# Graph Query Reformulation with Diversity

Davide Mottin
University of Trento
mottin@disi.unitn.eu

Francesco Bonchi
Yahoo Labs, Barcelona
bonchi@yahoo-inc.com

Francesco Gullo
Yahoo Labs, Barcelona
gullof@acm.org

## ABSTRACT

We study a problem of graph-query reformulation enabling explorative query-driven discovery in graph databases. Given a query issued by the user, the system, apart from returning the result patterns, also proposes a number of specializations (i.e., supergraphs) of the original query to facilitate the exploration of the results.

We formalize the problem of finding a set of reformulations of the input query by maximizing a linear combination of coverage (of the original query's answer set) and diversity among the specializations. We prove that our problem is hard, but also that a simple greedy algorithm achieves a $\frac{1}{2}$-approximation guarantee.

The most challenging step of the greedy algorithm is the computation of the specialization that brings the maximum increment to the objective function. To efficiently solve this step, we show how to compute the objective-function increment of a specialization linearly in the number of its results and derive an upper bound that we exploit to devise an efficient search-space visiting strategy.

An extensive evaluation on real and synthetic databases attests high efficiency and accuracy of our proposal.

## Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Selection process*

## Keywords

Graph queries, query reformulation, graph databases

## 1. INTRODUCTION

*Graph databases*, i.e., large collections of moderately-sized graphs, have recently attracted a great deal of attention in the data-mining/databases literature, fueled by a wide range of applications: screening and drug design from chemical compounds [23], motif discovery in protein structures [13], identification of objects and scenes in computer vision [28], scientific workflows [2], and so on [1]. One of the most studied primitives in this context is *subgraph-search queries* [33]: given a query graph, find all graphs that the query is subgraph isomorphic to. Answering this type of query is computationally hard, as it relies on the **NP**-complete subgraph-isomorphism problem.

*KDD'15,* August 10-13, 2015, Sydney, NSW, Australia.
© 2015 ACM. ISBN 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783343.

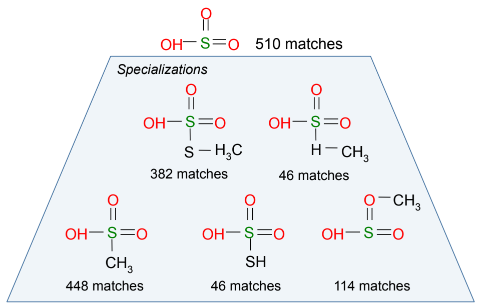


Figure 1: A graph query and a set of five reformulations (specializations) produced by our method on the real-world database `AIDS`.

Querying and mining graph databases is notoriously affected by the crucial *information-overload* (or *many-answers*) problem [18]: the answer to a query issued to large graph collections may return a considerable number of results that are hard to be processed by hand by a human being. In this paper we attempt to solve the information-overload problem in graph databases by proposing an exploratory approach where the user starts with a query to the graph database, and the system assists her by showing various *reformulations* of the query originally issued. Here by reformulation we mean a *specialization* (i.e., a supergraph) of the original query, in the spirit of providing the user with more specific queries that can help her refining her search of other relevant structures. In fact, as observed by several works focusing on query reformulation in different domains (e.g., keyword queries issued to search engines [5,25]), a natural approach when searching for information is to start with a general query for which the result set is unavoidably large, and then specialize the query in subsequent steps.

While query reformulation has been studied in contexts such as relational databases [18, 21], keyword search on structured data [34], or web search [9], to the best of our knowledge, this is the first work that focuses on query reformulation in graph databases.

**Example 1.** *In the example in Figure 1 the data analyst is searching for* `sulfonic acids`[1] *in the database.*

*Sulfonic acids are made of a* `sulfonyl hydroxide` *group (corresponding to the query in Figure 1) associated with some organic compound. Instead of directly searching independently for the various sulfonic acids, the data analyst issues the sulfonyl hydroxide group as a query, for which the database returns 510 graphs. In order to help the analyst in the exploratory search, the system proposes several specializations of the original query with the associated number of matching graphs. This immediately provides a high-level summary of what the database contains and guidance for the analyst in further inspecting more specific patterns.*

[1] `http://en.wikipedia.org/wiki/Sulfonic_acid`

Besides the bio/chemical domain, query reformulation in graph databases finds application in several other scenarios. As an example, in the context of scientific workflows, *provenance queries* are usually issued in order to find anomalies, i.e., sources of incorrect or partial information [2, 19]. These queries require to repeatedly check conditions in the workflows returned and may therefore be quite time consuming. Query reformulations (specializations) can be used in this case for easier and faster anomaly detection. Workflow analysis is also important to detect subprocesses that can be reused or optimized [26]: in this context, specializations can provide clear summaries to detect groups of similar subprocesses.

**Our proposal and contributions.** The specializations proposed for a given query must provide an effective high-level description of the original query results, in such a way that each specialization can somehow identify one of the different aspects contained in the result set. More specifically, a set of specializations should exhibit high *coverage* of the result set, and high *diversity* among them. The first requirement aims at guaranteeing that a large portion of the results of the original query are captured by the specializations, while the second requirement ensures a clear differentiation among the specializations, so that the user can discriminate and easily select the one(s) that better comply with her search goal.

Following the above intuition, we formalize the problem of finding a set of $k$ specializations that maximize a linear combination of coverage of the original query results and diversity among them. We prove that our problem is $\mathbf{NP}$-hard, but we also show that a simple greedy algorithm provides a $\frac{1}{2}$-approximation guarantee. However, quality guarantee does not imply efficiency. In fact, the key step of the greedy algorithm, i.e., finding the specialization leading to the maximum increment of the objective function, relies on a $\#\mathbf{P}$-complete problem. Our goal here is to solve such a step efficiently *while still guaranteeing optimality*, as this is needed to preserve the aforementioned approximation guarantee. To this end, we provide a number of technical insights into our problem, namely a fast computation of the objective-function increment and an upper bound on the maximum increment achievable, and exploit them to devise an efficient pruning strategy.

Our main contributions are summarized as follows:

- We formalize the novel problem of finding a set of $k$ reformulations (specializations) of an input query graph so as to maximize a linear combination of coverage and diversity.
- We show that our problem is $\mathbf{NP}$-hard, as well as that our coverage function is monotone submodular, and that our diversity measure is a pseudometric. These two properties allow us to adopt a greedy algorithm with provable quality guarantees.
- In order to guarantee efficiency, we devise a fast yet *exact* algorithm for the computation of the specialization leading to the maximum objective-function increment.
- We perform an extensive evaluation on real and synthetic graph databases. Results confirm high quality and efficiency achieved by our method.

**Roadmap.** The rest of the paper is organized as follows. In Section 2 we define the problem of graph query reformulation with diversity. Section 3 presents the proposed algorithm(s). In Section 4 we report our experiments. Finally, Section 5 overviews the related literature, while Section 6 concludes the paper.

## 2. PROBLEM STATEMENT

Let $\mathcal{D}$ be a *graph database* defined over a set of labels $L$. Each element of $\mathcal{D}$ is a labeled graph $G = (V, E, \ell)$, where $V$ is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $\ell : V \cup E \to L$ is a function that assigns a label from $L$ to each vertex in $V$ and each edge in $E$. For presentation clarity, we assume the graphs in $\mathcal{D}$ to be undirected; however, all our methods can straightforwardly handle directed graphs without any significant modifications.

A *graph isomorphism* between two graphs $G_1 = (V_1, E_1, \ell_1)$ and $G_2 = (V_2, E_2, \ell_2)$ is a bijective function $\mu : V_1 \to V_2$ such that: (i) $\ell_1(u) = \ell_2(\mu(u))$, for each $u \in V_1$, and (ii) for every $(u, v) \in E_1$, $(\mu(u), \mu(v)) \in E_2$ and $\ell_1(u, v) = \ell_2(\mu(u), \mu(v))$, and viceversa. If a graph isomorphism exists between $G_1$ and $G_2$, we say that $G_1$ and $G_2$ are *isomorphic*. If a graph isomorphism exists between $G_1$ and a subgraph of $G_2$, we say that $G_1$ is *subgraph isomorphic* to $G_2$, and we denote it by $G_1 \sqsubseteq G_2$.

A *query* $Q$ to a graph database $\mathcal{D}$ is a *connected* labeled graph. The result set to $Q$ is the set $\mathcal{R}_Q = \{G \in \mathcal{D} \mid Q \sqsubseteq G\}$ of all graphs in $\mathcal{D}$, which $Q$ is subgraph isomorphic to. We refer to $\mathcal{R}_Q$ as the *results* or the *result set* of $Q$.

Given a query $Q$, a *specialization* $Q'$ of $Q$ is a connected labeled graph such that $Q \sqsubseteq Q'$. We denote by $\mathbb{S}_Q$ the set of all possible specializations of $Q$ in the entire graph database $\mathcal{D}$, i.e., $\mathbb{S}_Q = \bigcup_{G \in \mathcal{D}} \{Q' \mid Q' \neq Q, Q \sqsubseteq Q' \sqsubseteq G,\ Q' \text{connected}\}$. It can be observed that, by definition, the result set of every specialized query $Q'$ is always a subset of the result set of the original query $Q$, i.e., $\mathcal{R}_{Q'} \subseteq \mathcal{R}_Q$.

Given a graph database $\mathcal{D}$ and a query $Q$, our goal is to find a set of $k$ specializations of $Q$ that captures well the result set $\mathcal{R}_Q$ of $Q$. More specifically, we aim at selecting a set of specializations of cardinality $k$ that exhibits $(i)$ high *coverage* of the result set $\mathcal{R}_Q$, and $(ii)$ high *diversity* among the subsets of $\mathcal{R}_Q$ identified by the single specializations. We next formalize these concepts.

The coverage of a set of specializations $\mathcal{Q}$ is defined as the number of results in $\mathcal{R}_Q$ captured by the specializations:

$$cov(\mathcal{Q}) = \left| \bigcup_{Q' \in \mathcal{Q}} \mathcal{R}_{Q'} \right|, \tag{1}$$

while the diversity between two queries $Q'$ and $Q''$ is defined as the number of uncommon results:

$$\begin{aligned} div(Q', Q'') &= |\mathcal{R}_{Q'} \cup \mathcal{R}_{Q''}| - |\mathcal{R}_{Q'} \cap \mathcal{R}_{Q''}| = \\ &= |\mathcal{R}_{Q'}| + |\mathcal{R}_{Q''}| - 2\,|\mathcal{R}_{Q'} \cap \mathcal{R}_{Q''}|. \end{aligned} \tag{2}$$

Overall, the function we aim at maximizing is:

$$f(\mathcal{Q}) = cov(\mathcal{Q}) + \lambda \sum_{Q', Q'' \in \mathcal{Q}} div(Q', Q''), \tag{3}$$

where $\lambda \in [0, 1]$ is a parameter that trades off between coverage and diversity, while also playing the role of a regularization factor in case of different scales of the two terms. Finally, the problem we tackle in this work is formally defined as follows:

**Problem 1.** *Given a graph database $\mathcal{D}$, a query $Q$, and an integer $k$, find a set $\mathcal{Q}^*$ of specializations of $Q$ such that:*

$$\mathcal{Q}^* = \operatorname*{arg\,max}_{\mathcal{Q} \subseteq \mathbb{S}_Q} f(\mathcal{Q}) \quad \textit{subject to} \quad |\mathcal{Q}| = k.$$

For $\lambda = 0$, Problem 1 corresponds to the well-known MAXIMUMCOVERAGE problem [12], which is known to be $\mathbf{NP}$-hard. As a result, Problem 1 is $\mathbf{NP}$-hard as well.

**Example 2.** *Figure 2 shows an example of Problem 1. The figure depicts a query $Q$, its corresponding result set $\mathcal{R}_Q = \{R_1, \ldots, R_5\}$, and four specializations $Q'_1, \ldots, Q'_4$. The subset of $\mathcal{R}_Q$ captured by each specialization is as follows: the first four results and the last four results form the result set of $Q'_1$ and $Q'_2$, respectively, while the results of $Q'_3$ and $Q'_4$ are $\{R_1, R_2\}$ and $\{R_3, R_4\}$, respectively. We also assume $\lambda = 0.3$.*

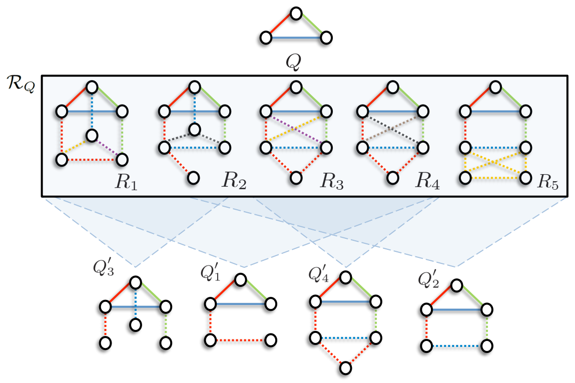


Figure 2: An instance of our problem.

*For $k = 2$, it can intuitively be observed that $\{Q'_3, Q'_4\}$ detect two of the main discriminating specialized queries arising from the result set $\mathcal{R}_Q$. The solution $\{Q'_1, Q'_2\}$ instead does not summarize $\mathcal{R}_Q$ equally well, as $Q'_1$ and $Q'_2$ identify two very general and not really informative patterns that are similar to one another and, as such, are both present in most of the results in $\mathcal{R}_Q$. This observation is acknowledged by the notions of coverage and diversity: $\{Q'_1, Q'_2\}$ have indeed slightly larger coverage than $\{Q'_3, Q'_4\}$, but they also exhibit much smaller diversity, which makes the latter solution preferable. Hence, this example shows that coverage and diversity are both critical in order to find a valid set of specializations. Our objective function $f$ captures this main finding: $f(\{Q'_3, Q'_4\})$ is larger than $f(\{Q'_1, Q'_2\})$, thus $\{Q'_3, Q'_4\}$ is preferred to $\{Q'_1, Q'_2\}$ according to $f$.*

## 3. ALGORITHMS

We next focus on how to solve Problem 1. We first discuss a naïve solution based on frequent subgraph mining (Section 3.1). Then, we shift the attention to the proposed approach: we prove some properties of our objective function $f$ (Section 3.2) , based on which we present a greedy algorithm exhibiting a $\frac{1}{2}$-approximation guarantee, while in Section 3.3 we treat the subproblem of finding the specialization that maximizes the *marginal potential gain* (defined next), which is the key step of the greedy algorithm.

### 3.1 A naïve approach

The objective function defined in Equation (3) reminds a notion of frequency: the more a specialization covers a result set, the more frequent that specialization is among the graphs in the result set.

This observation allows for defining a simple heuristic strategy to attack Problem 1, which is based on the well-known problem of *frequent subgraph mining* [14, 22, 32]: given a graph database and a threshold $\sigma$, find all subgraphs that are contained into at least $\sigma$ graphs of the database. In our context we do not have a threshold-based problem definition; however, a natural yet straightforward way of adapting the frequent-subgraph-mining problem to our context exists, and it corresponds to ask for the top-$k$ frequent subgraphs that are supergraph of the input query graph.

The advantage of using this approach as a solution to Problem 1 is that the literature on frequent subgraph mining can be reused almost as is, since the adaptation of existing frequent-subgraph-mining techniques to this variant of the problem is trivial. Unfortunately, this simple approach is not guaranteed to produce high-quality results. Indeed, while the notion of frequency is related to the notion of coverage, the notion of diversity is instead completely ignored. Taking into account diversity is crucial to find meaningful solutions, as clearly demonstrated in Example 2.

**Algorithm 1** Greedy

```
Input: A graph database $\mathcal{D}$, a query $Q$, an integer $k$
Output: A set of specialized queries $\mathcal{Q}$
1: $\mathcal{Q} \leftarrow \emptyset$
2: while $|\mathcal{Q}| < k$ do
3:     $Q^* \leftarrow \arg\max_{Q' \in \mathbb{S}_Q \setminus \mathcal{Q}} \Delta_f(\mathcal{Q}, Q')$        ▷ Equation (4)
4:     $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{Q^*\}$
```

The poor effectiveness of this naïve strategy is also supported by experimental evidence (see Section 4).

### 3.2 An approach with quality guarantees

The algorithm we propose as a more principled solution to Problem 1 relies on a recent result by Borodin *et al.* [6]. Given a universe of elements $U$, let $w : 2^U \rightarrow \mathbb{R}$ be a non-negative function measuring the quality of every subset of $U$, and $d : U \times U \rightarrow \mathbb{R}$ be a distance function between elements of $U$. Also, let $g$ be a set function defined as a linear combination of $w$ and the sum of pairwise distances computed according to $d$, i.e., for any $\hat{U} \subseteq U$, $g(\hat{U}) = w(\hat{U}) + \lambda \sum_{u',u'' \in \hat{U}} d(u', u'')$ (with $\lambda \in [0, 1]$ being a parameter). Finally, given a subset $U' \subset U$ and an element $u \notin U'$, let $\frac{1}{2}(w(U' \cup \{u\}) - w(U')) + \lambda(\sum_{u',u'' \in \hat{U}' \cup \{u\}} d(u', u'') - \sum_{u',u'' \in \hat{U}} d(u', u''))$ denote the *marginal potential gain*[2] of the function $g$. Borodin *et al.* [6] show that, for the problem of maximizing a set function like $g$ under a cardinality constraint, a greedy algorithm that iteratively selects the element maximizing the marginal potential gain achieves a $\frac{1}{2}$-approximation guarantee if $(i)$ the quality function $w$ is monotone submodular, and $(ii)$ the universe $U$ spans a metric space, i.e., $d$ is a metric. As shown in more detail next, this result holds even if $d$ is a *pseudometric*.

In our context, the universe $U$ corresponds to the set $\mathbb{S}_Q$ of all possible specializations of the input query $Q$, while the quality function corresponds to the coverage function $cov$ and the distance between two elements is measured in terms of the diversity $div$ between two specializations. Also, given a set of specializations $\mathcal{Q} \subseteq \mathbb{S}_Q$ and a specialization $Q' \in \mathbb{S}_Q \setminus \mathcal{Q}$, and denoting by $\Delta_{cov}(\mathcal{Q}, Q') = cov(\mathcal{Q} \cup \{Q'\}) - cov(\mathcal{Q})$ the marginal gain of the coverage term and by $\Delta_{div}(\mathcal{Q}, Q') = \sum_{\hat{Q} \in \mathcal{Q}} div(\hat{Q}, Q')$ the marginal gain of the diversity term, the marginal potential gain of our function $f$ is defined as

$$\Delta_f(\mathcal{Q}, Q') = \frac{1}{2}\Delta_{cov}(\mathcal{Q}, Q') + \lambda\Delta_{div}(\mathcal{Q}, Q'). \tag{4}$$

Thus, to exploit the above result by Borodin *et al.* [6] we need to prove that (1) the function $cov$ is monotone submodular, and (2) $div$ is a pseudometric. $cov(\cdot)$ is a classic coverage function, therefore it is submodular by construction. Moreover, $div$ is the symmetric difference[3] between sets, that is proved to be a pseudometric.

The proposed Greedy algorithm, whose pseudocode is shown as Algorithm 1, iteratively selects the specialization $Q^*$ that brings the maximum marginal potential gain to the objective function $f$, until $k$ specializations have been selected. The following result holds.

**Theorem 1.** *Greedy is a $\frac{1}{2}$-approximation algorithm for Problem 1.*

*Proof.* Looking at the proof of Theorem 1 in [6], it is easy to see that such a theorem remains true even in case of pseudometrics, as the indescernibility axiom is not exploited at all. □

[2]The $\frac{1}{2}$ is introduced in [6] to prove the approximation bound.

[3]http://en.wikipedia.org/wiki/Symmetric_difference

## 3.3 Maximizing the marginal potential gain

The proposed Greedy needs to face two main challenges:

- Finding the element that maximizes the marginal potential gain is very difficult in our context, as it corresponds to selecting a specialized query $Q^* \in \mathbb{S}_Q \setminus \mathcal{Q}$ that achieves the desired maximum objective-function increment, and, in the worst case, this requires to enumerate all possible subgraphs of each graph $G \in \mathcal{D}$ that $Q$ is subgraph isomorphic to. Such a problem corresponds to counting all subgraph-isomorphic structures in a graph and is known to be $\#\mathbf{P}$-complete [30].
- Our function $f$ is *non-monotone* in the size of the results: this makes the design of pruning strategies non-trivial at all (e.g., traditional downward-closure-based algorithms do not work).

In the following we present the proposed solution to the most critical step of the Greedy algorithm, i.e., finding the element that maximizes the marginal potential gain. We aim at solving this step efficiently while still guaranteeing optimality, as this is needed to preserve the approximation guarantee of Theorem 1. To this end, we devise an algorithm, Fast_MMPG, that visits the search space in a smart way based on two main findings: ($i$) an efficient computation of the marginal potential gain $\Delta_f$, and ($ii$) an upper bound on the maximum marginal potential gain achievable by a set of specializations which is used to substantially prune the search space.

**Computing $\Delta_f$ in linear time.** We show here how to efficiently solve the frequently-occurring subproblem of computing $\Delta_f$ when a specialization $Q'$ is given. According to Equation (4), $\Delta_f$ is a linear combination of $\Delta_{cov}$ and $\Delta_{div}$. The marginal gain $\Delta_{cov}$ can be computed by a simple scan of the results in $\mathcal{R}_{Q'}$, thus taking $\mathcal{O}(|\mathcal{R}_{Q'}|)$ time. As far as $\Delta_{div}$ concerns, a naïve computation would instead consider the results of all specializations in the current set $\mathcal{Q}$, thus requiring quadratic time. We show how, adopting a vector that keeps track of how many specializations (among the ones already computed) capture any result of the input query, the computation takes only $\mathcal{O}(|\mathcal{R}_{Q'}|)$ time as well.

Given a query $Q$, let $\{R_1, \ldots, R_n\}$ denote the graphs in its result set $\mathcal{R}_Q$. As stated above, the results of every specialized query $Q'$ of $Q$ is guaranteed to be a subset of $\mathcal{R}_Q$. Therefore, one can alternatively keep track of the results of $Q'$ by using a binary $n$-dimensional vector $\mathbf{x}_{Q'}$, where $\mathbf{x}_{Q'}[i] = 1$ if and only if $R_i \in \mathcal{R}_{Q'}$. Given a set of specializations $\mathcal{Q} \subseteq \mathbb{S}_Q$, let also $\mathbf{m}_\mathcal{Q} = \sum_{\hat{Q} \in \mathcal{Q}} \mathbf{x}_{\hat{Q}}$ be an $n$-dimensional integer vector whose $i$-th entry contains the number of specializations in $\mathcal{Q}$ having $R_i$ among their results. We hereinafter refer to $\mathbf{m}_\mathcal{Q}$ as the *multiplicity vector*. We also use $\|\mathbf{v}\|$ to denote the $L_1$-norm of a vector $\mathbf{v}$. In the next theorem we show how to exploit $\mathbf{m}_\mathcal{Q}$ and $\mathbf{x}_{Q'}$ to achieve the desired $\mathcal{O}(|\mathcal{R}_{Q'}|)$-time computation of $\Delta_{div}$.

**Theorem 2.** *Given a set of specializations $\mathcal{Q} \subseteq \mathbb{S}_Q$ and a specialization $Q' \in \mathbb{S}_Q \setminus \mathcal{Q}$, the marginal gain $\Delta_{div}(\mathcal{Q}, Q')$ is:*
$$\Delta_{div}(\mathcal{Q}, Q') = \|\mathbf{m}_\mathcal{Q}\| + |\mathcal{Q}| \times |\mathcal{R}_{Q'}| - 2\,\mathbf{m}_\mathcal{Q} \cdot \mathbf{x}_{Q'}.$$

*Proof.* $\Delta_{div}(\mathcal{Q}, Q')$ is equal to:
$$\begin{aligned}\Delta_{div}(\mathcal{Q}, Q') &= \sum_{\hat{Q}\in\mathcal{Q}} div(\hat{Q}, Q') = \\ &= \sum_{\hat{Q}\in\mathcal{Q}} \Big(|\mathcal{R}_{\hat{Q}}| + |\mathcal{R}_{Q'}| - 2|\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q'}|\Big) \\ &= \underbrace{\sum_{\hat{Q}\in\mathcal{Q}} |\mathcal{R}_{\hat{Q}}|}_{\|\mathbf{m}_\mathcal{Q}\|} + \underbrace{\sum_{\hat{Q}\in\mathcal{Q}} |\mathcal{R}_{Q'}|}_{|\mathcal{Q}|\times|\mathcal{R}_{Q'}|} - 2\sum_{\hat{Q}\in\mathcal{Q}} |\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q'}|. \qquad (5)\end{aligned}$$

By noting that the $i$-th entry of the vector $\mathbf{m}_\mathcal{Q}$ can alternatively be expressed as $\mathbf{m}_\mathcal{Q}[i] = \sum_{\hat{Q}\in\mathcal{Q}} \mathbb{1}[R_i \in \mathcal{R}_{\hat{Q}}]$ (where $\mathbb{1}[\cdot]$ is the indicator function), $\sum_{\hat{Q}\in\mathcal{Q}} |\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q'}|$ in (5) can be rewritten as:
$$\begin{aligned}\sum_{\hat{Q}\in\mathcal{Q}} |\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q'}| &= \sum_{\hat{Q}\in\mathcal{Q}} \sum_{\substack{i\in[1..n],\\ R_i\in\mathcal{R}_{Q'}}} \mathbb{1}[R_i \in \mathcal{R}_{\hat{Q}}] \\ &= \sum_{\substack{i\in[1..n],\\ R_i\in\mathcal{R}_{Q'}}} \underbrace{\sum_{\hat{Q}\in\mathcal{Q}} \mathbb{1}[R_i \in \mathcal{R}_{\hat{Q}}]}_{\mathbf{m}_\mathcal{Q}[i]} = \\ &= \sum_{i\in[1..n]} (\mathbf{x}_{Q'}[i] \times \mathbf{m}_\mathcal{Q}[i]) = \mathbf{m}_\mathcal{Q} \cdot \mathbf{x}_{Q'}. \quad (6)\end{aligned}$$
The proof is completed combining (5) and (6). $\square$

It is easy to see that the rewriting in Theorem 2 allows for computing $\Delta_{div}$ linearly in $|\mathcal{R}_{Q'}|$ as the terms $\|\mathbf{m}_\mathcal{Q}\|$ and $|\mathcal{Q}| \times |\mathcal{R}_{Q'}|$ are constant, while the scalar product $\mathbf{m}_\mathcal{Q} \cdot \mathbf{x}_{Q'}$ requires a scan of only the $|\mathcal{R}_{Q'}|$ non-zero entries of $\mathbf{x}_{Q'}$.

**Upper bound on $\Delta_f$.** Here we derive an upper bound on the marginal potential gain $\Delta_f$ (Equation (4)) exhibited by a set of specializations. Specifically, given a specialized query $Q'$, let $\mathcal{T}_{Q'}$ denote the set of all specializations that $Q'$ is subgraph isomorphic to, i.e., $\mathcal{T}_{Q'} = \{Q'' \in \mathbb{S}_Q \setminus \mathcal{Q} \mid Q' \sqsubseteq Q''\}$. We show how to bound the maximum marginal potential gain achievable by a specialization in $\mathcal{T}_{Q'}$ in a very efficient manner, that is by looking only at the specialization $Q'$. The ultimate goal is to exploit the resulting upper bound to early recognize (and skip) unnecessary portions of the search space.

We derive our upper bound by studying which results in $\mathcal{R}_Q$ should be captured by a specialization in $\mathcal{T}_{Q'}$ to achieve maximum marginal potential gain. To this end, let $\mathcal{R}_Q = \{R_1, \ldots, R_n\}$ denote the results of the original query $Q$. We assume that a set of specializations $\mathcal{Q} \subseteq \mathbb{S}_Q$ have been computed and kept track of the results identified by $\mathcal{Q}$ by the multiplicity vector $\mathbf{m}_\mathcal{Q}$ introduced above. Let $Q_1, Q_2 \in \mathbb{S}_Q \setminus \mathcal{Q}$ be two specializations such that their corresponding result sets differ by only one element, i.e., $\mathcal{R}_{Q_2} = \mathcal{R}_{Q_1} \cup \{R_j\}$. We want to study when the marginal potential gain brought by $Q_2$ is no less than the one given by $Q_1$, i.e., when $\phi_f = \Delta_f(\mathcal{Q}, Q_2) - \Delta_f(\mathcal{Q}, Q_1) \geq 0$. We focus here on a pair of specializations whose result sets differ by only one element in order to simplify the presentation of the theoretical results therein. This however does not result in any loss of generality, as shown later in Theorem 3.

We start our reasoning by showing how to profitably rearrange the quantities $\Delta_{cov}(\mathcal{Q}, Q_2) - \Delta_{cov}(\mathcal{Q}, Q_1)$ (Lemma 1) and $\Delta_{div}(\mathcal{Q}, Q_2) - \Delta_{div}(\mathcal{Q}, Q_1)$ (Lemma 2).

**Lemma 1.** *Let $Q_1, Q_2 \in \mathbb{S}_Q \setminus \mathcal{Q}$, s.t. $\mathcal{R}_{Q_2} = \mathcal{R}_{Q_1} \cup \{R_j\}$. It holds that* $\Delta_{cov}(\mathcal{Q}, Q_2) - \Delta_{cov}(\mathcal{Q}, Q_1) = \mathbb{1}[\mathbf{m}_\mathcal{Q}[j] = 0]$.

*Proof.*
$$\begin{aligned}&\Delta_{cov}(\mathcal{Q}, Q_2) - \Delta_{cov}(\mathcal{Q}, Q_1) = \\ &= \sum_{\substack{i\in[1..n],\\ R_i\in\mathcal{R}_{Q_2}}} \mathbb{1}[\mathbf{m}_\mathcal{Q}[i] = 0] - \sum_{\substack{i\in[1..n],\\ R_i\in\mathcal{R}_{Q_1}}} \mathbb{1}[\mathbf{m}_\mathcal{Q}[i] = 0] \\ &= \sum_{\substack{i\in[1..n],\\ R_i\in\mathcal{R}_{Q_1}}} \mathbb{1}[\mathbf{m}_\mathcal{Q}[i] = 0] + \mathbb{1}[\mathbf{m}_\mathcal{Q}[j] = 0] - \sum_{\substack{i\in[1..n],\\ R_i\in\mathcal{R}_{Q_1}}} \mathbb{1}[\mathbf{m}_\mathcal{Q}[i] = 0] \\ &= \mathbb{1}[\mathbf{m}_\mathcal{Q}[j] = 0] \qquad \square\end{aligned}$$

**Lemma 2.** *It holds that* $\Delta_{div}(\mathcal{Q}, Q_2) - \Delta_{div}(\mathcal{Q}, Q_1) = |\mathcal{Q}| - 2\mathbf{m}_\mathcal{Q}[j]$.

*Proof.*

$$\begin{aligned}
&\Delta_{div}(\mathcal{Q}, Q_2) - \Delta_{div}(\mathcal{Q}, Q_1) = \\
&\quad = \sum_{\hat{Q}\in\mathcal{Q}} div(\hat{Q}, Q_2) - \sum_{\hat{Q}\in\mathcal{Q}} div(\hat{Q}, Q_1) \\
&\quad = \sum_{\hat{Q}\in\mathcal{Q}} \Big(|\mathcal{R}_{\hat{Q}}| + \underbrace{|\mathcal{R}_{Q_2}|}_{|\mathcal{R}_{Q_1}|+1} - 2|\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q_2}|\Big) \\
&\qquad\quad - \sum_{\hat{Q}\in\mathcal{Q}} \Big(|\mathcal{R}_{\hat{Q}}| + |\mathcal{R}_{Q_1}| - 2|\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q_1}|\Big) \\
&\quad = |\mathcal{Q}| - 2\sum_{\hat{Q}\in\mathcal{Q}} \Big(|\mathcal{R}_{\hat{Q}} \cap (\mathcal{R}_{Q_1} \cup \{R_j\})| - |\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q_1}|\Big) \\
&\quad = |\mathcal{Q}| - 2\sum_{\hat{Q}\in\mathcal{Q}} \Big(|\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q_1}| + \mathbb{1}[R_j \in \mathcal{R}_{\hat{Q}}] - |\mathcal{R}_{\hat{Q}} \cap \mathcal{R}_{Q_1}|\Big) \\
&\quad = |\mathcal{Q}| - 2\underbrace{\sum_{\hat{Q}\in\mathcal{Q}} \mathbb{1}[R_j \in \mathcal{R}_{\hat{Q}}]}_{\mathbf{m}_{\mathcal{Q}}[j]} = |\mathcal{Q}| - 2\mathbf{m}_{\mathcal{Q}}[j]. \qquad \square
\end{aligned}$$

We now exploit Lemma 1 and 2 to show the desired condition about $\phi_f = \Delta_f(\mathcal{Q}, Q_2) - \Delta_f(\mathcal{Q}, Q_1) \geq 0$. We formally state this in the next lemma.

**Lemma 3.** *$\phi_f \geq 0$ if and only if $\mathbf{m}_{\mathcal{Q}}[j] \leq \frac{1}{2}|\mathcal{Q}|$.*

*Proof.* By Lemma 1 and 2 it results that:

$$\begin{aligned}
&\phi_f = \Delta_f(\mathcal{Q}, Q_2) - \Delta_f(\mathcal{Q}, Q_1) = \\
&\quad = \frac{1}{2}\left(\Delta_{cov}(\mathcal{Q}, Q_2) - \Delta_{cov}(\mathcal{Q}, Q_1)\right) + \\
&\qquad\qquad + \lambda\left(\Delta_{div}(\mathcal{Q}, Q_2) - \Delta_{div}(\mathcal{Q}, Q_1)\right) \\
&\quad = \frac{1}{2}\mathbb{1}[\mathbf{m}_{\mathcal{Q}}[j] = 0] + \lambda(|\mathcal{Q}| - 2\mathbf{m}_{\mathcal{Q}}[j]).
\end{aligned}$$

From the latter equality, it can be noted that, if $\mathbf{m}_{\mathcal{Q}}[j] = 0$, then $\phi_f = \frac{1}{2} + \lambda|\mathcal{Q}| > 0$. Otherwise, if $\mathbf{m}_{\mathcal{Q}}[j] > 0$, then $\phi_f = \lambda(|\mathcal{Q}| - 2\mathbf{m}_{\mathcal{Q}}[j])$, which is $\geq 0$ when $\mathbf{m}_{\mathcal{Q}}[j] \leq \frac{1}{2}|\mathcal{Q}|$. $\square$

Lemma 3 shows a condition about which results are "worth" to be captured by a specialization in order to achieve maximum marginal potential gain: ideally, the best specialization $Q^*$ should contain in its result set $\mathcal{R}_{Q^*}$ *all and only* those results $R_j$ whose corresponding multiplicity $\mathbf{m}_{\mathcal{Q}}[j]$ is no more than $\frac{1}{2}|\mathcal{Q}|$. To be precise, actually, the results $R_j$ such that $\mathbf{m}_{\mathcal{Q}}[j] = \frac{1}{2}|\mathcal{Q}|$ do not affect optimality: they can either be present in $\mathcal{R}_{Q^*}$ or not.

We exploit the above reasoning to derive our upper bound on the maximum marginal potential gain exhibited by a specialization in $\mathcal{T}_{Q'}$. We denote such an upper bound by $\overline{\Delta_f}(\mathcal{Q}, Q')$ and we formally state it in the next Theorem 3. Particularly, we express $\overline{\Delta_f}(\mathcal{Q}, Q')$ in terms of three $n$-dimensional binary vectors: $\mathbf{u}_{\mathcal{Q}}$ and $\mathbf{v}_{\mathcal{Q}}$, which keep track of the results in $\mathcal{R}_Q$ exhibiting null multiplicity and multiplicity no more than $\frac{1}{2}|\mathcal{Q}|$, respectively ($\mathbf{u}_{\mathcal{Q}}[i] = 1$ if and only if $\mathbf{m}_{\mathcal{Q}}[i] = 0$, and $\mathbf{v}_{\mathcal{Q}}[i] = 1$ if and only if $0 < \mathbf{m}_{\mathcal{Q}}[i] \leq \frac{1}{2}|\mathcal{Q}|$), and $\mathbf{x}_{Q^*}$, which keeps track of the results that the best specialization $Q^*$ should ideally capture and is defined as the Hadamard (i.e., element-wise) product between $\mathbf{v}_{\mathcal{Q}}$ and $\mathbf{x}_{Q'}$, i.e., $\mathbf{x}_{Q^*} = \mathbf{v}_{\mathcal{Q}} \circ \mathbf{x}_{Q'}$. The value of $\overline{\Delta_f}(\mathcal{Q}, Q')$ is as follows.

**Theorem 3.** *For a specialization $Q' \in \mathbb{S}_Q \setminus \mathcal{Q}$ it holds that*

$$\begin{aligned}
&\max_{Q''\in\mathcal{T}_{Q'}} \Delta_f(\mathcal{Q}, Q'') \leq \overline{\Delta_f}(\mathcal{Q}, Q') = \\
&\quad = \frac{1}{2}\mathbf{u}_{\mathcal{Q}} \cdot \mathbf{x}_{Q^*} + \lambda\left(\|\mathbf{m}_{\mathcal{Q}}\| + |\mathcal{Q}| \times \|\mathbf{x}_{Q^*}\| - 2\mathbf{m}_{\mathcal{Q}} \cdot \mathbf{x}_{Q^*}\right),
\end{aligned}$$

$$\mathbf{m}_{\mathcal{Q}} = \underbrace{\overset{R_1}{\boxed{0}}\overset{R_2}{\boxed{4}}\overset{R_3}{\boxed{2}}\overset{R_4}{\boxed{1}}\overset{R_5}{\boxed{0}}}_{<\frac{1}{2}|\mathcal{Q}|}\underbrace{\overset{R_6}{\boxed{5}}\overset{R_7}{\boxed{5}}}_{=\frac{1}{2}|\mathcal{Q}|}\underbrace{\overset{R_8}{\boxed{6}}\overset{R_9}{\boxed{6}}\overset{R_{10}}{\boxed{7}}}_{>\frac{1}{2}|\mathcal{Q}|} \quad (\|\mathbf{m}_{\mathcal{Q}}\| = 36)$$

$$\mathbf{u}_{\mathcal{Q}} = \boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}$$

$$\mathbf{v}_{\mathcal{Q}} = \boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{0}$$

$$\mathbf{x}_{Q'} = \underbrace{\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{1}}_{\mathcal{R}_{Q'}=\{R_1,R_2,R_3,R_6,R_8,R_9,R_{10}\}} \quad (\|\mathbf{x}_{Q'}\| = 7)$$

$$\underbrace{\mathbf{x}_{Q^*}}_{\mathbf{v}_{\mathcal{Q}}\circ\mathbf{x}_{Q'}} = \underbrace{\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{\mathcal{R}_{Q^*}=\{R_1,R_2,R_3,R_6\}} \quad (\|\mathbf{x}_{Q^*}\| = 4)$$

Figure 3: Illustration of the computation of the upper bound in Theorem 3.

*Proof.* By Lemma 3, we know how the result set of the best specialization $Q^*$ should be: the content of this best result set $\mathcal{R}_{Q^*}$ is expressed by the binary vector $\mathbf{x}_{Q^*} = \mathbf{v}_{\mathcal{Q}} \circ \mathbf{x}_{Q'}$. Based on this, Equation (4) can be rewritten as follows:

$$\begin{aligned}
&\overline{\Delta_f}(\mathcal{Q}, Q') = \Delta_f(\mathcal{Q}, Q^*) = \\
&\quad = \frac{1}{2}\Delta_{cov}(\mathcal{Q}, Q^*) + \lambda\Delta_{div}(\mathcal{Q}, Q^*) \\
&\quad = \frac{1}{2}\mathbf{u}_{\mathcal{Q}} \cdot \mathbf{x}_{Q^*} + \lambda\left(\|\mathbf{m}_{\mathcal{Q}}\| + |\mathcal{Q}| \times |\mathcal{R}_{Q^*}| - 2\mathbf{m}_{\mathcal{Q}} \cdot \mathbf{x}_{Q^*}\right) \\
&\quad = \frac{1}{2}\mathbf{u}_{\mathcal{Q}} \cdot \mathbf{x}_{Q^*} + \lambda\left(\|\mathbf{m}_{\mathcal{Q}}\| + |\mathcal{Q}| \times \|\mathbf{x}_{Q^*}\| - 2\mathbf{m}_{\mathcal{Q}} \cdot \mathbf{x}_{Q^*}\right),
\end{aligned}$$

where the third equality above derives from Theorem 2. $\square$

The computation of the upper bound $\overline{\Delta_f}(\mathcal{Q}, Q')$ is really fast: it takes $\mathcal{O}(|\mathcal{R}_{Q'}|)$ time, as all the terms of the expression in Theorem 3 either are constant or can be computed by considering the $|\mathcal{R}_{Q'}|$ non-zero entries of the vector $\mathbf{x}_{Q'}$.

**Example 3.** *Figure 3 shows an example of the computation of the upper bound $\overline{\Delta_f}$. We consider an input query $Q$ whose results are $\mathcal{R}_Q = \{R_1, \ldots, R_{10}\}$. We also assume that a set $\mathcal{Q}$ of $|\mathcal{Q}| = 10$ specializations have already been computed and that $\lambda = 0.5$. The integer vector $\mathbf{m}_{\mathcal{Q}}$ contains the multiplicity of the results in $\mathcal{Q}$, i.e., each entry $\mathbf{m}_{\mathcal{Q}}[i]$ contains the number of specializations in $\mathcal{Q}$ having $R_i$ among their results, while the binary vectors $\mathbf{u}_{\mathcal{Q}}$ and $\mathbf{v}_{\mathcal{Q}}$ keep track of the results in $\mathcal{R}_Q$ having null multiplicity and multiplicity $\leq \frac{1}{2}|\mathcal{Q}|$, respectively. For a given specialization $Q'$, the binary vectors $\mathbf{x}_{Q'}$ and $\mathbf{x}_{Q^*}$ denote the actual result set $R_{Q'}$ and the result set of the* ideal *specialization $Q^*$ that can be generated from $Q'$, respectively. According to Lemma 3, the result set of $Q^*$ is given by all and only the results of $Q'$ whose multiplicity is $\leq \frac{1}{2}|\mathcal{Q}|$, that is $\mathbf{x}_{Q^*} = \mathbf{v}_{\mathcal{Q}} \circ \mathbf{x}_{Q'}$.*
*The marginal gain of the coverage term $\Delta_{cov}(\mathcal{Q}, Q^*)$ is equal to the scalar product $\mathbf{u}_{\mathcal{Q}} \cdot \mathbf{x}_{Q^*}$; therefore, $\Delta_{cov}(\mathcal{Q}, Q^*) = 1$. According to Theorem 2, the marginal gain of the diversity term is $\Delta_{div}(\mathcal{Q}, Q^*) = \|\mathbf{m}_{\mathcal{Q}}\| + |\mathcal{Q}| \times |\mathbf{x}_{Q^*}| - 2\mathbf{m}_{\mathcal{Q}} \cdot \mathbf{x}_{Q^*} = 36 + 10 \times 4 - 2 \times 11 = 54$. As a result, the upper bound of $Q'$ is $\overline{\Delta_f}(\mathcal{Q}, Q') = \Delta_f(\mathcal{Q}, Q^*) = \frac{1}{2}\Delta_{cov}(\mathcal{Q}, Q^*) + \lambda\Delta_{div}(\mathcal{Q}, Q^*) = 27.5$, while the actual marginal potential gain of $Q'$ is $\Delta_f(\mathcal{Q}, Q') = \frac{1}{2}\Delta_{cov}(\mathcal{Q}, Q') + \lambda\Delta_{div}(\mathcal{Q}, Q') = \frac{1}{2}\mathbf{u}_{\mathcal{Q}} \cdot \mathbf{x}_{Q'} + \lambda(\|\mathbf{m}_{\mathcal{Q}}\| + |\mathcal{Q}| \times |\mathbf{x}_{Q'}| - 2\mathbf{m}_{\mathcal{Q}} \cdot \mathbf{x}_{Q'}) = 23.5$.*

**The Fast_MMPG algorithm** We exploit the findings discussed above in order to efficiently find a specialization exhibiting the maximum marginal potential gain. We assume our search space $\mathbb{S}_Q \setminus \mathcal{Q}$ organized as a tree $\mathcal{T}$, whose root corresponds to the input query $Q$, while the children of each node (specialization) $Q'$ correspond to all specializations generated by adding a single edge to $Q'$. According to this, a specialization can in principle have multiple fathers; this can however be avoided by borrowing standard mechanisms from frequent subgraph mining[4], hence we can safely assume that each specialization in $\mathcal{T}$ has a single father. As already anticipated, we also denote by $\mathcal{T}_{Q'}$ the subtree of $\mathcal{T}$ rooted at $Q'$.

The proposed Fast_MMPG algorithm, whose outline is reported as Algorithm 2, visits the various nodes $Q'$ of the tree $\mathcal{T}$ in non-increasing ordering of their upper-bound $\overline{\Delta_f}(\mathcal{Q}, Q')$, the rationale here is that larger upper bounds correspond to more promising subtrees. To this end, a priority queue $\mathbf{P}$ is used. At the beginning, $\mathbf{P}$ contains the children of the original query $Q$ (Line 2), and the algorithm processes the specializations in $\mathbf{P}$ until it becomes empty or the maximum upper bound therein does not exceed the best-so-far value $\Delta_f^*$ (Lines 3–13). For each specialization $Q'$ extracted from $\mathbf{P}$, the algorithm first computes the marginal potential gain $\Delta_f(\mathcal{Q}, Q')$ (according to Theorem 2), and uses it to possibly update $\Delta_f^*$ (Lines 5–8). $\Delta_f(\mathcal{Q}, Q')$ is also compared to the upper bound $\overline{\Delta_f}(\mathcal{Q}, Q')$ in order to early recognize whether it is worth to keep visiting the subtree $\mathcal{T}_{Q'}$ (Line 9): the visit goes ahead only if $\Delta_f(\mathcal{Q}, Q') < \overline{\Delta_f}(\mathcal{Q}, Q')$, otherwise $\mathcal{T}_{Q'}$ is entirely skipped. If the subtree $\mathcal{T}_{Q'}$ is not pruned, all children $Q''$ of $Q'$ are generated and, for each of them, the corresponding upper bound $\overline{\Delta_f}(\mathcal{Q}, Q'')$ is computed according to Theorem 3 (Line 10). All children $Q''$ having $\overline{\Delta_f}(\mathcal{Q}, Q'')$ no more than the best-so-far marginal potential gain $\Delta_f^*$ are discarded, while all others are added to $\mathbf{P}$ to be processed in a later iteration (Line 11).

The children of a (specialized) query $Q'$ (Lines 2 and 10) are generated according to the following strategy. For each result $R \in \mathcal{R}_{Q'}$, we keep track of all the subgraphs of $R$ that are isomorphic to $Q'$, and we expand each of those subgraphs by one step of BFS initiated in the vertices of that subgraph. A nice side effect of this strategy is that, for each children $Q''$ of $Q'$ yielded, we automatically have the corresponding result set $\mathcal{R}_{Q''}$ without running any further subgraph-search query on the database. As a result, the only subgraph-search query we need is the one to compute the results $\mathcal{R}_Q$ of the original query $Q$[5].

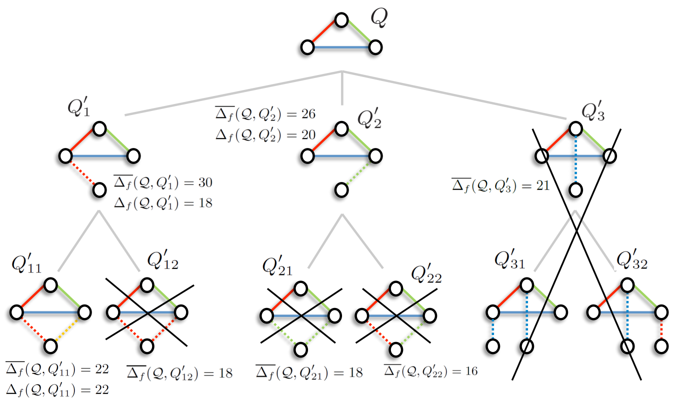
**Example 4.** *Figure 4 shows the execution of the Fast_MMPG algorithm on an example query $Q$ and the corresponding specialization tree. In the example we assume that a set of specializations $\mathcal{Q}$ have already been computed. The priority queue $\mathbf{P}$ is initialized with the children of $Q$, which, according to their upper bounds, follow the ordering $Q'_1 \rightarrow Q'_2 \rightarrow Q'_3$. In the first iteration, $Q'_1$ is extracted from $\mathbf{P}$ and the best-so-far value is set to $\Delta_f^* = \Delta_f(\mathcal{Q}, Q'_1) = 18$. Among the children of $Q'_1$, only $Q'_{11}$ is added to the priority queue as the upper bound of the other child $Q'_{12}$ is not $> \Delta_f^*$. The new ordering of the specializations in $\mathbf{P}$ is $Q'_2 \rightarrow Q'_{11} \rightarrow Q'_3$, thus $Q'_2$ is the next specialization to be processed. The value of $\Delta_f^*$ is updated and set to $\Delta_f^* = \Delta_f(\mathcal{Q}, Q'_2) = 20$, while all children of $Q'_2$ are pruned as their upper bound are less than $\Delta_f^*$. The next specialization processed is $Q'_{11}$, which leads to a new $\Delta_f^*$ equal to $\Delta_f(\mathcal{Q}, Q'_{11}) = 22$. After that, the algorithm terminates as the maximum upper-bound value in $\mathbf{P}$ (i.e., $\overline{\Delta_f}(\mathcal{Q}, Q'_3) = 21$) becomes smaller than $\Delta_f^*$: the subtree $\mathcal{T}_{Q'_3}$ is entirely skipped. The specialization ultimately outputted is $Q'_{11}$.*

The next theorem states the soundness of Fast_MMPG.

**Theorem 4.** *Algorithm 2 finds an optimal solution to the problem* $\arg\max_{Q' \in \mathbb{S}_Q \setminus \mathcal{Q}} \Delta_f(\mathcal{Q}, Q')$ *stated in Line 3 of Algorithm 1.*

[4]In our implementation we avoid to consider the same specialization multiple times by keeping track of the *DFS code* [32] of each specialization visited.

[5]We use the traditional subgraph-isomorphism algorithm by Ullmann [29] for this. Our work is however orthogonal to the method used for answering subgraph-search queries: for further speed-up, one can also resort to some indexing strategy [7, 8].


$Q$
$Q'_1$
$\overline{\Delta_f}(\mathcal{Q}, Q'_2) = 26$
$\Delta_f(\mathcal{Q}, Q'_2) = 20$
$Q'_2$
$Q'_3$
$\overline{\Delta_f}(\mathcal{Q}, Q'_1) = 30$
$\Delta_f(\mathcal{Q}, Q'_1) = 18$
$\overline{\Delta_f}(\mathcal{Q}, Q'_3) = 21$
$Q'_{11}$
$Q'_{12}$
$Q'_{21}$
$Q'_{22}$
$Q'_{31}$
$Q'_{32}$
$\overline{\Delta_f}(\mathcal{Q}, Q'_{11}) = 22$
$\Delta_f(\mathcal{Q}, Q'_{11}) = 22$
$\overline{\Delta_f}(\mathcal{Q}, Q'_{12}) = 18$
$\overline{\Delta_f}(\mathcal{Q}, Q'_{21}) = 18$
$\overline{\Delta_f}(\mathcal{Q}, Q'_{22}) = 16$
Figure 4: Illustration of the Fast_MMPG algorithm

**Algorithm 2** Fast_MMPG

```
Input: A graph database $\mathcal{D}$, a query $Q$, a set of specializations $\mathcal{Q}$
Output: A specialization $Q^* \in \mathbb{S}_Q \setminus \mathcal{Q}$ maximizing $\Delta_f(\mathcal{Q}, Q^*)$
 1: $\Delta_f^* \leftarrow -\infty$
 2: initialize $\mathbf{P}$ with {children of $Q$} $\setminus \mathcal{Q}$
 3: while $\mathbf{P}$ is not empty $\wedge \max(\mathbf{P}) > \Delta_f^*$ do
 4:     $Q' \leftarrow poll(\mathbf{P})$
 5:     if $\Delta_f(\mathcal{Q}, Q') > \Delta_f^*$ then
 6:         $Q^* \leftarrow Q'$
 7:         $\Delta_f^* \leftarrow \Delta_f(\mathcal{Q}, Q')$
 8:     if $|\mathcal{R}_{Q'}| > 0 \wedge \Delta_f(\mathcal{Q}, Q') < \overline{\Delta_f}(\mathcal{Q}, Q')$ then
 9:         $\mathcal{Q}' \leftarrow$ {children of $Q'$} $\setminus \mathcal{Q}$
10:         add $\{Q'' \in \mathcal{Q}' \mid \overline{\Delta_f}(\mathcal{Q}, Q'') > \Delta_f^*\}$ to $\mathbf{P}$
```

*Proof.* Let $\mathcal{T}' \subseteq \mathcal{T}$ denote the portion of $\mathcal{T}$ visited by the algorithm. According to Lines 5–7, the algorithm returns a specialization achieving the maximum marginal potential gain among all the ones in $\mathcal{T}'$. Then, the correctness of the algorithm follows from the fact that $\mathcal{T} \setminus \mathcal{T}'$ does not contain any specialization exhibiting a larger marginal potential gain. It is easy to see that the specializations $\mathcal{T} \setminus \mathcal{T}'$ discarded by the algorithm may only derive from the pruning rules exploited in Lines 8 and 10: without such rules the algorithm would instead consider every possible child of the current query $Q'$ and, overall, it would visit the entire tree $\mathcal{T}$. The correctness of such pruning rules is guaranteed by the correctness of the upper bound derived in Theorem 3. This completes the proof. □

## 4. EXPERIMENTS

In this section we empirically evaluate our approach by assessing its accuracy and efficiency on both real and synthetic graph databases, and comparing it to a number of baselines.

**Methods.** We recall that our method corresponds to the Greedy algorithm (Algorithm 1) equipped with the proposed Fast_MMPG (Algorithm 2) to perform the marginal potential gain maximization step. For the sake of brevity, in the following we denote our method by Fast_MMPG only.

We compare our Fast_MMPG to three baselines. The first baseline corresponds to the Greedy algorithm when equipped with a brute-force method to maximize the marginal potential gain, i.e., a method that visits the whole specialization search space without exploiting any finding devised in Section 3.3. Such a baseline, denoted by Greedy_BF, is mainly aimed at efficiency evaluation. The second baseline is the naïve method inspired by frequent subgraph mining and discussed in Section 3.1: this method selects the most frequent $k$ supergraphs of the input query as specializations.

|  |  | # vertices |  |  | # edges |  |  | # labels |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| *database* | *size* | *min* | *avg* | *max* | *min* | *avg* | *max* | *vert.* | *edges* | *dens.* |
| AIDS | 10k | 2 | 25 | 214 | 1 | 27 | 217 | 51 | 4 | 0.1 |
| Financial | 13k | 5 | 14 | 34 | 4 | 14 | 46 | 45 | 68 | 0.2 |
| Web | 18k | 2 | 8 | 48 | 1 | 7 | 53 | 639 | 8 | 0.3 |

Table 1: Characteristics of the real databases: number of graphs (size); *min*, *avg*, and *max* number of graph vertices/edges; number of vertex/edge labels; average density defined as $|E|/\binom{|V|}{2}$, where $|V|$ is the number of vertices and $|E|$ is the number of edges.

We refer to this baseline as k-freq. The third baseline is a method that discovers subgraph features from the graph database, indexes them, and, given a query $Q$, it returns the immediate (i.e., minimal) supergraphs of $Q$ among the features present in the index as specializations. In our implementation we extract discriminative yet frequent features according to the state-of-the-art methods defined in [33], while we use the well-known Lindex method [35] (with default parameter configuration) for indexing such features. We refer to this baseline as Lindex.

All methods are implemented in *Java* 1.7, and the experiments are performed on a $i686$ Intel Xeon E5-2440 2.40GHz, 125GB RAM machine over Linux kernel $v3.8.0$, which we limit to 30GB in all experiments. The graph database is loaded into main memory using the ParMol library [17]. We use a copy of Lindex kindly provided by the authors of [35] both for the Lindex baseline and to compute the DFS codes of all methods.

**Real databases.** We use real-world, publicly-available graph databases, whose main characteristics are shown in Table 1. AIDS[6] is a biological database extracted from the well-known *AIDS anti-viral screening* dataset[7] and traditionally used in the graph-mining/database literature [7, 16, 27, 31]. Vertices and edges represent atom and atom bonds, respectively. Financial[8] is a transaction workflow of loan-request processes submitted to a financial institute. Every vertex represents a subprocess while edges correspond to a resource exchanged between two processes. Web[9] is a workflow of web interactions between users and a recommender system for restaurants. A vertex is a restaurant and an edge is a browsing action.

**Synthetic databases.** We generate synthetic databases using the popular *GraphGen* graph generator [8][10]. We consider different database and graph sizes in order to better assess the scalability of our proposal (see Section 4.3).

**Query generation.** We generate random queries of various sizes. To ensure non-empty answers, we start from a vertex of a graph in the database (both sampled uniformly at random) and perform a DFS from that vertex until the desired size has been reached. DFS, compared to BFS, produces more diversified patterns. To have meaningful specializations, we discard queries having too few results (i.e., with number of results less than the number $k$ of output specializations). For each set of experiments and parameter configuration, we report results averaged over 10 random queries.

## 4.1 Ruling out Greedy_BF and Lindex

**Greedy_BF.** In Figure 5 we report the running time of our Fast_MMPG and the Greedy_BF baseline, using a synthetic database generated with GraphGen [8], where we vary the database size and set all other parameters to their default values.

[6] www.cs.ucsb.edu/~xyan/software.htm

[7] http://dtp.nci.nih.gov/docs/aids/aids_data.html.

[8] www.win.tue.nl/bpi/2012/challenge

[9] kdd.ics.uci.edu/databases/entree/entree.html

[10] http://www.cse.ust.hk/graphgen/.

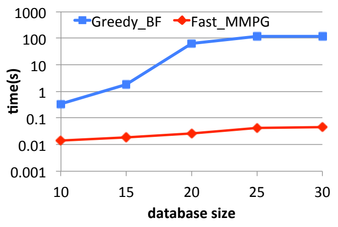
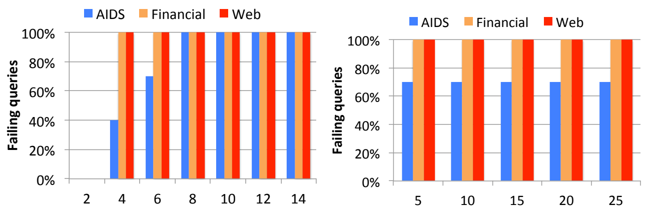


Figure 5: Running time of the proposed Fast_MMPG algorithm vs. the brute-force Greedy_BF baseline.



(a) query size ($|Q|$) (b) number of specializations (k)

Figure 6: Queries for which the Lindex baseline returns no specializations with varying (a) query size ($|Q|$), and (b) number of specializations ($k$).

It is easy to see that Greedy_BF is not suitable for any real-world settings: on even very small databases of 30 graphs it is four orders of magnitude slower than Fast_MMPG, and we could not run it on larger databases due to its excessive running time. For this reason, we avoid to report efficiency results for Greedy_BF in the remainder. In terms of accuracy, we recall that both Fast_MMPG and Greedy_BF employ the greedy scheme in Algorithm 1 and they both also optimally solve the sub-problem of maximizing the marginal potential gain (for the optimality of our Fast_MMPG see Theorem 4). As a consequence, they yield exactly the same results.

**Lindex.** All the indexes on graph databases proposed in the literature are expressively designed to split a query graph in features of smaller size, in order to speed-up subgraph-search queries. Thus, such indexes usually work well if the task is to find small-sized subgraphs of the query graph, while being less suited for the task of finding supergraphs. As a result, for queries of size exceeding the size of the largest feature in the index, the Lindex baseline would inevitably output an empty answer. As the features indexed are usually of very small size, this actually happens very often.

Indeed, Figure 6 shows the percentage of queries for which no specializations are found by Lindex. It can be observed that Lindex fails in finding specializations in most cases, e.g., 100% of the times for queries of size larger than 2 on the Financial and Web graphs. This confirms that Lindex is not really suitable for the problem of graph query reformulation we tackle in this work. For this purpose (and due to lack of space), we avoid reporting further details on Lindex in the remainder of this section.

## 4.2 Performance with varying parameters

Here we discuss the results of the evaluation on real datasets. We perform tests with varying the main parameters involved in the process: $(i)$ size (i.e., number of edges) $|Q|$ of the input query, $(ii)$ value of the regularization factor $\lambda$ (Equation (3)), and $(iii)$ number of output specializations $k$. We vary these parameters in the following ranges: $|Q| \in [2, 14]$, $\lambda \in [0, 0.5]$, $k \in [5, 25]$. While varying one parameter, we keep the other two fixed to (around) their median values, i.e., we set $|Q| = 6$, $\lambda = 0.3$, and $k = 10$.

We report running times (Figures 7) and quality (in terms of objective-function value, Tables 2) of the proposed Fast_MMPG algorithm and the baseline k-freq. Due to space limits, for running times, we report results for two databases only (AIDS and Web). We however observe similar results on the dataset omitted.

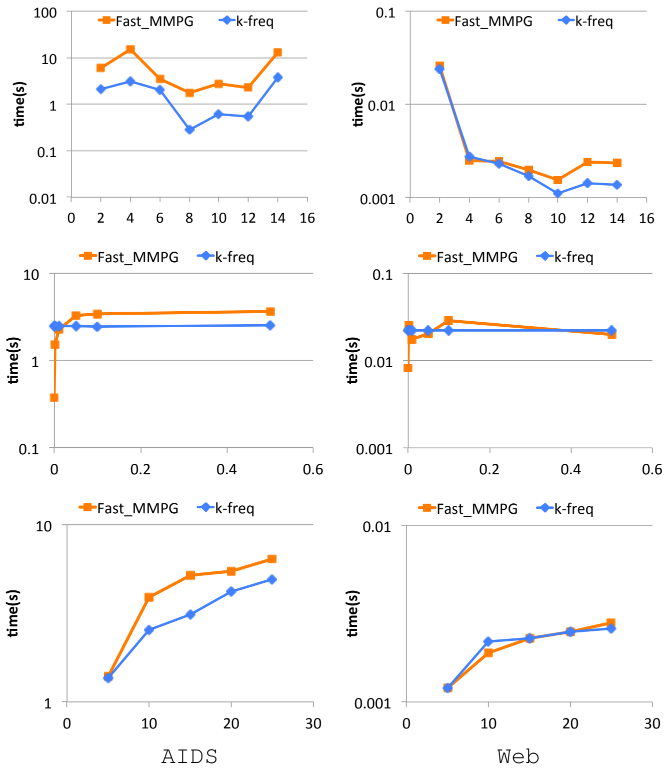


Figure 7: Running times of the proposed Fast_MMPG algorithm and the k-freq baseline on the real datasets with varying query size $|Q|$ (first row), varying regularization factor $\lambda$ (second row), and varying number of specializations $k$ (third row), on datasets AIDS (left) and Web (right).

**Query size ($|Q|$).** Figure 7 (first row) reports the efficiency of Fast_MMPG varying the query size. It can be observed that our algorithm can easily handle all real databases, with running times ranging from a few milliseconds (Web) to a few seconds (AIDS). The difference in time observed through the various databases is mainly due to the size of the graphs therein: the graphs in AIDS are indeed generally larger than the other two databases. While in most cases times decrease as query size increases (conveying smaller result sets), this also depends on query type: a more complex query can have smaller support set than a query with larger size but simpler structure. The latter observation is supported by AIDS dataset. Finally, the running times of our method are comparable to the baseline k-freq: although this baseline employs a much simpler scheme than our Fast_MMPG, and, as such, is expected to run faster, we instead do not observe this in practice. The reason is that, even though k-freq may visit less specializations than Fast_MMPG, the ones visited by k-freq are top-frequent specializations for which a quite large number of subgraph isomorphisms need to be performed. On the contrary, our Fast_MMPG algorithm does not necessarily visit top-frequent specializations as diversity is also considered.

The quality results are shown in Table 2 (a), where we report the objective-function values of Fast_MMPG and k-freq, along with the percentage gain achieved by Fast_MMPG over k-freq (last row). In general, Fast_MMPG evidently outperforms k-freq, with gain ranging from up to 40% on Web to 52% on AIDS (17% and 34% on average). The reason of the larger improvement exhibited on AIDS is likely due to the type of graphs contained in the database: the graphs in AIDS are larger and more diversified than the workflow graphs in Financial and Web, which are instead more similar to one another and thus capture less results.

Finally, we generally observe that the objective-function value decreases as the query size increases. This is expected since the objective-function value is directly proportional to the number of query results, and, clearly, the larger the query, the fewer the results.

**Regularization factor ($\lambda$).** The running time of Fast_MMPG with varying $\lambda$, shown in Figure 7 (second row), follows a fluctuating trend for smaller values of $\lambda$, while converging for $\lambda > 0.1$. In all cases, the running time is again very small: it ranges from 3.2ms (Web) to 3.6s (AIDS). Again, our Fast_MMPG is really close to the baseline k-freq.

As far as accuracy reported in Table 2 (b), as expected, larger $\lambda$ values lead to larger improvements by our Fast_MMPG over the baseline. The motivation is that a larger value of $\lambda$ steers the objective function towards the maximization of diversity. Since coverage is implicitly captured, as a side effect, by the top-$k$ frequent specializations output by the k-freq baseline, the latter is unsurprisingly closer to Fast_MMPG for values of $\lambda$ close to zero. However, we restate that a value of $\lambda$ too small is not generally a good choice, because it would practically correspond to ignore diversity, which instead plays a key role, as extensively discussed above.

**Number of specializations ($k$).** The efficiency of Fast_MMPG with varying the number of output specializations is reported in Figure 7 (third row). Clearly, the running time is increasing as $k$ increases. However, the trends on all datasets are low exponential in $k$, which attests the scalability of our method with respect to the output size. Again, Fast_MMPG is really close to k-freq.

The quality results reported in Table 2 (c) show that, while being better than the baseline in all settings, Fast_MMPG exhibits (slightly) decreasing gain as $k$ increases. This is reasonable as the greedy scheme of Fast_MMPG implies that the maximum marginal-gain value gets progressively smaller as more specializations are added to the solution.

## 4.3 Scalability

We test the scalability of our Fast_MMPG algorithm (and the k-freq baseline) on synthetic databases. Particularly, we analyze the efficiency performance from two main perspectives, that is varying $(i)$ the database size (i.e., number of graphs), and $(ii)$ the average size (expressed as number of edges) of the graphs in the database. To this end, we use *GraphGen* [8] to generate databases of sizes in the range [25K, 250K] and average graph sizes in [10, 50]. All other parameters are set to the default values suggested by the generator. The main parameter values are: number of vertex labels (20), number of edge labels (20), average graph density (0.3), consistently observed on our real datasets.

The results of this evaluation are reported in Figure 8. The main message of the left figure is that Fast_MMPG can handle query specialization in a database of 250K graphs in a few seconds. More specifically, the trend is linear in the database size: the running time ranges from almost 0.1s (25K graphs) to 2.2s (250K graphs). This attests full scalability of Fast_MMPG on very large database sizes.

The scalability of Fast_MMPG is also confirmed by the experiment with varying the graph size: our algorithm takes less than one second for handling databases with average graph size of 50. We remark that this value is far beyond the graph size that is commonly encountered in real-world scenarios (in our three real-world databases, the average graph size is 4, 14 and 27 as reported in Table 1). However, here Fast_MMPG (and k-freq too) is more sensitive to changes than the previous experiment: this is expected since, like most methods on querying graph databases, our technique relies on subgraph-isomorphism, whose running time is notoriously

Table 2: Quality (in terms of the objective function $f$ defined in Equation (3)) of the proposed Fast_MMPG algorithm and the k-freq baseline on real databases with (a) varying the query size $|Q|$, (b) varying regularization factor $\lambda$, and (c) varying number of specializations $k$.

(a)

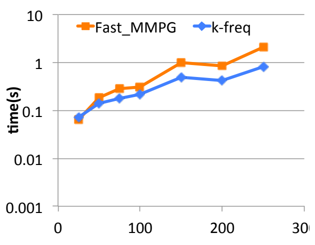
| | AIDS | | | | | | | Financial | | | | | | | Web | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|Q\|$ | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 2 | 4 | 6 | 8 | 10 | 12 | 14 |
| Fast_MMPG | 30 086 | 14 121 | 5 917 | 1 478 | 1 803 | 1 624 | 1 087 | 34 135 | 6 437 | 1 555 | 1 487 | 1 147 | 686 | 389 | 675 | 103 | 73 | 65 | 23 | 21 | 12 |
| k-freq | 22 818 | 11 435 | 4 336 | 795 | 1 317 | 786 | 621 | 33 865 | 5 800 | 1 224 | 830 | 956 | 581 | 274 | 557 | 88 | 43 | 63 | 19 | 19 | 10 |
| *gain* (%) | 24 | 19 | 27 | 46 | 27 | 52 | 43 | 1 | 10 | 21 | 44 | 17 | 15 | 30 | 18 | 15 | 40 | 3 | 18 | 8 | 15 |

(b)

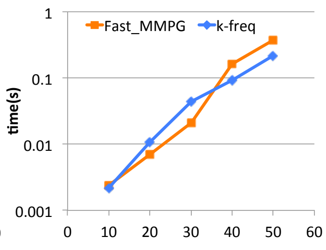
| | AIDS | | | | | Financial | | | | | Web | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\lambda$ | 0 | 0.01 | 0.05 | 0.1 | 0.5 | 0 | 0.01 | 0.05 | 0.1 | 0.5 | 0 | 0.01 | 0.05 | 0.1 | 0.5 |
| Fast_MMPG | 433 | 613 | 1 345 | 2 260 | 9 566 | 201 | 244 | 422 | 649 | 2 461 | 4 | 5.2 | 9.3 | 14.3 | 54.7 |
| k-freq | 409 | 540 | 1 063 | 1718 | 6 954 | 188 | 222 | 360 | 533 | 1 914 | 4 | 5 | 8.4 | 12.7 | 46.6 |
| *gain* (%) | 6 | 12 | 21 | 24 | 27 | 7 | 9 | 15 | 18 | 22 | 0 | 3 | 9 | 11 | 15 |

(c)

| | AIDS | | | | | Financial | | | | | Web | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | 5 | 10 | 15 | 20 | 25 | 5 | 10 | 15 | 20 | 25 | 5 | 10 | 15 | 20 | 25 |
| Fast_MMPG | 1 791 | 5 917 | 12 462 | 21 235 | 32 029 | 629 | 1 555 | 2 904 | 4 630 | 6 645 | 12 | 35 | 68 | 99 | 134 |
| k-freq | 1 373 | 4 336 | 9 709 | 17 061 | 25 667 | 535 | 1 224 | 2 241 | 3 410 | 5 400 | 7 | 30 | 62 | 92 | 123 |
| *gain* (%) | 23 | 27 | 22 | 20 | 20 | 15 | 21 | 23 | 26 | 19 | 41 | 14 | 8 | 7 | 8 |

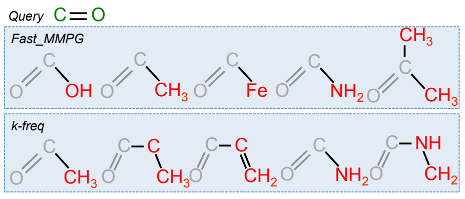


(a) database size (in thousands)

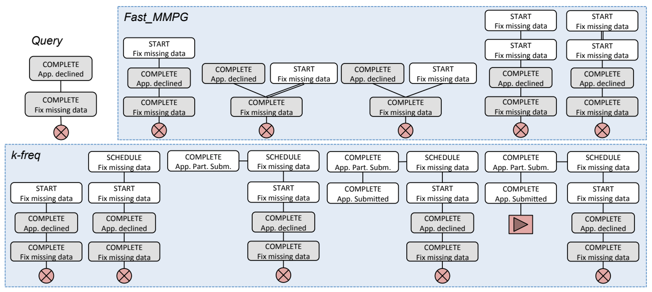


(b) average graph size

Figure 8: Scalability of the proposed Fast_MMPG algorithm on synthetic databases: (a) running time vs. database size (avg graph size set to 30); (b) running time vs. average graph size (database size set to 10K).



(a) AIDS



(b) Financial

Figure 9: Examples queries on the AIDS and Financial datasets and the specializations returned by Fast_MMPG and k-freq ($k = 5$).

more affected by the size of the graph than the number of graphs, as larger graphs typically lead to more isomorphisms.

## 4.4 Qualitative evaluation

We provide here some visual examples of the results produced by our Fast_MMPG algorithm and the k-freq baseline. Figure 9a shows a query issued to the AIDS dataset. The query corresponds to a well-known chemical compound, i.e., *formaldehyde*. The specializations output by our Fast_MMPG correspond to chemical compounds that span the search space horizontally, thus showing non-overlapping molecules, among which one can recognize two very common compounds of formaldehyde, namely *formamide* (fourth specialization) and *acetone* (fifth specialization). On the other hand, the k-freq specializations are very close to each other (some of them are even subgraphs of other specializations, e.g., first and fourth specialization): such specializations are therefore much less informative than the ones found by our Fast_MMPG.

The second example on the Financial database (Figure 9b) shows that methods based only on frequency (like k-freq) are not suitable for capturing the various (diverse) alternatives from the results of a query. Indeed, once a frequent structure (specialization) has been encountered, all other specializations are most likely generated starting from there, meaning that every subsequent specialization is a supergraph of the previous one. This is exactly what happens with the specializations output by k-freq for the example in Figure 9b. Instead, our Fast_MMPG returns specializations whose common structure mostly corresponds to the query itself.

## 5. RELATED WORK

**Querying graph databases.** Efficient query answering (without reformulation) in graph databases has been long studied. The types of query considered include *full-graph search* [4], whose goal is to retrieve all graphs that are isomorphic to the input query; *subgraph/supergraph search* [7, 8, 33], which finds all graphs that are supergraph/subgraph of the input query; *generalized subgraph search* [16], which is a generalization of subgraph search where exact edge matching is replaced with the notion of path matching constrained by a path length; *similarity search* [27], which aims at returning all graphs that are similar enough to a given query.

This work treats subgraph-search queries and studies for the first time the problem of finding specializations for this type of query.

**Query reformulation.** Query reformulation (also known as *query modification* or *query rewriting*) is a classic problem in data mining, databases, and information retrieval, whose main goal is to provide the user with a set of alternative queries that may better capture her search intent. Depending on the specific goal, query reformulation is also referred to as *query refinement* [18], *query relaxation* [21], *query recommendation/suggestion* [3], or *query expansion* [3].

Query reformulation has been studied for web-search queries [9], as well as structured queries [18, 21] and keyword queries [34] on structured databases. As far as web search, query reformulation has also been used as a tool for a different problem, i.e., *result diversification*, that is the problem of selecting a subset of query results that are diverse from each other [10]. Diversity (along with coverage) is also part of our objective function, but with a different purpose: we want to find other patterns that identify diverse subsets of the results of the original query rather than simply selecting a number of diverse query results.

However, none of those contributions deal with graph queries: to the best of our knowledge, this is the first work focusing on the problem of query reformulation in graph databases.

**Graph pattern mining.** Finding patterns in graph databases is a well-studied problem. Research in this field has mainly focused on *frequent subgraph mining* [14, 22, 32], which aims at finding all structures (usually subgraphs, but also trees or paths) that occur frequently in the graphs of the database, and *optimal graph pattern mining*, that is the problem of finding substructure(s) that maximize a given quality function [20, 31].

A naïve approach to graph query reformulation might be to resort to existing frequent-subgraph-mining methods: find the supergraphs of the query that appear frequently in the database and just interpret them as specializations (see Section 3.1). However, we experimentally show the superiority of our proposal over this frequent-subgraph-based approach in Section 4.

**Result-set reduction in graph databases.** The problem of reducing the answers to a query issued to a graph database has also received some attention. Existing solutions rely on either clustering the graphs in the result set [11, 15] (not to be confused with the problem of clustering the vertices of a *single* graph) or returning top-$k$ representative results [24].

That problem follows the general line of overcoming information overload, but is only marginally related to the problem tackled in this work. Rather than aiming at reducing the result set, our query-reformulation problem indeed asks for something more, i.e., we want to output a set of reformulated (i.e., more specific) queries that can help the user better comprehend the results and refine her search. Existing approaches to result-set reduction cannot instead output query reformulations, as our problem requires.

## 6. CONCLUSIONS

This paper studies the problem of query reformulation in graph databases. Given a graph database and a query graph, the goal is to produce a set of *specializations*, i.e., queries that are more specific than the original query and, as such, capture a subset of its results and represent a useful guidance for refining the user search.

We formalize our problem by asking for a set of $k$ specializations of the input query so as to maximize both coverage and diversity. We characterize the hardness of the problem and show that it allows a greedy algorithm with provable quality guarantees. We also devise a principled strategy to efficiently solve the most critical step of the greedy algorithm, i.e., finding the specialization maximizing the marginal potential gain. Experiments on both real-world and synthetic graph databases attest that our method runs in real-time, scales well on large databases, and provides high-quality results.

## 7. REFERENCES

[1] C. C. Aggarwal and H. Wang. *Managing and Mining Graph Data*. Springer, 2010.

[2] M. K. Anand, S. Bowers, and B. LudÃd'scher. Techniques for efficiently querying scientific workflow provenance graphs. In *EDBT*, pages 287–298, 2010.

[3] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search, Second edition*. Pearson Education Ltd., 2011.

[4] S. Berretti, A. D. Bimbo, and E. Vicario. Efficient matching and indexing of graph models in content-based retrieval. *TPAMI*, 23(10):1089–1105, 2001.

[5] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna. Query reformulation mining: models, patterns, and applications. *Inf. Retr.*, 14(3), 2011.

[6] A. Borodin, H. C. Lee, and Y. Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *PODS*, pages 155–166, 2012.

[7] J. Cheng, Y. Ke, A. W.-C. Fu, and J. X. Yu. Fast graph query processing with a low-cost index. *VLDBJ*, 20(4):521–539, 2011.

[8] J. Cheng, Y. Ke, W. Ng, and A. Lu. Fg-index: Towards verification free query processing on graph databases. In *SIGMOD*, 2007.

[9] V. Dang and B. W. Croft. Query reformulation using anchor text. In *WSDM*, pages 41–50, 2010.

[10] M. Drosou and E. Pitoura. Disc diversity: Result diversification based on dissimilarity and coverage. *PVLDB*, 6(1):13–24, 2012.

[11] M. Ferrer, E. Valveny, F. Serratosa, I. Bardají, and H. Bunke. Graph-based k-means clustering: A comparison of the set median versus the generalized median graph. In X. Jiang and N. Petkov, editors, *CAIP*, volume 5702, pages 342–350. 2009.

[12] D. S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., 1997.

[13] J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A. Tropsha. Mining protein family specific residue packing patterns from protein structure graphs. In *RECOMB*, pages 308–315, 2004.

[14] J. Huan, W. Wang, J. Prins, and J. Yang. SPIN: mining maximal frequent subgraphs from graph databases. In *KDD*, 2004.

[15] X. Huang, H. Cheng, J. Yang, J. X. Yu, H. Fei, and J. Huan. Semi-supervised clustering of graph objects: A subgraph mining approach. In *DASFAA*, pages 197–212, 2012.

[16] W. Lin, X. Xiao, J. Cheng, and S. S. Bhowmick. Efficient algorithms for generalized subgraph query processing. In *CIKM*, 2012.

[17] T. Meinl, M. Wörlein, O. Urzova, I. Fischer, and M. Philippsen. The parmol package for frequent subgraph mining. *ECEEASST*, 1, 2007.

[18] C. Mishra and N. Koudas. Interactive query refinement. In *EDBT*, pages 862–873, 2009.

[19] P. Missier, N. W. Paton, and K. Belhajjame. Fine-grained and efficient lineage querying of collection-based workflow provenance. In *EDBT*, pages 299–310, 2010.

[20] S. Morishita and J. Sese. Transversing itemset lattices with statistical metric pruning. In *SIGMOD*, pages 226–236, 2000.

[21] D. Mottin, A. Marascu, S. B. Roy, G. Das, T. Palpanas, and Y. Velegrakis. A probabilistic optimization framework for the empty-answer problem. *PVLDB*, 6(14):1762–1773, 2013.

[22] S. Nijssen and J. N. Kok. A quickstart in frequent structure mining can make a difference. In *KDD*, pages 647–652, 2004.

[23] S. A. Rahman et al. Small molecule subgraph detector (smsd) toolkit. *J. Cheminformatics*, 1:1–12, 2009.

[24] S. Ranu, M. Hoang, and A. Singh. Answering top-k representative queries on graph databases. In *SIGMOD*, pages 1163–1174, 2014.

[25] S. Y. Rieh and H. Xie. Analysis of multiple query reformulations on the web: the interactive information retrieval context. *Inf. Process. Manage.*, 42(3), 2006.

[26] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and re-using workflows with vstrails. In *SIGMOD*, 2008.

[27] H. Shang, X. Lin, Y. Zhang, J. X. Yu, and W. Wang. Connected substructure similarity search. In *SIGMOD*, pages 903–914, 2010.

[28] A. Shokoufandeh and S. J. Dickinson. Graph-theoretical methods in computer vision. In *Theoretical Aspects of Computer Science*, 2000.

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

[30] L. G. Valiant. The complexity of computing the permanent. *TCS*, 8(2):189–201, 1979.

[31] X. Yan, H. Cheng, J. Han, and P. S. Yu. Mining significant graph patterns by leap search. In *SIGMOD*, pages 433–444, 2008.

[32] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.

[33] X. Yan, P. S. Yu, and J. Han. Graph indexing based on discriminative frequent structure analysis. *TODS*, 30(4):960–993, 2005.

[34] J. Yao, B. Cui, L. Hua, and Y. Huang. Keyword query reformulation on structured data. In *ICDE*, 2012.

[35] D. Yuan and P. Mitra. Lindex: a lattice-based index for graph databases. *VLDBJ*, 22(2):229–252, 2013.