Subgraph Search in Large Graphs with Result Diversification

Huiwen Yu* Dayu Yuan*

Abstract

The problem of subgraph search in large graphs has wide applications in both nature and social science. The subgraph search results are typically ordered based on graph similarity score. In this paper, we study the problem of ranking the subgraph search results based on diversification. We design two ranking measures based on both similarity and diversity, and formalize the problem as an optimization problem. We give two efficient algorithms, the greedy selection and the swapping selection with provable performance guarantee. We also propose a novel local search heuristic with at least 100 times speedup and a similar solution quality. We demonstrate the efficiency and effectiveness of our approaches via extensive experiments.

Keyword: subgraph search; result diversification; submodular; local search.

1 Introduction

Graphs are useful models for representing complex structures in nature and human society. The subgraph search problem, i.e. finding a subgraph mapping in a target graph which is similar to the query graph, is an important primitive for searching and extracting information in the graph database. The problem has applications in cheminformatics, bioinformatics [27] where one wants to find a specific component from a compound, image patch search [7], RDF query in semantic web [6], and social network such as the facebook graph search [1].

We ask the question on how to efficiently find the top k similar subgraph mappings and effectively *rank* the results to improve the search quality. It has been shown that ranking search results based on not only similarity but also *diversity* reduces information redundancy and provides the user with broader options [5]. Consider the expert search problem [19], where the user specifies a set of skills and desired relationship between the experts, and the question is to find groups of experts satisfying the requirements. For example, the user might want to find three people who know each other and have strong background in circuit design, embedded system and robotics respectively, so that they can form a team for robots design competition. Suppose we search for a group of three people from a social graph. We represent the query as a triangle with one label on each node representing a skill of the corresponding person. A ranking scheme based on similarity returns all subgraph mappings satisfying the requirement. The top results might each contain people from a same institute, of the same gender and major. However, it is usually good to have candidates with different backgrounds and other relevant skills. Diversification plays an important role in building such type of ranking systems.

Diversifying search result is a hard problem in general. First, it is a challenging task to design effective diversity measure. A good measure should consider the diversity factor by maintaining the similarity guarantee. Second, the problem is usually NP-hard and thus computationally difficult. In this paper, we consider the problem of ranking subgraph search results based on diversification. We design two diversity measures. One is based on the content similarity and the other is based on the information coverage. We formalize the ranking problem as an optimization problem and propose two ranking algorithms with provable performance guarantee. For any graph similarity measure and subgraph search algorithm, our ranking algorithms can be used as a final step to order the search results.

On the other hand, in the general settings of the query diversification problem, a typical assumption is that finding all similar results is not a hard problem. In the case of the subgraph search problem, finding similar subgraph mappings measured by e.g. edit distance [14], or finding exactly-matched mappings (the subgraph isomorphism problem) are both NP-hard. Hence, it is desirable to eliminate generating all similar mappings before returning the top results with diversification. In this paper, we design a heuristic solution to achieve this goal.

To summarize, our contributions in this paper are as follows. (1) We design two diversity measures based on content and coverage respectively. We formalize the problem of ranking subgraph search results based on diversification as an optimization problem. (2) We present two efficient polynomial-time algorithms with provable performance guarantee. These two algorithms are based on greedy selection and swapping selection respectively. The swapping selection consumes less space and is faster than the greedy selection, while the solution quality of the greedy selection is generally better. (3) We give a novel heuristic based on local search which combines the subgraph search step and the ranking step. This heuristic performs at least 100 times faster and yet achieves similar solution quality compared to the greedy selection and the swapping selection. Moreover,

*Department of Computer Science and Engineering, Pennsylvania State University, USA. hwyu, duy113@cse.psu.edu. The first author is supported in part by NSF Grant CCF-0964655 and CCF-1320814.

[1] https://www.facebook.com/about/graphsearch

we conduct extensive experiments on both real-world and synthetic data which demonstrate the efficiency and the effectiveness of our methods.

This paper is organized as follows. We give basic definitions and notations in Section 2. We formally define the problem and show the "diminishing return" property of the new diversity measures in Section 3. We present three algorithms to the ranking problem in Section 4. Experimental results are presented in Section 5, and followed by conclusion in Section 6.

1.1 Related works We review previous works on subgraph search algorithms and query diversification. The exact subgraph matching problem or the subgraph isomorphism problem is NP-complete. Lee et al. give detailed comparisons of several well-known exact subgraph matching algorithms [20]. Algorithms for inexact subgraph matching problem are heuristic-based [17, 18, 22, 27, 28, 39]. For example, TALE [28] by Tian et al. first locates mappings of a subset of "important" nodes and then expands them to the whole graph. For specific applications, there are matching algorithms for RDF query [6], image patch search [7], graph nodes with multiple labels [34], labels with ontology structure [33], querying probabilistic database [38]. These algorithms rank the subgraph search results by similarity score.

Diversifying the search results has been addressed recently to improve the search quality (see survey [10]). The widely-used diversity measures are based on content [36], novelty [1] or coverage [21]. There are two commonly-used heuristics for the problems of diversifying the search results. One is based on the swapping heuristic and the other is based on the greedy selection [36]. Tong et al. first formalize the problem in an optimization point of view [29]. There are more efficient methods [4, 31], and new measures proposed recently [8, 9, 11] to further improve the search quality. On the other hand, Gollapudi et al. show that it is impossible to satisfy many diversity requirements simultaneously [15]. Angel et al. [2] and Fraternali et al. [13] study data access methods to efficiently find the top diversified results. Borodin et al. [3] consider the max-sum diversification under matroid constraint. The idea of diversification has also been applied to other application domains, such as query reformulation [35], skyline query [30], feature selection in the graph classification [41], question recommendation [16], news display [26], clique enumeration [32]. Fan el al. [12] also study the problem of diversifying the subgraph search results but with ranking functions which do not have the "diminishing return" property (see Section 3).

2 Preliminaries

We represent a graph G by a triple (V, E, ℓ), where V is the set of nodes, E is the set of edges, and ℓ is a label function. In this paper, we consider the case where only nodes have labels. Our technique can be extended to the case where edges also have labels. Given a collection of labels L, ℓ is a function from the node set V to a subset of L. Given a query graph $Q = (V_q, E_q, \ell)$ and a target graph $G = (V, E, \ell)$, the subgraph isomorphism problem, or the exact subgraph matching problem is to find a one-to-one mapping $\phi : V_q \to \phi(V_q) \subseteq V$ such that for any node $v \in V$, $\ell(v) = \ell(\phi(v))$, and for any two nodes $(u, v) \in E_q$, $(\phi(v), \phi(u)) \in E$. It is usually more useful to consider inexact subgraph matching in most applications. There are different graph similarity measures for inexact matching problem, e.g. the graph edit distance [14], the maximum common subgraph [40]. Our ranking algorithms take any graph similarity measure. We assume the similarity score is normalized within $[0, 1]$. Let γ be the similarity threshold constant. We consider mappings with a similarity score at least γ as similar mappings. Further, we denote the shortest path distance of nodes v and u in the graph G by $dist_G(v, u)$. Let $dist_G(v, S)$ be the distance of a node v to a set $S \subseteq V$, which is defined as $\min_{u \in S} dist_G(v, u)$. Let $N_G(v)$ be the neighbors of v and $N^i_G(v)$ be the set of nodes within distance i to v. We might omit G in these notations if it is clear from the context.

To introduce the diversity measures on graphs, we first define a similarity measure of two subgraph mappings H_1, H_2 in G by their *label coverage*. We denote the *label coverage vector* of a subgraph H in G by $\vec{l}_H$. The ith entry of the label coverage vector represents the coverage weight of label i "covered" by H. Let n_L be the universal length of a label coverage vector, i.e. $n_L = |L|$. If H does not cover a label, the corresponding coverage weight is 0.

DEFINITION 2.1. (LABEL SIMILARITY) *The label similarity of two subgraphs H_1 and H_2 in the target graph G is defined as*

$$sim_L(H_1, H_2) = \frac{\sum_{i=1}^{n_L} \min\{\vec{l}_{H_1}(i), \vec{l}_{H_2}(i)\}}{\sum_{i=1}^{n_L} \max\{\vec{l}_{H_1}(i), \vec{l}_{H_2}(i)\}}. \tag{2.1}$$

Notice that this similarity of label coverage is used to define the diversity of the subgraph search results. We use the graph similarity measure (Example 2.1) to compare the topology along with the vertex labels of two subgraphs. To define the coverage weight of a label, we assume that this weight decreases with respect to the "strength" of the feature/label. We mainly consider one definition in this paper as follows. Given parameters $\alpha \in (0, 1)$ and an integer h, we say a label l is covered by a set S if there is a node v within distance h from S and $l \in \ell(v)$. The weight of this label l is defined to be $\alpha^{dist(v,S)}$. We can interpret this definition using the idea of information propagation. The influence of a node u to v decreases with respect to the distance of u to v by a factor of α. If the diversity of a set H is determined by the distinct features/labels it can access, we define the weight

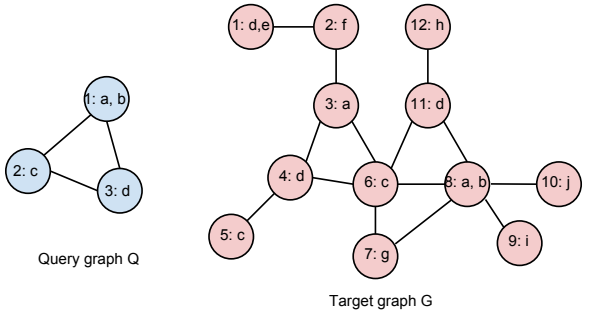

Figure 1: Illustrative figure for Example 2.1. The description in a node: the number is the index of the node and the letters are node labels.

of a label l in $\vec{l}_H$ to be $\alpha^{dist(l,S)}$, where $dist(l,S)$ is the minimum distance of any node v to S with v containing the label l if $dist(l,S) \leq h$. Otherwise, we consider the weight of l to be 0. Another way to define the coverage weight of a label is $\sum_{l \in \vec{l}_v : dist(v,S) \leq h} \alpha^{dist(v,S)}$. In this case, repetitions of one feature increase the diversity. In a computational point of view, those two definitions have similar complexity. We consider only the first definition in the paper.

DEFINITION 2.2. (DIVERSITY BY LABEL COVERAGE) *Given parameters $\alpha \in (0,1)$, $h \geq 0$. For any subgraph H of G, the label coverage diversity of H with respect to G is defined as,*

$$div_G(H) = \sum_{l \in L : dist(l,H) \leq h} \alpha^{dist(l,H)}. \tag{2.2}$$

EXAMPLE 2.1. *In this example, we use a simple graph (Figure 1) to illustrate all definitions introduced in this section. Let $\alpha = 0.5$. We represent a subgraph by its vertex set. Consider the subgraph mappings $M_1 = (8,6,11)$, $M_2 = (3,6,4)$ and $M_3 = (8,6,4)$ and $M_4 = (3,5,4)$ of the query graph Q in the target graph G. We compute $\vec{l}_{M_2}$. We omit labels of weight 0. If $h = 0$, $\vec{l}_{M_2} = (a,1; c,1; d,1)$ with diversity 3. If $h = 1$, $\vec{l}_{M_2} = (a,1; b,0.5; c,1; d,1; f,0.5; g,0.5)$ with diversity 4.5. If $h = 2$, $\vec{l}_{M_2} = (a,1; b,0.5; c,1; d,1; e,0.25; f,0.5; g,0.5; h,0.25; i,0.25; j,0.25)$ with diversity 5.5. Consider the case of $h = 1$ again. $\vec{l}_{M_1} = (a,1; b,1; c,1; d,1; g,0.5; h,0.5; i,0.5; \ j,0.5)$. Then $sim_L(M_1,M_2) = 4/6.5 = 0.615$.*

3 Problem fomulation

We are ready to define the measures for ranking subgraph search results based on diversification. Given a target graph G and a query graph Q, suppose a subgraph search algorithm returns k subgraph mappings $S_1, ..., S_k$ in G. Let s_i be the similarity score of S_i to Q. We define two diversity measures which incorporate both the similarity and the diversity. The first measure is based on content similarity (diversity measure f_1) and the second one is based on information/label coverage (diversity measure f_2).

1. $f_1 = 2\sum_{i=1}^{k} s_i - \lambda_1 \sum_{i=1}^{k} \sum_{j \neq i} sim_L(S_j, S_i)$, $\lambda_1 \leq \gamma / \max_i\{\sum_{j \neq i} sim_L(S_j, S_i)\}$.
2. $f_2 = \sum_{i=1}^{k} s_i + \lambda_2 \sum_{i=1}^{k} s_i \cdot div_G(S_i)$, $\alpha \leq \gamma$.

The content-based or the coverage-based measure has been considered as two important measures for diversity [10]. The novelty here is the extension of the definitions to the subgraph search problem. For f_1, we can think of $1 - sim_L(\cdot,\cdot)$ as the distance metric commonly used in the content-based definitions. For f_2, we add a factor s_i to the diversity score to prevent selecting mappings of low similarity score but with high diversity. However in this way, we need to make a little modification to the computation of the diversity when a label l belongs to several mappings. We assign l to the mapping which has the smallest distance to l, and if there are more than one such mappings, we pick the one with the highest similarity score. The parameter $\lambda_1(\lambda_2)$ balances the weight of similarity and diversity. Namely, the larger $\lambda_1(\lambda_2)$ implies the more emphasis on diversity.

EXAMPLE 2.1 (continued). We compute the f_1, f_2 score on the instance in Example 2.1. We use a *graph similarity measure* generalized from the maximum common subgraph measure [40]. Let $\phi : Q \to H \subseteq G$. The similarity score of a node $v \in Q$ to $\phi(v)$ is the Jaccard similarity of their labels, namely $|\ell(v) \cap \ell(\phi(v))|/|\ell(v)|$. The similarity score of an edge $(u,v) \in Q$ to $(\phi(u), \phi(v))$ is 1 if $(\phi(u), \phi(v))$ is adjacent in H or 0 otherwise. The similarity score of H to Q is the sum of the similarity scores of all nodes and edges in Q after normalization. We have the similarity scores of M_1, M_2, M_3, M_4 which are 1, 0.92, 0.83, 0.75 respectively. Let $h = 1$. Consider the top-2 mappings, M_1, M_2 and M_1, M_3. $\vec{l}_{M_3} = (a,1; b,1; c,1; d,1; g,0.5; i,0.5; j,0.5)$. $sim_L(M_1, M_3) = 5.5/6 = 0.917$. Using the f_1 measure and set $\lambda_1 = 0.8$, $f_1(M_1, M_2) = 3.35$ which is larger than $f_1(M_1, M_3) = 2.93$. Using f_2 measure and set $\lambda_2 = 0.2$. For M_1, M_2, $div_G(M_1) = 6$, $div_G(M_2) = 0.5$ and $f_2(M_1, M_2) = 3.21$. For M_1, M_3, $div_G(M_1) = 6$, $div_G(M_3) = 0$ and $f_2(M_1, M_3) = 3.03$. Hence, under either diversity measure, we prefer the M_1, M_2 pair. Visually, M_1, M_2 are relatively far apart in G than M_1, M_3.

Before presenting the ranking algorithms for measure f_1 or f_2, we first prove an important property of f_1, f_2 in the following theorem. Theorem 3.1 shows that both functions f_1, f_2 have a "diminishing return" property which is similar to the monotone submodularity. As a direct implication of

Theorem 3.1, the problem of maximizing f_1 or f_2 is NP-hard. The proof is omitted due to space constraint.

THEOREM 3.1. *Let S_Q be a collection of subgraphs in G. Let f be f_1 or f_2. For any subcollection $S, S' \subseteq S_Q$ and $S \subseteq S'$, for any subcollection $R \subset S_Q$ and $R \cap S' = \emptyset$, we have $f(S) \leq f(S')$, and $f(S \cup R) - f(S) \geq f(S' \cup R) - f(S')$.*

4 Top-k similar results with diversification

In this section, we study the problem of finding the top-k similar subgraph mappings with diversification measured by f (f is f_1 or f_2). Since both the subgraph search problem and the ranking with diversification problem are NP-hard in general, we should not expect any exact solution to solve either problem in polynomial time. Our goal is to design algorithms to approximately solve the problem in an efficient way and attain high accuracy. We first give a close-to-optimal solution using the greedy selection. We then modify this approach to be more space-efficient using the swapping selection. These two algorithms are common approaches in the general search diversification problem [10]. Finally, we give a very fast heuristic based on local search.

4.1 Greedy selection The problem of finding the top-k similar mappings with diversification guarantee can always be solved in two steps. The first step is finding all similar subgraph mappings and the second step is ranking those mappings based on some criteria. Theorem 3.1 suggests that the objective functions f_1, f_2 have a similar property to the monotonically increasing submodularity. This property indicates that we can use the standard greedy algorithm to select the top-k mappings and this approach has a close-to-optimal approximation ratio $1 - 1/e \approx 0.632$ [23]. Let $\mathcal{C}_k$ be the collection of the top-k mappings, which is initialized to be empty. In each iteration of the greedy selection, a mapping S which maximizes the quantity $f(\mathcal{C}_k \cup S) - f(\mathcal{C}_k)$ is selected. Suppose there are in total M similar mappings, there are at most n_l labels with positive weight in any label coverage vector, the time complexity of the greedy selection is $O(kn_lM)$. Since we need to store all similar mappings explicitly, the space complexity is $O(|V| + |E| + M(|V_q| + |E_q|))$. When the number of similar mappings is not too large, the greedy selection is very efficient. On the other hand, it could consume too much space and take a long time to finish the computation. We present the algorithm in Algorithm GR and summarize the approximation ratio of this algorithm in Claim 4.1.

CLAIM 4.1. *Algorithm GR has an approximation ratio $1 - 1/e$, which is optimal assuming $P \neq NP$.*

4.2 Swapping selection One of the drawbacks of Algorithm GR is that we need to generate all similar subgraph mappings before selecting the top-k results. This approach is not scalable when the number of candidate mappings is very large. A natural way to reduce the space consumption is to generate and select the top-k candidates simultaneously. Nemhauser et al. [24] show that a simple polynomial-time swapping-based selection algorithm for the problem of maximizing monotone submodular functions has an approximation ratio 1/2. The algorithm might iterate through the whole candidate set multiple times. More specifically, suppose at time t the algorithm selects a set S of k elements (in our case, an element is a subgraph mapping), if there exists a set of r elements T with $T \cap S = \emptyset$, and r elements $S' \subseteq S$ such that $f((S \setminus S') \cup T) > f(S)$, the algorithm replaces S' by T. Inspired by this strategy, we always keep k mappings $\mathcal{C}_k$ as a candidate solution. Each time the subgraph search algorithm generates a new candidate mapping H, we check if there exists a mapping H' from $\mathcal{C}_k$, such that $f((\mathcal{C}_k \setminus H') \cup H) > \beta f(\mathcal{C}_k)$, for some $\beta \geq 1$. We replace H' with H if such a mapping exists. However, for the sake of reducing time complexity, in experimentation we only go through the entire collection of similar mappings once. We remark that for a cover-based diversity measure f_2, we can show in a similar way as [25] that a one-pass swapping-based algorithm has an approximation ratio 1/4 by choosing $\beta = 2$. However, it is not clear if there is any theoretical guarantee of the solution quality for one-pass or constant-pass swapping-based algorithm for objective function f_1. Following the notations in Section 4.1, the time of deciding whether a swap should be performed is $O(n_l)$ and if so, there is $O(kn_l)$ extra time to update the current solution. Suppose the algorithm goes through the entire collection of similar mappings T times. The time complexity for the whole algorithm is $O(kTn_lM)$. The algorithm requires only space for storing the current solution. Thus the space complexity is $O(|V| + |E| + k(|V_q| + |E_q|))$. So far, we only select the top-k results without ranking them. We could rank the results greedily. However, one drawback of the swapping-based algorithm is that the top-$k'(< k)$ results drawn from the top-k results do not have any performance guarantee [37]. We present the algorithm in Algorithm SW and summarize the approximation ratio of this algorithm in Claim 4.2.

Algorithm GR Greedy selection

1: **Input:** A collection $\mathcal{Q}$ of similar mappings returned by any subgraph search algorithm.
2: Initialize a collection of the top-k mappings $\mathcal{C}_k \leftarrow \emptyset$.
3: **for** $i \leftarrow 1$ to k **do**
4: $\quad H_i \leftarrow \arg_{S \in \mathcal{Q} \setminus \mathcal{C}_k} \max f(\mathcal{C}_k \cup \{S\}) - f(\mathcal{C}_k)$.
5: $\quad \mathcal{C}_k \leftarrow \mathcal{C}_k \cup \{H_i\}$.
6: **end for**
7: **return** $\mathcal{C}_k$.

CLAIM 4.2. *For $\beta = 1$, Algorithm SW has an approximation ratio $1/2$. For $\beta = 2$, $T = 1$, Algorithm SW has an approximation ratio $1/4$ for diversity measure f_2.*

Algorithm SW Swapping selection

```
1: Input: Query Q. A subgraph search algorithm A.
2: Initialize a collection of the top-k mappings C_k ← ∅.
3: for t ← 1 to T do
4:     Use A to generate a similar mapping of Q with
   similarity score at least γ,
5:     if |C_k| < k then
6:         C_k ← C_k ∪ {H}.
7:     else
8:         H' ← arg min f(C_k) − f(C_k \ H').
9:         if f((C_k \ H') ∪ H) > βf(C_k) then
10:            Replace H' with H.
11:        end if
12:    end if
13: end for
14: return C_k ranked greedily.
```

4.3 Local search While Algorithm GR and Algorithm SW both have the flexibility that they can be used to rank the results produced by any subgraph search algorithm, one problem of them is that they need to compute the label coverage vector for every similar mapping. When the number of potential similar mappings increases, evaluating every mapping can be time-consuming. In this section, we present a heuristic to combine the subgraph search step and the ranking step. The key observation to speedup the algorithm is that, intuitively mutually far apart mappings lead to large diversity score. For f_1, if the label coverage vectors of two mappings S_i and S_j do not contain any common label, then $sim_L(S_i, S_j) = 0$. The set of nodes within distance h of S_i and S_j must have no intersection. Thus, S_i and S_j are "far" from each other in the graph G. For f_2, we have $div(S_i \cup S_j) \leq div(S_i) + div(S_j)$ because of a possible intersection of the label covers of S_i, S_j. We might still prefer to choose S_i and S_j which are not close. Hence, to maximize f_1 or f_2, we should skip evaluating mappings which are clustered together. Namely, after finding one candidate mapping, the algorithm should jump to some "far-apart" location to look for the next candidate.

Although we try to keep the subgraph search step as a black box, to better explain the details of the local search procedure, we first summarize the common approach of doing subgraph similarity search. A typical paradigm for subgraph similarity search is first finding candidate matchings of a node or a small unit of the query graph, then aggregating those candidate matchings based on some specific function. For a given query graph Q, we denote a list of candidate matchings of a node $v \in Q$ by $cand(v)$. The candidate matchings of v contain nodes in G which are "similar" to v. The node similarity can be measured by comparing the two corresponding nodes [28], or comparing an r-hop neighborhood (usually $r \leq 2$) of them [17, 18]. Then $cand(v)$ is sorted in the descending order of this measure.

Let's get back to our heuristic approach of choosing "far-apart" mappings. A natural way to implement this intuition is using local search. Local search is a widely-used strategy for solving optimization problems. In many situations, it is proved to be very efficient and accurate. Our general search strategy starts from finding an arbitrary similar mapping greedily. Namely, we always try node candidate matchings with the largest node similarity score. We call this mapping an anchor mapping A (with vertex set V_A). We then perform local search around the anchor by picking a small subset of nodes S, and search around the neighborhood of $V_A \setminus S$ to improve the incremental diversity score with the guarantee that the similarity of the new mapping is at least γ. Suppose we find a mapping H. To find the next mapping, we search for the anchor mapping by avoiding nodes in H unless it does not succeed, we relax this restriction by picking an arbitrary similar mapping.

For f_1, we can continue using the node candidate matchings sorted with respect to node similarity measure. As for diversity f_2, this strategy might not suffice because our objective function is maximizing the "coverage", thus ensuring only large mutual distance is not enough. We propose two strategies to adapt to this case. The first one is to find more than k candidate mappings and then use the greedy selection (Algorithm GR) to pick the top k results. The second one is to rank node candidate matchings by considering the diversity score, e.g. using $f_2(v)$. The intuition can be explained as follows. Suppose we ignore the graph similarity constraint. We can define the diversified coverage (DC) of a subset $H \subseteq V$ as a weighted set. An element represents a distinct label l with weight $\alpha^{dist(l,H)}$. We define the "union", $\uplus$ of two DC sets S, T as the standard set union of elements, for every element $e \in S \uplus T$, the weight of e is the maximum weight of e in S and T. It is easy to show that this function g defined on $\uplus$ is submodular. Hence, we can employ the greedy algorithm to aggregate candidate lists and obtain a sufficiently diversified result. The algorithm is presented in Algorithm LS.

Assume the greedy aggregation of an anchor mapping typically succeeds in T_a steps and we perform T steps local search around the anchor. The time complexity of Algorithm LS is $O(k(T_a + n_l T))$. The space complexity is similar as Algorithm SW, which is $O(|V| + |E| + k(|V_q| + |E_q|))$.

4.4 Optimization We compute an index of the target graph G off-line so that the algorithms can retrieve useful information efficiently. The index we use is standard. First,

Algorithm LS Local search heuristic

Input: Query Q, a search order π, node candidate matchings $\{cand(v), v \in V_q\}$.
Initialize a collection of the top-k mappings $\mathcal{C}_k \leftarrow \emptyset$.
for $i \leftarrow 1$ to k **do**
  $\triangleright$ Find an anchor mapping.
  Initialize anchor mapping A to be empty.
  for v according to the search order π **do**
    while unmatched node not found **do**
      $u \leftarrow$ the first node in $cand(v)$ which has not been matched.
      if an upper bound of the similarity score of $A \cup \{u\}$ is at least γ **then**
        $A \leftarrow A \cup \{u\}$.
        break.
      end if
    end while
  end for
  if the anchor mapping is not found **then**
    Pick an arbitrary mapping A with similarity score at least γ.
  end if
  $\triangleright$ Local search.
  for $t \leftarrow 1$ to T **do**
    Randomly pick a subset of p nodes S from V_q.
    if there is a subset of p nodes T in the r-hop neighborhoods of $V_A \setminus S$ such that $f(\mathcal{C}_k \cup A) < f(\mathcal{C}_k \cup (A \setminus S \cup T))$, and the similarity score of $(A \setminus S) \cup T$ is at least γ **then**
      $A \leftarrow A \setminus S \cup T$.
    end if
  end for
  $\mathcal{C}_k \leftarrow \mathcal{C}_k \cup A$.
  Mark nodes in A as matched.
end for
return $\mathcal{C}_k$.

construct an inverted index for every label by maintaining a hash set of nodes in G which contain this label. Second, for every node $u \in G$, we maintain an r-hop neighbors of u. Let n be the number of nodes in G. The time complexity for setup the inverted index is $O(|L|n)$. Let d_g be the average degree of nodes in G. It takes $O(d_g^r \cdot n)$ time to find all r-hop neighbors for every node. The space complexity for maintaining the index is $O((|L| + d_g^r) \cdot n)$.

One important quantity considered in this paper is the label coverage vector. Equipped with the label and the neighbor index, we can compute the label coverage vector in time linear to the number of labels with positive weight in this vector. Consider a subset S as an example. Suppose we implement the label coverage vector and the neighbor of a node using hash set. When we insert a new node v to S, we scan each node $u \in N^r(v)$ and check if $u \in N^r(S)$. We also scan $\vec{l}_u$ to update $\vec{l}_S$.

5 Experiments

In this section, we present results of empirical evaluations of Algorithm GR, SW, LS. We use results obtained by the greedy selection (Algorithm GR) as the ground truth for comparison.

5.1 Experiments setup We test our algorithms on both real-world data and synthetic data. The statistics of those datasets are summarized in Table 1.

DBLP[2]. We construct a co-authorship graph from the DBLP dataset. The nodes are authors. We add an edge between two nodes if the corresponding authors have participated in at least one published work. The label of a node is the set of places where the corresponding author has publications, and the weight of this label is the number of publications at that place by this author. In this way, the DBLP graph contains 1036991 nodes, 3491030 edges and 6748 labels.

IMDB[3]. We construct a co-casting graph from the IMDB dataset. The nodes are actors and actresses. We add an edge between two nodes if the corresponding actors/actresses have co-casted in at least five plays. The label of a node is a set of genres such that the actor/actress performs in at least one play of that genre. The total types of genres is 30. We keep only nodes of at least 10 labels (with repetition). In this way, the IMDB graph contains 199634 nodes, 8212206 edges and 30 labels.

Synthetic data. We use a collaboration network ca-HepTh (hep), ca-AstroPh (AstroPh), and a product co-purchasing network amazon0302 (amazon) from the network datasets[4] as the target graphs, and generate synthetic labels. We generate the labels in the following two ways. First, assign a uniformly random number l from 1 to l_m to every node in G, and then sample l labels from a label set L uniformly at random to this node. Second, assign a random number l to the node v, where l is proportional to the degree of v plus a constant bias, and then sample l labels from a label set L uniformly at random to v.

Query graphs. We generate query graphs of q nodes in two ways. First, extract a subgraph from G. Starting from a random node of G, we use the random breadth-first search to select the neighborhoods until q nodes are selected. For the selected nodes, we randomly pick a small number of labels and assign to the query graph. Second, generate a random graph of q nodes and assign a small number of labels to every node uniformly at random from L. In our experiments, we use only the first method to generate queries for the real-

[2] http://dblp.uni-trier.de/xml/
[3] http://www.imdb.com/interfaces
[4] http://snap.stanford.edu/data/

Table 1: Statistics of the graph data.

Data	$\|V\|$	$\|E\|$	deg_{max}	$\overline{deg}$	File(Mb)
DBLP	1036991	3491030	5571	6.73	75.1
IMDB	199634	8212206	6695	82.3	33.2
hep	9877	51971	65	5.26	0.66
AstroPh	18722	396160	504	21.1	5.3
amazon	262111	1234877	420	6.87	17.2

world data as using the second method usually results in queries which have only a small number of matchings in G. We use the second method to generate queries for the synthetic data.

Implementation. We implement the procedure of generating node candidate matchings based on [18], which is demonstrated to outperform many other approaches. We use the graph similarity measure defined in Example 2.1. To efficiently aggregate node matchings and find subgraph mappings, we travers the graph in depth-first order and use pruning rules.

System. The experiments are conducted on a server machine with a 2.67GHz Intel Xeon 4-core processer, running Linux 2.6.18 and gcc 4.1.2. The system has 256KB L2 cache per core, 8192KB L3 cache, and 24G main memory. We implement all algorithms in C++, and use the STL implementation for all data structures.

5.2 Experimental results - Efficiency and solution quality of Algorithm GR, SW, LS on synthetic data. We use a small dataset, the hep data to compare the performance of the three algorithms. We use the first method to generate labels on nodes, with frequency 10 from in total 100 labels. We set the size of the query graphs from 3 to 9, fix $k = 10$, $\alpha = 0.5$, $\gamma = 0.8$, $\lambda_1 = 0.09$, $\lambda_2 = 0.1$. The result is given in Figure 2. For any size of the query graph and any diversity measure, we conclude that on average the Algorithm GR (greedy selection) performs the best in solution quality, followed by Algorithm LS (local search heuristic), then Algorithm SW (swapping selection). Algorithm LS is worse than Algorithm GR by at most 10% and very close on average. Algorithm SW is 15% worse. As we observe from the results, the diversity scores of the top-k mappings returned by Algorithm LS also exhibit a "greedy" style. As for Algorithm SW, they are more uniform. For the time complexity, Algorithm GR and Algorithm SW are similar, while Algorithm LS is 2-scales faster. The time complexity of Algorithm GR and SW depends on the number of similar mappings. Thus, their running time on smaller queries ($k = 4$, f_1 measure) can be higher than larger queries. On the other hand, the running time of Algorithm LS is not very sensitive

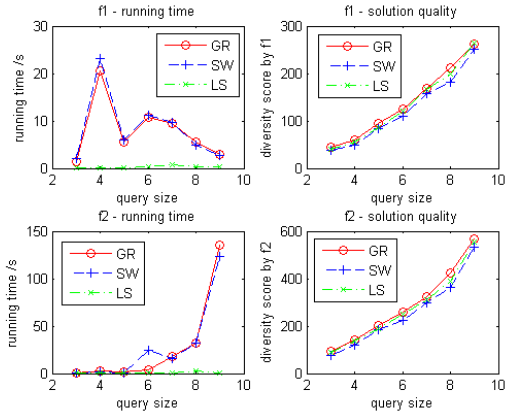

Figure 2: Performance of Algorithm GR, SW, LS with respect to the query size on the hep data.

to the size of the query graph.

We then use the astroph10 and the amazon5.400 data to test the performance of the three algorithms. We have observed from the hep data a large gap of time complexity between Algorithm GR, SW and Algorithm LS. The difference becomes more significant for these larger datasets. For example, in a test on astroph10 with query of size 5 and diversity measured by f_2, Algorithm LS finishes in 0.37s while Algorithm GR in 203s and Algorithm SW in 178s. In another test on amazon5.400 with query of size 4 and diversity measured by f_2, Algorithm LS finishes in 0.96s while Algorithm GR in 467s and Algorithm SW in 402s. For larger query, Algorithm GR, SW are more time-consuming. For example, in a test on the real-world dataset DBLP with query size 3 and diversity measured by f_2, Algorithm SW finishes in 4.1 hours, and Algorithm GR cannot finish within 5 hours, while Algorithm LS for about 3 minutes. Based on these results, for the astroph10, amazon5.400 and the DBLP, IMDB data, we do not test performance of Algorithm SW and we only compare solution quality of Algorithm GR and LS. We also modify the Algorithm GR by early terminating the program if 5000 similar mappings are found, while the total number of similar mappings is usually at least 10^6.

The results of the astroph10 and the amazon5.400 data are presented in Figure 3. We can see that Algorithm GR and Algorithm LS have very close solution quality for large size query. For small query, Algorithm GR is usually 10% better.

- Performance of Algorithm LS with different heuristics. As we point out in Section 4.3, for Algorithm LS, we can try different heuristic to obtain better solution quality for diversity measure f_2. We compare the performance of the three heuristics, the first one is presented in Algorithm LS ($LS = k$), the second one is first finding $k' > k$ mappings then selecting the top-k of them by the greedy selection ($LS > k$), the third one is sorting node matching candidate

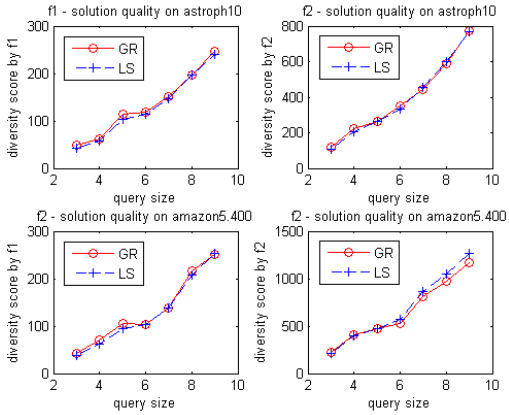

Figure 3: Solution quality of Algorithm GR, LS with respect to the query size on the astroph10 and the amazon5.400 data.

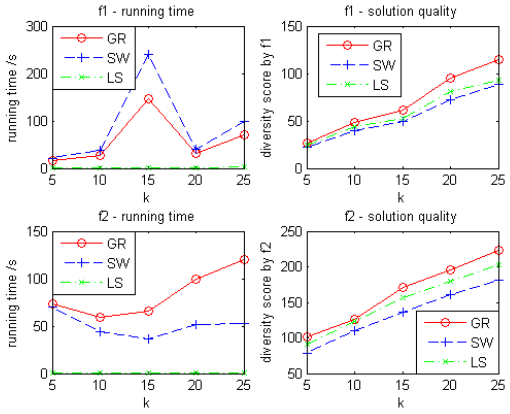

Figure 4: Performance of Algorithm GR, SW, LS with respect to k on the astroph10 data.

Table 2: Comparison of the three heuristics for the local heuristic Algorithm LS by solution quality. The best one is highlighted.

	astroph10			amazon5.400		
query size	3	4	5	3	4	5
$LS = k$	111	192	283	215	**358**	535
$LS > k$	**117**	**204**	**309**	**220**	351	530
$LS - f_2$	108	194	291	216	353	**536**

lists with respect to $f_2(v)$ ($LS - f_2$). We compare the three heuristics on the astroph10 and the amazon5.400 data, varying query size from 3 to 5 and set $k' = 2k$. The result is given in Table 2. We can see that there is no big difference among the three heuristics. $LS > k$ has better solution quality on average than the other two. $LS = k$ and $LS - f_2$ wins in different queries. Interestingly, the running time of the three heuristics are also very close. Although in the testing data, there is no significant advantage of employing the $LS > k$ or the $LS - f_2$ heuristic than $LS = k$, we believe they are more reasonable heuristics for optimizing f_2, and it is interesting to find data or analytically justify this intuition.

- Parameter k. We use the astroph10 data to test the efficiency and solution quality of Algorithm GR, SW, LS with respect to k. We use the query graphs of size 3, set k from 5 to 25 with step 5, $\alpha = 0.5$, $\epsilon = 0.2$, $\gamma = 0.8$, $\lambda_1 = 0.1$, $\lambda_2 = 0.1$. Experiments are repeated 3 times for each value of k and every time with a new generated query graph. The results are given in Figure 4. The running time of Algorithm GR and LS increases with respect to k. For Algorithm SW, the running time is larger for bigger k but there exhibits more oscillations. We also observe that for some query ($k = 15$, f_1 measure), the running time can be much larger. This is because sometimes we might have a lot more node matching candidates. For the solution quality, Algorithm GR outperforms Algorithm LS, Algorithm LS is better than Algorithm SW. The gap increases for larger k, and is more significant for the f_1 measure.

- Solution quality of Algorithm GR, LS on real-world data. We compare the solution quality of the modified Algorithm GR and Algorithm LS on the real-world data DBLP and IMDB. We set the size of the query graphs from 3 to 9, fix $k = 5$, $\alpha = 0.5$, $\gamma = 0.8$, $\lambda_1 = 0.09$, $\lambda_2 = 0.08$. The results are given in Figure 5. We observe again a very close solution quality of Algorithm GR and LS. Due to the simplified implementation of Algorithm GR, as it does not generate all similar mappings, the quality of Algorithm GR is sometimes worse than Algorithm LS.

We remark that although the running time of Algorithm LS is much faster than Algorithm GR, SW, on these datasets where the number of node matching candidates can be very large (10^4 for example), Algorithm LS might still require a scale of 10^3s time to finish the computation.

6 Conclusion

We study the problem of ranking subgraph search results with diversification guarantee. We propose two diversity measures and formalize the problem as an optimization problem. We propose two algorithms to efficiently find top-k similarity mappings with provable performance guarantee and one very fast heuristic by combining the subgraph search step and the ranking step. Experiments demonstrate the efficiency and effectiveness of our methods.

References

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, 2009.

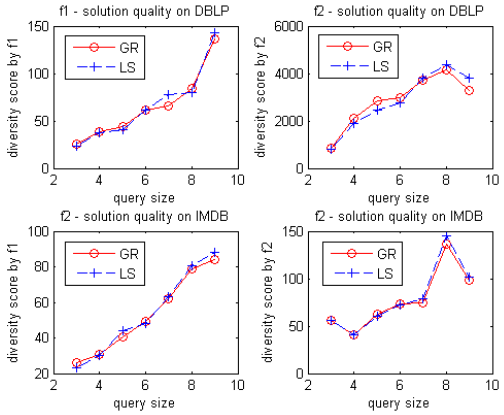

Figure 5: Solution quality of Algorithm GR, LS with respect to query size on the DBLP and the IMDB data.

[2] A. Angel and N. Koudas. Efficient diversity-aware search. In *SIGMOD*, 2011.

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

[4] G. Capannini, F. M. Nardini, R. Perego, and F. Silvestri. Efficient diversification of web search results. In *PVLDB*, 2011.

[5] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

[6] Z. Chong, H. Chen, Z. Zhang, H. Shu, G. Qi, and A. Zhou. Rdf pattern matching using sortable views. In *CIKM*, 2012.

[7] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, 2009.

[8] V. Dang and W. B. Croft. Diversity by proportionality: an election-based approach to search result diversification. In *SIGIR*, 2012.

[9] Z. Dou, S. Hu, K. Chen, R. Song, and J.-R. Wen. Multi-dimensional search result diversification. In *WSDM*, 2011.

[10] M. Drosou and E. Pitoura. Search result diversification. *SIGMOD Rec.*, 39(1):41–47, 2010.

[11] M. Drosou and E. Pitoura. Disc diversity: result diversification based on dissimilarity and coverage. In *PVLDB*, 2013.

[12] W. Fan, X. Wang, and Y. Wu. Diversigied top-k graph pattern matching. In *PVLDB*, 2014.

[13] P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Top-k bounded diversification. In *SIGMOD*, 2012.

[14] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Anal. Appl.*, 13(1):113–129, Jan. 2010.

[15] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, 2009.

[16] D. P. Idan Szpektor, Yoelle Maarek. When relevance is not enough: Promoting diversity and freshness in personalized question recommendation. In *WWW*, 2013.

[17] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao. Neighborhood based fast graph search in large networks. In *SIGMOD*, 2011.

[18] A. Khan, Y. Wu, C. C. Aggarwal, and X. Yan. Nema: Fast graph search with label similarity. In *PVLDB*, 2013.

[19] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *KDD*, 2009.

[20] J. Lee, W.-S. Han, R. Kasperovics, and J.-H. Lee. An in-depth comparison of subgraph isomorphism algorithms in graph databases. In *PVLDB*, 2013.

[21] K. Liu, E. Terzi, and T. Grandison. Highlighting diverse concepts in documents. In *SDM*, 2009.

[22] S. Ma, Y. Cao, J. Huai, and T. Wo. Distributed graph pattern matching. In *WWW*, 2012.

[23] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operational research Programming*, 3(3):177–188, 1978.

[24] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[25] B. Saha and L. Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *SDM*, 2009.

[26] P. I. Sofiane Abbar, Sihem Amer-Yahia and S. Mahabadi. Efficient computation of diverse news. In *WWW*, 2013.

[27] Y. Tian, R. C. Mceachin, C. Santos, D. J. States, and J. M. Patel. Saga: a subgraph matching tool for biological graphs. *Bioinformatics*, 23(2):232–239, 2007.

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

[29] H. Tong, J. He, Z. Wen, R. Konuru, and C.-Y. Lin. Diversified ranking on large graphs: an optimization viewpoint. In *KDD*, 2011.

[30] G. Valkanas, A. N. Papadopoulos, and D. Gunopulos. Skydiver: a framework for skyline diversification. In *EDBT*, 2013.

[31] M. R. Vieira, H. L. Razente, M. C. N. Barioni, M. Hadjieleftheriou, D. Srivastava, C. Traina, and V. J. Tsotras. On query result diversification. In *ICDE*, 2011.

[32] J. Wang, J. Cheng, and A. W.-C. Fu. Redundancy-aware maximal cliques. In *KDD*, 2013.

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

[34] J. Yang, S. Zhang, and W. Jin. Delta: indexing and querying multi-labeled graphs. In *CIKM*, 2011.

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

[36] C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT*, 2009.

[37] H. Yu and D. Yuan. Set coverage problems in a one-pass data stream. In *SDM*, 2013.

[38] Y. Yuan, G. Wang, L. Chen, and H. Wang. Efficient subgraph similarity search on large probabilistic graph databases. In *PVLDB*, 2012.

[39] S. Zhang, J. Yang, and W. Jin. Sapper: Subgraph indexing and approximate matching in large graphs. In *PVLDB*, 2010.

[40] Y. Zhu, L. Qin, J. X. Yu, Y. Ke, and X. Lin. High efficiency and quality: large graphs matching. In *CIKM*, 2011.

[41] Y. Zhu, J. X. Yu, H. Cheng, and L. Qin. Graph classification: a diversified discriminative feature selection approach. In *CIKM*, 2012.