Data graph G *matches* pattern graph Q via *graph simulation*, denoted by $Q \trianglelefteq G$, if there exists a binary *match relation* $R \subseteq V_Q \times V$ in G for Q such that

(1) for each $(u, v) \in R$, $f_Q(u) = f(v)$, *i.e.,* u and v have the same label; and (2) for each $u \in V_Q$, there exists $v \in V$ such that (a) $(u, v) \in R$, and (b) for each edge (u, u') in E_Q, there exists an edge (v, v') in E such that $(u', v') \in R$ and (u, u') and (v, v') share the same label, *i.e.,* $f_Q(u, u') = f(v, v')$.

Intuitively, graph simulation preserves the label match and the child relationships between pattern and data graphs, and it relaxes the topological 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, *i.e.,* a set of disjoint rooted trees, 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 each node in V_T to a distinct label in the label set Σ_V. Observe that each node in T has a unique label as taxonomy records the relationships among labels.

A taxonomy graph T defines a specialization-generation hierarchy for labels in the corresponding data graphs. The distance from node u to u' in T, denoted by $\mathsf{dist}_T(u, u')$, is the number of edges in the shortest path from u to u' if u and u' are connected in T; and is $+\infty$ otherwise. We denote by $\mathsf{desc}_T(u)$ the set of descendants of u in T. We also write $\mathsf{dist}_T(u, u')$ and $\mathsf{desc}_T(u)$ as $\mathsf{dist}_T(f_T(u), f_T(u'))$ and $\mathsf{desc}_T(f_T(u))$, respectively, when labels are concerned, since nodes in the taxonomy carry distinct labels.

We are now ready to define taxonomy simulation.

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 exists a binary *match relation* $R^T \subseteq V_Q \times V$ in G for Q such that

(1) for each $(u, v) \in R^T$, $f(v) \in \mathsf{desc}_T(f_Q(u))$; and

(2) for each node $u \in V_Q$, there exists $v \in V$ such that (a) $(u, v) \in R^T$ and (b) for each edge $(u, u') \in E_Q$, there exists an edge (v, v') in E such that $(u', v') \in R^T$ and $f_Q(u, u') = f(v, v')$.

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

When $Q \trianglelefteq_T G$, one can verify there exists a *unique maximum* match relation R_M^T in G for Q *w.r.t.* T, following [16]. 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. One can verify 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 in Q_1 to {exhibition_hall}, {river} and {take_away_food, restaurant} in G_1, respectively. □

B. Answering Taxonomy Simulation Queries

We next study the evaluation of taxonomy simulation queries and prove the following.

Theorem 1: *Given a pattern Q, data graph G and taxonomy T, it takes cubic time to decide whether $Q \trianglelefteq_T G$ and compute the maximum match relation for Q in G.* □

As a proof we give such a taxonomy simulation algorithm.

Algorithm TSim. The algorithm, denoted by TSim and shown in Fig. 3, extends the algorithm (denoted by gsim) for graph simulation [16] such that we check node label containment (condition (1) in the definition of taxonomy simulation) instead of identical label matching (line 2; underlined). The tricky part is to carry out this step efficiently so that we do not need to search T for each node u in V_Q.

Algorithm TSim achieves this with the following steps.

(1) As an *offline* preprocessing, it encodes each node label in G in a bit string ℓ^T of length $|V_T|$ such that each position in the string stands for the presence of a label. For each label x, the position for x is 1 in ℓ^T and all other positions are 0.

(2) It constructs the *taxonomy pattern graph* Q^T and uses it instead of Q. Here Q^T shares the same structure with Q, except

Input: data $G(V, E, f)$, pattern $Q(V_Q, E_Q, f_Q)$, taxonomy T.
Output: The maximum match relation R_M^T for Q in G *w.r.t.* T.
1. **for each** $u \in V_Q$ **do**
2. $\quad \mathsf{sim}(u) := \{v \in V \mid f(v) \in \mathsf{desc}_T(f_Q(u))\}$;
3. **while** there are changes **do**
4. $\quad$ **for each** (u, u') in E_Q and **each** $v \in \mathsf{sim}(u)$ **do**
5. $\quad\quad$ **if** there exists no edge (v, v') in G such that $v' \in \mathsf{sim}(u')$ and $f(v, v') = f_Q(u, u')$ **then**
6. $\quad\quad\quad \mathsf{sim}(u) = \mathsf{sim}(u) \setminus \{v\}$;
7. $\quad$ **if** $\mathsf{sim}(u) = \emptyset$ **then return** $\emptyset$;
8. $R_M^T := \{(u, v) | u \in V_Q, v \in \mathsf{sim}(u)\}$; **return** R_M^T.

Fig. 3. Algorithm TSim

that for each node u in Q^T, its label is a bit string $\ell_Q^T(u)$ of length $|V_T|$ similar to the encodings in G such that, for each label l' in $\mathsf{desc}_T(f_Q(u))$, the position in $\ell_Q^T(u)$ that stands for l' is 1, and all other positions are 0.

With this, algorithm TSim conducts line 2 in Fig. 3 in $O(|V_T|)$ time without accessing edges of T, as follows: it checks whether there exists position in $\ell_Q^T(u)$ & $\ell^T(v)$ assigned with 1 and puts v in $\mathsf{sim}(u)$ if so. Here $\ell^T(v)$ is the label encoding of $f(v)$ in G and & is the bitwise AND operation.

Correctness & Complexity. The correctness of TSim is guaranteed by the following: for any pattern node u and data node v, if $f(v) \in \mathsf{desc}_T(f_Q(u))$, then $\ell_Q^T(u)$ & $\ell^T(v)$ contains at least one position that is assigned with 1. Algorithm TSim is in $O(|V_Q||T| + |V_Q||V||V_T| + |Q||G|)$ time. Indeed, (1) the construction of the taxonomy pattern graph Q^T is in $O(|V_Q||T|)$ time; (2) the initialization (lines 1-2) is in $O(|V_Q||V||V_T|)$ time with a one-time $O(|V_T|^2)$ offline computation for encoding node labels; and (3) the remaining part can be implemented in $O(|Q||G|)$ time, the same as gsim [16].

III. Relaxing Taxonomy Simulation

As discussed in Section I, though it is able to identify more sensible matches than taxonomy subgraph isomorphism, when querying real-life graphs, taxonomy simulation still comes short for patterns designed with no prior information about the data graphs, and can fail to identify any sensible match to medium-size patterns. We next propose to extend taxonomy simulation pattern graphs by further making use of the hierarchical information among the labels in the taxonomy.

Below we first define relaxations for taxonomy simulation queries. We then propose two measures to rank relaxations.

A. Relaxations for Taxonomy Simulation

Label relaxation. A *label relaxation* δ *w.r.t.* a taxonomy T is of form $l \to l'$ such that l' is a parent 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 $\mathsf{dist}_T(l', l) \leq \mu$; 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.

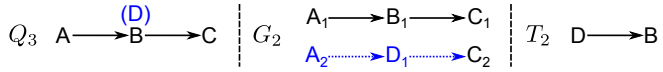


Fig. 4. Relaxation ratio

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

Relaxed pattern. Given pattern graph Q and pattern relaxation Δ for Q *w.r.t.* T, the *relaxed pattern* of Q *w.r.t.* Δ, denoted by $Q \oplus \Delta$, is the pattern derived from Q by replacing each occurrence of l with l' in Q for each $l \to l'$ in Δ.

Example 4: Recall Q_2 and T_1 from Example 2. When $\mu = 2$, $\Delta = \{\delta = \mathsf{newspaper} \to \mathsf{media}\}$ 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. □

B. Ranking 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 certain functions to rank the relaxations and compute the top-k pattern relaxations based on the functions.

Below we design two functions: *topological ranking* to measure pattern relaxations in terms of their relaxation distances *w.r.t.* Q, with G and T taken into consideration together; and *diversified-topological ranking* which combines the topological function with a diversification function. Based on the functions, we introduce two top-k pattern relaxation problems.

(1) Topological ranking. In real world, one typically wants to capture more match results while ensures the results are sensible to the patterns. 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 in details.

(a) Relaxation ratio. The *relaxation ratio* of a label relaxation $\delta = l \to l'$ *w.r.t.* taxonomy T for pattern graph $Q(V_Q, E_Q, f_Q)$, 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 (including u itself), 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 [28]. 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. 4. 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

$\Delta = \{\mathsf{B} \to \mathsf{D}\}$ on node B in Q_3 turns data node D_1 to be 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, $\mathsf{rank}_{Q_3}(\mathsf{C}) = 3 > \mathsf{rank}_{Q_3}(\mathsf{B}) = 2 > \mathsf{rank}_{Q_3}(\mathsf{A}) = 1$. □

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 a label relaxation $\delta = l \to l'$ *on* Q *in* G is defined as $\frac{|\mathsf{cand}_G^T(l)|}{|\mathsf{cand}_G^T(l')|}$, 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 one 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* (topKPR), formulated as follows.

Top-k pattern relaxation problem. 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 topKPR is to find a k-set $S \subseteq \mathcal{U}_\mu(Q, T)$, such that

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

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

Example 6: Consider Q_2, 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 $k = \mu = 2$. The label relaxations and part of possible pattern relaxations are listed below.

Pattern relaxations	$\Gamma(Q_2, \Delta_i)$
$\Delta_1 = \{\delta_1 = \text{newspaper} \to \text{media}\}$	1.359
$\Delta_2 = \{\delta_2 = \text{museum} \to \text{cultural\_center}\}$	2.175
$\Delta_3 = \{\delta_3 = \text{river} \to \text{natural\_place}\}$	3.695
$\Delta_4 = \{\delta_4 = \text{river} \to \text{body\_of\_water}\}$	4.077
$\Delta_5 = \{\delta_5 = \text{restaurant} \to \text{leisure\_center}\}$	7.249
$\Delta_6 = \{\delta_6 = \text{restaurant} \to \text{architecture..}\}$	14.778
...	...

Observe the following. (a) One can verify that S=$\{\Delta_1, \Delta_2\}$ is the top-2 pattern relaxations for topKPR. (b) $\Gamma(Q_2, \Delta_3)$ is less than $\Gamma(Q_2, \Delta_4)$, even though $\mathsf{dist}_T(\text{natural\_place}, \text{river}) = 2$ 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, although the relaxation ratio ranks relaxations with smaller distance higher. □

(2) Diversified topological ranking. It is desirable that the relaxed patterns not only are close to original patterns, but also are as diverse as possible, so that they can provide more information in their match results. Below we extend the topological ranking function with diversification.

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 and their descentants in T. 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 depart from the searching goal of Q_2, turning a scenery trip to a 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. We next introduce the *diversified top-k pattern relaxation problem*, denoted by $\mathsf{topKPR}^{\mathsf{DF}}$, which is defined along the same lines as problem topKPR. More specifically, given Q, G, T, and two integers μ and k, $\mathsf{topKPR}^{\mathsf{DF}}$ is to find a k-set $S \subseteq \mathcal{U}_\mu(Q, T)$ (recall $\mathcal{U}_\mu(Q, T)$ from the topKPR problem), such that

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

That is, $\mathsf{topKPR}^{\mathsf{DF}}$ 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 again. 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. □

IV. Finding Top-k Relaxations

In this section, we develop algorithms for the two top-k pattern relaxation problems. We focus on problem topKPR in Section IV-A first, and move on to topKPR$^{\mathsf{DF}}$ in Section IV-B.

A. Finding Top-k Relaxations for topKPR

We first present the main result for problem topKPR. Below 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 2: *There exists an algorithm that computes the top-k μ-bounded pattern relaxation for Q* w.r.t. *T in G within $O(\mu k|V_Q|^2)$-time, independent of $|G|$ and $|T|$.* □

As a proof, we next give such an algorithm. It utilizes the *Lawler's procedure* for computing top-k answers to optimization problems [17]. Lawler's procedure has the following *property*. For an optimization problem that can be formulated in integer programming with n 0-1 variables and the optimal (top-1) solution can be found in $c(n)$ time, 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 Relax$^{\mathsf{TF}}$. The algorithm, denoted by Relax$^{\mathsf{TF}}$, is shown in Fig. 5. It uses (a) a list L_{TR} to store the top-k pattern relaxations identified so far; and (b) a priority queue $\mathcal{Q}$ to cache candidate top-k pattern relaxations, which will be used to divide and branch the search space, *i.e.,* a collection of all relevant candidate label relaxations. 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 the top-1 pattern relaxation Δ_1 within $\mathcal{L}_1$ via procedure topRel, which is then pushed into $\mathcal{Q}$ together with $\mathcal{L}_1$ (line 4). It then iteratively searches the remaining $k-1$ relaxations by reducing to top-1 relaxation search within sub-collections of $\mathcal{L}_1$ (lines 5-11), until $\mathcal{Q}$ becomes empty (line 5), or the top-k results are already found (line 6). Each time it pops out pattern relaxation Δ_K with minimum topological ranking value, together with the collection of candidate label relaxations from which Δ_k is found, say $\mathcal{L}_K$ (line 7). It then puts Δ_K to L_{TR} as the K-th best relaxation (line 8). After that, it adopts the Lawer's procedure, denoted by LawlerBranch (defer to the full version due to limited space; cf. [17]), to generate sub-collections $\mathcal{L}^{s1}$, …, $\mathcal{L}^{sm}$ of candidate label relaxations from $\mathcal{L}_K$ and Δ_K (line 9). It then finds the top-1 pattern relaxation Δ_i in each sub-collection $\mathcal{L}^{si}$ via procedure topRel. (lines 10-11). It returns L_{TR} if all top-k relaxations are found (line 12).

Procedure topRel. Given Q, G and a collection $\mathcal{L}$ of candidate label relaxations, procedure topRel is also shown in Fig. 5.

Input: Q, G, T, two positive integers μ and k.
Output: A list L_{TR} of top-k pattern relaxations.
1. $L_{\mathsf{TR}} := [\,]$; $\mathcal{Q}$:= *nil*; $K := 1$;
2. **for each** label ℓ_i in Q **do** */* assume Q has labels ℓ_1, …, ℓ_m */*
3. generate a list L_i of label relaxations for ℓ_i bounded by μ in T;
4. $\mathcal{L}_1 := (L_1, \ldots, L_m)$; Δ_1 := topRel$(Q, G, \mathcal{L}_1)$; $\mathcal{Q}$.push$(\langle \Delta_1, \mathcal{L}_1 \rangle)$;
5. **while** $\mathcal{Q} \neq \emptyset$ **do**
6. **if** $K = k + 1$ **then break**;
7. $(\Delta_K, \mathcal{L}_K)$:= $\mathcal{Q}$.pop(); */*Δ_K has minimum $\Gamma(Q, \Delta_K)$ in $\mathcal{Q}$*/*
8. append Δ_K to L_{TR};
9. sub-collections $\mathcal{L}^{s_1}, \ldots, \mathcal{L}^{s_m}$:= LawlerBranch$(\Delta_K, \mathcal{L}_K)$;
10. **for** i in $[1, m]$ **do** Δ_i := topRel$(Q, G, \mathcal{L}^{s_i})$; $\mathcal{Q}$.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 with $\mathcal{L}$ *w.r.t.* $\Gamma(Q, \Delta)$.
1. **for each** i in $[1, m]$ **do** $\delta^i_{\min} := \arg\min_{\delta \in L_i} \gamma_Q(\delta) \cdot \mathcal{I}^G_Q(\delta)$;
2. $\Delta := \{\delta^1_{min}, \delta^2_{min}, \ldots, \delta^m_{min}\}$; **return** Δ;

Fig. 5. Algorithm Relax$^{\mathsf{TF}}$

It generates the top-1 pattern relaxation by selecting label relaxations δ with minimum $\gamma_Q(\delta) \cdot \mathcal{I}^G_Q(\delta)$ for each label in Q.

Correctness & Complexity. The correctness of Relax$^{\mathsf{TF}}$ is guaranteed by the property of Lawler's procedure. Algorithm Relax$^{\mathsf{TF}}$ is in $O(\mu k|V_Q|^2)$ time. Indeed, topRel is in $O(\mu|V_Q|^2)$ and the total time for branching candidate label relaxations is $O(k|V_Q|^2)$. Thus by the property of Lawler's procedure, Relax$^{\mathsf{TF}}$is in $O(k \cdot \mu|V_Q|^2) + O(k|V_Q|^2) = O(\mu k|V_Q|^2)$.

B. Top-k Diversified Relaxations for topKPR$^{\mathsf{DF}}$

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

Theorem 3: *(1) The decision problem of* topKPR$^{\mathsf{DF}}$ *is* NP*-complete. (2) The optimization problem of* topKPR$^{\mathsf{DF}}$ *is* APX*-hard to approximate.* □

Proof sketch: We prove that the decision problem of topKPR$^{\mathsf{DF}}$ is in NP by providing an NP algorithm: first guess a k-element set S and then check whether $S \subseteq \mathcal{U}_\mu(Q, T)$ and $F(Q, S) \leq B$ in PTIME. Here B is the bound on $F(Q, S)$ used by the decision version of topKPR$^{\mathsf{DF}}$. The NP-hardness is verified by a reduction from the k-clique problem [6]. The APX-hardness follows from an approximation-preserving reduction from the maximum clique problem, which is shown APX-hard [6] (see [2] for a complete proof). □

Despite of the hardness and inapproximability, we develop an algorithm for topKPR$^{\mathsf{DF}}$, denoted by Relax$^{\mathsf{DF}}$, by reducing topKPR$^{\mathsf{DF}}$ 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 [21].

Algorithm Relax$^{\mathsf{DF}}$. We next give algorithm Relax$^{\mathsf{DF}}$ by presenting the reduction from topKPR$^{\mathsf{DF}}$ to maxDP. The subtlety is that topKPR$^{\mathsf{DF}}$ is a minimization problem while maxDP is maximization. Nonetheless, we guarantee that optimal solutions to maxDP give us optimal solutions to topKPR$^{\mathsf{DF}}$.

Given Q, G, T, μ and k, algorithm Relax$^{\mathsf{DF}}$ 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_{Δ_1}, u_{Δ_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 Relax$^{\mathsf{DF}}$ simply returns k pattern relaxations encoded by V_k for maxDP.

Proposition 4: *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* topKPR$^{\mathsf{DF}}$. □

Proposition 4 ensures the following: Relax$^{\mathsf{DF}}$ *always returns the top-k pattern relaxations for* topKPR$^{\mathsf{DF}}$ *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 topKPR$^{\mathsf{DF}}$.

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 VII for details).

V. Answering Pattern Relaxations

In this section, we further study the evaluation of the top-k relaxed patterns produced by algorithms in Section IV.

Given Q, G, T and k pattern relaxations Δ_1, …, Δ_k, we aim to compute the answers to the relaxed patterns $Q \oplus \Delta_1$, …, $Q \oplus \Delta_k$ 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 that maximally utilizes computation sharing among the relaxed patterns *w.r.t.* T.

Algorithm evalPR. The algorithm, denoted by evalPR and shown in Fig. 6, works in two phases: (1) constructing the *minimum pairing tree*, a structure to organize the evaluation of relaxed patterns while maximizing shared computation; and

Input: Q, G, T, k pattern relaxations Δ_1, …, Δ_k;
Output: answers to $Q \oplus \Delta_1$, …, $Q \oplus \Delta_k$ in G.
1. compute the minimum pairing tree $\mathcal{T}$ of Δ_1, …, Δ_k for Q;
2. set u to the root of $\mathcal{T}$; $\mathcal{Q}$:=∅; $\mathcal{Q}$.push(u);
3. **while** $\mathcal{Q}$ is not empty **do**
4. u := $\mathcal{Q}$.pop();
5. u.ans := BDeval(pre(u), u); */* bounded decremental evaluation*/*
6. $\mathcal{Q}$.push(post(u));
7. **return** $\{u.\mathsf{ans} \mid u$ is a leaf of $\mathcal{T}\}$

Fig. 6. Algorithm evalPR

(2) bounded decremental evaluation for taxonomy simulation that carries out the computation in a *bounded* manner.

(1) Minimum pairing tree construction. The *minimum pairing tree* $\mathcal{T}$ of the k pattern relaxations Δ_1, …, Δ_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 Δ_1, …, Δ_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 Δ_1, …, Δ_n is a set P of $\lceil \frac{n}{2} \rceil$ pattern relaxations Δ'_1, …, $\Delta'_{\lceil \frac{n}{2} \rceil}$ 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 disjoint 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

$$\sum_{\Delta'\in P}\sum_{\Delta_i\in\Delta', i=1,2}\sum_{u\in V_Q} |\mathsf{cand}_G^T(f_{Q\oplus\Delta'}(u)) \setminus \mathsf{cand}_G^T(f_{Q\oplus\Delta_i}(u)|$$

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 $\mathsf{cand}_G^T(l)$ is the set of nodes v in G such that $f(v) \in \mathsf{desc}_T(l)$.

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.* relaxations in S, the total computation can be minimized. The minimum pairing tree organizes such minimization recursively.

Note that the minimum pairing tree $\mathcal{T}$ can be built in $O(k^{2.5} \log k)$ time, by invoking $\lceil \log k \rceil$ times of the algorithm for the *maximum weighted matching problem* [22].

Example 9: Recall the label relaxations δ_1,δ_2,…,δ_6 in the form in Example 6. When $k = 8$ and $\mu = 2$, one can verify that Δ_1,Δ_2,…,Δ_8 are the top-8 pattern relaxations *w.r.t.* topological ranking function, as shown in the leaves in Fig. 7. Algorithm evalPR firstly constructs the minimum pairing tree as shown in Fig. 7. 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_2. □

(2) Bounded decremental evaluation. After constructing $\mathcal{T}$ of Δ_1, …, Δ_k (line 1), evalPR then evaluates the relaxed patterns

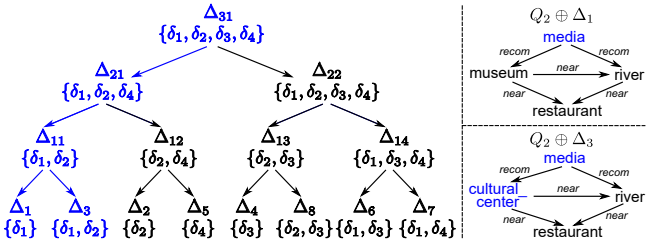


Fig. 7. Minimum pairing tree

of Q *w.r.t.* relaxations in the nodes of $\mathcal{T}$, from root to leaves (lines 2-6). More specifically, for each node u, it evaluates the relaxed pattern *w.r.t.* relaxation in u by *reusing* the answers to the relaxed patterns in the father node $\mathsf{pre}(u)$ of u in $\mathcal{T}$ via procedure BDeval (line 5). It finally returns the answers to the relaxed patterns in leaves of $\mathcal{T}$ (line 7).

Here 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 [13]. Following [13], 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 conducts the evaluation process. For example, evalPR evaluates $Q_2 \oplus \Delta_1$ and $Q_2 \oplus \Delta_3$, which are also shown in Fig. 7, based on the computation from root Δ_{31}, to Δ_{21}, and to Δ_{11} sequentially. Firstly, (a) evalPR finds the connected component containing $\mathsf{newspaper}_1$ in G_1 is the match result to $Q_2 \oplus \Delta_{31}$, and (b) further to $Q_2 \oplus \Delta_{21}$. Then (c) evalPR removes nodes $\mathsf{body\_of\_water}$ and $\mathsf{newspaper}_2$ from the answer in (b) as the match result to $Q_2 \oplus \Delta_{11}$. Finally, (d) evalPR removes nodes castle and $\mathsf{newspaper}_1$ from the answer in (c) as the match result to $Q_2 \oplus \Delta_1$, and (e) also finds the answer in (c) is exactly the match result to $Q_2 \oplus \Delta_3$.

One can find that the decremental process not only finds the match result for $Q_2 \oplus \Delta_1$ in G_1, but also finds the match result for $Q_2 \oplus \Delta_3$ meanwhile. Moreover, following the order organized by the minimum pairing tree, the above process also shares certain computation for evaluating $\Delta_2, \Delta_4, \ldots, \Delta_8$. □

VI. Explaining Relaxations

In Section V, 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? Below we answer this question, by studying the *match-relaxation explanation problem*.

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 results $(Q \oplus \Delta)(G)$ 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 results to $Q \oplus \Delta$ in G using label relaxations in Δ. In particular, when v is already in the match results 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 $\Delta_3 = \{\delta_1, \delta_2\}$ from Example 10, and the match result to the relaxed pattern $Q_2 \oplus \Delta_3$ in G_1 is $\{\mathsf{newspaper}_1, \mathsf{television\_show}, \mathsf{castle}, \mathsf{river}, \mathsf{exhibition\_hall}, \mathsf{take\_away\_food}, \mathsf{restaurant}\}$. One can verify that (1) the minimum explanation for the matched node $\mathsf{television\_show}$ *w.r.t.* Δ_3 is a subset of Δ_3, *i.e.,* $\mathcal{E}^m_{\Delta_3}(\mathsf{television\_show}) = \{\delta_1\}$; and (2) the minimum explanation for the matched node $\mathsf{newspaper}_1$ *w.r.t.* Δ_3 is Δ_3 itself, *i.e.,* $\mathcal{E}^m_{\Delta_3}(\mathsf{newspaper}_1) = \{\delta_1, \delta_2\}$. □

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 computed in Section IV and their match results $(Q \oplus \Delta_1)(G), \ldots, (Q \oplus \Delta_k)(G)$ in G computed in Section V, an integer $i \in [1, k]$, and a node v in $(Q \oplus \Delta_i)(G)$, the *match-relaxation explanation problem*, denoted by 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 IV) and their answers in G (Section V).

Depending on the ranking functions used for computing the top-k pattern relaxations, we study two instances of MRE, denoted by $\mathsf{MRE}_{\mathsf{TF}}$ and $\mathsf{MRE}_{\mathsf{DF}}$, respectively, to explain top-k relaxations identified by algorithm $\mathsf{Relax}^{\mathsf{TF}}$ with the topological ranking function and algorithm $\mathsf{Relax}^{\mathsf{DF}}$ with the diversified topological ranking function in Section IV, respectively.

Instance $\mathsf{MRE}_{\mathsf{TF}}$. The minimum explanation can be found in linear time for $\mathsf{MRE}_{\mathsf{TF}}$, based on the property below.

Proposition 5: *For any Q, G, T and top-k pattern relaxations $\Delta_1, \ldots, \Delta_k$ identified by $\mathsf{Relax}^{\mathsf{TF}}$, 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 (see [2] for a proof). It gives us a *linear* time algorithm, denoted by $\mathsf{expPR}^{\mathsf{TF}}$, for $\mathsf{MRE}_{\mathsf{TF}}$ 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 $\mathsf{expPR}^{\mathsf{TF}}$ is *optimal* since it only parses the input in linear time.

Instance $\mathsf{MRE}_{\mathsf{DF}}$. We next study $\mathsf{MRE}_{\mathsf{DF}}$ to explain the top-k diversified pattern relaxations identified by $\mathsf{Relax}^{\mathsf{DF}}$.

Due to the diversification component in the ranking function for problem $\mathsf{topKPR}^{\mathsf{DF}}$, Proposition 5 does not hold anymore. Worse still, the problem is intractable, as shown below.

Theorem 6: *The decision problem of $\mathsf{MRE}_{\mathsf{DF}}$ is NP-complete.* □

Proof sketch: To show it is in NP, we give an NP algorithm: first guess a subset Δ' of one of the k pattern relaxations, and then check whether $|\Delta'| \leq B$ and v is in match relation $(Q \oplus \Delta')(G)$ in PTIME. Here B is the bound on the size of

explanations used by the decision version of $\mathsf{MRE}_{\mathsf{DF}}$. To show it is NP-hard, we give a reduction from the minimum set cover problem, which is NP-complete [24] (see [2] for a proof). □

Nonetheless, we provide a *parameterized* algorithm, denoted by $\mathsf{expPR}^{\mathsf{DF}}$, that 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 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 $\mathsf{expPR}^{\mathsf{DF}}(M)$ 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 $\delta = l \to l'$ in Δ_j in a descending order by $\mathsf{cand}_G^T(l')$. Each time it checks whether $v \in (Q \oplus (\Delta_j \backslash \{\delta\}))(G)$ (via TSim in Section II if $(Q \oplus (\Delta_j \backslash \{\delta\}))(G)$ is not computed by evalPR in Section V), 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.

VII. Experimental Study

Using real-life and synthetic data, we verify the effectiveness and efficiency of taxonomy simulation, pattern relaxation of taxonomy simulation, and match-relaxation 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 built-in taxonomy graph with 735 concepts (nodes), where edges indicate an is-a relation, which forms a rooted forest with average height 2.29 (maximum height 6).

(2) YAGO [4] consists of (i) a data graph with 5.13M nodes and 5.39M edges; and (ii) a built-in taxonomy graph with 6488 concepts (nodes), which is also a rooted forest with average height 3.27 (maximum height 13).

(3) Synthetic data was produced by a graph-tool library [3]. The synthetic graphs $G(V_G, E_G, f_G)$ was controlled by 3 parameters: $|V_G|$, $|E_G|$, and the number $|L|$ of labels. We also generated taxonomy graphs for the synthetic graphs sharing the same label set and controlled by parameter $|V_T|$.

Pattern generator. We implemented a generator for producing random pattern graphs $Q(V_Q, E_Q, f_Q)$, controlled by 4 parameters: $|V_Q|$, $|E_Q| = \lfloor \alpha |V_Q| \rfloor$, 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 algorithms in C++: (a) our algorithms TSim, $\mathsf{Relax}^{\mathsf{TF}}$, $\mathsf{Relax}^{\mathsf{DF}}$, evalPR, $\mathsf{expPR}^{\mathsf{TF}}$ and $\mathsf{expPR}^{\mathsf{DF}}$; (b) our algorithms $\mathsf{evalPR}^{\mathsf{TF}}$ and $\mathsf{evalPR}^{\mathsf{DF}}$ for using evalPR to answer top-k relaxed patterns for topKPR and $\mathsf{topKPR}^{\mathsf{DF}}$, respectively; (c) algorithms $\mathsf{TSim}^{\mathsf{TF}}$ and $\mathsf{TSim}^{\mathsf{DF}}$ for using TSim k times to answer top-k relaxed patterns for topKPR and $\mathsf{topKPR}^{\mathsf{DF}}$, respectively; (d) algorithm gsim [16] for answering graph simulation queries; (e) algorithm $\mathsf{Relax}^{\mathsf{CMP}}$ using a ranking function from [29] for generating top-k relaxations, which ranks relaxations according to the total relaxation distance on the taxonomy graph between the relaxed and original pattern labels; and (f) algorithm $\mathsf{TSim}^{\mathsf{CMP}}$ for answering the k relaxed patterns found by $\mathsf{Relax}^{\mathsf{CMP}}$ 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. We next present our findings. In all the experiments, 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.

Exp-1: Effectiveness. We first evaluated the effectiveness.

(1) Effectiveness of taxonomy simulation. We evaluated the quality of the matches found by taxonomy simulation and its effectiveness in capturing meaningful matches.

Quality. To evaluate the accuracy of matches found by taxonomy simulation, 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.,* entities of persons, 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

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

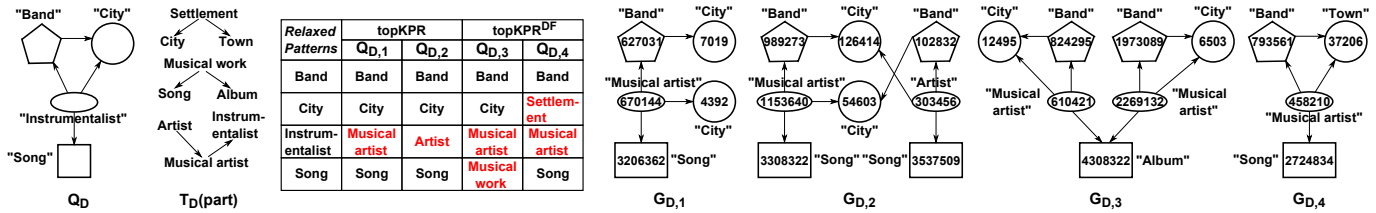
where $\mathsf{valid}(u, v)$ is 1 if data node v in G is an entity satisfying the conditions maintained for u in Q (we also used WordNet and Wiki to observe synonymous entities); and is 0 otherwise. Note that $\mathsf{acc}(S, Q, G)$ is in the range $[0, 1]$ and is 1 if S contains accurate answers only. 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 of the 30 patterns and found that their average accuracy is 0.98 and 0.94 on DBpedia and YAGO, respectively.

Quantity. We also quantified the effectiveness by comparing the average number of matches found by TSim and gsim. We used the pattern generator to generate patterns. Varying $|V_Q|$ of Q from 2 to 10, we report the results on DBpedia and YAGO in Figures 9(a) and 9(b). Observe the following. (i) TSim can identify more matches than gsim, as expected, in all cases. (ii) The effect is more evident on small patterns, *e.g.,* TSim finds matches 1.4 times more than gsim when $|V_Q|$ is 4 on DBpedia. (iii) Both TSim and gsim find very few matches for medium-size patterns on DBpedia and YAGO, *e.g.,* when $|V_Q| \geq 6$, they both cannot identify any matches. This also confirms the need for studying relaxation of taxonomy simulation.

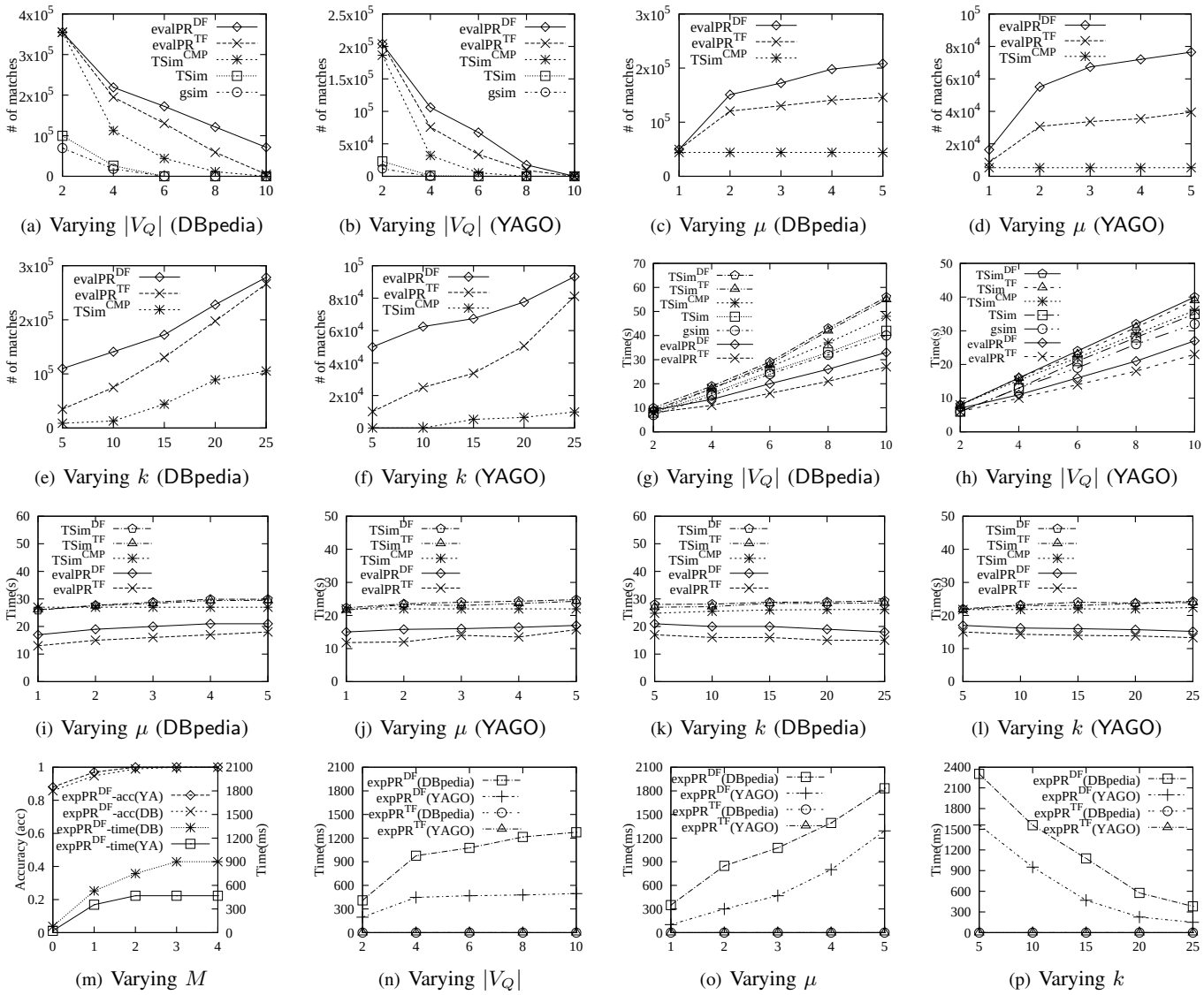
(2) Effectiveness of pattern relaxation. We also evaluated the effectiveness of pattern relaxation.

Quality. We first evaluated the quality of relaxed patterns. Using the same setting in (1), we inspected match relations to the top-5 relaxed patterned of the 30 patterns, and found that the



Relaxed Patterns	topKPR		topKPR^{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

Fig. 8. Real-life taxonomy simulation relaxation and matches on DBpedia



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

(e) Varying k (DBpedia) (f) Varying k (YAGO) (g) Varying $|V_Q|$ (DBpedia) (h) Varying $|V_Q|$ (YAGO)

(i) Varying μ (DBpedia) (j) Varying μ (YAGO) (k) Varying k (DBpedia) (l) Varying k (YAGO)

(m) Varying M (n) Varying $|V_Q|$ (o) Varying μ (p) Varying k

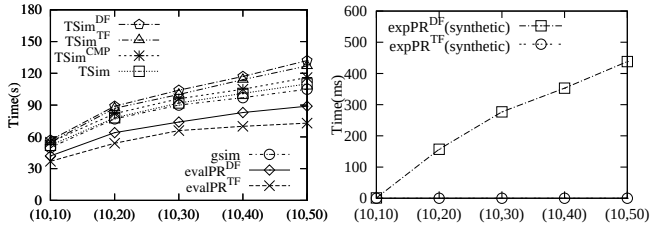
Fig. 9. Effectiveness and efficiency of taxonomy simulation relaxation and explanation on real-life data

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.

We further illustrate the quality with an example case.

As shown in Fig. 8, pattern graph Q_D is to find all "city" items in DBpedia that (a) each 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 data graph G_D, nodes are entities with unique id from different domains, with labels indicating their domains, and they only match the nodes of Q_D with the same geometry shapes, *e.g.*,, circles, ellipses, squares and pentagons.

Observe that there are no match results for Q_D on DBpedia via TSim. We next examine its relaxations. Using the taxonomy T_D in Fig. 8, we computed the top-2 relaxed patterns of Q_D and their match results in G, also shown in Fig. 8. Here $Q_{D,1}$



(a) Varying $|G|$ (synthetic) (b) Varying $|G|$ (synthetic)

Fig. 10. Scalability of taxonomy simulation relaxation and explanation

and $Q_{D,2}$ are top-2 relaxed patterns for topKPR, and $Q_{D,3}$ and $Q_{D,4}$ are top-2 relaxed patterns for topKPR$^{\mathsf{DF}}$. Part of match results to 4 relaxed patterns are $G_{D,1}$, $G_{D,2}$, $G_{D,3}$ and $G_{D,4}$, respectively. One can verify that they all are sensible matches and can capture the intention of pattern Q_D. However, such sensible matches cannot be returned when using Q_D directly. We ascribe the phenomenon to the inaccurate specification of patterns from users versus the numerous labels residing in knowledge graphs, *e.g.,* 735 built-in labels for DBpedia.

Quantity. We then further evaluated the effectiveness by comparing the average number of matches found by the k relaxed patterns returned by Relax$^{\mathsf{TF}}$, Relax$^{\mathsf{DF}}$ and Relax$^{\mathsf{CMP}}$, respectively, using evalPR$^{\mathsf{TF}}$, evalPR$^{\mathsf{DF}}$and TSim$^{\mathsf{CMP}}$.

(a) Impact of $|V_Q|$. Varying the number $|V_Q|$ of nodes in Q from 2 to 10, we report the results in Figures 9(a) and 9(b) for DBpedia and YAGO, respectively. Observe the following. (i) The number of matches found by evalPR$^{\mathsf{TF}}$, evalPR$^{\mathsf{DF}}$ and TSim$^{\mathsf{CMP}}$ 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, evalPR$^{\mathsf{TF}}$ and evalPR$^{\mathsf{DF}}$ consistently find more matches than TSim$^{\mathsf{CMP}}$. This verifies the effectiveness of information ratio in topological ranking function, which enables the use of information of data graphs in ranking. (iii) evalPR$^{\mathsf{DF}}$ identifies more matches than evalPR$^{\mathsf{TF}}$. This is because the diversification function tends to generate further 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 9(c) and 9(d), tell us the following. (i) Even when μ is small, both evalPR$^{\mathsf{TF}}$ and evalPR$^{\mathsf{DF}}$ are able to find sensible matches on the two datasets, and the quantity are much more than TSim$^{\mathsf{CMP}}$, *e.g.,* they found 8525 and 16351 matches when $\mu = 1$ on YAGO, respectively, compared to 5320 by TSim$^{\mathsf{CMP}}$. (ii) Both evalPR$^{\mathsf{TF}}$ and evalPR$^{\mathsf{DF}}$ make more use of larger μ as they can take more factors on the data graphs and taxonomy into consideration, *e.g.,* evalPR$^{\mathsf{TF}}$ and evalPR$^{\mathsf{DF}}$ found 39478 and 76502 matches when $\mu = 5$ on YAGO, while TSim$^{\mathsf{CMP}}$ 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 shown in Figures 9(e) and 9(f). We find that both evalPR$^{\mathsf{TF}}$ and evalPR$^{\mathsf{DF}}$ can find sensible matches even when k is small. For example, evalPR$^{\mathsf{TF}}$ identified 10100 matches when $k = 5$ on YAGO, and even more for evalPR$^{\mathsf{DF}}$, while TSim$^{\mathsf{CMP}}$ found no matches.

(3) Effectiveness of explanation. To evaluate the effectiveness of explanation, we randomly selected 100 matched nodes to the relaxed patterns found by Relax$^{\mathsf{TF}}$ and Relax$^{\mathsf{DF}}$, respectively, and used algorithms expPR$^{\mathsf{TF}}$ and expPR$^{\mathsf{DF}}$ (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 expPR$^{\mathsf{TF}}$ is always 1, we only report results for expPR$^{\mathsf{DF}}$ in Figure 9(m). We find that the explanations computed by expPR$^{\mathsf{DF}}$ 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.

Exp-2: Efficiency. In the second set of experiments, we evaluated the efficiency of our algorithms. All top-k pattern relaxation algorithms Relax$^{\mathsf{TF}}$, Relax$^{\mathsf{DF}}$ and Relax$^{\mathsf{CMP}}$ terminate within 2s on all cases. Below we focus on the efficiency of relaxation evaluation and explanation.

(1) Efficiency of taxonomy simulation. Using the same setting as Exp-1(1), we tested the efficiency of taxonomy simulation vs. graph simulation, by comparing TSim and gsim. The results are shown in Figures 9(g) and 9(h), and tell us the following. (i) Algorithm TSim scales well with big graphs, *e.g.,* it takes no more than 45s for pattern graphs with 10 nodes on data graphs with millions of nodes. (ii) The running time of TSim and gsim is almost equal in all cases.

(2) Efficiency of pattern relaxation. We compared the evaluation time of evalPR$^{\mathsf{TF}}$, evalPR$^{\mathsf{DF}}$, TSim$^{\mathsf{TF}}$, TSim$^{\mathsf{DF}}$ and TSim$^{\mathsf{CMP}}$, using the same setting as Exp-1(2). We report the findings in Figures 9(g), 9(h), 9(i), 9(j), 9(k) and 9(l), which tell us the following. (i) evalPR$^{\mathsf{TF}}$ and evalPR$^{\mathsf{DF}}$ improved TSim$^{\mathsf{TF}}$ and TSim$^{\mathsf{DF}}$ significantly, *e.g.,* evalPR$^{\mathsf{TF}}$ is 1.6 times faster than TSim$^{\mathsf{TF}}$when $k = 5$ on DBpedia. The improvement increases when $|V_Q|$ or k grows larger. (ii) Algorithm TSim$^{\mathsf{DF}}$ is the slowest among all cases and TSim$^{\mathsf{CMP}}$ is the fastest. This is because TSim$^{\mathsf{TF}}$ and TSim$^{\mathsf{DF}}$ encode the dataset information into ranking functions, such that they both find many more meaningful matches than others, while TSim$^{\mathsf{CMP}}$ always returns empty results even with medium size patterns, *e.g.,* queries Q on YAGO with $|V_Q| = 6$.

(3) Efficiency of explanation. Using the same setting as in Exp-1(3) with $M = 4$ by default, we evaluated the efficiency of our explanation algorithms expPR$^{\mathsf{TF}}$ and expPR$^{\mathsf{DF}}$ on DBpedia and YAGO. We report the results in Figures 9(m), 9(n), 9(o) and 9(p). Observe the following. (i) Both expPR$^{\mathsf{TF}}$ and expPR$^{\mathsf{DF}}$ 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 expPR$^{\mathsf{DF}}$ increases when M, $|V_Q|$ and μ increase, as expected. It decreases when k increases. Indeed, when k is large, more pattern relaxations are generated, such that the probability of invoking TSim by expPR$^{\mathsf{DF}}$ is small.

We also find that λ with less values lead to diverse and more match results to relaxed patterns. Indeed, evalPR$^{\mathsf{DF}}$ ($\lambda = 0.5$) finds more matches than evalPR$^{\mathsf{TF}}$ ($\lambda = 1$) in all cases. This

is because evalPR$^{\mathsf{DF}}$ tends to generate further relaxed ancestor labels for the diversification in relaxed patterns. Accordingly, evalPR$^{\mathsf{DF}}$ is a bit slower than evalPR$^{\mathsf{TF}}$.

Exp-3: Scalability. We also evaluated the scalability of the algorithms on synthetic data. Fixing $|V_G|$ of G to be 10M, $|L| = |V_T| = 2K$, we varied $|E_G|$ of G from 10M to 50M, corresponding to (10, 10) to (10, 50) in Figures 10(a) and 10(b), Observe the following. (i) All algorithms scale well with larger $|G|$. The slowest algorithm TSim$^{\mathsf{DF}}$ terminates within 130s in all cases, while its optimized version evalPR$^{\mathsf{DF}}$ finishes within 90s, as shown in Figure 10(a). (ii) Better still, the running time of evalPR$^{\mathsf{TF}}$ and evalPR$^{\mathsf{DF}}$ are not sensitive to the increment of $|G|$, as shown in Figure 10(a). (iii) expPR$^{\mathsf{TF}}$ and expPR$^{\mathsf{DF}}$ takes less than 1ms and 0.5s in all cases, respectively. The running time of expPR$^{\mathsf{DF}}$ increases when $|G|$ increases as shown in Figure 10(b). The findings are consistent with above.

Summary. From the experiments we find the following. (1) Taxonomy simulation captures high quality matches that cannot be found by graph simulation, *e.g.,* TSim returns 1.4 times more sensible matches than gsim for Q with 4 nodes on DBpedia with almost equal time. (2) Our top-k ranking functions and algorithms are effective in finding relaxations, such that more sensible matches can be identified, *e.g.,* 11.9 times more for Q with 4 nodes on DBpedia. (3) Our optimization on the evaluation of the top-k relaxed patterns is effective for large Q, and all evaluation algorithms scale well with larger G. (4) Our algorithms can explain matches to the relaxed patterns accurately and efficiently, *e.g.,* they explain relaxations with accuracy above 85% even without access to the data graphs, and achieve 99% accuracy when two data accesses are allowed on both DBpedia and YAGO, in 2.4s.

VIII. Conclusion

We have proposed a package of techniques for relaxing graph pattern matching queries. We have defined taxonomy simulation by combining taxonomy with graph simulation, to relax graph pattern matching semantically and structurally. We have proposed a notion of relaxation for taxonomy simulation, to enable the use of taxonomy information from both the "upward" and "downward" directions. We have designed practical functions to rank relaxations of taxonomy simulation patterns, developed practical algorithms to compute and answer top-k relaxed patterns, and to explain the matches captured by the relaxations. We have experimentally verified the effectiveness and efficiency of the techniques.

There are several issues that are of interests for future research. One is to study relaxing and explaining taxonomy-based subgraph isomorphism queries. Another is to consider the dynamic setting where data graphs change over time.

References

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

[2] Full version. http://homepages.inf.ed.ac.uk/ycao/full.pdf.

[3] Synthetic data. http://projects.skewed.de.

[4] YAGO. http://www.mpi-inf.mpg.de/yago.

[5] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, and D. Toman. Structure and content scoring for xml. In *PVLDB*, 2005.

[6] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. Complexity and approximation: Combinatorial optimization problems and their approximability properties. 1999.

[7] N. Bidoit, M. Herschel, and A. Tzompanaki. Efficient computation of polynomial explanations of why-not questions. In *CIKM*, 2015.

[8] A. Borodin, H. C. Lee, and Y. Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *PODS*, 2012.

[9] A. Cakmak and G. Ozsoyoglu. Taxonomy-superimposed graph mining. In *EDBT*, 2008.

[10] J. Cheney, L. Chiticariu, and W.-C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4), 2009.

[11] O. Corby, R. Dieng-Kuntz, C. Faron-Zucker, and F. L. Gandon. Searching the semantic web: Approximate query processing based on ontologies. *Intelligent Systems*, 21(1), 2006.

[12] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. Weikum. Language-model-based ranking for queries on rdf-graphs. In *CIKM*, 2009.

[13] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu. Graph pattern matching: From intractability to polynomial time. *PVLDB*, 3(1):1161–1172, 2010.

[14] C. Freire, W. Gatterbauer, N. Immerman, and A. Meliou. The complexity of resilience and responsibility for self-join-free conjunctive queries. *PVLDB*, 2015.

[15] A. Gangemi. Ontology design patterns for semantic web content. In *ISWC*, 2005.

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

[17] E. L. Lawler. 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.

[18] E. Little, K. Sambhoos, and J. Llinas. Enhancing graph matching techniques with ontologies. In *Information Fusion*, 2008.

[19] F. Mandreoli, R. Martoglia, and W. Penzo. Flexible query answering on graph-modeled data. In *EDBT*, 2009.

[20] D. Martinenghi and R. Torlone. Taxonomy-based relaxation of query answering in relational databases. *The VLDB Journal*, 23(5), 2014.

[21] A. Martínez-Gavara, V. Campos, M. Laguna, and R. Martí. Heuristic solution approaches for the maximum minsum dispersion problem. *Journal of Global Optimization*, 2016.

[22] S. Micali and V. V. Vazirani. An $o(|e||v|^{a/2})$ algorithm for finding maximum matching in general graphs. In *SFCS*, 1980.

[23] D. Mottin, A. Marascu, S. B. Roy, G. Das, T. Palpanas, and Y. Velegrakis. A probabilistic optimization framework for the empty-answer problem. In *PVLDB*, 2013.

[24] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[25] N. A. Reshef, A. Barger, Y. Dubinsky, I. Guy, and S. K. Davidson. Bon voyage: Social travel planning in the enterprise. In *CSCW*, 2012.

[26] S. Roy and D. Suciu. A formal approach to finding explanations for database queries. In *SIGMOD*, 2014.

[27] T. Tran, P. Cimiano, S. Rudolph, and R. Studer. *Ontology-based interpretation of keywords for semantic search*. Springer, 2007.

[28] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge university press, 1994.

[29] Y. Wu, S. Yang, and X. Yan. Ontology-based subgraph querying. In *ICDE*, 2013.

[30] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, and G. Weikum. Relationship queries on extended knowledge graphs. In *WSDM*, 2016.

[31] X. Yan, P. S. Yu, and J. Han. Substructure similarity search in graph databases. In *SIGMOD*, 2005.

[32] S. Yang, Y. Wu, H. Sun, and X. Yan. Schemaless and structureless graph querying. In *VLDB*, 2014.

APPENDIX

(A) Algorithms.

Procedure LawlerBranch **in algorithm** Relax$^{\mathsf{TF}}$

The adopted Lawler's procedure for dividing collections of label relaxations in algorithm Relax$^{\mathsf{TF}}$ is shown Fig. 11.

Input: A pattern relaxation $\Delta = \{\delta^1, \delta^2, \ldots, \delta^m\}$ and a collection
$\mathcal{L} = \{\mathcal{L'}_1, \ldots, \mathcal{L'}_m\}$ of candidate label relaxations;
Output: m sub-collections of candidate label relaxations.
1. **for** i **from** 1 **to** m **do**
2. **for** j **from** 1 **to** $i-1$ **do** $\mathcal{L}^i_j := \{\delta^j\}$;
3. $\mathcal{L}^i_i := \mathcal{L'}_i \setminus \{\delta^i\}$;
4. **for** j **from** $i+1$ **to** m **do** $\mathcal{L}^i_j := \mathcal{L'}_j$;
5. $\mathcal{L}^i$:= $(\mathcal{L}^i_1, \ldots, \mathcal{L}^i_m)$;
6. **return** $\langle \mathcal{L}^1, \ldots, \mathcal{L}^m \rangle$;

Fig. 11. Procedure LawlerBranch in algorithm Relax$^{\mathsf{TF}}$

(B) Proofs

Proof of Theorem 3

(1) We first show that the decision problem of topKPR$^{\mathsf{DF}}$ is in NP, and then show it is NP-hard. Note that the decision problem of topKPR$^{\mathsf{DF}}$ is stated as follows:

- *Input*: pattern Q, taxonomy T, natural numbers μ, k and B.
- *Question*: whether there exists a k-element set S of μ-bounded pattern relaxations for Q *w.r.t.* T such that $\sum_{\Delta \in S} F(Q, \Delta) \leq B$.

Upper bound. We prove that topKPR$^{\mathsf{DF}}$ is in NP by giving an NP algorithm, working as follows.

(a) Guess a k-element set S of pattern relaxations for Q *w.r.t.* T.

(b) Check whether S consists of μ-bounded pattern relaxations only and $\sum_{\Delta \in S} F(Q, S) \leq B$. Return "Yes" if so; otherwise go to step (a).

Lower bound. We show it is NP-hard by reduction from the k-clique problem. An instance of the clique problem consists of a graph $G(V, E)$ and a positive integer K, and is to decide whether G has a k-clique. Given an instance $G(V, E)$ and K of the clique problem, we construct an instance of the topKPR$^{\mathsf{DF}}$ problem, namely, pattern graph Q, data graph G', taxonomy graph T, positive integer k, constants $\lambda \in [0, 1]$ and μ and real number B, such that G has a K-clique if and only if there exists a k-set S of μ-bounded pattern relaxations for Q *w.r.t.* T with $F(Q, S) \leq B$.

(a) The pattern graph Q consists of a single node u, with label $f_Q(u) = A_0$.

(b) The taxonomy graph $T(V_T, E_T, f_T)$ is constructed as follows:

- V_T consists of $|V| + |E| + 1$ ndoes u_0, u_1, $\ldots u_{|V|}$, w_1, $\ldots$, $w_{|E|}$.
- E_T consists of (u_0, u_1), $\ldots$, $(u_0, u_{|V|})$, and (u_i, w_j) for each pair $(v_i, e_j) \in V \times E$ such that v_i is not a node of edge e_j in G.
- $f_T(u_i) = A_i (i \in [0, |V|])$; $f_T(w_j) = B_j (j \in [1, |E|])$, where A_0, $\ldots$, $A_{|V|}$, B_1, $\ldots$, $B_{|E|}$ are distinct labels.

Intuitively, u_1, $\ldots$, $u_{|V|}$ encode nodes of G and w_1, $\ldots$, $w_{|E|}$ encode edges of G.

(c) The data graph G' is exactly the same as the taxonomy graph T.

(d) Let $\lambda = 0$; $\mu = 1$; $k = K$; and $B = 0$.

We next show that G has a K-clique if and only if there exists a k-set S of 1-bounded pattern relaxations for Q *w.r.t.* T with $F(Q, S) \leq B$.

$\boxed{\Rightarrow}$ Assume that G has a K-clique $G_K = (V_K, E_K)$. Then we construct K-pattern relaxations S as follows. For each node v_i in V_K, include a pattern relaxatioin $\{A_0 \to A_i\}$ in S. Observe the following. For any two pattern relaxations $\Delta_i = \{A_0 \to A_i\}$ and $\Delta_j = \{A_0 \to A_j\}$ in S, $\theta(\Delta_i, \Delta_j) = 0$. For any pattern relaxations $\Delta_p = \{A_0 \to A_p\} \notin S$, $\Delta_q = \{A_0 \to A_q\} \notin S$, $\theta(\Delta_i, \Delta_p) > 0$ and $\theta(\Delta_p, \Delta_q) > 0$. Therefore, $F(Q, S) = 0 = B$.

$\boxed{\Leftarrow}$. Assume that there exists a set S of k 1-bounded pattern relaxations for $\mathcal{Q}$ *w.r.t.* T with $F(Q,S) \leq B$. Then $F(Q,S) = 0$. We show that S encodes a K-clique for G. Consider the subset $V_K \subseteq V$ of nodes in G such that, v_i of G is in V_K if and only if $\{A_0 \to B_i\}$ is in S. Observe that V_K is a K-clique of G. Indeed, for any two nodes v_i and v_j in V_K, since $\Delta_i = \{A_0 \to A_i\}$ and $\Delta_j = \{A_0 \to A_j\}$ are in S and $\theta_Q(\Delta_i, \Delta_j) = 0$, by the construction of T, we know that there must exist an edge e of G such that both v_i and v_j are connected to e. Thus V_K induces a K-clique of G.

(2) One can verify that the above reduction construction also gives us an AP-reduction from the maximum clique problem to the topKPR$^{\mathsf{DF}}$ problem. Since the maximum clique problem is APX-hard [6], so is the topKPR$^{\mathsf{DF}}$ problem.

Proof of Proposition 5 By algorithm Relax$^{\mathsf{TF}}$, if Δ is returned as the k-th pattern relaxation, then for any subset $\Delta' \subsetneq \Delta$, Δ' must also be returned prior to Δ. Thus for any v in $(Q \oplus \Delta_i)(G)$, there must exists $j \in [1, i]$ such that $\Delta_j \subsetneq \Delta_i$ and Δ_j is a minimum explanation for v in Q *w.r.t.* T.

Proof of Theorem 6 We first show that MRE$_{\mathsf{DF}}$ is in NP, and then show it is NP-hard. Note that the decision problem of MRE$_{\mathsf{DF}}$ is as follows.

- *Input*: pattern graph Q, data graph G, taxonomy graph T, k pattern relaxations $\Delta_1, \ldots, \Delta_k$ for Q *w.r.t.* T that can be returned by Relax$^{\mathsf{DF}}$, integer i, node v in $(Q \oplus \Delta_i)(G)$, and integer B.
- *Questioin*: whether there exists an explanation $\mathcal{E}_{\Delta_i}(v)$ for v *w.r.t.* Δ_i such that $|\mathcal{E}_{\Delta_i}(v)| \leq B$.

Upper bound. We show that MRE$_{\mathsf{DF}}$ is in NP by giving an NP algorithm, working as follows.

(a) Guess a subset Δ' of Δ_i.

(b) Check whether $v \in (Q \oplus \Delta')(G)$ and $|\Delta'| \leq B$. Return "Yes" if so; otherwise go to the step (a).

The algorithm is obviously correct. It is in NP since there are most exponential in $|\Delta|$ many guesses in step (a) and step (b) is in PTIME in $|Q|$, $|G|$, $|T|$ and $|\Delta|$.

Lower bound. We next show that it is NP-hard by reduction from the *set cover* problem. An instance of the set cover problem consists of a set U of n elements $e_1, \ldots, e_n$, a collection F of m subsets $S_1, \ldots, S_m$ of U, and an integer k. It is to decide whether there exists a subset C of F such that $\bigcup_{S \in C} S = U$ and $|C| \leq k$.

Given an instance U, F and K of the set cover problem, we construct an instance of MRE$_{\mathsf{DF}}$, namely, pattern graph Q, data graph G, taxonomy graph T, k pattern relaxations $\Delta_1, \ldots, \Delta_k$, integer $p \in [1, k]$, node $v \in (Q \oplus \Delta_p)(G)$ and integer B, such that U has a cover C of cardinality no larger than K if and only if there exists an explanation $\Delta'(v)$ for v *w.r.t.* Δ_p such that $|\Delta'(v)| \leq B$. The construction is given as follows.

(1) Pattern graph $Q(V_Q, E_Q, f_Q)$ is defined as follows:

- V_Q consists of $1 + m + n$ nodes: u_U, $u_{S_1}, \ldots, u_{S_m}$, $u_{e_1}, \ldots, u_{e_n}$.
- E_Q consists of $n + \sum_{S \in F} |S|$ edges: (u_U, u_{e_i}) for each $i \in [1, n]$, (u_{S_i}, u_{e_j}) for each $i \in [1, m]$ and each j such that $e_j \in S_i$.
- $f_Q(u_U) = l_U$, $f_Q(u_{S_i}) = l_{S_i}$ for each $i \in [1, n]$, $f_Q(u_{e_i}) = l_{e_i}$ for each $i \in [1, n]$.

(2) Data graph $G(V, E, f)$ is constructed as follows:

- V contains nodes v_U, $v_{S_1}, \ldots, v_{S_m}$, $v_{e_1}, \ldots, v_{e_n}$, w_U, $w_{S_1}, \ldots, w_{S_m}$, $w_{e_1}, \ldots, w_{e_n}$;
- E contains (v_U, v_{e_i}) and (w_U, w_{e_i}) for each $i \in [1, n]$, (v_{S_i}, v_{e_j}) and (w_{S_i}, w_{e_j}) for each $i \in [1, m]$ and each j such that $e_j \in S_i$;
- $f(u_U) = f(w_U) = l_U$, $f(u_{S_i}) = l_{S_i}$ and $f(w_{S_i}) = l'_{S_i}$ for each $i \in [1, m]$, $f(u_{e_j}) = f(w_{e_j}) = l_{e_j}$ for each $j \in [1, n]$.

(3) Taxonomy graph $T(V_T, E_T)$ is constructed as follows (we use labels to denote nodes in V_T):

- V_T contains $l'_{S_1}, \ldots, l'_{S_m}$, $l_{S_1}, \ldots, l_{S_m}$;
- E_T contains (l'_{S_i}, l_{S_i}) for each $i \in [1, m]$.

(4) Let $k = 2$.

(5) Let $\Delta_1 = \emptyset$ and Δ_2 be $\{(l_{S_i} \to l'_{S_i}) \,|\, i \in [1, m]\}$. Note that this can always possible be returned by Relax$^{\mathsf{DF}}$ by adding dummy nodes to G and T.

(6) Let $p = 2$ and v be w_U. Note that w_U is in the match results $(Q \oplus \Delta_2)(G)$.

(7) Let $B = K$.

We next show that there exists an explanation $\Delta(v)$ for v *w.r.t.* T with $|\Delta(v)| \leq K$ if and only if there exists $C \subseteq F$ such that $\bigcup_{S \in C} S = U$ and $|C| = K$.

$\boxed{\Rightarrow}$ Assume that there exists an explanation $\Delta(v) \subseteq \Delta_2$ with $|\Delta(v)| \leq K$. Then consider C constructed as follows: for each label relaxation $l_{S_i} \to l'_{S_i}$ in $\Delta(v)$, include set S_i in C. Thus $|C| = |\Delta(v)| \leq K$. Since $w_U \in (Q \oplus \Delta(v))(G)$, nodes w_{e_1}, …, w_{e_n} must be matched by $Q \oplus \Delta(v)$. Thus by the matching semantics, for each w_{e_i}, there at least one node w_{e_j} that is matched. Thus C is a cover of U, *i.e.,* U has a K-cover.

$\boxed{\Leftarrow}$ Assume that U has a K-cover $C = \{S_{i_1}, \ldots, S_{i_K}\}$. Then construct $\Delta(v)$ as follows: for each $S_{i_j} (j \in [1, K])$ in C, include $l_{S_{i_j}} \to l'_{S_{i_j}}$ in $\Delta(v)$. Then $w \in (Q \oplus \Delta(v)(G)$. Indeed, since C is a cover of U, nodes w_{e_1}, …, w_{e_n} are connected to $w_{i_j} (j \in [1, K])$. By the matching semantics, $w \in (Q \oplus \Delta(v))(G)$, *i.e.,* $\Delta(v)$ is an explanation for v in $\mathcal{Q}$ *w.r.t.* T, and $|\Delta(v)| = |C| = K$.