Near-optimal supervised feature selection among frequent subgraphs

Marisa Thoma* Hong Cheng† Arthur Gretton‡ Jiawei Han§ Hans-Peter Kriegel*
Alex Smola¶ Le Song‖ Philip S. Yu** Xifeng Yan†† Karsten Borgwardt‡‡

Abstract

Graph classification is an increasingly important step in numerous application domains, such as function prediction of molecules and proteins, computerised scene analysis, and anomaly detection in program flows.

Among the various approaches proposed in the literature, graph classification based on frequent subgraphs is a popular branch: Graphs are represented as (usually binary) vectors, with components indicating whether a graph contains a particular subgraph that is frequent across the dataset.

On large graphs, however, one faces the enormous problem that the number of these frequent subgraphs may grow exponentially with the size of the graphs, but only few of them possess enough discriminative power to make them useful for graph classification. Efficient and discriminative feature selection among frequent subgraphs is hence a key challenge for graph mining.

In this article, we propose an approach to feature selection on frequent subgraphs, called *CORK*, that combines two central advantages. First, it optimizes a submodular quality criterion, which means that we can yield a near-optimal solution using greedy feature selection. Second, our submodular quality function criterion can be integrated into gSpan, the state-of-the-art tool for frequent subgraph mining, and help to prune the search space for discriminative frequent subgraphs even *during* frequent subgraph mining.

1 Introduction.

A typical graph classification problem has the following formulation: given a set of training graphs associated with labels $\{G_i, y_i\}_{i=1}^n$, $y_i \in \{\pm 1\}$, the task is to learn a classifier that predicts the labels of unclassified structures. The resulting classification algorithm has a wide variety of real world applications.

In biology and chemistry, for example, graph classification quantitatively correlates chemical structures with biological and chemical processes, such as active or inactive to virus, toxic or non-toxic to human beings [19]. This makes graph classification scientifically and commercially valuable (eg. in searching for new drugs). In computer vision, images can be abstracted as graphs, where nodes are spatial entities and edges are their mutual relationships. A classical recognition system can query ARG model databases by the scene and identify the type of foreground objects. In software engineering, a program can also be modeled as a graph, where program blocks are nodes and flows of the program are edges. Static and dynamic analysis of program behaviors can then be carried out in these graphs. For instance, anomaly detection of control flows is essentially a graph classification problem.

Recent studies have proposed several effective graph classification methods, including substructure (or so-called fragment) based approaches [17, 8, 31], kernel-based approaches [16, 22, 12], and boosting methods [20]. In the substructure based approach, each graph is treated as a set of subgraph features, eg. acyclic substructures [31] and frequent substructures [17, 8, 7], which share the same concept as a convolution kernel. In [7], the framework of frequent pattern-based classification is examined, where a classification model is built in the feature space of frequent itemsets. There are two reasons for adapting frequent graphs in graph classification. First, it is computationally difficult to enumerate all of the substructures existing in a large graph dataset, while it is possible to mine frequent patterns due to the recent development of efficient graph mining algorithms. Second, the discriminative power of very infrequent substructures is bounded by a low value due to their limited coverage in the dataset. Therefore, it is a good approximation to take frequent substructures as features to build classification models.

However, the vast number of substructures poses three new challenges.

1. Redundancy: Most frequent substructures are slightly different from each other in structure and are highly correlated with the class labels.

2. Significance: While low-support features are not representative of the graphs, frequent subgraphs

*Institute for Informatics, Ludwig-Maximilians-Universität München

†Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong

‡Max Planck Institute for Biological Cybernetics, Tübingen

§University of Illinois at Urbana-Champaign

¶Yahoo! Research, Santa Clara, California

‖School of Computer Science, Carnegie Mellon University

**University of Illinois at Chicago, Chicago, Illinois

††Department of Computer Science, University of California at Santa Barbara

‡‡University of Cambridge and Max-Planck-Institutes for Developmental Biology and Cybernetics, Tübingen

are not necessarily useful for classification. Statistical significance rather than frequency of the graph patterns should be used for evaluation of their discriminative power.

3. Efficiency: Very frequent subgraphs are not useful since they are not discriminative between classes. Therefore, frequent subgraph based classification usually sets up a pretty low frequency threshold, resulting in thousands or even millions of features. Given such a tremendous number of features, a complicated feature selection mechanism is likely to fail.

Consequently, we need an efficient algorithm to select discriminative features among a large number of frequent subgraphs. In earlier work [7], we adopted a heuristic approach and demonstrated that it could outperform methods using low dimensional features.

Goal Our goal is to define an efficient near-optimal approach to feature selection among frequent subgraphs generated by gSpan [34]. The key idea is to pick frequent subgraphs that greedily maximise a *submodular* quality criterion, thereby guaranteeing that the greedy solution to the feature selection problem is close to the global optimal solution [23]. To make this approach efficient, we integrate it into gSpan, the state-of-the-art tool for frequent subgraph mining, and derive pruning criteria that allow us to narrow down the search space when looking for discriminative subgraphs.

Unlike its predecessors that use heuristic strategies for feature selection (such as [7]) or do not provide optimality guarantees, we define a principled, near-optimal approach to feature selection on frequent subgraphs that can be integrated into gSpan [34]. An excellent wrapper approach to this problem has recently been published by [28]. Our approach differs from Tsuda's in two ways: Our feature selection method is independent from the choice of the classifier (filter method) and we can provide optimality guarantees for our solution.

2 Near-optimal feature selection among frequent subgraphs

In the following we will first define the optimization problem we want to tackle (Section 2.1) and then we will review the essential ingredients of our graph feature selector: first, submodularity and its use in feature selection (Section 2.2); second, gSpan, the method to find frequent subgraphs (Section 2.3).

Notation We are given a dataset $\mathcal{G} = \mathcal{A} \cup \mathcal{B}$ of graphs that each belong to one of two classes $\mathcal{A}$ or $\mathcal{B}$.

As a notational convention, the *vertex set* of a graph $G \in \mathcal{G}$ is denoted by $V(G)$ and the *edge set* by $E(G)$. A label function, l, maps a vertex or an edge to a label. A graph G is a subgraph of another graph G' if there exists a subgraph isomorphism from G to G', denoted by $G \sqsubseteq G'$. G' is called a super-graph of G.

Definition 2.1. (Subgraph Isomorphism) *A subgraph isomorphism is an injective function $f : V(G) \to V(G')$, such that*

1. $\forall u \in V(G)$, $l(u) = l'(f(u))$, and

2. $\forall (u, v) \in E(G)$, $(f(u), f(v)) \in E(G')$ and $l(u, v) = l'(f(u), f(v))$,

where l and l' are the label function of G and G', respectively. f is called an embedding of G in G'.

Given a graph database $\mathcal{G}$, $\mathcal{G}_{G_1}$ is the number of graphs in $\mathcal{G}$ where G is a subgraph and $\mathcal{G}_{G_0}$ is the number of graphs in $\mathcal{G}$ of which G is *not* a subgraph. $\mathcal{G}_{G_1}$ is called the *(absolute) support*, denoted by *support*(G). A graph G is *frequent* if its support is no less than a minimum support threshold, min_sup. As one can see, the frequent graph is a relative concept: whether or not a graph is frequent depends on the value of min_sup.

2.1 Combinatorial optimization problem

The problem of feature selection among frequent subgraphs can be cast as a combinatorial optimization problem. We denote by $\mathcal{S}$ the full set of features, which in our case corresponds to the frequent subgraphs generated by gSpan. We use these features to predict class membership of individual graph instances: clearly, only a subset $\mathcal{T} \subseteq \mathcal{S}$ of features will be relevant. We denote the relevance of a set of frequent subgraphs for class membership by $q(\mathcal{T})$, where q is a criterion measuring the discriminative power of $\mathcal{T}$. It is computed by restricting the graphs to the features in $\mathcal{T}$. Feature selection can then be formulated as:

$$\mathcal{S}^\dagger = \arg\max_{\mathcal{T} \subseteq \mathcal{S}} q(\mathcal{T}) \qquad \text{s.t.} \quad |\mathcal{T}| \leq s \tag{2.1}$$

where $|\cdot|$ computes the cardinality of a set and s is the maximally allowed number of selected features.

Unfortunately, solving this problem optimally requires us to search all possible subsets of features exhaustively. The common remedy is to resort to heuristic alternatives, the solutions of which cannot be guaranteed to be globally optimal or even close to the global optimal solution. Hence the key point in this article is to employ a heuristic approach which *does* allow for these quality guarantees, namely a greedy strategy which achieves *near-optimal* results.

2.2 Feature Selection and Submodularity

Assume that we are measuring the discriminative power

$q(\mathcal{S})$ of a set of frequent subgraphs $\mathcal{S}$ in terms of a quality function q. A near-optimality solution is reached for a *submodular* quality function q when used in combination with greedy feature selection. Greedy forward feature selection consists in iteratively picking the feature that – in union with the features selected so far - maximises the quality function q over the prospective feature set. In general, this strategy will not yield an optimal solution, but it can be shown to yield a near-optimal solution if q is submodular:

Definition 2.2. (Submodular set function) *A quality function q is said to be* **submodular** *on a set $\mathcal{D}$ if for $\mathcal{T}' \subset \mathcal{T} \subseteq \mathcal{D}$ and $X \in \mathcal{D}$*

$$q(\mathcal{T}' \cup \{X\}) - q(\mathcal{T}') \geq q(\mathcal{T} \cup \{X\}) - q(\mathcal{T}) \qquad (2.2)$$

If q is submodular and we employ greedy forward feature selection, then we can exploit the following theorem from [23]:

Theorem 2.1. *If q is a submodular, nondecreasing set function on a set $\mathcal{D}$ and $q(\emptyset) = 0$, then greedy forward feature selection is guaranteed to find a set of features $\mathcal{T} \subseteq \mathcal{D}$ such that*

$$q(\mathcal{T}) \geq \left(1 - \frac{1}{e}\right) \max_{\mathcal{U} \subseteq \mathcal{D}:\ |\mathcal{U}|=s} q(\mathcal{U}) \ , \qquad (2.3)$$

where s is the number of features to be selected.

As a direct consequence, the result from greedy feature selection achieves at least $\left(1 - \frac{1}{e}\right) \approx 63\%$ of the score of the optimal solution to the feature selection problem. This is referred to as being *near-optimal* in the literature (e.g. [13]).

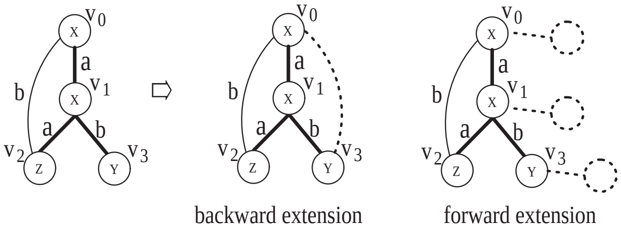

Figure 1: gSpan: Rightmost Extension

2.3 gSpan

If we found a useful submodular criterion for feature selection on frequent subgraphs, we could yield a near-optimal solution to problem (2.1). But how do we determine the frequent subgraphs in the first place? For this purpose, we use the frequent subgraph algorithm gSpan [34], which we introduce in the following.

The discovery of frequent graphs usually consists of two steps. In the first step, we generate frequent subgraph candidates, while in the second step, we check the frequency of each candidate. The second step involves a subgraph isomorphism test, which is NP-complete. Fortunately, efficient isomorphism testing algorithms have been developed, making such testing affordable in practice. Most studies of frequent subgraph discovery pay attention to the first step; that is, how to generate as few frequent subgraph candidates as possible, and as fast as possible.

The initial frequent graph mining algorithms, such as AGM [15], FSG [21] and the path-join algorithm [30], share similar characteristics with the Apriori-based itemset mining [1]. All of them require a join operation to merge two (or more) frequent substructures into one larger substructure candidate. To avoid this overhead, non-Apriori-based algorithms such as gSpan [34], MoFa [2], FFSM [14], and Gaston [24] adopt the pattern-growth methodology, which attempts to extend graphs from a single subgraph directly. For each discovered graph G, these methods recursively add new edges until all the frequent supergraphs of G have been discovered. The recursion stops once no frequent graph can be generated any more.

gSpan introduced a sophisticated extension method, which is built on depth first search (DFS) tree. Given a graph G and a DFS tree T, we call the starting vertex in T, v_0, the $r$oot, and the last visited vertex, v_n, the $r$ightmost vertex. The straight path from v_0 to v_n is called the $r$ightmost path. Figure 1 shows an example. The darkened edges form a DFS tree. The vertices are discovered in the order v_0, v_1, v_2, v_3. The vertex v_3 is the rightmost vertex. The rightmost path is $v_0 \sim v_1 \sim v_3$.

This method, called rightmost extension, restricts the extension of new edges in a graph as follows: Given a graph G and a DFS tree T, a new edge e can be added between the rightmost vertex and other vertices on the rightmost path (*backward extension*); or it can introduce a new vertex and connect to vertices on the rightmost path (*forward extension*). If we want to extend the graph in Figure 1, the backward extension candidate can be (v_3, v_0). The forward extension candidates can be edges extending from v_3, v_1, or v_0 with a new vertex introduced. Since there could be multiple DFS trees for one graph, gSpan establishes a set of rules to select one of them as representative so that the backward and forward extensions will only take place on one DFS tree.

Overall, new edges are only added to the vertices along the rightmost path. With this restricted extension, gSpan reduces the generation of the same graphs. However, it still guarantees the completeness of enumerating all frequent subgraphs. For a detailed description of gSpan, see [34]. Algorithm 2.1 outlines the pseudocode of gSpan. $G \diamond_r e$ means that an edge is extended from graph G using backward or forward extension. $G \neq \mathit{dfs}(G)$ checks whether G has been discovered before, where $\mathit{dfs}(G)$ is the canonical form of graph G [34].

ALGORITHM 2.1. gSpan(G, $\mathcal{G}$, min_sup, $\mathcal{S}$)

Input: A graph G, a graph dataset $\mathcal{G}$, and min_sup.
Output: The set of frequent subgraphs $\mathcal{S}$.

```
if $G \neq dfs(G)$, then
    return;
insert $G$ into $\mathcal{S}$;
set $C$ to $\emptyset$;
scan $\mathcal{G}$ once, find all the edges $e$ such that $G$ can be
  rightmost extended to $G \diamond_r e$;
  insert $G \diamond_r e$ into $C$ and count its frequency;
for each frequent $G \diamond_r e$ in $C$ do
    Call gSpan($G \diamond_r e$, $\mathcal{G}$, min_sup, $\mathcal{S}$);
return;
```

Once we have determined the frequent subgraphs using gSpan, a natural way of representing each graph G is in terms of a binary indicator vector of length $|\mathcal{S}|$:

DEFINITION 2.3. (INDICATOR VECTOR) *Given a graph G_j from a dataset $\mathcal{G}$ and a set of frequent subgraph features $\mathcal{S}$ discovered by gSpan. We then define an indicator vector $v^{(j)}$ for G_j as*

$$v_i^{(j)} = \begin{cases} 1 & \text{if } \mathcal{S}_i \sqsubseteq G_j \ (\mathcal{S}_i \text{ is a subgraph of } G_j) \\ 0 & \text{otherwise} \end{cases}, \tag{2.4}$$

where $v_i^{(j)}$ is the i-th component of $v^{(j)}$ and $\mathcal{S}_i$ is the i-th graph in $\mathcal{S}$.

2.4 Definition of CORK

DEFINITION 2.4. *Let $\mathcal{G}$ be a dataset of binary vectors, consisting of two classes $\mathcal{G} = \mathcal{A} \cup \mathcal{B}$. Let $\mathcal{D}$ denote the set of features of the data objects in $\mathcal{G}$, and let X be a single feature from $\mathcal{D}$,* i.e., $X \in \mathcal{D}$.

DEFINITION 2.5. (CORRESPONDENCE) *A pair of data objects $(v^{(i)}, v^{(j)})$ is called a* **correspondence** *in a set of features indicated by indices $\mathcal{U} \subseteq \{1, \ldots, |\mathcal{D}|\}$ (or, w.r.t. to a set of features $\mathcal{U}$) iff*

$$(v^{(i)} \in \mathcal{A}) \wedge (v^{(j)} \in \mathcal{B}) \wedge \forall d \in \mathcal{U} : (v_d^{(i)} = v_d^{(j)}), \tag{2.5}$$

DEFINITION 2.6. (CORK) *We define a quality criterion q, called* **CORK** *(*Correspondence-based Quality Criterion*), for a subset of features $\mathcal{U}$ as*

$$q(\mathcal{U}) = (-1) * \textit{number of correspondences in } \mathcal{U} \tag{2.6}$$

THEOREM 2.2. *q is submodular.*

Proof. For q to be submodular, adding feature $X \in \mathcal{D}$ with $\mathcal{D}[x] = X$ to a feature set $\mathcal{T}' \subseteq \mathcal{T} \subseteq \mathcal{S}$ has to increase $q(\mathcal{T}')$ at least as much as adding feature X increases $q(\mathcal{T})$. This is equivalent to stating that whenever adding feature X removes one correspondence of $\mathcal{T}$, at least one correspondence of $\mathcal{T}'$ has to be eliminated as well.

Let us first state that an instance pair $(v^{(i)}, v^{(j)})$, that is a correspondence in $\mathcal{T}$ must also be a correspondence in $\mathcal{T}'$. Note that the opposite is not necessarily true.

Furthermore, whenever adding a feature X to $\mathcal{T}$ removes this correspondence from $\mathcal{T}$, this means that $v_x^{(i)} \neq v_x^{(j)}$, since the other features in $\mathcal{T}$ must match. Therefore, the two formerly corresponding feature patterns for $(v^{(i)}, v^{(j)})$ cannot match in $\mathcal{T}' \cup \{X\}$ either. Thus, if a feature X eliminates a correspondence from $\mathcal{T}$, this very correspondence (possibly together with further correspondences) is also removed from $\mathcal{T}'$, and we satisfy the submodularity condition of Equation 2.2.

This submodular criterion can be turned (by adding the constant $|\mathcal{A}| \cdot |\mathcal{B}|$) into a submodular set function fulfilling the conditions of Theorem 2.1.

We can now use q for greedy forward feature selection on a pre-mined set $\mathcal{S}$ of frequent subgraphs in $\mathcal{G}$ and receive a result set $\mathcal{T} \subseteq \mathcal{S}$ with a guaranteed quality bound. However, the success of $\mathcal{T}$ depends strongly on the setting of min_sup. If the support is chosen too low we can quickly generate too many features for a selection run finishing within reasonable time. Setting min_sup too high can cause the loss of all meaningful features. In the following, we want to introduce a selection approach which directly mines only discriminative subgraphs, which is *nested in gspan* and which can act independently from a frequency threshold.

2.5 Pruning gSpan's search space via CORK

gSpan exploits the fact that the frequency of a subgraph $S \in \mathcal{S}$ is an upper bound for the frequency of all of its supergraphs $T \sqsupseteq S$ (all subgraphs containing S) when pruning the search space for frequent subgraphs. In a similar way, we show that from the CORK-value of a subgraph S, we can derive an upper bound for the CORK-values of all of its supergraphs, that allows us to further prune the search space.

THEOREM 2.3. *Let $S, T \in \mathcal{S}$ be frequent subgraphs, and T be a supergraph of S. Let $\mathcal{A}_{S_1}$ denote the number of graphs in class $\mathcal{A}$ that contain S ('hits'), $\mathcal{A}_{S_0}$ the number of graphs in $\mathcal{A}$ that do not contain S ('misses')* *(and define $\mathcal{B}_{S_0}$, $\mathcal{B}_{S_1}$ analogously). Then*

$$q(\{S\}) = -\left(\mathcal{A}_{S_0} * \mathcal{B}_{S_0} + \mathcal{A}_{S_1} * \mathcal{B}_{S_1}\right) \tag{2.7}$$

and

$$q(\{T\}) \leq q(\{S\}) + \max \left\{ \begin{matrix} \mathcal{A}_{S_1} \cdot (\mathcal{B}_{S_1} - \mathcal{B}_{S_0}) \\ (\mathcal{A}_{S_1} - \mathcal{A}_{S_0}) \cdot \mathcal{B}_{S_1} \\ 0 \end{matrix} \right\} \tag{2.8}$$

Proof. Equation 2.7 arises from the definition of the criterion: CORK counts the inter-class pairs of graphs that both contain S or both do not contain S. We note that the gSpan pruning criterion is also valid for each class:

$$\mathcal{A}_{S_1} \geq \mathcal{A}_{T_1} \wedge \mathcal{B}_{S_1} \geq \mathcal{B}_{T_1} \tag{2.9}$$

If we thus want to asses how many correspondences may be eliminated by T, we can take into account, that T can never create new hits but can only decrement the number of hits in both classes. Naturally, the best improvement for S is made, when T eliminates all hits in one of the two classes and maintains the hits in the other class. When all hits of T disappear from $\mathcal{A}$, $\mathcal{A}_{S_0}$ increases by $\mathcal{A}_{S_1}$ and thus:

$$\begin{aligned} q(\{T\}) &= -\left((\mathcal{A}_{S_0} + \mathcal{A}_{S_1}) \cdot \mathcal{B}_{S_0} + 0 \cdot \mathcal{B}_{S_1}\right) = \\ &= -(\mathcal{A}_{S_0} + \mathcal{A}_{S_1}) \cdot \mathcal{B}_{S_0} \end{aligned} \tag{2.10}$$

Same holds for the elimination of all hits from $\mathcal{B}$:

$$\begin{aligned} q(\{T\}) &= -\left(\mathcal{A}_{S_0} \cdot (\mathcal{B}_{S_0} + \mathcal{B}_{S_1}) + \mathcal{A}_{S_1} \cdot 0\right) = \\ &= -\mathcal{A}_{S_0} \cdot (\mathcal{B}_{S_0} + \mathcal{B}_{S_1}) \end{aligned} \tag{2.11}$$

The third bounding case to be considered occurs if T changes nothing at all, *i.e.,* $q(\{T\}) = q(\{S\})$. Our maximal CORK value of T is thus

$$\begin{aligned} q(\{T\}) \leq \max \left\{ \begin{matrix} -|\mathcal{A}| \cdot \mathcal{B}_{S_0} \\ -\mathcal{A}_{S_0} \cdot |\mathcal{B}| \\ q(\{S\}) \end{matrix} \right\} &= \\ \overset{\text{eq. 2.7}}{=} q(\{S\}) + \max \left\{ \begin{matrix} \mathcal{A}_{S_1} \cdot (\mathcal{B}_{S_1} - \mathcal{B}_{S_0}) \\ (\mathcal{A}_{S_1} - \mathcal{A}_{S_0}) \cdot \mathcal{B}_{S_1} \\ 0 \end{matrix} \right\} &\blacksquare \end{aligned} \tag{2.12}$$

We can now use inequality (2.8) to provide an upper bound for the CORK values of supergraphs of a given subgraph S and exploit this information for pruning the search space in a branch-and-bound fashion.

Inequality (2.8) can be directly applied in the first iteration of greedy selection. For later iterations of greedy selection, we can define a similar bound for pruning. For this purpose, we need the concept of equivalence classes.

Definition 2.7. (Equivalence Classes) *Given a dataset $\mathcal{G}$ of graphs represented as binary indicator vectors over the feature set $\mathcal{U}$. Then the equivalence class of an indicator vector $v^{(i)}$ is defined as the set*

$$\{v^{(j)} | \forall d \in \mathcal{U}:\ v_d^{(i)} = v_d^{(j)}\} \tag{2.13}$$

Now let $\mathcal{P} \subseteq 2^{\mathcal{U}}$ be the set of all unique binary indicator vectors occurring in $\mathcal{G}$ with $|\mathcal{P}| = l$. Each of these unique indicator vectors forms an equivalence class $\mathbf{E}_c$ *($c \in \{1, ..., l\}$) containing all indicator vectors which are identical to indicator vector $\mathcal{P}_c$.*

We denote by

$$\mathcal{A}_{\mathcal{P}_c} = \left| \{v^{(i)} \in \mathcal{A} \,|\, \forall d \in \mathcal{U}:\ v_d^{(i)} = \mathcal{P}_c[d]\} \right| \tag{2.14}$$

the number of instances in $\mathcal{A}$ of equivalence class $\mathbf{E}_c$ and by

$$\mathcal{B}_{\mathcal{P}_c} = \left| \{v^{(i)} \in \mathcal{B} \,|\, \forall d \in \mathcal{U}:\ v_d^{(i)} = \mathcal{P}_c[d]\} \right| \tag{2.15}$$

the number of instances in $\mathcal{B}$ of equivalence class $\mathbf{E}_c$.

The number of correspondences for a feature set $\mathcal{U} \subseteq \{1, \ldots, |\mathcal{D}|\}$ can be calculated by adding up the correspondences of their equivalence classes $\mathbf{E}_c$ in $\mathcal{U}$:

$$q(\mathcal{U}) = (-1) \cdot \left(\sum_{\mathcal{P}_c \in \mathcal{P}} \mathcal{A}_{\mathcal{P}_c} \cdot \mathcal{B}_{\mathcal{P}_c} \right) \tag{2.16}$$

The bound of Equation (2.8) can then be extended into:

$$\begin{aligned} q(\mathcal{U} \cup \{T\}) \leq q(\mathcal{U} \cup \{S\}) + \\ \sum_{\mathcal{P}_c \in \mathcal{P}} \max \left\{ \begin{matrix} \mathcal{A}_{\mathcal{P}_c \cup \{S_1\}} \cdot (\mathcal{B}_{\mathcal{P}_c \cup \{S_1\}} - \mathcal{B}_{\mathcal{P}_c \cup \{S_0\}}) \\ (\mathcal{A}_{\mathcal{P}_c \cup \{S_1\}} - \mathcal{A}_{\mathcal{P}_c \cup \{S_0\}}) \cdot \mathcal{B}_{\mathcal{P}_c \cup \{S_1\}} \\ 0 \end{matrix} \right\} \end{aligned} \tag{2.17}$$

The main difference to (2.8) is that in later iterations of greedy selection, we only have to consider those graphs which are part of a correspondence (rather than all graphs).

We can now define our feature mining process in Algorithm 2.2: We initialize the set of selected subgraphs as an empty set $\mathcal{S}^\dagger$ and follow a recursive operation. In step 2, we require the next best subgraph S with $q(\mathcal{S}^\dagger \cup \{S\}) = \max_{T \in \mathcal{S}} q(\mathcal{S}^\dagger \cup \{T\})$. It can be obtained by simply running gSpan, always maintaining the currently best subgraph S according to q. Whenever in the course of mining, we reach a subgraph T with $q(\mathcal{S}^\dagger \cup \{T'\}) < q(\mathcal{S}^\dagger \cup \{S_0\})$ for any supergraph $T' \sqsupseteq T$ according to the bound defined in (2.17), we can prune the branches originating from S. As long as the resulting subgraph S actually improves $q(\mathcal{S}^\dagger)$, it is accepted as a discriminative feature and we start looking for the next best subgraph.

Algorithm 2.2. gSpan$_{\text{CORK}}(\mathcal{G}$, min_sup)

```
Input: Graph set $\mathcal{G}$, min_sup.
Output: Set of discriminative, frequent subgraphs $\mathcal{S}^\dagger$.

1: $\mathcal{S}^\dagger = \emptyset$;
2: $S$ = best subgraph according to $q(\mathcal{S}^\dagger \cup \{S\})$;
3: if $q(\mathcal{S}^\dagger \cup \{S\}) > q(\mathcal{S}^\dagger)$, then
4:     $\mathcal{S}^\dagger = \mathcal{S}^\dagger \cup \{S\}$;
5:     goto 2;
6: return $\mathcal{S}^\dagger$;
```

In contrast to the definition in Equation 2.1, this setting does not require a threshold s for the maximal number of subgraphs since it terminates automatically, when no new discriminative subgraph is found. In our experiments, we further noticed that on most datasets, CORK provides such a strong bound that it is even possible to omit the support threshold min_sup and still receive a discriminative set of (not necessarily frequent) subgraphs within a reasonable amount of time.

3 Experimental Evaluation

In this section, we conduct experiments to examine the effectiveness and efficiency of CORK in finding discriminative frequent subgraphs.

3.1 Datasets To evaluate our algorithm, we employed the 9 real-world datasets summarized in Table 1:

- Anti-cancer screen datasets (NCI): we use 8 datasets collected from the PubChem website as in [31]. They are selected from the bioassay records for cancer cell lines. Each of the anti-cancer screens forms a classification problem, where the class labels on these datasets are either active or inactive as a screen for anti-cancer activity. The active class is extremely rare compared to the inactive class. For a detailed description, please refer to [31] and the website, http://pubchem.ncbi.nlm.nih.gov. Each dataset can be retrieved by submitting queries in the above website.

  In order to have a fair comparison in those unbalanced datasets, each dataset has been resampled by forming 5 data subsets with balanced classes, where excessive instances from the larger class have been removed.

- Dobson and Doig (DD) [9] molecule data set: it consists of 1178 proteins, which can again be divided up into two classes: 691 enzymes and 487 non-enzymes. An extracted graph's vertices represent the C$_\alpha$ atoms of the corresponding protein's amino acids. Together with all distinct special conformations, they sum up to 82 vertex labels and are connected if they are at least within 6 Å of each other in the 3D protein structure. In order to retrieve edge labels, discretizing those distances would be possible, but prone to arbitrary thresholding. Consequently, edge labels are omitted. Even in this compacted form, with an average size of 285 vertices and 716 edges, these proteins are larger and stronger connected than the molecules from the NCI screening.

Dataset $\mathcal{G}$	$\lvert\mathcal{G}\rvert$	avg$\lvert V(G)\rvert$	avg$\lvert E(G)\rvert$	$\lvert\mathcal{L}_V\rvert$	$\lvert\mathcal{L}_E\rvert$
NCI1	4117	29.8	32.3	43	3
NCI33	3298	30.1	32.6	39	3
NCI41	3108	30.2	32.8	28	3
NCI47	4068	29.8	32.4	44	3
NCI81	4812	29.1	31.6	44	3
NCI109	4149	29.5	32.1	44	3
NCI145	3911	29.6	32.1	37	3
NCI330	4608	24.9	26.6	47	3
DD	1178	284.3	715.7	82	1

Table 1: Topologies of used graph sets:
$|\mathcal{G}|$: size of the dataset
avg$|V(G)|$: average number of vertices per graph
avg$|E(G)|$: average number of edges per graph
$|\mathcal{L}_V|$: number of vertex labels
$|\mathcal{L}_E|$: number of edge labels

In the experiments on these datasets, our CORK procedure selected between 15 and 66 subgraphs of sizes varying between 2 and 12 vertices (=atoms or amino acids), around 5% of which contain cycles. This means that subgraph mining procedures restricted to subclasses of graphs like trees [17] or graphs of restricted size [32, 25, 31, 27], which have been developed for less complex outputs and faster runtimes, would not enable us to produce results similar to those of gSpan, the graph miner we use.

3.2 Comparison to filter approaches CORK is a filter, hence in the first experiment, we assessed whether CORK selects subgraphs that generalise well on classification benchmarks, comparing it to state-of-the-art filter methods for subgraph selection.

We use 10-fold cross-validation for classification. Each dataset is partitioned into ten parts evenly. Each time, one part is used for testing and the other nine are combined for frequent subgraph mining, feature selection and model learning. In our current implementation, we use LIBSVM [6] to train a C-SVM classifier based on the selected features. C is optimised within a

		PC		Delta		IG		SC		CORK	
Dataset	$\left\|\mathcal{S}^\dagger\right\|$	AUC	Std	AUC	Std	AUC	Std	AUC	Std	AUC	Std
NCI1	57	0.685	0.053	0.724	0.025	0.712	0.024	0.690	0.026	**0.769**	0.023
NCI33	53	0.667	0.049	0.718	0.027	0.698	0.027	0.681	0.029	**0.759**	0.028
NCI41	49	0.689	0.059	0.722	0.023	0.748	0.028	0.732	0.037	**0.763**	0.027
NCI47	56	0.709	0.054	0.728	0.022	0.698	0.026	0.687	0.025	**0.779**	0.024
NCI81	64	0.670	0.071	0.711	0.022	0.731	0.724	0.720	0.024	**0.770**	0.022
NCI109	56	0.696	0.061	0.716	0.026	0.749	0.025	0.719	0.028	**0.774**	0.023
NCI145	55	0.688	0.068	0.717	0.029	0.733	0.035	0.698	0.027	**0.773**	0.029
NCI330	66	0.695	0.044	0.699	0.027	0.676	0.028	0.660	0.025	**0.769**	0.023
DD	15	0.605	0.051	**0.800**	0.038	0.674	0.048	0.694	0.039	0.778	0.038

Table 2: Classification AUC values (and standard deviation (Std)) for filter approaches on the 8 NCI graph datasets and on the DD graphs (PC = Pearson's Correlation Coefficient, Delta = the Delta method, IG = Information Gain, SC = Sequential Cover, CORK = Correspondence-based Quality Criterion). The number of features $\left|\mathcal{S}^\dagger\right|$ was determined by CORK selection on frequent subgraphs with min_sup 10%; best results are shown in bold.

range of seven values $\{10^{-6}, 10^{-4}, 10^{-2}, 1, 10^2, 10^4, 10^6\}$ / (size of the dataset) by cross-validation on the *training* dataset only. We employ a linear kernel on the selected graph features, and normalise the resulting kernel matrix K via $K_{normalised}(i,j) = \frac{K(i,j)}{\sqrt{K(i,i)K(j,j)}}$. We repeat the whole experiment 10 times and we report average results from these 10 runs.

We compare CORK to four state-of-the-art filter methods. Three of them are rankers using Pearson's Correlation Coefficient, the Delta Criterion which is closely related to MoSS [3] and Information Gain as a ranking criterion, and the last setting is the Sequential Cover method [8].

Pearson's Correlation The Pearson's Correlation Coefficient (PC) is commonly used in microarray data analysis [29, 10], where discriminative genes for phenotype prediction need to be selected from thousands of uninformative ones. As a selection criterion, the squared correlation between the occurrence pattern and the class label pattern is calculated for each feature independently and the k top-scoring features are selected.

Delta criterion The difference among subgraph frequencies in different classes is another popular feature selection criterion. For instance, in [3], Borgelt et. al. introduced MoSS, an approach for mining discriminative subgraphs. Their approach is designed for pharmacological screenings which specifically aim for characterizing the positive class. Thus, the idea is to accept only subgraphs which are frequent in the positive group, and infrequent in the complement. From this, we derive the following delta criterion as

$$q_{\text{delta}}(S) = \max\left(\mathcal{A}_{S_1} - \mathcal{B}_{S_1}, \mathcal{B}_{S_1} - \mathcal{A}_{S_1}\right) \ , \tag{3.18}$$

which can be used as a ranker criterion, in a similar way as (PC).

Information Gain As a final ranking method, we compare CORK to the Information Gain (IG), an entropy-based measure, which is frequently used in feature selection [36, 26].

Sequential Cover Algorithm 3.1 outlines the sequential cover method (SC). Frequent graphs are first ranked according to their relevance measure such as information gain, Fisher score, or confidence. In this experiment, we use confidence as the relevance measure. If a top-ranked frequent subgraph covers some of uncovered training instances, it will be accepted and removed from the feature set $\mathcal{S}$. The algorithm terminates if either all instances are covered or $\mathcal{S}$ becomes empty. SC can be executed multiple times to make several covers on the instances.

Algorithm 3.1. Sequential Cover (SC)

Input: Set of frequent subgraphs $\mathcal{S}$, training dataset $\mathcal{G}$
Output: Selected set of subgraphs $\mathcal{S}^\dagger$

```
Sort subgraphs in $\mathcal{S}$ in decreasing order of the chosen
  relevance measure;
while ($\mathcal{G} \neq \emptyset \wedge \mathcal{S} \neq \emptyset$)
    $S$ = first subgraph of $\mathcal{S}$;
    If $S$ covers at least one graph in $\mathcal{G}$
        $\mathcal{S}^\dagger = \mathcal{S}^\dagger \cup \{S\}$;
    $\mathcal{S} = \mathcal{S} \setminus \{S\}$;
    for each graph $G \in \mathcal{G}$ covered by $S$
        $\mathcal{G} = \mathcal{G} \setminus \{G\}$;
return $S^\dagger$
```

The results of the filter experiments are displayed in Table 2. We show the number of selected subgraphs $\left|\mathcal{S}^\dagger\right|$ among frequent subgraphs of min_sup 10%, together with the area under the receiver operating characteristic curve (AUC) and its standard deviation (Std) over the 100 experiments conducted per value. We observe that in all but one dataset, CORK detects the best feature combination for the 2-class classification problems at hand.

3.3 Other target sizes The number of selected features $\left|\mathcal{S}^\dagger\right|$ is an important parameter in feature selection. In order to demonstrate the fairness of our evaluation, Figure 2 displays a screening over the number of selected features for the tested filter approaches on NCI330. We see that the number of 66 subgraphs selected by CORK does not represent the optimal number of features for any of the criteria. However, in all cases, the improvement for larger feature sets increases slower for the higher set sizes. Moreover, CORK returns the best results for all tested feature sizes $\left|\mathcal{S}^\dagger\right| \geq 25$.

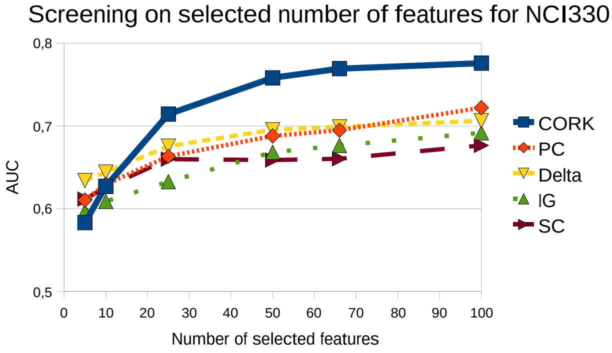

Figure 2: Screening over the number of selected features $\left|\mathcal{S}^\dagger\right|$ for CORK selection, the rankers using Pearson's Correlation, the Delta method and Information Gain, and Sequential Cover Selection.

3.4 Experimental runtime analysis In our third experiment, we evaluated the runtime performance of nested feature selection (i.e. *during* mining) versus un-nested feature selection (i.e. *after* mining). We run nested CORK on two datasets (the DD dataset and the NCI1 screening in Figures 3 and 4) and record the number of correspondences and the number of subgraphs examined per iteration.

In the DD experiment (Figure 3), we observe that in the beginning, we achieve a steep decrease in the number of correspondences, whilst enumerating a comparable number of subgraphs for the first 10 iterations and thus maintaing an almost constant runtime per iteration. In the end, CORK prunes a larger percentage of the enumerated subgraphs and the iterations speed up. The enumeration stops when all instances from the two classes are separated.

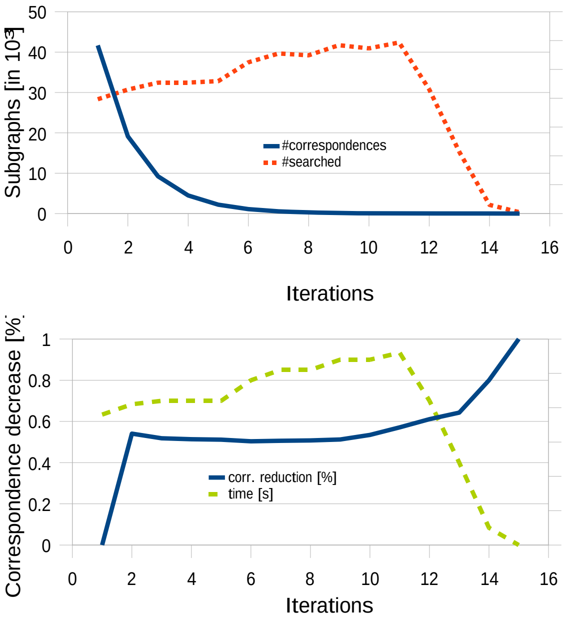

Figure 3: Nested feature mining experiment for the DD dataset (min_sup is set to 10%): each iteration corresponds to one selected feature. **Upper plot:** number of subgraphs (in 10^3) enumerated for the selection of one feature (in red, left scale) and number of correspondences (in 10^3) present at each iteration (in blue, right scale). **Lower plot:** percentaged decrease in the number of correspondences due to the current feature (in blue, left scale) and runtime per iteration (in green, right scale).

This attractive behaviour can be observed if there exists a (small) subset of subgraph features that eliminates all correspondences. In the other, unseparable case, CORK alone is not able to fully separate the two classes. This does not present a problem in un-nested feature selection, as the procedure simply ends when no new useful features can be identified. However, in the gSpan-nested setting, it may happen, that the complete DFS search tree has to be searched in order to discover that there is no better subgraph. This is illustrated in Figure 4, where the search space cannot be completely resolved with 11 correspondences remaining.

A way out of this problem is to allow CORK to terminate even if not all correspondences have been

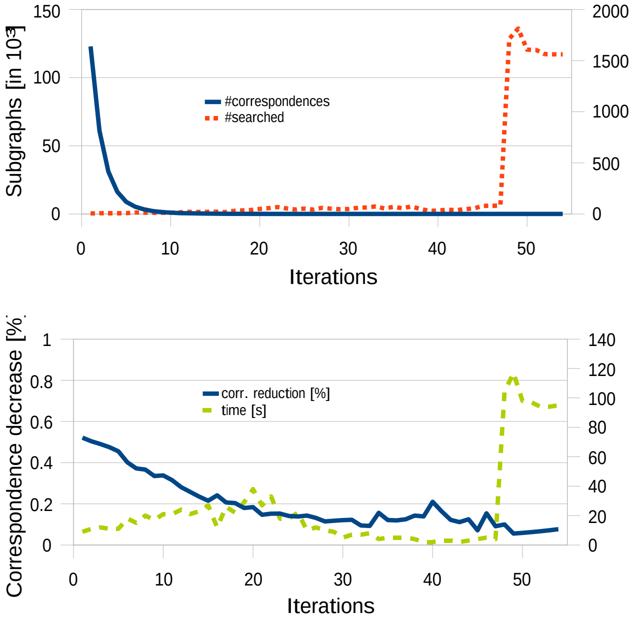

Figure 4: Nested CORK feature mining experiment for the NCI1 dataset. Same setting and types of figures as in Figure 3. After selecting 55 features, 11 correspondences remain (upper plot).

resolved, i.e. to introduce a *tolerance threshold* on the number of remaining correspondences.

3.5 Impact of tolerance threshold for correspondences

In our fourth experiment, we assessed the impact of employing a tolerance threshold t that leads to the termination of CORK, i.e. CORK feature selection ends once the number of correspondences falls below t. As demonstrated in Section 3.4, in later iterations on unseparable datasets, expensive subgraph mining results in relatively few resolved correspondences.

				time [min, s]	
t	#selected	AUC	Std	nested	un-nested
10000	5	0.748	0.044	4’51”	8’05”
1000	8	0.765	0.044	6’21”	14’14”
100	11	0.772	0.043	10’15”	18’33”
10	13	0.776	0.037	10’44”	19’39”
0	15	**0.778**	0.038	10’49”	20’02”

Table 3: Nested CORK versus un-nested CORK feature selection on the DD dataset with varying tolerance thresholds t. The un-nested runtimes are omitting the 20 minutes needed for the initial enumeration of frequent subgraphs.

In order to improve the effectiveness of CORK and to prevent overfitting by meaningless features, we define a tolerance threshold t on the number of correspondences that lead to the termination of the nested mining procedure.

We used the same setting as for the validation runs in Section 3.2 on the DD dataset. Both CORK selection variants are stopped as soon as they result in less than t correspondences. The results are displayed in Table 3.

This summary shows a slight advantage in accuracy of the lower tolerance thresholds 100 and 10, however, the additional runtime does not seem to be worth such an improvement over the quicker alternative of using a threshold of 1000 correspondences. The by far lower runtimes of the nested experiments further demonstrate the power of nested feature selection over the convential un-nested variants.

3.6 Comparison to wrapper approaches

The last experiment compares CORK to state-of-the-art wrapper approaches. These wrapper approaches allegedly outperform filter-based approaches in graph mining [28], hence we wanted to get a feeling for the difference in performance. We used the same experimental setup as in Section 3.2 and compare CORK to LAR-LASSO and decision-tree based classifiers (Table 4).

The LAR-LASSO method by [28] is a nested feature selection approach as well. Unlike CORK, it is a wrapper method as it minimizes a loss-function based on an active set of currently selected features using path regularization.

Another class of discriminative pattern mining approaches for graph mining was proposed by [37] and [11] who use a decision-tree like classifer. For a given dataset, [11] iteratively mine for the most meaningful feature according to information gain, and split this dataset into two separate problems. They proceed until the subproblems are solved or are of a smaller size than a given threshold. Using the gSpan extension CloseGraph [35], they have published experiments on the NCI screenings which we compare to ours in Table 4. Note, however, that the experiments of [11] have been conducted on the complete graph sets, while ours are resulting from balanced subsets of the whole dataset.

4 Discussion

Similar to information theoretic criteria used for decision trees, CORK measures the quality of a set of features to separate target classes using a specific set of features. Applying an efficient greedy forward selection scenario, CORK clearly performs best in comparison to other filter methods (see Table 2). It is not surprising that in such a vast space of interdependent features, feature combinations are more valuable than the simple ranking approach we used with Pearson's Correlation, the Delta method and the Information Gain. The Sequential Cover method at least takes into account that all instances should be covered by the selected set of features, yet, can never compete with CORK. We have been rather surprised by the mightiness of the Delta method since it actually scored better than Pearson Correlation. However, the complexity of the problem obviously requires the consideration of the various features' interdependence. CORK respects this interdependence by iteratively picking the subgraph feature which optimally complements the set of features selected so far (in terms of resolving correspondences).

Among the wrapper methods (see Table 4), CORK usually scores better than the model-based search tree approaches M^{b}T and DT M^{b}T (from [11]), even though these employ by far more subgraphs than CORK. Let us note, that those two feature selectors usually perfom slightly better than the simple ranker approach also employing Information Gain (cf. Tables 2 and 4). This criterion can be submodular, given certain preconditions [18]. This, however, is not the case here, since subgraphs are neither independent nor do they represent a subset of features mined previously. Thus, our less complex selection criterion still leads to higher quality results.

CORK cannot yet fully compete with the LAR-LASSO wrapper approach by [28]. It seems, however, to be more successful in matters of time on the Dobson & Doig problem, consisting of significantly larger graphs (see Table 1). This observation suggests that CORK pruning may be a useful alternative for datasets of large graphs. Furthermore, CORK as a filter method is useful when searching for features irrespective of a specific classifier.

One potentially helpful extension of our method was used in [28]. They store the DFS search tree for a set of previously mined frequent subgraphs. When restricting the mining procedure to a fixed min_sup value, this entails much shorter mining times, since gSpan only has to be called once per feature selection step and not several times. Still, the feasibility of this approach obviously depends on the size of the DFS tree that has to be stored.

We are currently exploring other submodular criteria [4] for subgraph feature selection that might allow for even higher levels of prediction accuracy. However, in preliminary experiments we have not yet found a more complex criterion permitting a feasible runtime.

An interesting question for future research is to find optimality guarantees for the horizontal leap search strategy for pattern mining which was recently proposed in [33], or to speed up CORK by employing this search strategy while maintaining its theoretical properties.

Another exciting question is whether our results on the optimality of supervised feature selection can be transfered to techniques for unsupervised feature selection on frequent subgraphs [5]. We are positive that this is possible (S. Nijssen, personal communication (2008)).

Acknowledgements

This research has been supported in part by the THESEUS Program in the MEDICO Project, which is funded by the German Federal Ministry of Economics and Technology under the grant number 01MQ07020. The responsibility for this publication lies with the authors. The authors would like to thank Siegfried Nijssen for fruitful discussions.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487–499, 1994.

[2] C. Borgelt and M. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proc. 2002 Int. Conf. on Data Mining (ICDM'02)*, pages 211–218, 2002.

[3] C. Borgelt, T. Meinl, and M. Berthold. Moss: a program for molecular substructure mining. In *OSDM '05: Proceedings of the 1st international workshop on open source data mining*, pages 6–15, New York, NY, USA, 2005. ACM.

[4] E. Boros, T. Horiyama, T. Ibaraki, K. Makino, and M. Yagiura. Finding essential attributes from binary data. *Ann. Math. Artif. Intell.*, 39(3):223–257, 2003.

[5] B. Bringmann and A. Zimmermann. One in a million: picking the right patterns. *Knowledge and Information Systems*, 2008.

[6] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[7] H. Cheng, X. Yan, J. Han, and C. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proc. of ICDE*, Istanbul, Turkey, 2007.

[8] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1036–1050, 2005.

[9] P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *J Mol Biol*, 330(4):771–783, Jul 2003.

[10] L. Ein-Dor, O. Zuk, and E. Domany. Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer. *Proc. Natl. Acad. Sci. USA*, 103(15):5923–5928, Apr 2006.

[11] W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. S. Yu, and O. Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In Y. Li, B. Liu, and S. Sarawagi, editors, *KDD*, pages 230–238. ACM, 2008.

[12] H. Fröhlich, J. Wegner, F. Sieker, and A. Zell. Optimal assignment kernels for attributed molecular graphs. In *Proc. Intl. Conf. Machine Learning*, pages 225–232, Bonn, Germany, 2005.

[13] C. Guestrin, A. Krause, and A. Singh. Near-optimal sensor placements in gaussian processes. In *Proc. Intl. Conf. Machine Learning*, Bonn, Germany, 2005.

[14] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. In *Proc. 2003 Int. Conf. Data Mining (ICDM'03)*, pages 549–552, 2003.

[15] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proc. 2000 European Symp. Principle of Data Mining and Knowledge Discovery (PKDD'00)*, pages 13–23, 2000.

[16] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proc. of ICML*, pages 321–328, Washington, DC, 2003.

[17] S. Kramer, L. Raedt, and C. Helma. Molecular feature mining in HIV data. In *Proc. of KDD*, pages 136–143, San Francisco, CA, 2001.

[18] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Uncertainty in Artificial Intelligence UAI'05*, 2005.

[19] H. Kubinyi. Drug research: myths, hype and reality. *Nature Reviews: Drug Discovery*, 2:665–668, 2003.

[20] T. Kudo, E. Maeda, and Y. Matsumoto. An application of boosting to graph classification. In *Advances in Neural Information Processing Systems 17 (NIPS'04)*, pages 729–736, Vancouver, BC, Dec. 2004.

[21] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proc. 2001 Int. Conf. Data Mining (ICDM'01)*, pages 313–320, 2001.

[22] P. Mahë, N. Ueda, T. Akutsu, J. Perret, and J. Vert. Extensions of marginalized graph kernels. In *Proc. of ICML*, pages 552–559, Alberta, Canada, 2004.

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

[24] S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. In *Proc. 2004 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'04)*, pages 647–652, 2004.

[25] N. Przulj. Biological network comparison using graphlet degree distribution. In *2006 European Conference on Computational Biology (ECCB)*, September 2006.

[26] P. Radivojac, Z. Obradovic, A. K. Dunker, and S. Vucetic. Feature selection filters based on the permutation test. In *Pedreschi (Eds.), Machine Learning: ECML 2004, 15th European Conference on Machine Learning*, pages 334–346. Springer, 2004.

[27] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *AISTATS*, 2009.

[28] K. Tsuda. Entire regularization paths for graph data. In *Proc. Intl. Conf. Machine Learning*, pages 919–926, 2007.

[29] L. J. van 't Veer, H. Dai, M. J. van de Vijver, Y. D. He, A. A. M. Hart, et al. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–536, 2002.

[30] N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph patterns from semistructured data. In *Proc. 2002 Int. Conf. on Data Mining (ICDM'02)*, pages 458–465, 2002.

[31] N. Wale and G. Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. In *Proc. of ICDM*, pages 678–689, Hong Kong, 2006.

[32] S. Wernicke. A faster algorithm for detecting network motifs. In *WABI*, pages 165–177, 2005.

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

[34] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proc. 2002 Int. Conf. on Data Mining (ICDM'02)*, pages 721–724, 2002.

[35] X. Yan and J. Han. CloseGraph: mining closed frequent graph patterns. In *KDD*, pages 286–295, 2003.

[36] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. Intl. Conf. Machine Learning*, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[37] A. Zimmermann and B. Bringmann. CTC - correlating tree patterns for classification. In *Proc. 2005 Int. Conf. Data Mining (ICDM05)*, pages 833–836, 2005.

		Filter		Wrapper				
		CORK			M^bT AUC values		LAR SVM	
Dataset	$\left\|\mathcal{S}^\dagger\right\|$	AUC	Std	$\left\|\mathcal{S}^\dagger\right\|$ M^bT	M^bT	DT M^bT	AUC	Std
NCI1	57	0.769	0.023	77	0.685	0.74	0.805	0.021
NCI33	53	0.759	0.028	344	0.743	0.745	0.792	0.024
NCI41	49	0.763	0.027	376	0.765	0.763	0.802	0.025
NCI47	56	0.779	0.024	587	0.708	0.727	0.809	0.023
NCI81	64	0.770	0.022	685	0.696	0.723	0.792	0.021
NCI109	56	0.774	0.023	605	0.699	0.746	0.808	0.022
NCI145	55	0.773	0.029	491	0.747	0.752	0.807	0.022
NCI330	66	0.769	0.023	n.a.			0.797	0.020
DD	15	0.778	0.038	n.a.			0.789	0.039

Table 4: Classification AUC values (with standard deviation (Std)) on the 8 NCI graph datasets and of the DD graphs (CORK = Correspondence-based Quality Criterion, M^bT and DT M^bT = Model based search tree approaches – results taken from [11], LAR-SVM = features selected (the same number $\left|\mathcal{S}^\dagger\right|$ as CORK) by LAR-LASSO evaluated via SVM). Frequency threshold for frequent subgraphs is 10%.