Independent Informative Subgraph Mining for Graph Information Retrieval

Bingjun Sun
Department of Computer Science and Engineering
Pennsylvania State University
University Park,PA 16802,USA
sunbingjun@hotmail.com

Prasenjit Mitra
College of Information Sciences and Technology
Pennsylvania State University
University Park,PA 16802,USA
pmitra@ist.psu.edu

C. Lee Giles
College of Information Sciences and Technology
Pennsylvania State University
University Park,PA 16802,USA
giles@ist.psu.edu

ABSTRACT

In order to enable scalable querying of graph databases, intelligent selection of subgraphs to index is essential. An improved index can reduce response times for graph queries significantly. For a given subgraph query, graph candidates that may contain the subgraph are retrieved using the graph index and subgraph isomorphism tests are performed to prune out unsatisfied graphs. However, since the space of all possible subgraphs of the whole set of graphs is prohibitively large, feature selection is required to identify a good subset of subgraph features for indexing. Thus, one of the key issues is: given the set of all possible subgraphs of the graph set, which subset of features is the *optimal* such that the algorithm retrieves the smallest set of candidate graphs and reduces the number of subgraph isomorphism tests? We introduce a graph search method for subgraph queries based on subgraph frequencies. Then, we propose several novel feature selection criteria, Max-Precision, Max-Irredundant-Information, and Max-Information-Min-Redundancy, based on *mutual information*. Finally we show theoretically and empirically that our proposed methods retrieve a smaller candidate set than previous methods. For example, using the same number of features, our method improve the precision for the query candidate set by 4%-13% in comparison to previous methods [25, 26]. As a result the response time of subgraph queries also is improved correspondingly.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation,Retrieval models*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Feature evaluation and selection*

General Terms

Algorithms, Design, Experimentation

CIKM'09, November 2–6, 2009, Hong Kong, China.

Keywords

Graph mining, feature selection, index pruning, graph search

1. INTRODUCTION

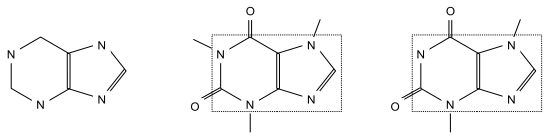
Graphs have been used to represent structured data for a long time. Recently, more and more structured data, such as chemical molecule structures [20, 22], DNA and protein structures [10], social networks [5], social and citation networks [14], and XML documents [29], are identified and studied. Large number of such databases are available on the Web. Data mining and search methods for structured data are needed for users to quickly identify a small subset of relevant data for further analysis and experiments.

End-users pose graph queries to a graph database and seek to retrieve its *support*, i.e., all graphs of which the queried graph is a subgraph (Figure 1). Graph query answering has been addressed while determining chemical structure similarity [17]. The crux of the problem lies in the complexity of subgraph isomorphism. However, since subgraph isomorphism is NP-complete [26], it is prohibitively expensive to scan all graphs in real time. When the number of graphs are large, to enable fast querying, instead of isomorphism tests on the fly, we need to index the graphs offline to allow for fast graph retrieval. Indexing all possible subgraphs of graphs is impossible because of their sheer number. Thus, intelligent indexing techniques and index pruning techniques for graph databases are essential to enable scalable querying. Graph database querying is decomposed into three stages [26]: 1) graph mining, 2) graph indexing, and 3) graph querying. First, from the set of graphs in the database certain subgraph are extracted and selected. Then each selected subgraph is converted into a canonical string, and each graph is mapped into a linear space of subgraphs. Second, a graph index is built using the canonical strings of subgraphs. Finally, for a given subgraph query, all the indexed subgraphs of the query are determined, and the index is looked up with these subgraphs to obtain a candidate set of graphs containing all the indexed subgraphs. Subgraph isomorphism tests are performed on the candidate set to find all graphs that contain the query graph. This candidate set must be small for the graph query if the graph retrieval is in real-time. To keep the candidate set small as well as keep the index size reasonable, we need to select subgraphs judiciously for indexing. We address the issue of finding the set of subgraphs to index such that the candidate set of subgraphs on which the subgraph isomorphism is computed is the smallest.

All previous approaches [13, 11, 15, 25, 26] try to find an appropriate way to discover a set of subgraphs that contains as much *information* as possible to achieve a high precision of query answers. However, no previous work proposes any criterion to measure the information contained in the set of subgraphs. Most of them only assume *frequent subgraphs* are more informative than *infrequent subgraphs* [13, 11, 15, 25], where a *frequent subgraph* is a subgraph that occurs more frequently in a graph database than a threshold value. Some of them find frequent subgraphs independently and ignore the redundancy between subgraphs [13, 11, 15]. For example, for a set of graphs D, all subgraphs G_i' of a frequent subgraph G' are also frequent. If G' occurs every time when G_i' occurs, and does not occur independently very often, then G_i' is redundant. Selecting G_i' after having selected G' cannot increase the information contained in the feature set as expected. In other words, two informative features with high redundancy contain less over-all information than two informative features with low redundancy. Some previous works propose heuristics to avoid selecting redundant features to some extent [25, 6, 26]. However, all these methods only remove the redundancy partially. We propose novel methods that remove the redundancy as much as possible while indexing such that while looking up the index, fewer false positives are selected as candidates on which the full graph isomorphism has to be performed.

All previous works select subgraphs independently or sequentially, but none of them propose any criteria to measure the information contained in a set of subgraphs. Methods that select subgraphs independently ignore the redundancy between subgraph features. Sequential subgraph selection methods find frequent subgraphs from the smallest to the largest size and avoid selecting a subgraph when its supergraph has been selected. However, they do not consider highly correlated subgraphs that are not a supergraph-subgraph pair. Previous efforts [25, 6] focus on two criteria, i.e., *informative subgraph principle* and/or *irredundant subgraph principle*, although they have different heuristics to estimate information and redundancy. However, they are not shown theoretically to have an optimal or near optimal performance; we do so for our methods. This is the major contribution in this paper.

We propose a subgraph selection criterion based on *mutual information* called Max-Irredundant-Information (MII), which is an approximation of Max-Precision, which can measure the over-all information content of a subgraph set and attempts to maximize the precision of retrieved subgraphs over which subgraph isomorphism has to be computed. However, computing the information content of all possible subset of selected subgraphs is expensive. Thus, we propose a method Max-Information-Min-Redundancy (MImR), which approximates the MII method, and combined with a greedy feature selection algorithm is much more computationally efficient. Our approach is different from previous work in that: 1) we optimize the evaluation criterion of precision directly, 2) we use a probabilistic model based on mutual information to approximately optimize precision, and 3) we combine the informative and irredundant principles naturally using a probabilistic model and find an approximation method to find the near-optimal subgraph feature set to index. The proposed methods are expected to outperform previous ones, because we prove theoretically that they can achieve the *optimal* or *near-optimal* precision. Furthermore, Yan, Yu, and Han [26] only consider the occurrence of subgraphs in graphs as binary features for search. We utilize the subgraph frequencies (i.e., the number of times each subgraph occurs) in each graph to prune out more graphs that do not contain the query graph.



(a) Query (b) Caffeine (c) Thesal

Figure 1: Subgraph query (a) and its support (b, c)

Our major contributions are as follows:

- We show a framework for graph mining, indexing, and search for querying graph databases, which can be applied to data repositories of chemical structures, protein structures, etc.
- We propose a new criterion to filter out highly correlated subgraphs by mining *independent frequent subgraphs*.
- We propose several novel criteria for subgraph selection with corresponding algorithms based on directly optimizing precision and mutual information. We also apply a greedy forward subgraph selection method based on these criteria.
- We introduce a search method for subgraph queries. Our method reduces candidate graph sets on which full isomorphism has to be performed by 4%-13% than a previous approach [26], so that query time is reduced correspondingly.

The rest of this paper is organized as follows. In Section 2, we review related works. In Section 3, we describe our graph search process. In Section 4, we first propose an algorithm for mining independent frequent subgraphs. Then we propose three criteria to further reduce the set of selected independent, frequent subgraphs. Finally ,we describe a greedy algorithm for subgraph selection. Section 5 presents experimental results validating our algorithms. Conclusions and future work are discussed in Section 6.

2. RELATED WORK

Prior work on querying graphs fall into three categories: graph pattern mining, indexing, and search. Work on graph mining extract features from graphs to map them into a linear space for indexing. Three types of features can be used, *paths* only [18], *trees* only [28], and *subgraphs* [26]. Most previous approaches index subgraphs, because paths and trees are special subgraphs that lose much structural information due to their limited representative capabilities. After the subgraph features are extracted, *canonical labeling* or an *isomorphism test* is used to determine if two graphs are isomorphic to each other. A canonical label is a unique string corresponding to a graph, such that there is a one-to-one mapping function between graphs and canonical labeling strings. Both canonical labeling and isomorphism tests are NP-complete [26, 13]. Canonical strings of subgraphs based on canonical labeling are used for graph indexing and search.

Since there are too many subgraphs to index, feature selection is required. A naive idea is to select frequent subgraphs only [8]. There are several algorithms for mining

Table 1: Notations used throughout

Term	Description	Term	Description
G	graph	FG	candidate subgraph set
G'	subgraph	S	selected subgraph set
D	graph set	n	# candidate subgraphs
D_G	support of G	m	# selected subgraphs
G_q	graph query	$F_{G' \subset G}$	frequency of G' in G
Q	query set	s_q	$\{G'_q \| G'_q \in S, G'_q \subseteq G_q \in Q\}$
Term		Description	
$D_{G'_q, \geq F_{G'_q \subseteq G_q}}$		$\{G \| G \in D_{G'_q}, F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}\}$	
$\forall G'_q$	$\forall G'_q, G'_q \subseteq G_q \wedge G'_q \in S, F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}$		

frequent subgraphs, AGM [11], FSG [13], gSpan [26], FFSM [10], Gaston [15], and others [2, 1]. Worlein, et al., [24] provide a quantitative comparison of the subgraph miners. Fischer and Meinl provide an overview of graph-based molecular data mining [9]. Indexing the full set of frequent subgraphs results in very large indices, because many of the frequent subgraphs are redundant. Previous works usually use the following approaches to reduce the set of indexed subgraphs: 1) mining closed frequent subgraphs, where only each subgraph that is not 100% correlated with any of its supergraphs are selected [25], 2) mining for each frequent subgraph that has correlations with all of its supergraphs lower than a threshold [6], 3) sequentially selecting subgraphs from the smallest size to the largest size based on subgraphs' frequency and discrimination [26], and 4) using paths composed of small basic structures, such as cycles, crosses, and chains, instead of vertices and edges [12].

For chemical structure search, previous methods fall into four categories: 1) *full structure search*: search the exactly matched structure as the query graph, 2) *substructure search*: find structures that contain the query graphs [19, 26], 3) *full structure similarity search*: retrieve structures that are similar to the query graph [17], and 4) *substructure similarity search*: find the structures that contain a substructure that is similar to the query graph [27].

Previous works on index pruning for IR usually prune postings of irrelevant terms in each document [4, 3, 7]. Criteria in *information theory* are applied to measure the information of each term in each document. However, most previous works focus on selecting informative terms without considering redundancy [4, 3]. Moura, et al., consider local information of phrases to keep consistent postings of correlated terms, instead of global information for feature selection [7]. Peng, et al., propose feature selection focusing on supervised learning [16]. The generic goal is to find the optimal subset from the set of feature candidates so that selected features are 1) correlated to the class distribution, and 2) uncorrelated to each other. We extend the idea of feature selection to graph search (see in Section 4.2).

3. PROBLEM FORMALIZATION

In this section, we introduce preliminary notations, and then give an overview of how a subgraph query is processed. Table 1 lists the notations used throughout the paper.

3.1 Preliminaries

We consider only *connected labeled undirected (sub)graphs*. Relevant notations are given as follows:

Definition 1. **Labeled Undirected Graph:** A labeled undirected graph is a 5-tuple, $G = \{V, E, L_V, L_E, l\}$, where V is a set of vertices. Each $v \in V$ is an unique ID representing a vertex, $E \subseteq V \times V$ is a set of edges where $e = (u, v) \in E, u \in V, v \in V$, L_V is a set of vertex labels, L_E is a set of edge labels, and $l : V \cup E \rightarrow L_V \cup L_E$ is a function assigning labels to vertices and edges on the graph.

Definition 2. **Connected Graph:** A path $p(v_1, v_{n+1}) = (e_1, e_2, ..., e_n), e_1 = (v_1, v_2), e_2 = (v_2, v_3), ..., e_n = (v_n, v_{n+1})$, $e_i \in E_G, i = 1, 2, ..., n$, on a graph G is a sequence of edges connecting two vertices $v_1 \in V_G, v_{n+1} \in V_G$. A graph G is connected iff $\forall u, v \in V_G$, a path $p(u, v)$ always exists. The *size* of a graph $Size(G)$ is the number of edges in G.

Definition 3. **Subgraph and Connected Subgraph:** A subgraph G' of a graph G is a graph, i.e., $G' \subseteq G$, where $V_{G'} \subseteq V_G$, and $E_{G'} \subseteq E_G$ where $\forall e = (v_1, v_2) \in E_{G'}$, the two vertices $v_1, v_2 \in V_{G'}$. G is the supergraph of G', or we say G contains G'. A subgraph G' of a graph G is a connected subgraph if and only if it is a connected graph.

Note that if we change the IDs of vertices on a graph, the graph still keeps the same structure. Thus, a graph isomorphism test is required to identify whether two graphs are isomorphic (i.e., the same) to each other [26]. Another method to achieve the function of graph isomorphism tests is to use canonical labels of graphs [26]. Usually if there is a method to sequentialize all isomorphic graphs of the same graph into different strings, then the minimum or maximum string is the canonical labeling. Two graphs are isomorphic to each other, if and only if their strings of canonical labeling are the same. Thus, strings of canonical labeling of subgraph features can be used to index graphs for fast search. We provide two definitions below:

Definition 4. **Graph Isomorphism and Subgraph Isomorphism:** A graph isomorphism between two graphs G and G', is a bijective function $f : V_G \rightarrow V_{G'}$ that maps each vertex $v \in V_G$ to a vertex $v' \in V_{G'}$, i.e., $v' = f(v)$, such that $\forall v \in V_G, l_G(v) = l_{G'}(f(v))$, and $\forall e = (u, v) \in E_G, (f(u), f(v)) \in E_{G'}$ and $l_G(u, v)) = l_{G'}((f(u), f(v))$. Since f is a bijective function, a bijective function $f' : V_{G'} \rightarrow V_G$ also exists. A subgraph isomorphism between two graphs G' and G is the graph isomorphism between two graphs G' and G'', where G'' is a subgraph of G.

Definition 5. **Canonical labeling:** A canonical labeling $CL(G)$ is a unique string to represent a graph G, where given two graphs G and G', G is isomorphic to G' iff $CL(G) = CL(G')$.

As mentioned before, both isomorphism tests and canonical labeling can be used to determine if two graphs are isomorphic. The canonical labelings of selected subgraph features are indexed for graph search.

3.2 Answering Subgraph Queries

In this section, we first provide some definitions, and then introduce an algorithm to answer subgraph queries.

Definition 6. **Support, Support Graph, and Subgraph Query:** Given a data set D of graphs G, the *support* of subgraph G', $D_{G'}$, is the set of all graphs G in D that contain G', i.e., $D_{G'} = \{G | G \in D, G' \subseteq G\}$. Each graph in $D_{G'}$ is a *support graph* of G'. $|D_G|$ is the number of support graphs in D_G. A *subgraph query* G_q seeks to find the support of G_q, D_{G_q}.

Algorithm 1 Graph Search of Subgraph Query

Algorithm: GSSQ(G_q,S,$Index_D$):
Input: Query Subgraph G_q, indexed subgraph set S, and index of the graph set D, $Index_D$.
Output: Support of G_q, D_{G_q}.
1. **if** G_q is indexed, find D_{G_q} using $Index_D$; **return** D_{G_q};
2. $D_{G_q} = \{\emptyset\}$;
   find all subgraphs of G_q, $G'_q \in S$ with $F_{G' \subseteq G_q}$;
3. **if** no G'_q is found, $D_{G_q} = D$;
4. **else for all** G'_q **do**
5. Find $D_{G'_q, \geq F_{G'_q \subseteq G_q}}$ using $Index_D$,
   then $D_{G_q} = D_{G_q} \cap D_{G'_q}$;
6. **for all** $G \in D_{G_q}$ **do**
7. **if** subgraphIsomorphism(G_q, G)==false, remove G;
8. **return** D_{G_q};

Like words are indexed to support document search, subgraphs are indexed to support graph search. Note that subgraphs may overlap with each other. We define subgraph frequency as follows:

Definition 7. **Embedding and Subgraph Frequency:** An *embedding* of a subgraph G' in a graph G, i.e., $Emb_{G' \subseteq G}$, is an instance of $G' \subseteq G$. The frequency of a subgraph G' in a graph G, i.e., $F_{G' \subseteq G}$, is the number of embeddings of G' in G. Embeddings may overlap.

Algorithm 1 shows how a subgraph query can be answered. First, if the query graph G_q is indexed, its support is returned directly. Otherwise, the algorithm identifies all G'_q, i.e., all indexed subgraphs of G_q, with their corresponding frequency $F_{G'_q \subseteq G_q}$, then finds all graphs G from the index where G satisfies $F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}$, and finally performs subgraph isomorphism tests on each G to identify whether it contains G_q. For each selected subgraph $G'_q \subseteq G_q$, each support graph G of G_q must also be a support graph of each graph query G'_q.

Note that besides using subgraph frequencies, we also can use binary features to represent if a subgraph occurs in a graph or not. Many previous works use binary features [25, 26]. In this case, $F_{G'_q \subseteq G} = \{0, 1\}$. Thus, $F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}$ means that if for each subgraph G'_q occurring in G_q, G'_q also occurs in G, then G is a candidate of the support of the query G_q. This is because each occurrence of G'_q in G_q also occurs in each support graph of G_q, G. Thus, the intersection of the support of each G'_q with $F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}$ must contain all support graphs of G_q. As mentioned before, how to extract and select subgraphs for indexing and querying is the key issue. A subgraph G' is *frequent* if the size of its support $|D_{G'}| \geq F_{min}$, the minimum threshold of subgraph frequency.

4. SUBGRAPH MINING

In this section, we describe the algorithm to discover *independent* frequent subgraphs from a set of graphs. Then we introduce three feature selection criteria, MP, MII, and MImR, and finally, we propose a greedy algorithm to select irredundant and informative subgraph features sequentially from the discovered independent frequent subgraphs for graph indexing and search.

Algorithm 2 Independent Frequent Subgraph Mining

Algorithm: IFGM(D,F_{min},F_{max},$Corr_{max}$):
Input: Set of graphs D,
minimal and maximal threshold of frequency F_{min} and F_{max},
maximal threshold of correlation $Corr_{max}$.
Output: Set of Independent Frequent Subgraphs FG, each subgraph has a list of support graphs with corresponding frequencies.
1. **Initialization**: $FG = \{\emptyset\}$, and
   find frequent vertex set $FV = \{v | F_{min} \leq F_v \leq F_{max}\}$.
2. **for all** $v \in FV$ **do**
3. Find the set of all one-edge extensions of v, L;
4. **searchSubgraph**(v,$path$,L);
5. **return** FG;

Subprocedure: searchSubgraph(G,T,L):
Input: A graph G, its type $T \in \{path, tree, cyclic\}$, and its extension set L.
Output: FG.
1. $Dep_G = false$;
2. **for all** $l \in L$ **do**
3. $G' = G + l$ and find T';
4. Find the frequency of G' in D, $F_{G'}$;
5. **if** $Corr(G, G') \geq Corr_{max}$, **then** $Dep_G = true$;
6. **if** $Dep_G == false$, **then** put G into FG;
7. **for all** $l \in L'$ **do**
8. **if** $F_{min} \leq F_{G'} \leq F_{max}$
9. Find the set of all one-edge extensions of G', L';
10. searchSubgraph(G',T',L');

4.1 Independent Frequent Subgraph Mining

Different from previous works [15, 25, 26] using binary features of subgraphs, our proposed independent frequent subgraph mining algorithm also discovers subgraph frequencies and uses them to measure the correlation of subgraphs. Very frequent subgraphs are like stop words. They are not very informative because a large number of graphs will contain them and they do not reduce the candidate set of subgraphs on which subgraph isomorphism has to be performed significantly. We also consider very infrequent to be not very useful because we assume that queries containing these infrequent subgraphs will be infrequent. If we have a real query log and these infrequent subgraphs appear in frequent queries, this assumption can be easily relaxed without much bearing on the rest of the treatise. So, we define a lower bound and an upper bound of frequencies for frequent subgraph mining, and we remove subgraphs that have any highly *correlated* supergraphs. Thus, rather than only identifying the support graphs of each subgraph, we also find the subgraph frequencies on each support graph to compute correlation of two subgraphs G'_i, G' where $G'_i \subseteq G'$, by using the correlation of two random variables $\mathbf{G'_i}$ and $\mathbf{G'}$, i.e.,

$$Corr(G'_i, G') = \frac{Cov(\mathbf{G'_i}, \mathbf{G'})}{SD(\mathbf{G'_i})SD(\mathbf{G'})}, \quad (1)$$

where $Cov(G'_i, G')$ is the covariance of G'_i and G', $SD(G')$ is the standard deviation of G'. The random variable $\mathbf{G'}$ (similar to the random variable $\mathbf{G'_i}$) is a variable representing which graph G the subgraph G' occurs in. Its probability distribution $p(\mathbf{G'} = G)$ represents the likelihood of G' occurring in G. It is estimated using $p(\mathbf{G'} = G) =$

$F_{G' \subseteq G} / \sum_{G_i \in D} F_{G' \subseteq G_i}$. Note that unlike in previous work [25, 6], which only uses the number of support graphs, we use the correlation utilizing subgraph frequencies on support graphs. For example, if two subgraphs $G'_i \subseteq G'$ always appear on the same graphs G, previous methods only select G'. However, if $F_{G'_i \subseteq G} \gg F_{G'_i \subseteq G'} \times F_{G' \subseteq G}$, i.e., G'_i has more embeddings on G than the embeddings of G'_i in G via the embeddings of G', G'_i is still useful to index in addition with G' and thus, our algorithm indexes G'_i.

4.2 Irredundant Informative Subgraph Selection

The independent frequent subgraphs discovered by algorithm 2 can be used for graph indexing. However, the correlation in Equation (1) is used only to pre-filter highly correlated subgraphs. Partial redundancies between subgraphs still exist. Thus, we propose a feature selection approach for pruning the index further . Consider a matrix of subgraph frequencies in all graphs. Each subgraph feature G' has a list of support graphs G with a frequency $F_{G' \subseteq G}$, and correspondingly a graph has a list of subgraphs G' with $F_{G' \subseteq G}$. Then we can have the joint probability distribution $P(\mathbf{G}, \mathbf{F})$, where $\mathbf{G}$ is a random variable with outcomes of all the graphs $G \in D$, $\mathbf{F}$ is a random variable with outcomes of all the subgraph features $G' \in FG$. This joint distribution is computed using $p(G, G') = F_{G' \subseteq G}/Z$, where Z is a constant to normalize all subgraph frequencies into probabilities.

Max-Precision

Our goal is to select a set of features using which the algorithm can optimize the precision of the candidate graphs among all the retrieved candidates for all queries. Thus, given the possible user generated graph query set Q, we can find the support graphs of each query $G_q \in Q$, where each support graph is considered as *relevant* to G_q. Since the possible user generated graph query set is hard to obtain without user logs, we use a pseudo graph query set Q for feature selection that is generated randomly from the set of all the discovered subgraphs FG. Thus, the Max-Precision (MP) problem to select the optimal subgraph set S_{opt} is defined as follows:

$$S_{opt} = \arg\max_S Prec(S), \text{ where } Prec(S)$$

$$= \frac{1}{|Q|} \sum_{G_q \in Q} \frac{|D_{G_q}|}{|\bigcap_{G'_q \subseteq G_q \wedge G'_q \in S} D_{G'_q, \geq F_{G'_q \subseteq G_q}}|}$$

$$= \frac{1}{|Q|} \sum_{G_q \in Q} p(G_q \subseteq G | \forall G'_q) = \frac{1}{|Q|} \sum_{G_q \in Q} \frac{p(G_q \subseteq G, \forall G'_q)}{p(\forall G'_q)}$$

$$\approx [\prod_{G_q \in Q} \frac{p(G_q \subseteq G)}{p(\forall G'_q)}]^{1/|Q|}, \quad (2)$$

where $S = \{G'_1, G'_2, ..., G'_m\}$, $D_{G'_q, \geq F_{G'_q \subseteq G_q}}$ is the set of support graphs of G'_q where $\forall G, F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}$ (note that $D_{\emptyset, \geq F_{\emptyset \subseteq G_q}}$=D), and in this paper, we let

$$\forall G'_q = \forall G'_q, G'_q \subseteq G_q \wedge G'_q \in S, F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}.$$

Then $p(G_q \subseteq G | \forall G'_q)$ is the conditional probability that a graph G contains G_q given $F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}$ for all G'_q such that $G'_q \in G_q$, $G'_q \in S$. The last term in Equation (2) uses the geometric mean to approximate the arithmetic mean.

However, even when we have the possible user generated graph query set Q with a probability distribution of each query, finding the optimal subgraph set S that maximizes $Prec(S)$ is computationally expensive, since for each possible subset of subgraphs we have to compute $Prec(S)$. Even greedy algorithms are expensive (shown in Section 4.3 and 5.3). We desire a more time-efficient algorithm. We show below how to use an approximation method to select the set of subgraphs to index.

Max-Irredundant-Information

As mentioned in the introduction, we need to combine the informative and irredundant principles together. In order to do so, we propose a mutual-information-based strategy as follows. The mutual information (MI) $MI(X;Y)$ is a quantity to measure the dependency among two or more random variables [21, 23]. For the case of two random variables, we have

$$MI(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}.$$

Obviously, when random variables X and Y are independent, $I(X;Y) = 0$. In our case, the pair of outcomes are two different subgraphs G and G', where $G' \subset G$. Mathematically,

$$PMI(x;y) = \log \frac{p(x,y)}{p(x)p(y)}.$$

We use PMI to measure the dependence of a subgraph feature G'_q and a query graph G_q in the set of retrieved graphs G, or dependence of a pair of subgraphs, G'_i and G'_j. We call this scheme the Max-Irredudant-Information (MII) that has the following form,

$$S_{opt} = \arg\max_S(IrreduInfo(S)), \text{ where } IrreduInfo(S)$$

$$= \sum_{G_q \in Q} \log \frac{p(G_q \subseteq G, \forall G'_q)}{p(G_q \subseteq G)p(\forall G'_q)} = -\sum_{G_q \in Q} \log p(\forall G'_q). \quad (3)$$

THEOREM 1. *MII is equivalent to MP with geometric mean.*

PROOF. For MP using the geometric mean,

$$S_{opt} = \arg\max_S(Prec(S)) = \arg\max_S(\log(Prec(S)))$$

$$= \arg\max_S \sum_{G_q \in Q} [\log p(G_q \subseteq G) - \log p(\forall G'_q)].$$

$$= \arg\max_S [-\sum_{G_q \in Q} \log p(\forall G'_q)],$$

which is the same as MII. Thus, they are equivalent. □

Both MP and MII are computationally expensive, because for each possible subset of features we have to compute $IrreduInfo(S)$. MP and MII both combine the informative and irredundant principles naturally, where selected subgraphs should be informative (have a high pruning power, i.e., $|D|/|D_{G'_q, \geq F_{G'_q \subseteq G_q}}|$), but also each pair of subgraphs should be irredundant of each other. Thus, we decompose this problem into two subproblems: *Max-Information* and *Min-Redundancy*, in order to reduce the time complexity.

Max-Information

If all $G' \in S$ are independent of each other, then we have

$$IrreduInfo(S) = -\sum_{G_q \in Q} \sum_{\forall G'_q} \log p(G'_q).$$

$IrreduInfo(S)$ is the sum of each subgraph's pointwise contribution. We propose a pointwise-pruning-power-based criterion to measure the information contained by each subgraph as follows:

$$Info(G'_q) = \sum_{G_q \in Q \wedge G'_q \subseteq G_q} \log \frac{p(G_q \subseteq G, G'_q)}{p(G_q \subseteq G)p(G'_q)}$$

$$= - \sum_{G_q \in Q \wedge G'_q \subseteq G_q} \log p(G'_q)$$

$$= \sum_{G_q \in Q \wedge G'_q \subseteq G_q} \log \frac{|D|}{|D_{G'_q, \geq F_{G'_q \subseteq G_q}}|}, \quad (4)$$

where we use $p(G'_q)$ to represent $p(G'_q, F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q})$, and $D_{G'_q, \geq F_{G'_q \subseteq G_q}}$ is the support graph set where each support graph G has at least $F_{G'_q \subseteq G_q}$ embeddings of G'_q, i.e., $F_{G'_q \subseteq G} \geq F_{G'_q \subseteq G_q}$. Thus, the first goal of subgraph selection is to find a subset of subgraphs S with m subgraphs G'_i that maximizes the sum of information scores of each G'_i, called Max-Information (MI), is defined as follows:

$$\max_S Info(S), \text{ where } Info(S) = \frac{1}{m} \sum_{G'_i \in S} Info(G'_i).$$

Min-Redundancy

Using PMI, we can define the dependence of a pair of subgraphs G'_i and G'_j. Two subgraphs G'_i and G'_j are *positively dependent* if $p(G'_i|G'_j) > p(G'_j)$, *negatively dependent* if $p(G'_i|G'_j) < p(G'_j)$, and *independent* otherwise. They are *irrelevant* if they are negatively dependent or independent. Thus, we define *redundance* of two subgraphs as follows:

$$Redu(G'_i; G'_j) = \sum_{G_q \in Q \wedge G'_i \subseteq G_q \wedge G'_j \subseteq G_q} \log \frac{p(G'_i, G'_j)}{p(G'_i)p(G'_j)}.$$

If two subgraphs are irredundant, i.e., have a low redundancy score, together they are more informative than two redundant subgraphs. Thus, another goal of subgraph selection is to find a subset of subgraphs S with m subgraphs G'_i that minimizes the redundancy of the selected subgraphs, i.e., the sum of mutual information of each pair G'_i and G'_j, called Min-Redundance, defined as follows:

$$\min_S Redu(S),$$

$$\text{where } Redu(S) = \frac{2 \sum_{G'_i, G'_j \in S, i \neq j} Redu(G'_i, G'_j)}{m(m-1)}. \quad (5)$$

Max-Information-Min-Redundancy

We need to obtain Max-Information and Min-Redundancy in the selected subgraph set, but considering all the selected subgraphs together as in MP or MII is computationally expensive. Thus, we propose a global criterion that combines the two constraints, Max-Information and Min-Redundancy, and is significantly more efficient computationally, called Max-Information-Min-Redundancy (MImR), as follows:

$$S_{opt} = \arg\max_S (Info(S) - Redu(S)). \quad (6)$$

In practice, usually normal feature selection algorithms using *first-order* incremental search can be used to find the near-optimal subgraph set. Suppose we have selected $k-1$ subgraphs and want to select the next subgraph. Then the local optimal feature G'_k is selected to maximize the following function:

$$MP : \max_{G'}(Prec(S_k)), or$$

$$MII : \max_{G'}(IrreduInfo(S_k)), or$$

$$MImR : \max_{G'}(Info(S_k|S_{k-1}) - Redu(S_k|S_{k-1}))$$

$$= \max_{G'}(Info(G') - \frac{1}{k-1} \sum_{G'_i \in S_{k-1}} Redu(G', G'_i)). \quad (7)$$

Now we show that for the first-order incremental search, MImR is an approximation to MII. First we define *pointwise entropy* as $PH(x) = -\log p(x)$ and *joint pointwise entropy* as $PH(x, y) = -\log p(x, y)$. It is easy to verify that

$$IrreduInfo(S) = \sum_{G_q \in Q} [PH(G_q \subseteq G) + PH(\forall G'_q) - PH(G_q \subseteq G, \forall G'_q)] \quad (8)$$

We define *pointwise total correlation* $PC(S)$ as follows:

$$PC(S) = \sum_{G_q \in Q} \log \frac{p(\forall G'_q)}{\prod_{G'_q} p(G'_q)}$$

$$= \sum_{G_q \in Q} [\sum_{G'_q} PH(G'_q) - PH(\forall G'_q)] \quad (9)$$

and $PC(S, Q)$ as follows:

$$PC(S, Q) = \sum_{G_q \in Q} \log \frac{p(G_q \subseteq G, \forall G'_q)}{p(G_q \subseteq G) \prod_{G'_q} p(G'_q)}$$

$$= \sum_{G_q \in Q} [PH(G_q \subseteq G) + \sum_{G'_q} PH(G'_q) - PH(G_q \subseteq G, \forall G'_q)] \quad (10)$$

Then by subtracting (9) from (10) and substituting the difference into (8) we have

$$IrreduInfo(S) = PC(S, Q) - PC(S). \quad (11)$$

Thus, MII is equivalent to simultaneously maximizing the first term and minimizing the second term at the left hand side of Equation (11).

It is easy to show that the first term,

$$PC(S, Q) = \sum_{G_q \in Q} [\sum_{G'_q} PH(G'_q) - PH(\forall G'_q | G_q \subseteq G)] \leq \sum_{G_q \in Q} \sum_{G'_q} PH(G'_q),$$

is maximized only if all the variables in $\{S, Q\}$ are the most dependent. Thus, if all $m-1$ subgraphs in S have been selected, the mth subgraph that is the most dependent on Q should be selected for MII, because it can maximize $PC(S, Q)$. Note that this is the same as the Max-Info strategy. The second term $PC(S) > 0$ if subgraphs are positively dependent, < 0 if negatively dependent, and $= 0$ if all the subgraphs are independent. Thus, if all $m-1$ subgraphs in S have been selected, the mth subgraph that is the most pairwise negatively dependent on each selected subgraph in S should be selected for MII, which can minimize $PC(S)$. This is same as the Min-Redu strategy. Thus, MImR is a combination of Max-Info and Min-Redu and an approximation to MII.

Algorithm 3 Irredundant Informative Subgraph Selection

Algorithm: IIGS(FG, m):
Input:
Candidate set of subgraphs FG, and
Number of features to select m.
Output:
Set of Irredundant Informative Subgraphs S.
1. **Initialization**: $S = \{\emptyset\}$, $k = 1$.
2. **while** $k \geq m$, do
3. scan all $G' \in FG$ and
find $G_{opt} = \arg\max_{G'}(Equation(7))$;
4. move G_{opt} from FG to S;
5. $k++$;
7.**return** $IIFG$;

4.3 Subgraph Selection Algorithm

It is easy to shown that finding the optimal solution for the MP or MII problem is as expensive as:

$$O(\frac{n!}{m!(n-m)!}|Q| \cdot avg|s_q| \cdot (|D| + avg|D_{G'_q \in s_q, \geq F_{G'_q \subseteq G_q}}|)),$$

where $s_q = \{G'_q | G'_q \in S, G'_q \subseteq G_q \in Q\}$ is the set of possible subgraphs in G_q and $avg|s_q|$ is the average size of s_q.

We use forward selection in this work (Algorithm 3). Forward selection is a greedy algorithm using *first-order* incremental search, i.e., every time only the best subgraph from the rest of the candidate subgraph set is added to the selected subgraph set. Initially, the algorithm finds the most informative subgraph. Then, every time when a new subgraph is added, the rest of the subgraphs in the candidate set are evaluated. The one that maximizes Equation (7) is selected. The algorithm repeats this until m subgraphs are selected.

For MP and MII, computational complexity of the first-order incremental selection is

$$O(n^2|Q| \cdot avg|s_q| \cdot (|D| + avg|D_{G'_q \in s_q, \geq F_{G'_q \subseteq G_q}}|)).$$

For MImR, the computational complexity of the first-order incremental selection involves three parts,

1) $O(|Q| \cdot avg|s_q| \cdot |D|)$

for pre-computing information scores of all features,

2) $O(|Q| \cdot (avg|s_q| \cdot |D| + avg|s_q|^2 \cdot avg|D_{G'_q \in s_q, \geq F_{G'_q \subseteq G_q}}|))$

for pre-computing pairwise dependence scores of all feature pairs, and

3) $O(n^2)$

for subgraph selection, which is much faster than that of MP or MII. Because $avg(|s_q|)$ can be viewed as a constant compared with other numbers, forward subgraph selection based on MImR is quadratic, while for MP or MII it is quartic. Another advantage of the *first-order* incremental search is that we only need to run the algorithm once to select m subgraphs, and we know what are the best $k \leq m$ subgraphs without re-running the algorithm every time when the number of selected subgraphs is changed.

5. EXPERIMENTAL EVALUATION

In this section, we evaluate our proposed approaches and compare the experimental results with two previous methods. Our results show that:

- Our proposed subgraph selection and graph search method can improve the precision of return candidates (number of graphs containing the query subgraph/number of graphs on which on which subgraph isomorphism has to be computed) by about 4%-13% when compared to existing methods. Correspondingly, the overall response time for online subgraph queries is improved.
- The algorithm based on MImR has a much lower computational cost than those based on MP and MII, and MImR can achieve a reasonably high precision that is only slightly lower than that of MP or MII.

5.1 Experimental Data Set

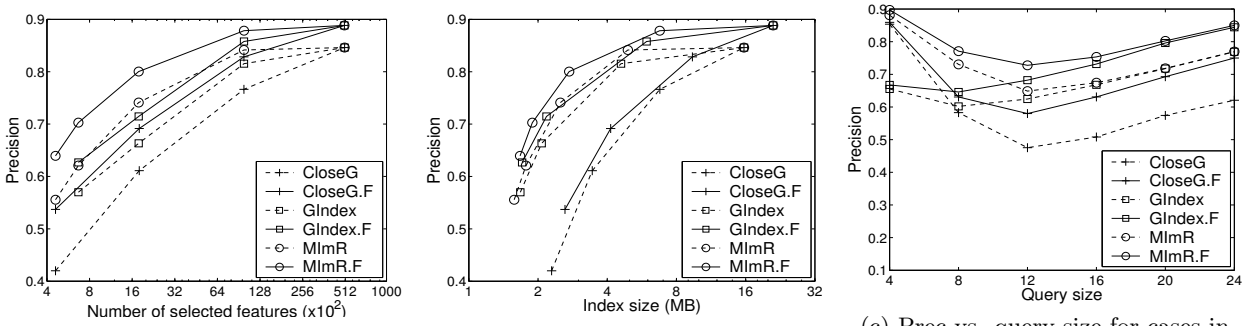
We use the same real data set and testing query set as those used by Yan, et al., [26]. It is a NCI/NIH HIV/AIDS antiviral screen data set that contains 43,905 chemical structures. The experimental subset contains 10,000 chemical structures randomly selected from the whole set and the query set contains 6000 randomly generated queries, i.e., 1000 queries per $Size(G_q) = \{4, 8, 12, 16, 20, 24\}$. Although we only use chemical structures for experiments, our approach is applicable to any structures that can be represented by graphs, such as DNA sequences or XML files.

5.2 Evaluated Subgraph Selection Methods

We evaluate average precisions of returned graphs for all queries using different subgraph selection methods. When we compute the average precision of returned structures, we only count queries with non-empty supports in the data set. In our experiment, we evaluate six methods. First, we evaluate three methods without considering subgraph frequencies, including two previous methods, *CloseG* [25] and *GIndex* [26], and our proposed method, *MImR*. They use *binary features* that only consider the occurrence of subgraphs in graphs. Each subgraph feature takes a binary value of 1 or 0 to represent the occurrence of a subgraph in a graph or not, respectively. Then we evaluate three methods considering subgraph frequencies. We extend the *CloseG* method to *CloseG.F*, the *GIndex* method to *GIndex.F*, and propose a *MImR.F*, which is an extension of our *MImR* method, respectively. They use numerical features of subgraph frequencies in graphs, i.e., *frequency features*. *CloseG* and *CloseG.F* [25] just select *frequent closed subgraphs* independently that have $D_G \geq F_{min} = \{1000, 500, 200, 100\}$ ($m = \{460, 1795, 9846, 50625\}$ respectively) without considering redundancy. *GIndex* and *GIndex.F* [26] select subgraphs from the candidate subgraph set where $D_G \geq 100$. It scans subgraphs from the smallest to the largest avoiding to select redundant supergraphs of selected subgraphs. A discriminative score is defined and computed for each subgraph candidate to measure the redundancy. If its score is larger than a threshold $D_{min} = \{7.0, 3.1, 1.09, 1\}$ ($m = \{667, 1779, 9855, 50625\}$ respectively) then the subgraph is selected. *MImR.F* is our proposed method in Section 4.2 and *MImR* is similar but uses binary features. Five variations on the number of selected subgraphs are evaluated, $m = \{460, 667, 1779, 9855, 50625\}$. *MP* and *MII* are only evaluated using the data set with 100 structures because they are forbiddingly expensive in practice. We randomly sample 30,000 subgraphs as the training query set with the same distributions as that of the testing query set, and then use it for subgraph selection in *MImR* and *MImR.F*.

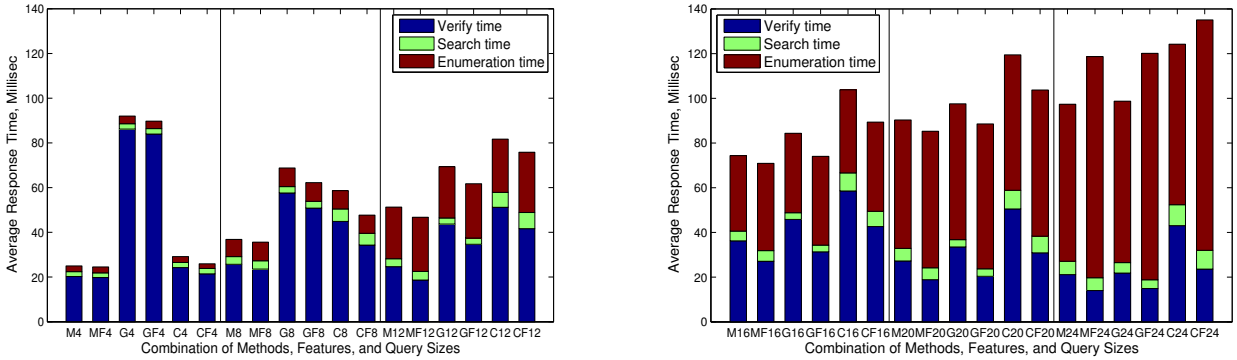
5.3 Precision of Returned Results

We show precision of returned results for the testing query set in Figure 2. Because each subgraph selection method



(a) Prec vs. # selected feature (b) Prec vs. index size (c) Prec vs. query size for cases in Table 2

Figure 2: Average precision of graph search for subgraph queries



(a) Small query sizes (4, 8, 12) (b) Large query sizes (16, 20, 24)

Note: For combination of Methods, Features, and Query Sizes, M=MImR, G=GIndex, C=CloseG, MF=MImR.F, GF=GIndex.F, CF=CloseG.F, 4=Query size of 4. For example, M4 is MImR using binary features for queries with size of 4, and CF12 is CloseG.F using frequency features for queries with size of 12.

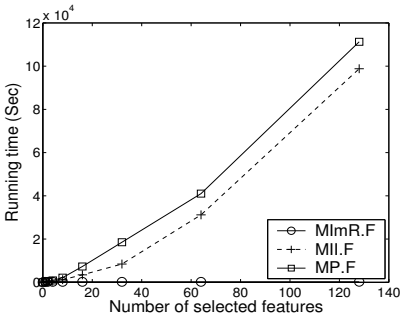
Figure 3: Response time of subgraph queries for cases in Table 2

can select different numbers of subgraphs for indexing by adjusting parameters, we show the curves of precision versus different values of the selected subgraph number m and the index size in Figure 2. We also present the precision curve versus the query graph size $|G_q|$ to illustrate the effect of different query sizes in Figure 2. To evaluate the precision versus the index size, we first index the canonical string of each selected subgraph. If a subgraph has a frequency larger than one, we then index the frequency. Thus, using numerical features has a larger index size than using binary features given the same number of selected subgraphs.

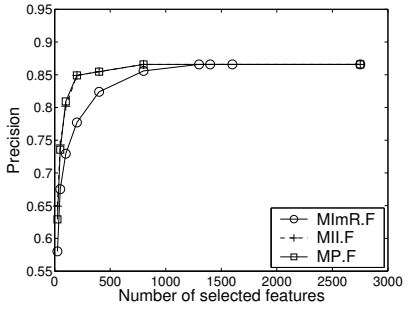
In Figure 2 (a), we can observe that given the same number of selected subgraphs, *GIndex* improves the average precision compared with *CloseG*. Our proposed approach *MImR* can improve the precision by about 4%-13%. This illustrates that our probabilistic model for subgraph selection works better than the previous method proposed by Yan, et al., [26]. We also can see that *CloseG.F*, *GIndex.F*, and *MImR.F* can improve the average precision compared with *CloseG*, *GIndex*, and *MImR* by using subgraph frequencies as features (Figure 2 (a)). This demonstrates that using subgraph frequencies for subgraph queries can improve the precision by about 4%-12%. In Figure 2 (b), we can see that using subgraph frequencies as features will increase the index size by about 6%-30%, since more information of frequencies are indexed. We can see that given the same index size, *CloseG.F*, *GIndex.F*, and *MImR.F* also have higher precision than *CloseG*, *GIndex*, and *MImR*, even though not as much as that in the cases in Figure 2 (a). In Figure 2 (c), the curves are of the precisions vs. query sizes for the cases in Table 2. We can observe that *CloseG* and *CloseG.F* have higher precision for small queries, while *GIndex* and *GIndex.F* have higher precision for large queries. *MImR* and *MImR.F* are more balanced than either and always have precisions above their precisions. The p-values in Table 2 of 1-sided T-tests show that the improvement is significant with a confidence level of at least 99.9%.

5.4 Response Time of Subgraph Queries

In this section, we show the overall response time for subgraph queries using different subgraph selection methods. The search process to answer subgraph queries works as follows. Every time when a subgraph query is entered, the algorithm first generated the canonical string of the query graph and check if this query graph is indexed. If the query is indexed, the support of the query is retrieved without verification using subgraph isomorphism. If the query is not indexed, its subgraphs are enumerated, and among those subgraphs, indexed ones are use to scan the index and find candidate sets for each subgraph and compute the intersection for all candidate sets. Finally, subgraph isomorphism



(a) Time vs. # selected feature



(b) Prec vs. # selected feature

Figure 5: Comparison of MP, MII, and MImR

Method	#Feature	Index size	Precision
CloseG	1795	3.44MB	61.10%
CloseG.F	1795	4.14MB	69.15%
GIndex	1779	2.07MB	66.32%
GIndex.F	1779	2.18MB	71.47%
MImR	1779	2.50MB	74.11%
MImR.F	1779	2.73MB	80.03%

Methods	P-value
GIndex vs CloseG	0.000
MImR vs GIndex	0.000
GIndex.F vs CloseG.F	0.001
MImR.F vs GIndex.F	0.000
CloseG.F vs CloseG	0.000
GIndex.F vs GIndex	0.000
MImR.F vs MImR	0.000

Table 2: Average precision and 1-sided T-test for feature selection methods

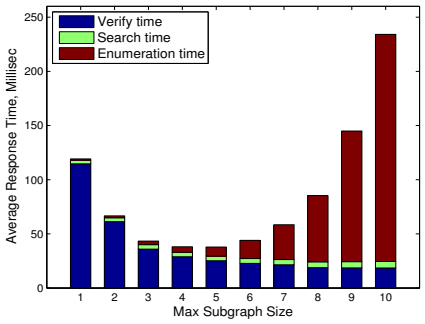


Figure 4: Effect of max enumerated subgraph size on response time for query size of 20 & MImR.F

tests are performed on all retrieved candidates to prune out unsatisfied graphs.

Thus, response time of a subgraph query involves 1) *enumeration time* to identify if the query graph is indexed or not, and enumerate the subgraphs of the query if the query is not indexed (note that enumeration time also includes the time of finding canonical strings of subgraphs), 2) *search time* to retrieve candidates from the graph index, and 3) *verification time* to prune out unsatisfied graphs that are not super-graphs of the query from the candidates using subgraph isomorphism tests. Enumeration time first depends on how many queries are indexed and then depends on query sizes. Usually large queries have long enumeration time. Search time includes time to retrieve candidate sets, and time to compute the set intersection. If the query is indexed, then no set intersection is computed. Moreover, if the query is indexed, verification time is zero. Otherwise, it depends on the number of returned candidates. Thus, if the precision of returned candidates is higher, then verification time is shorter.

We illustrate average response times for the cases in Table 2, in Figure 3. We set the max enumerated subgraph size of queries as 8. From Figure 3, we can observe that the average over-all response time of subgraph queries using MImR is always the best in comparison with the other two methods, GIndex and CloseG, for different query sizes, irrespective of whether binary features or frequency features are used in the index and the search process. However, the improvement of response time is significant for small queries, while for large queries, it is not significant, because the enumeration time dominates the over-all response time. We also can observe that the search time is always a small part of the over-all response time, while enumeration time and verification time change much for different cases and affect the over-all response time significantly. Usually for small queries, the major part of response time is the verification time, while for large queries, the major part is enumeration time. This is because small queries usually have large supports containing more supergraphs so that as a result the candidate set is also very large to verify, although each subgraph isomorphism test for small queries is not expensive. In comparison, large queries usually have small candidates to verify, but the enumeration time is expensive for large queries, because it increases exponentially when the query size increases.

Similar to precision curves in Figure 2 (c), we can observe that 1) for GIndex and GIndex.F, the verification time is long for small queries but short for large queries, 2) for CloseG and CloseG.F, the verification time is short for small queries but long for large queries, 3) for MImR and MImR.F, the verification time is short for all queries. Moreover, using frequency features can achieve a shorter verification time than using binary features for all the cases. This is consistent with the precision curves in Figure 2 (c). Although using frequency features can achieve a shorter verification time than using binary features, it requires a longer enumeration time because the exact occurrence number of each subgraph is needed to be identified for frequency features. Thus, the over-all response time is shorter using binary features than using frequency features for large queries, because the enumeration time dominates the response time for large queries. However, the over-all response time is shorter using frequency features than binary features for small queries, because the verification time contributes more than the enumeration time to the response time.

Because the subgraph enumeration process uses Algorithm 3, we can set the maximum subgraph size to enumerate. If we set a smaller maximum subgraph size, we can expect a shorter enumeration time but a longer verification time, because fewer subgraphs are used to retrieve candidates. We adjust the maximum subgraph size and evaluate MImR.F with the query size of 20, and show the results in Figure 4. We can observe that for queries with graph size of 20, if we use MImR.F, the optimal maximum subgraph size for subgraph enumeration is 5. Below that, verification time increases significantly, while above that, enumeration time increases much. Thus, we can achieve better response times than those in Figure 3, if we tune and find the best maximum subgraph size for subgraph enumeration.

5.5 Time Complexity of Subgraph Selection Methods

To compare the time complexity for our proposed methods, *MP.F*, *MII.F*, and *MImR.F*, we select a data set with 100 chemical structures. We use a testing set of 100 queries with 20 queries per $Size(G_q) = \{4, 8, 12, 16, 20\}$, and a training set of 500 queries with the same distribution. We use the forward feature selection for the three methods. We show the time complexity and the average precision of query answers in Figure 5. The results demonstrate that *MImR.F* is significantly more computationally efficient than *MP.F* and *MII.F*. *MImR.F* achieves only a slightly worse precision in comparison to *MP.F* or *MII.F*. Note that *MP.F* and *MII.F* may not achieve the optimal performance since we use forward feature selection. To find the better solution using a global method of feature selection, we expect more significant computational costs. We can observe from Figure 5 (b) that the precision increases while more subgraphs are selected. After a certain number of subgraphs have been selected, the precision reaches the highest value, i.e., adding more subgraphs cannot increase the information contained in the subgraph set for the testing query set. Thus, if the user query set is known, we can find this point and stop adding useless subgraphs to the index after this point. A better subgraph selection method can achieve the highest precision with a smaller number of subgraphs. Note that there is no overfitting problem for the task of subgraph queries. This is because in the subgraph query problem, given that a query graph is a subgraph of a graph, all the subgraph features of this query graph are always subgraphs of the same graph. Adding more features can always prune out more (at least zero) unsatisfied candidates and improve (at least maintain) the precision of subgraph queries.

6. CONCLUSIONS AND FUTURE WORK

We proposed a novel probabilistic model to determine a near-optimal set of subgraphs to index in order to answer a given set of queries given a constraint on the maximum number of subgraphs that can be indexed (typically due to space and performance limitations). We consider subgraph frequencies in graphs to improve the precision of our method further. We introduce several criteria for subgraph selection, including Max-Precision (MP), a method that directly optimizes the precision of query answers, Max-Irredundant-Information (MII) and Max-Information-Min-Redundancy (MImR) that are based on a probabilistic model using mutual information. We show theoretically that MImR and MII are approximations of MP. We also propose a greedy feature selection algorithm using MImR that works well in practice. Experiments show that our proposed approaches perform significantly better than previous methods. Future work will use real queries from user logs to select features and evaluate precision.

7. ACKNOWLEDGMENTS

We acknowledge the partial support of NSF Grant 0535656 and 0845487.

8. REFERENCES

[1] B. Berendt. Using and learning semantics in frequent subgraph mining. In *Proc. WEBKDD*, 2005.

[2] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proc. ICDM*, 2002.

[3] S. Buttcher and C. L. A. Clarke. A document-centric approach to static index pruning in text retrieval systems. In *Proc. CIKM*, 2006.

[4] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. Maarek, and A. Soffer. Static index pruning for information retrieval systems. In *Proc. SIGIR*, 2001.

[5] B. Chen, Q. Zhao, B. Sun, and P. Mitra. Temporal and social network based blogging behavior prediction in blogspace. In *Proc. ICDM*, 2007.

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

[7] E. S. de Moura, C. F. dos Santos, D. R. Fernandes, A. S. Silva, P. Calado, and M. A. Nascimento. Improving web search efficiency via a locality based static pruning method. In *Proc. WWW*, 2005.

[8] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In *Proc. SIGKDD*, 1998.

[9] I. Fischer and T. Meinl. Graph based molecular data mining - an overview. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[10] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomophism. In *Proc. ICDM*, 2003.

[11] A. Inokuchi. Mining generalized substructures from a set of labeled graphs. In *Proc. ICDM*, 2004.

[12] H. Jiang, H. Wang, P. S. Yu, and S. Zhou. Gstring: A novel approach for efficient search in graph databases. In *Proc. ICDE*, 2007.

[13] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proc. ICDM*, 2001.

[14] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[15] S. Nijssen and J. N. Kok. A quickstart in frequent structure mining can make a difference. In *Proc. SIGKDD*, 2004.

[16] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on PAMI*, 27(8):1226–1238, 2005.

[17] J. W. Raymond, E. J. Gardiner, and P. Willet. Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6), 2002.

[18] D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proc. PODS*, 2002.

[19] S. Srinivasa and S. Kumar. A platform based on the multi-dimensional data model for analysis of bio-molecular structures. In *Proc. VLDB*, pages 975–986, 2003.

[20] B. Sun, P. Mitra, and C. L. Giles. Mining, indexing, and searching for textual chemical molecule information on the web. In *Proc. WWW*, 2008.

[21] B. Sun, P. Mitra, H. Zha, C. L. Giles, and J. Yen. Topic segmentation with shared topic detection and alignment of multiple documents. In *Proc. SIGIR*, 2007.

[22] B. Sun, Q. Tan, P. Mitra, and C. L. Giles. Extraction and search of chemical formulae in text documents on the web. In *Proc. WWW*, 2007.

[23] B. Sun, D. Zhou, H. Zha, and J. Yen. Multi-task text segmentation and alignment based on weighted mutual information. In *Proc. CIKM*, 2006.

[24] M. Worlein, T. Meinl, I. Fischer, and M. Philippsen. A quantitative comparison of the subgraph miners mofa, gspan, ffsm, and gaston. In *Proc. PKDD*, 2005.

[25] X. Yan and J. Han. Closegraph: Mining closed frequent graph patterns. In *Proc. SIGKDD*, 2003.

[26] X. Yan, P. S. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *Proc. SIGMOD*, 2004.

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

[28] P. Zhao, J. X. Yu, and P. S. Yu. Graph indexing: Tree + delata >= graph. In *Proc. VLDB*, 2007.

[29] Q. Zhao, L. Chen, S. S. Bhowmick, and S. Madria. Xml structural delta mining: issues and challenges. *Data and Knowledge Engineering*, 2006.