Discriminative frequent subgraph mining with optimality guarantees

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

July 7, 2010

Abstract

The goal of frequent subgraph mining is to detect subgraphs that frequently occur in a dataset of graphs. In classification settings, one is often interested in discovering *discriminative* frequent subgraphs, whose presence or absence is indicative of the class membership of a graph. 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.

In a graph classification problem, we are given a set of training graphs $\{G_1, \ldots, G_n\}$ with class labels $\{G_i, y_i\}_{i=1}^n$, $y_i \in \{1, \ldots, K\}$. Given these training examples, our task is to train a classifier for correctly predicting the labels of unclassified test graphs.

Such graph classification algorithms have 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 in an anticancer screen, toxic or non-toxic to human beings [21]. This makes graph classification scientifically and commercially valuable (e.g. in drug discovery). In computer vision, images can be abstracted as graphs, where nodes are spatial entities and edges are their mutual relationships. Such models can be used to identify the type of foreground objects in an image. 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 research in graph classification comprises three branches:

- first, the family of *frequent pattern approaches* [19, 10, 8]. Each graph is represented by its frequent subgraphs, i.e., its set of subgraphs that occur in at least $\sigma\%$ of all graphs in the database. This frequent pattern approach is also referred to as the (frequent) substructure or fragment approach, and we will use these terms interchangeably.

- second, the family of approaches that consider *all subgraphs* of a certain type in a graph [18, 36, 30]. For instance, the graph kernels by [18, 30] belong to this class and they count common walks and subtree patterns in two graphs, respectively.

- third, the family of wrapper approaches that select informative subgraphs for classification during the training phase. Typical instances of this family are the boosting approach by [22] and the lasso-approach by [33].

In this article, we are concerned with the first of these three families, the family of frequent subgraph approaches. There are two reasons for adapting frequent subgraphs 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 extremely infrequent substructures is small due to their limited coverage in the dataset.

*Institute for Informatics, Ludwig-Maximilians-Universität München
†Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong
‡School of Computer Science, Carnegie Mellon University
§University of Illinois at Urbana-Champaign
¶Yahoo! Research, Santa Clara, California
‖University of Illinois at Chicago, Chicago, Illinois
**Department of Computer Science, University of California at Santa Barbara
††Max Planck Institute for Developmental Biology and Max Planck Institute for Biological Cybernetics, Tübingen

Therefore, it is a promising approach to use frequent substructures as features in classification models.

However, the vast number of substructures poses three challenges.

1. Redundancy: Most frequent substructures only differ slightly in structure and co-occur in the same graphs.

2. Statistical significance: Frequency alone is not a good measure of the discriminative power of a subgraph, as both frequent and infrequent subgraphs may be uniformly distributed over all classes. Only frequent subgraphs whose presence is statistically significantly correlated with class membership are promising contributors for classification.

3. Efficiency: Very frequent subgraphs are not useful for classification due to lack of discriminative power. Therefore, frequent subgraph based classification usually sets an extremely low frequency threshold, resulting in thousands or even millions of features. Given such a tremendous number of features, any runtime or memory-intensive feature selection algorithm will fail.

Consequently, we need an efficient algorithm to select discriminative features among a large number of frequent subgraphs. In [32], we introduced a near-optimal approach to feature selection among frequent subgraphs generated by gSpan [39] for two-class problems. Our method greedily chooses frequent subgraphs according to the *submodular* quality criterion CORK (Correspondence-based Quality Criterion). The use of a submodular function in a greedy approach ensures a solution close to the optimal solution [24]. We furthermore showed that CORK can be integrated into gSpan, the state-of-the-art tool for frequent subgraph mining.

Other approaches use heuristic strategies for feature selection (such as [8, 13]) or do not provide optimality guarantees [22, 29, 28, 33, 38, 17]. We will present an overview on related algorithms in Section 3.1.

Goal The goal of this paper is to refresh the idea of near-optimal feature selection in subgraph patterns and to introduce improvements for future use. As a review of [32] we will first formalize the optimization problem to be solved (Section 2.1) and then we will summarize 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). We will review our selection criterion CORK for two-class problems in Section 2.4, and explain its integration as additional pruning criterion into pattern growth based graph miners like gSpan in Section 2.6.

Many applications for graph learning actually define more than the commonly-used two classes: Biological molecules can be categorized into a wide catalog of functional or structural classes, social network communities are involved with various topics and process flows can be analyzed with respect to multiple attributes. As a new contribution, we will thus generalize CORK to multi-class problems in Section 2.7.

Finally, for increasing the flexibility of our algorithm, in Section 2.8, we will also provide an extension for using the proposed pruning approach on pre-mined graphs. After a review of related work in Section 3 we thoroughly evaluate the proposed algorithms in Section 4 on 11 real-world datasets and conclude with a discussion and outlook in Section 5.

2 Near-optimal feature selection among frequent subgraphs

We formalize the given dataset as a collection of graphs $\mathcal{G} = \cup_{i=i}^{K} \mathbf{K}_i$ that each belong to one of the K classes $\mathbf{K}_i$. In this paper we exclude overlapping classes, however, the proposed selection approach can be easily extended to graphs with multiple labels.

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 $G \sqsubseteq G'$. Accordingly, G' is called a super-graph of G ($G' \sqsupseteq G$). Due to its importance for this article, we here recite the definition of a subgraph isomorphism.

Definition 2.1. (Subgraph Isomorphism) *A subgraph isomorphism is an injective function $f : V(G) \rightarrow V(G')$, 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 $\mathcal{G}$, we denote by $\mathcal{G}_{G_1}$ the number of graphs in $\mathcal{G}$ of which G is a subgraph and by $\mathcal{G}_{G_0}$ the number of graphs in $\mathcal{G}$ of which G is *not* a subgraph. $\mathcal{G}_{G_1}$ is called the *(absolute) support*. A graph G is *frequent* if its support is no less than a minimum support threshold, σ. Hence, the frequent graph is a relative concept: whether or not a graph is frequent depends on the value of σ and on the number of elements $|\mathcal{G}|$ contained in $\mathcal{G}$.

2.1 Combinatorial optimization problem Feature selection among frequent subgraphs can be defined as a combinatorial optimization problem. We denote by $\mathcal{D}$ the full set of features, which in our case will correspond to the frequent subgraphs generated by gSpan. When using these features to predict the class membership of individual graph instances, clearly, only a subset $\mathcal{E} \subseteq \mathcal{D}$ of features will be relevant. We denote the relevance of a feature set for class membership by $q(\mathcal{E})$, where q is a quality criterion measuring the discriminative power of $\mathcal{E}$. It is computed by restricting the dataset's representation to the features in $\mathcal{E}$. We then formulate feature selection as:

$$\mathcal{D}^{\ddagger} = \arg\max_{\mathcal{E}\subseteq\mathcal{D}} q(\mathcal{E}) \qquad \text{s.t.} \quad |\,\mathcal{E}\,| \leq s \tag{2.1}$$

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

The optimal solution of this problem would require us to search all possible subsets of features exhaustively. Due to the exponential number of all feature combinations this approach is prohibitive for large feature sets like frequent subgraphs. 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 $q(\mathcal{E})$ of a feature set $\mathcal{E}$ 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 $\mathcal{E}' \subseteq \mathcal{E} \subseteq \mathcal{D}$ and $X \in \mathcal{E}$*

$$q(\mathcal{E}' \cup \{X\}) - q(\mathcal{E}') \geq q(\mathcal{E} \cup \{X\}) - q(\mathcal{E}) \tag{2.2}$$

If q is submodular and we employ greedy forward feature selection, then we can exploit the following theorem from [24]:

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

$$q(\mathcal{E}^{\dagger}) \geq \left(1 - \frac{1}{e}\right) \max_{\mathcal{E}\subseteq\mathcal{D}:\ |\mathcal{E}|=s} q(\mathcal{E}) \ , \tag{2.3}$$

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 property is referred to as being *near-optimal* in the literature (e.g. [14]).

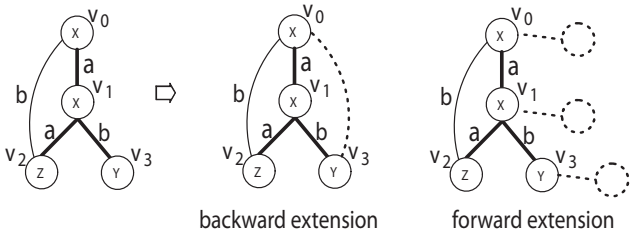


Figure 1: gSpan: Rightmost Extension

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 [39], which we will outline 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 [16], FSG [23] and the path-join algorithm [35], 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 [39], MoFa [3], FFSM [15], and Gaston [25] adopt the pattern-growth methodology, which attempts to generate candidate graphs from a *single* graph. 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 more frequent graph can be generated.

gSpan introduced a sophisticated extension method, which is built on a depth first search (DFS) tree. Given a graph G we label the root, i.e. the starting vertex of the DFS tree, as v_0, and the last visited vertex as v_n. v_n is also called the *rightmost vertex*. Consequently, the straight path from v_0 to v_n is the *rightmost 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, thus v_3 is the rightmost vertex. The rightmost path is (v_0, v_1, v_3).

This method, called rightmost extension, restricts the extension of new edges in a graph as follows: For a given graph and a DFS tree, 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 originating from a vertex on the rightmost path (*forward extension*). As we do not allow duplicate connections, the only legal backward extension candidate of the graph in Figure 1 is (v_3, v_0). The forward extension candidates can be edges from v_3, v_1, or v_0 introducing a new vertex. Since there may 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 in one DFS tree. One of those rules is the restriction of newly generated edges to the vertices along the rightmost path. Another rule, the minimality test, checks whether the currently examined graph has not been treated before. For a detailed description of gSpan, see [39].

ALGORITHM 2.1. gSPAN(G, $\mathcal{G}$, σ, $\mathcal{S}$)
Input: Graph G, graph dataset $\mathcal{G}$, threshold σ, set of subgraphs $\mathcal{S}$
Output: The set of frequent subgraphs $\mathcal{S}$.

1: **if** $G \neq$ DFS(G), **then**
2: **return** $\mathcal{S}$ // *G is not minimal*
3: insert G into $\mathcal{S}$
4: set C to $\emptyset$
5: scan $\mathcal{G}$ once: find all the edges e such that G can be rightmost extended to $G \diamond_r e$
6: insert $G \diamond_r e$ into C and count its frequency
7: **for each** frequent $G \diamond_r e$ in C **do**
8: Call gSPAN($G \diamond_r e$, $\mathcal{G}$, σ, $\mathcal{S}$)
9: **done**
10: **return** $\mathcal{S}$

Algorithm 2.1 outlines the pseudocode of gSpan. $G \diamond_r e$ denotes that an edge e extends graph G via rightmost extension. Step 1 is the minimality test, where DFS(G), the canonical form of graph G [39] is compared to the edge order of G. Therefore, G is only proceeded at the first encounter.

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_i from a dataset $\mathcal{G}$ and a set of frequent subgraph features $\mathcal{S}$ discovered by gSpan. We then define an indicator vector $v^{(i)}$ for G_i as*

$$v_d^{(i)} = \begin{cases} 1 & \text{if } \mathcal{S}_d \sqsubseteq G_i \quad (\mathcal{S}_d \text{ is a subgraph of } G_i) \\ 0 & \text{otherwise} \end{cases}, \tag{2.4}$$

where $v_d^{(i)}$ is the d-th component of $v^{(i)}$ and $\mathcal{S}_d$ is the d-th graph in $\mathcal{S}$.

2.4 Definition of CORK We now define our feature selection criterion q for two-class problems. It will be generalized to multi-class problems in Section 2.7.

DEFINITION 2.4. *Let $\mathcal{G}$ be a dataset of binary vectors, consisting of two disjunct classes $\mathcal{G} = \mathcal{A} \cup \mathcal{B}$. Let $\mathcal{D}$ denote a set of features of the data objects in $\mathcal{G}$, represented by indicator vector $v^{(i)}$ for graphs $G_i \in \mathcal{G}$.*

As we aim to separate the two classes, we pay specific attention to pairs of inter-class instances with the same pattern in the given feature set. These instance pairs are *correspondences*:

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. a set of features $\mathcal{U}$) iff*

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

where $v_d^{(i)}$ is the value of feature d in vector $v_{(i)}$.

Our quality criterion consequently punishes the number of correspondences remaining for feature set $\mathcal{D}$.

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

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

THEOREM 2.2. *q is submodular.*

Proof. For q to be submodular, adding feature $X \in \mathcal{D}$ to a feature set $\mathcal{E}' \subseteq \mathcal{E} \subseteq \mathcal{D}$ has to increase $q(\mathcal{E}')$ at least as much as adding feature X to $\mathcal{E}$ increases $q(\mathcal{E})$. This law of diminishing returns is obviously fulfilled if removing a correspondence from $\mathcal{E}$ by adding feature X also results in a correspondence being eliminated in $\mathcal{E}'$ by adding feature X.

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

In the following, let x be the index of feature X in $\mathcal{D}$. Whenever adding a feature X to $\mathcal{E}$ removes the above correspondence from $\mathcal{E}$, this means that $v_x^{(i)} \neq v_x^{(j)}$, since the other features in $\mathcal{E}$ must match. Therefore, the two formerly corresponding feature patterns for $(v^{(i)}, v^{(j)})$ cannot match in $\mathcal{E}' \cup \{X\}$ either. Thus, if a feature X eliminates a correspondence from $\mathcal{E}$, this very correspondence (possibly together with further correspondences) is also removed from $\mathcal{E}'$, 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.

2.5 Computation of CORK The CORK value for one feature X in a dataset of the classes $\mathcal{A}$ and $\mathcal{B}$ can be computed as the number of inter-class pairs of objects that both contain X (with $\mathcal{A}_{X_1}$ instances in $\mathcal{A}$ and $\mathcal{B}_{X_1}$ instances in $\mathcal{B}$) or that both do not contain X ($\mathcal{A}_{X_0}$ and $\mathcal{B}_{X_0}$ objects).

$$q(\{X\}) = -\left(\mathcal{A}_{X_0} \cdot \mathcal{B}_{X_0} + \mathcal{A}_{X_1} \cdot \mathcal{B}_{X_1}\right) \tag{2.7}$$

For feature sets CORK can be efficiently computed by recursively dividing the dataset into equivalence classes:

Definition 2.7. (Equivalence Classes) *Given a two-class dataset $\mathcal{G} = \mathcal{A} \cup \mathcal{B}$ represented as binary indicator vectors over the feature set $\mathcal{U}$. Let $\mathcal{P} \subseteq 2^{\mathcal{U}}$ be the set of all unique binary indicator vectors occurring in $\mathcal{G}$ with $|\mathcal{P}| = l$. Then the* equivalence class *of an indicator vector $v^{(i)} \in \mathcal{G}$ is defined as the set*

$$\{v^{(j)} | v^{(j)} \in \mathcal{G} \ \wedge \ \forall\, d \in \mathcal{U} :\ v_d^{(i)} = v_d^{(j)}\} \tag{2.8}$$

Each of these unique indicator vectors $\mathcal{P}_c$ forms an equivalence class $\mathbf{E}_c (c \in \{1, ..., l\})$ containing all graphs of with an indicator vector equal to $\mathcal{P}_c$.

We denote by

$$\mathcal{A}_{\mathcal{P}_c} = \left|\{v^{(i)} \in \mathcal{A} \mid \forall d \in \mathcal{U} :\ v_d^{(i)} = \mathcal{P}_c[d]\}\right| \tag{2.9}$$

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

$$\mathcal{B}_{\mathcal{P}_c} = \left|\{v^{(i)} \in \mathcal{B} \mid \forall d \in \mathcal{U} :\ v_d^{(i)} = \mathcal{P}_c[d]\}\right| \tag{2.10}$$

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

In each greedy iteration step, those equivalence classes can be efficiently split into hits and misses. The CORK score for a feature set $\mathcal{U} \subseteq \{1, \ldots, |\mathcal{D}|\}$ can thus be calculated by adding up the correspondences of all occurring equivalence classes $\mathbf{E}_c$ in $\mathcal{U}$:

$$q(\mathcal{U}) = (-1) \cdot \left(\sum_{\mathcal{P}_c \in \mathcal{P}} \mathcal{A}_{\mathcal{P}_c} \cdot \mathcal{B}_{\mathcal{P}_c}\right) \tag{2.11}$$

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 $\mathcal{S}^\dagger \subseteq \mathcal{S}$ of discriminative subgraphs with a guaranteed quality bound. However, the success of $\mathcal{S}^\dagger$ strongly depends on the choice of the minimum support σ. If σ is chosen too low, we can quickly generate too many features for the selection step to finish in a reasonable runtime. Setting σ too high can cause the loss of all informative features. In the following, we will introduce a selection approach which directly mines only discriminative subgraphs, which is *nested in gSpan* and which can act independently from a frequency threshold.

2.6 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. We will show how to derive an upper bound for the CORK-values of all supergraphs of a subgraph S, which allows us to further prune the search space.

Let us emphasize that this technique can also be applied in other graph miners which employ a kind of hierarchical subgraph pattern growth [3, 15, 25] or Apriori-based join [16, 23, 15]. The only necessary pre-condition for including CORK as pruning step is a supergraph relation ($T \sqsupseteq S$) for patterns mined at a later stage.

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*

$$q(\{T\}) \leq q(\{S\}) + \max \left\{ \begin{array}{c} \mathcal{A}_{S_1} \cdot (\mathcal{B}_{S_1} - \mathcal{B}_{S_0}) \\ (\mathcal{A}_{S_1} - \mathcal{A}_{S_0}) \cdot \mathcal{B}_{S_1} \\ 0 \end{array} \right\} \tag{2.12}$$

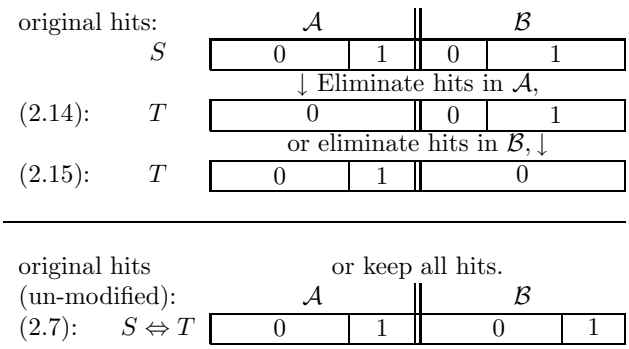


Figure 2: Possible change scenarios for the number of hits of supergraphs T for given hit distributions of $S \sqsubseteq T$: Hits ("1") can change into misses ("0"). The resulting extreme cases are illustrated for eliminating all hits from $\mathcal{A}$ (2.14) or from $\mathcal{B}$ (2.15), or for the case where keeping all hits is the best choice as in (2.7)

Proof. We note that the gSpan pruning criterion is also valid for each class:

$$\mathcal{A}_{S_1} \geq \mathcal{A}_{T_1} \wedge \mathcal{B}_{S_1} \geq \mathcal{B}_{T_1} \quad . \tag{2.13}$$

If we 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. This is illustrated in the first two cases of Figure 2. When all hits of T disappear from $\mathcal{A}$, $\mathcal{A}_{S_0}$ increases by $\mathcal{A}_{S_1}$ and thus:

$$\begin{aligned} q(\{T\}) &= -\left((\mathcal{A}_{S_0} + \mathcal{A}_{S_1}) \cdot \mathcal{B}_{S_0} + 0 \cdot \mathcal{B}_{S_1}\right) = \\ &= -(\mathcal{A}_{S_0} + \mathcal{A}_{S_1}) \cdot \mathcal{B}_{S_0} = -\,|\mathcal{A}| \cdot \mathcal{B}_{S_0} \end{aligned} \tag{2.14}$$

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

$$\begin{aligned} q(\{T\}) &= -\left(\mathcal{A}_{S_0} \cdot (\mathcal{B}_{S_0} + \mathcal{B}_{S_1}) + \mathcal{A}_{S_1} \cdot 0\right) = \\ &= -\,\mathcal{A}_{S_0} \cdot (\mathcal{B}_{S_0} + \mathcal{B}_{S_1}) = -\,\mathcal{A}_{S_0} \cdot |\mathcal{B}| \end{aligned} \tag{2.15}$$

Finally, we observe a third scenario when T does not cause any change at all, i.e., $q(\{T\}) = q(\{S\})$. This provides an additional bound if the decrease of hits in any class results in more correspondences than for S alone (cf. the last case in Figure 2). Our maximal CORK value of T is thus

$$\begin{aligned} q(\{T\}) &\leq \max \left\{ \begin{array}{c} -\,|\mathcal{A}| \cdot \mathcal{B}_{S_0} \\ -\,\mathcal{A}_{S_0} \cdot |\mathcal{B}| \\ q(\{S\}) \end{array} \right\} = \\ &\overset{\text{eq. 2.7}}{=} q(\{S\}) + \max \left\{ \begin{array}{c} \mathcal{A}_{S_1} \cdot (\mathcal{B}_{S_1} - \mathcal{B}_{S_0}) \\ (\mathcal{A}_{S_1} - \mathcal{A}_{S_0}) \cdot \mathcal{B}_{S_1} \\ 0 \end{array} \right\} \square \end{aligned} \tag{2.16}$$

We can now use inequality (2.12) to provide an upper bound for the CORK values of supergraphs T of a given subgraph S and exploit this information for pruning the search space in a branch-and-bound fashion.

Inequality (2.12) can be directly applied in the first iteration of greedy selection. For later iterations of greedy selection, we can define a similar bound on a set of features.

The bound of Equation 2.12 then extends to:

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

The main difference to (2.12) 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 thus define an additional pruning bound for subgraph enumeration:

Definition 2.8. (CORK Upper Bound) *Given a subgraph set $\mathcal{U}$ and a subgraph S. The CORK value of any supergraph T of $S (T \sqsupseteq S)$ cannot exceed the bound $MAX_{CORK}(\mathcal{U}, S)$:*

$$\begin{aligned} MAX_{CORK}(\mathcal{U}, S) &= q(\mathcal{U} \cup \{S\}) + \\ &\sum_{\mathcal{P}_c \in \mathcal{P}} \max \left\{ \begin{array}{c} \mathcal{A}_{\mathcal{P}_c \cup \{S_1\}} \cdot \left(\mathcal{B}_{\mathcal{P}_c \cup \{S_1\}} - \mathcal{B}_{\mathcal{P}_c \cup \{S_0\}}\right) \\ \left(\mathcal{A}_{\mathcal{P}_c \cup \{S_1\}} - \mathcal{A}_{\mathcal{P}_c \cup \{S_0\}}\right) \cdot \mathcal{B}_{\mathcal{P}_c \cup \{S_1\}} \\ 0 \end{array} \right\} . \end{aligned} \tag{2.18}$$

Algorithm 2.2. gSpan$_{\text{CORK}}(\mathcal{G}, \sigma = 0)$
Input : Graph set $\mathcal{G}$, optional threshold σ.
Output: Set of discriminative (frequent) subgraphs $\mathcal{S}^\dagger$.

1: $\mathcal{S}^\dagger = \emptyset$
2: S = best subgraph for $q(\mathcal{S}^\dagger \cup \{S\})$ *// gSpan call*
3: **if** $q(\mathcal{S}^\dagger \cup \{S\}) > q(\mathcal{S}^\dagger)$, **then**
4: $\mathcal{S}^\dagger = \mathcal{S}^\dagger \cup \{S\}$ *// S is an improvement*
5: **goto** 2
6: **return** $\mathcal{S}^\dagger$

The new feature mining process is defined in Algorithm 2.2:[1] 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 running gSpan, always maintaining the currently best subgraph S according to q. Whenever in the course of mining, we reach a subgraph T with

[1] An implementation of gSpan$_{\text{CORK}}$ is available at http://www.dbs.ifi.lmu.de/~thoma/pub/sam2010/sam2010.zip .

$\text{MAX}_{\text{CORK}}(\mathcal{S}^\dagger, T) \leq q(\mathcal{S}^\dagger \cup \{S\})$, we can prune all branches originating from T. Else, the candidate subgraph S might still be replaced by any of T's children. 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.

In contrast to the definition in Equation 2.1, this setting does not require a selection threshold s for the maximal number of features (subgraphs) since it automatically terminates 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 σ and still receive a discriminative set of (not necessarily frequent) subgraphs within a reasonable amount of time.

2.7 CORK for multi-class problems So far, we have restricted our attention to settings with two classes. Now, we will show how to extend $\text{gSpan}_{\text{CORK}}$ to multi-class problems. The key challenges here are to extend CORK's definition for handling multiple classes, and to then prove that this multi-class CORK (mcCORK) is still submodular and that it can still be integrated into gSpan.

Definition 2.9. (pairwise CORK) *Assume we are given a graph dataset $\mathcal{G} := \cup_{i=1}^{K} \mathbf{K}_i$ with K disjunct classes. $q_{i,j}(\mathcal{U})$ shall denote the CORK value restricting the dataset to classes $\mathbf{K}_i$ and $\mathbf{K}_j$ for a feature set $\mathcal{U}$. Then* pairwise *multi-class CORK ($mcCORK_{pw}$) is defined as*

$$mcCORK_{pw}(\mathcal{U}) := \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} q_{i,j}(\mathcal{U}) = (-1) \cdot \sum_{\mathcal{P}_c \in \mathcal{P}} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \mathbf{K}_{i,\mathcal{P}_c} \cdot \mathbf{K}_{j,\mathcal{P}_c} \ , \tag{2.19}$$

i.e., as the sum over CORK values for all pairs of classes, where $\mathbf{K}_{i,\mathcal{P}_c}$ is the number of matches of pattern $\mathcal{P}_c$ for $\mathcal{U}$ in class i and $\mathbf{K}_{j,\mathcal{P}_c}$ is the number of $\mathcal{P}_c$'s matches in class j, respectively.

Note that we restrict our definition to non-overlapping class labels. Of course, if a graph G belongs to multiple classes, $q_{i,j}(\mathcal{U})$ can be modified such that G is not considered when calculating the overall occurrences per equivalence class. This can be achieved using an additional counter for each equivalence class which is raised whenever a hit also belongs to another class and which is later subtracted from the equivalence class count. However, as structured output is not the focus of this paper, we will pause this line of thought for now.

Since pairwise CORK requires a quadratic runtime in the number of classes, we now show the ranking equivalence of pairwise CORK with the linear variant *1-vs.-rest* CORK.

Definition 2.10. (1-vs.-rest CORK) *Assume we are given a graph dataset $\mathcal{G} := \cup_{i=1}^{K} \mathbf{K}_i$ with K disjunct classes. $q_i(\mathcal{U})$ shall denote the CORK value for a dataset consisting of class $\mathbf{K}_i$ and its complement ($\mathbf{K}_{\neg i} = \cup_{j=1, j\neq i}^{K} \mathbf{K}_j$) as second class for a feature set $\mathcal{U}$. Then* 1-vs.-rest *multi-class CORK ($mcCORK_{1vr}$) is defined as*

$$mcCORK_{1vr}(\mathcal{U}) := \sum_{i=1}^{K} q_i(\mathcal{U}) = (-1) \cdot \sum_{\mathcal{P}_c \in \mathcal{P}} \sum_{i=1}^{K} \mathbf{K}_{i,\mathcal{P}_c} \cdot \mathbf{K}_{\neg i,\mathcal{P}_c} \ . \tag{2.20}$$

Lemma 2.1. *1-vs.-rest CORK and pairwise CORK result in the same ranking of feature sets.*

Proof. As the classes i to K are disjunct and since CORK does not use relative hit frequencies, the pairwise approach can be reduced to 1-vs.-rest as follows:

$$\begin{aligned}
\text{mcCORK}_{\text{1vr}}(\mathcal{U}) &= (-1) \cdot \sum_{\mathcal{P}_c \in \mathcal{P}} \sum_{i=1}^{K} \mathbf{K}_{i,\mathcal{P}_c} \cdot \mathbf{K}_{\neg i,\mathcal{P}_c} \\
&= (-1) \cdot \sum_{\mathcal{P}_c \in \mathcal{P}} \sum_{i=1}^{K} \left(\mathbf{K}_{i,\mathcal{P}_c} \cdot \left(-\mathbf{K}_{i,\mathcal{P}_c} + \sum_{j=1}^{K} \mathbf{K}_{j,\mathcal{P}_c} \right) \right) \\
&= (-1) \cdot \sum_{\mathcal{P}_c \in \mathcal{P}} \left(\sum_{i=1}^{K} \sum_{j=1}^{K} \mathbf{K}_{i,\mathcal{P}_c} \cdot \mathbf{K}_{j,\mathcal{P}_c} \ - \ \sum_{i=1}^{K} \mathbf{K}_{i,\mathcal{P}_c}^2 \right) \\
&= (-1) \cdot \sum_{\mathcal{P}_c \in \mathcal{P}} \left(2 \cdot \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \mathbf{K}_{i,\mathcal{P}_c} \cdot \mathbf{K}_{j,\mathcal{P}_c} \right) \\
&= 2 \cdot \text{mcCORK}_{\text{pw}}(\mathcal{U})
\end{aligned}$$

□

We next show the submodularity of this multi-class extension of CORK.

Theorem 2.4. *mcCORK is submodular.*

Proof. Both pairwise and 1-vs.-rest mcCORK are sums of pairwise CORK values. As pairwise CORK was shown to be submodular in Theorem 2.2, mcCORK is a sum of submodular functions. As submodular functions are closed under addition, mcCORK is also submodular. □

For the standard application of CORK-based greedy feature selection, we can hence replace two-class CORK by multi-class CORK, and perform multi-class feature selection with the same optimality guarantees. The question that remains to be answered is whether we can still perform nested feature selection with CORK in multi-class settings, that is whether we can integrate multi-class CORK into gSpan. For this purpose, we require a bound akin to equation (2.18). Since this bound is computed for all encountered frequent subgraphs, we define the bound for the faster 1-vs.-rest mcCORK variant.

THEOREM 2.5. *Let* $MAX_{CORK(i)}(\mathcal{U}, S)$ *denote the CORK upper bound for the subgraph set* $\mathcal{U}$ *and a subgraph* S *for class* $\mathbf{K}_i$ *and its complement* $\mathbf{K}_{\neg i} = \cup_{j=1, j\neq i}^{K} \mathbf{K}_j$. *Then*

(2.21)
$$mcCORK_{1vr}(\mathcal{U} \cup \{T\}) \leq \sum_{i=1}^{K} MAX_{CORK(i)}(\mathcal{U}, S) \ ,$$

where T *is any supergraph of* S $(T \sqsupseteq S)$.

Proof. $\text{mcCORK}(\mathcal{U} \cup \{T\})$ is a sum of pairwise CORK values $q_i(\mathcal{U} \cup \{T\})$, each of which can be upper-bounded by $\text{MAX}_{\text{CORK}(i)}(\mathcal{U}, S)$. As a consequence, the sum of these upper bounds

$$\sum_{i=1}^{K} \text{MAX}_{\text{CORK}(i)}(\mathcal{U}, S) \tag{2.22}$$

provides an upper bound for the sum of pairwise CORK values

$$\sum_{i=1}^{K} q_i(\mathcal{U} \cup \{T\}) \ , \tag{2.23}$$

that is an upper bound for $\text{mcCORK}_{1\text{vr}}(\mathcal{U} \cup \{T\})$. □

Inequality (2.21) can be used for pruning subtrees in gSpan's DFS search tree, if the upper bound on mcCORK in this subtree is less than the subgraph with maximum mcCORK score encountered so far.

2.8 Using pre-mined subgraphs The $\text{gSPAN}_{\text{CORK}}$ algorithm introduced in Section 2.6 is intended to speed up subgraph enumeration procedures which aim at generating features for classification. However, some datasets already allow for fast subgraph enumeration even without explicitly giving additional pruning criteria such as CORK. Furthermore, one could choose to use an alternative kind of enumeration, not necessarily targeting frequent subgraphs [19, 31, 36]. We now show that given an enumeration of subgraphs, we can convert Algorithm 2.2 into an offline approach depicted in Algorithm 2.3.

ALGORITHM 2.3. $\text{OFFLINE\_SELECT}_{\text{CORK}}(\mathcal{S})$
Input : List of subgraphs $\mathcal{S}$ with occurrence patterns $v^{(i)}_{\text{index of } S}$ for all $i \in \{1, \ldots, |\mathcal{G}|\}$
Output: Set of discriminative subgraphs $\mathcal{S}^\dagger$.

1: Generate DFS Codes for the graphs of $\mathcal{S}$
2: Sort $\mathcal{S}$ lexicographically in ascending order
3: $\mathcal{N}$ = integer array of size $|\mathcal{S}|$ // *map siblings*
4: Fill $\mathcal{N}$ s.t. $\mathcal{N}[i]$ is the position of the next element in $\mathcal{N}$ of which $\mathcal{S}[i]$ is not a prefix
5: $\mathcal{S}^\dagger = \emptyset$
6: $S = \text{NULL}$ // *next best subgraph*
7: $i = 0$
8: **while** $i < |\mathcal{S}|$ **do**
9: **if** $q(\mathcal{S}^\dagger \cup \{\mathcal{S}[i]\}) > q(\mathcal{S}^\dagger \cup \{S\})$, **then**
10: $S = \mathcal{S}[i]$
11: **if** $\text{MAX}_{\text{CORK}}(\mathcal{S}^\dagger, \mathcal{S}[i]) \leq q(\mathcal{S}^\dagger \cup \{S\})$, **then**
12: $i = \mathcal{N}[i]$ // *prune the children of* $\mathcal{S}[i]$
13: **else**
14: i++
15: **done**
16: **if** $q(\mathcal{S}^\dagger \cup \{S\}) > q(\mathcal{S}^\dagger)$, **then**
17: $\mathcal{S}^\dagger = \mathcal{S}^\dagger \cup \{S\}$
18: **goto** 6
19: **return** $\mathcal{S}^\dagger$

We first require a conversion of the subgraph enumeration into the canonical form of DFS Codes, such that the subgraphs can be sorted in the same lexicographical order as used by the gSpan traversal (step 2). Then we use this sorting to form a mapping $\mathcal{N}$ of each subgraph at sorting position i to the first subgraph index $> i$ which does not have the DFS Code of $\mathcal{S}[i]$ as a prefix (step 4). If $\mathcal{S}$ is the result of a gSpan run, $\mathcal{N}$ simply points from any DFS Code to the next DFS Code with a lower or equal number of edges. For treating other enumerations, an actual prefix test may become necessary. We now know that all elements of $\mathcal{S}$ from $i+1$ to $\mathcal{N}[i]$ are children of $\mathcal{S}[i]$ in the DFS Search Tree traversal, and thus supergraphs of $\mathcal{S}[i]$. While now traversing $\mathcal{S}$, looking for the next best subgraph according to CORK, in step 12 we skip those graphs if they can be pruned according to the CORK Upper Bound (2.18).

Using pre-mined subgraphs instead of applying the nested approach of Algorithm 2.2 can be a strong runtime advantage over $\text{gSPAN}_{\text{CORK}}$ if

1. the number of frequent subgraphs is relatively low, since then the complete enumeration can be faster

than repeated enumerations of bounded DFS code trees,

2. or if the frequent subgraphs are especially large, thus they repeatedly slow down the DFS code minimality test.

3 Related Work

In this article, we combine two components to achieve our goal of efficient feature selection among frequent subgraphs with quality guarantees: i) frequent subgraph mining and ii) a submodular quality function. We review related work on both of these components in the following.

3.1 Discriminative Subgraph Mining Discriminative frequent subgraph mining has evolved into a major direction in graph mining research over recent years. We here summarize prominent contributions to this branch of graph mining.

LeapSearch [38] speeds up subgraph mining by heuristically exploiting the fact that structurally similar subgraph patterns tend to have similar frequencies and statistical significance scores, resulting in orders of magnitude speed-up in comparison with state-of-the-art methods.

gBoost [22, 29], is a nested boosting approach, which repeatedly mines a set of frequent subgraphs while optimizing an LPBoost problem. This becomes feasible by iteratively refining pruning bounds which restrict the search space. In [28] Saigo et al. propose a faster version of gBoost using partial least squares regression on frequent subgraphs (gPLS).

The MoSS subgraph mining approach by Borgelt et al. [4] explicitly mines subgraphs which are frequent in the target class and infrequent in the control class. In [17] Jin et al. propose COM, a method for discriminative mining frequent subgraphs based on co-occurrence patterns. Using only one subgraph mining cycle, they iteratively grow a set of rules from the subgraphs mined so far, which is also designed for identifying a target class. Comparatively to MoSS they also use a minimum support threshold for rules involving the target class and a maximum support threshold for rules with patterns matching the control class.

An excellent wrapper approach to the problem of discriminate frequent subgraph mining was published by Koji Tsuda [33]. He uses the LASSO algorithm for mining salient features while exploiting pruning criteria on the used search path. Our approach differs from Tsuda's in two ways: Our feature selection method is a filter method and hence independent from the choice of the classifier and we can provide optimality guarantees for our solution.

Another class of discriminative pattern mining approaches for graph mining was proposed by [41] and [13] who use a decision-tree-like classifier. For a given dataset, [13] iteratively mine for the most meaningful feature according to the 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.

3.2 Related work on correspondences While we here present the first integration of a submodular quality function into the frequent subgraph mining process, there is related work on the quality function we employ. Correspondences were referred to as inconsistencies in Dash et al. [9] and used to define another, non-submodular quality criterion. In [5] Boros et al. derived CORK from families of Hamming distance measures as

(3.24)
$$\theta(\mathcal{U}) = \sum_{v^{(i)} \in \mathcal{A},\ v^{(j)} \in \mathcal{B}} \begin{cases} 1 & \text{if } \exists\ d \in \mathcal{U} :\ v_d^{(i)} \neq v_d^{(j)} \\ 0 & \text{else} \end{cases}$$

They recognized its beneficial greedy selection properties and evaluated other, more involved submodular set functions on small datasets with at most 125 features. We examined whether one of these other submodular set functions could be integrated into gSpan for efficient subgraph mining. However, it turned out that only CORK can be represented in terms of equivalence classes which allows for its efficient computation.

4 Experimental Evaluation

In this section, we conduct experiments to examine the effectiveness and efficiency of CORK in finding discriminative frequent subgraphs. After introducing the used graph datasets we will compare CORK to a number of other filter approaches. We first use the number of features selected by CORK as parametrization for all filters and later analyze how the competitors perform for a larger variety of selected features. We continue with a runtime analysis of the nested algorithm gSpan$_{\text{CORK}}$, followed by an improvement recommendation involving an additional threshold. We conclude the experimental section with a comparison to some of the wrapper approaches introduced in Section 3.1.

4.1 Datasets To evaluate our algorithm, we employed the 11 real-world datasets summarized in Table 1:[2]

[2] All datasets (overall size 23.4 MB) are available at http://www.dbs.ifi.lmu.de/~thoma/pub/sam2010/data.zip .

Dataset $\mathcal{G}$	$\lvert\mathcal{G}\rvert$	$\lvert V(G)\rvert$	$\lvert E(G)\rvert$	$\lvert\mathcal{L}_V\rvert$	$\lvert\mathcal{L}_E\rvert$	K
NCI1	4117	29.8	32.3	43	3	2
NCI33	3298	30.1	32.6	39	3	2
NCI41	3108	30.2	32.8	28	3	2
NCI47	4068	29.8	32.4	44	3	2
NCI81	4812	29.1	31.6	44	3	2
NCI109	4149	29.5	32.1	44	3	2
NCI145	3911	29.6	32.1	37	3	2
NCI330	4608	24.9	26.6	47	3	2
DD	1178	284.3	715.7	82	1	2
DD6C	664	357.9	909.7	63	1	6
AIDS	5621	27.6	29.7	44	4	3

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

- Anti-cancer screen datasets (NCI): we use 8 datasets collected from the PubChem website as in [36]. 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 in a screen for anti-cancer activity. The active class is extremely rare compared to the inactive class. For a detailed description, please refer to [36] 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 on those unbalanced datasets, each dataset has been re-sampled by forming 5 data subsets with balanced classes, where excessive instances from the larger class have been removed.

- Dobson and Doig (DD) [11] molecule data set: it consists of 1178 proteins, which can again be divided up into two classes: 691 enzymes and 487 non-enzymes. The vertices of an extracted graph represent the C_α atoms of the amino acids of the corresponding protein. 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 more densely connected than the molecules from the NCI screening.

EC	Name	Count
1	Oxidoreductases	145
2	Transferases	175
3	Hydrolases	214
4	Lyases	66
5	Isomerases	37
6	Ligases	27

Table 2: DD6C class distribution: Number of instances of the DD dataset by EC number.

- EC-number groups for DD (DD6C): We furthermore use the DD dataset for differentiating the examples of the enzymes class into their EC numbers [2], a hierarchical categorization system for enzymes. We distinguish between the 6 basic classes, thus transferring the dataset DD into a new dataset DD6C consisting of 664 enzymes that could be mapped to an EC number. Among the remaining enzymes 25 could not be mapped and 2 caused duplicate matches and were thus excluded from DD6C. The topology of this new dataset reveals that the non-enzymes in the original DD dataset appear to be smaller on average than the enzymes which also appear in the DD6C dataset. We thus consider the DD6C problem as harder than the DD problem, not only because of the additional classes, but also because of less pronounced variations between the classes. The class distribution is summarized in Table 2.

- AIDS antiviral screen data (AIDS): it contains the activity test information of $43,850$ chemical compounds. Each chemical compound is labeled as either active (CA), moderately active (CM) or inactive (CI) with respect to the HIV virus. Among these compounds, 423 belong to CA, 1081 are of CM, and the rest is in Class CI. This dataset is publicly available on the website of the Developmental Therapeutics Program (http://dtp.nci.nih.gov/docs/aids/aids_data.html). As with the NCI datasets, we have transformed this data into a slightly more balanced form of 10 splits, combining the active (CA) and moderately active (CM) compounds with samples of the inactive compounds (CI). The average number of compounds per split is shown in Table 1.

In the experiments on these datasets, our CORK procedure selected between 11 and 66 subgraphs of sizes

varying between 2 and 12 vertices (=atoms or amino acids), approximately 5% of which contain cycles. This means that subgraph mining procedures restricted to sub-classes of graphs like trees [19] or graphs of restricted size [37, 26, 36, 31], 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 mining approach we use.

4.2 Comparison to filter approaches CORK is a filter method. 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 [7] to train a C-SVM classifier based on the selected features. C is optimised within a 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 KM via $KM_{\text{norm}}(i,j) = \frac{KM(i,j)}{\sqrt{KM(i,i)KM(j,j)}}$. 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 [4] and Information Gain as a ranking criterion, and the fourth comparison partner is the Sequential Cover method [10].

Pearson's Correlation The Pearson's Correlation Coefficient (PC) is commonly used in microarray data analysis [34, 12], 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 (i.e., 1 to K) is calculated for each feature independently and a pre-defined number of the top-scoring features are selected.

Delta criterion The difference among subgraph frequencies in different classes is another popular feature selection criterion. For instance, the MoSS mining approach by Borgelt et al. [4] 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

$$q_{\text{delta}}(S) = \max\left(\mathcal{A}_{S_1} - \mathcal{B}_{S_1}, \mathcal{B}_{S_1} - \mathcal{A}_{S_1}\right) , \tag{4.25}$$

which can be used as a ranker criterion, in a similar way as PC. We extend it to multi-class by taking the difference between the number of hits in the class with the maximum frequency and the remaining average hit count per class:

$$q_{\text{delta MC}}(S) = \max_{i\in\{1,\ldots,K\}} \left(\mathbf{K}_{i,S_1} - \frac{1}{K-1} \sum_{j=1, j\neq i}^{K} \mathbf{K}_{j,S_1} \right) \tag{4.26}$$

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 [40, 27]:

$$q_{\text{IG}}(S) = \sum_{i\in\{0,1\}} \sum_{j=1}^{K} p(S=i, C=\mathbf{K}_j) \log_2 \frac{p(S=i, C=\mathbf{K}_j)}{p(S=i)\cdot p(C=\mathbf{K}_j)} , \tag{4.27}$$

where C is the class variable of the tested objects.

Sequential Cover Algorithm 4.1 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:

$$q_{\text{conf}}(S) = \max_{i\in\{1,\ldots,K\}} \frac{\mathbf{K}_{i,S_1}}{\sum_{j=1}^{K} \mathbf{K}_{j,S_1}} \tag{4.28}$$

If a top-ranked frequent subgraph covers some of the 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 4.1. 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 $\mathcal{S}$;
  $\mathcal{S} = \mathcal{S} \setminus \{S\}$;
  if S covers at least one graph in $\mathcal{G}$ **then**
    $\mathcal{S}^\dagger = \mathcal{S}^\dagger \cup \{S\}$;
    for each graph $G \in \mathcal{G}$ covered by S
      $\mathcal{G} = \mathcal{G} \setminus \{G\}$;
return $S^\dagger$

		PC		Delta		IG		SC		CORK	
Dataset	$\lvert\mathcal{S}^\dagger\rvert$	AUC	Std	AUC	Std	AUC	Std	AUC	Std	AUC	Std
NCI1	57	0.682	0.052	0.724	0.025	0.712	0.024	0.690	0.026	**0.769**	0.023
NCI33	53	0.682	0.053	0.718	0.027	0.698	0.027	0.681	0.029	**0.759**	0.028
NCI41	49	0.681	0.058	0.722	0.023	0.748	0.028	0.732	0.037	**0.763**	0.027
NCI47	56	0.714	0.052	0.728	0.022	0.698	0.026	0.687	0.025	**0.779**	0.024
NCI81	64	0.668	0.068	0.711	0.022	0.731	0.022	0.720	0.024	**0.770**	0.022
NCI109	56	0.699	0.061	0.716	0.026	0.749	0.025	0.719	0.028	**0.774**	0.023
NCI145	55	0.684	0.070	0.717	0.029	0.733	0.035	0.698	0.027	**0.773**	0.029
NCI330	66	0.692	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

(a) Classification AUC values (and standard deviation (Std)) for the 8 NCI graph datasets and on the two-class DD graphs.

			PC		Delta		IG		SC		CORK	
Dataset	$\lvert\mathcal{S}^\dagger\rvert$	Val.	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
DD6C	14	$\widehat{\text{AUC}}_{\text{pw}}$	0.719	0.018	0.703	0.015	0.715	0.009	0.715	0.027	**0.723**	0.018
		Accuracy	0.341	0.047	0.324	0.033	0.323	0.099	0.355	0.044	**0.359**	0.050
AIDS	55	$\widehat{\text{AUC}}_{\text{pw}}$	0.829	0.001	0.829	0.001	0.829	0.001	0.829	0.002	**0.832**	0.006
		Accuracy	0.733	0.001	0.733	0.001	0.733	0.001	0.733	0.001	**0.735**	0.005

(b) Multi-class average pairwise AUC estimates ($\widehat{\text{AUC}}_{\text{pw}}$) and classification accuracies (both with standard deviation (Std)) for filter approaches on the DD6C and the AIDS graphs

Table 3: Classification evaluation of filter methods (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 $\sigma = 10\,\%$; best results are shown in bold.

The results of the filter experiments are displayed in Table 3. Note that for better comparability, the number of selected features for all experiments was determined via CORK. Potential disadvantages for the other selection approaches are addressed in the next section. Table 3a shows the number of selected subgraphs $\left|\mathcal{S}^\dagger\right|$ among frequent subgraphs of σ 10%, together with the average area under the receiver operating characteristic curve (AUC) and its standard deviation (Std) over 100 conducted experiments. We observe that in all but one dataset, CORK detects the best feature combination for the two-class classification problems at hand.

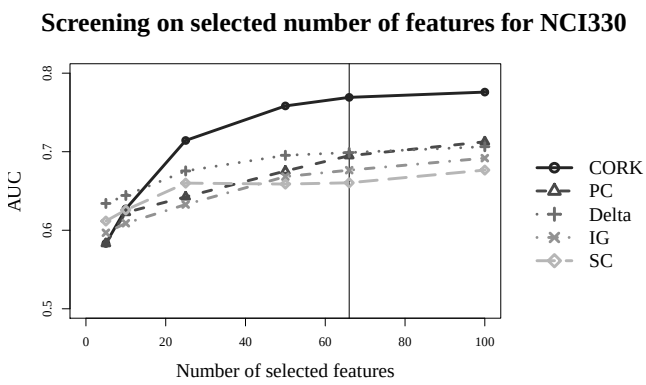
Table 3b compares the multi-class filter selectors on the multi-class datasets DD6C and AIDS by their average pair-wise AUC estimate

$$\widehat{\text{AUC}}_{\text{pw}}(\mathcal{G},\mathcal{U}) = \sum_{i=1}^{K} \frac{\left|\left\{d^{\mathcal{U}}_{i,j}(G_a) = i \mid G_a \in \mathbf{K}_i, j \in \{1,\ldots,K\} \setminus i\right\}\right|}{(K-1)\cdot|\mathcal{G}|} \tag{4.29}$$

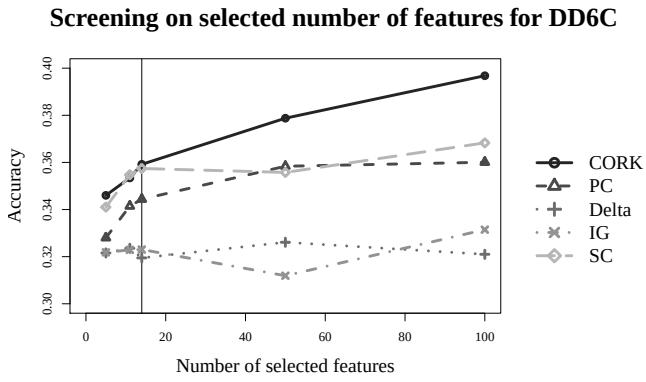
as the fraction of pairwise inter-class decisions in the dataset $\mathcal{G}$ where the decision function $d^{\mathcal{U}}_{i,j}$ votes for the correct class based on the selected subgraphs $\mathcal{U}$. For further orientation, we provide the classification accuracy. As can be seen, CORK performs best for both datasets, although there are no significant differences in accuracy compared to other methods.

It is not surprising that in the vast space of interdependent features spanned by frequent subgraphs, 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 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).

4.3 Other target sizes The number of selected features $\left|\mathcal{S}^\dagger\right|$ is an important parameter in feature selec-



(a) NCI330 dataset (two classes)



(b) Multi-class dataset DD6C

Figure 3: Screening of the feature quality over the number of selected features $\left|\mathcal{S}^{\dagger}\right|$ for CORK selection, Pearson's Correlation, the Delta method, Information Gain, and Sequential Cover Selection. The vertical line marks the number of features originally chosen by CORK.

tion. CORK suggests an automatic bound for the number of selected features, however, the selection procedure can be terminated earlier or restarted for determining fewer or more features. In order to demonstrate the fairness of our evaluation, Figure 3 displays screenings over the number of selected features for the tested filter approaches on the two-class problem NCI330 and the multi-class problem DD6C. We see that the number of subgraphs selected by CORK does not represent the optimal number of features for any of the criteria or datasets. However, in all cases, the larger the feature sets get, the smaller the increases in accuracy by adding more features. Moreover, CORK returns the best results for all tested feature sizes above the recommended number of features.

4.4 Experimental runtime analysis In our third experiment, we evaluated the runtime performance of nested feature selection, i.e. features are acquired *during* mining, as opposed to un-nested feature selection which takes place *after* mining. We run nested CORK on two complete datasets (the DD dataset and the NCI1 screening in Figure 4) and record the number of correspondences and the number of subgraphs examined per iteration. Since previous mining experiments have been handled on training subsets, the number of iterations is slightly elevated ($16 > 15$ and $64 > 57$) as opposed to Table 3.

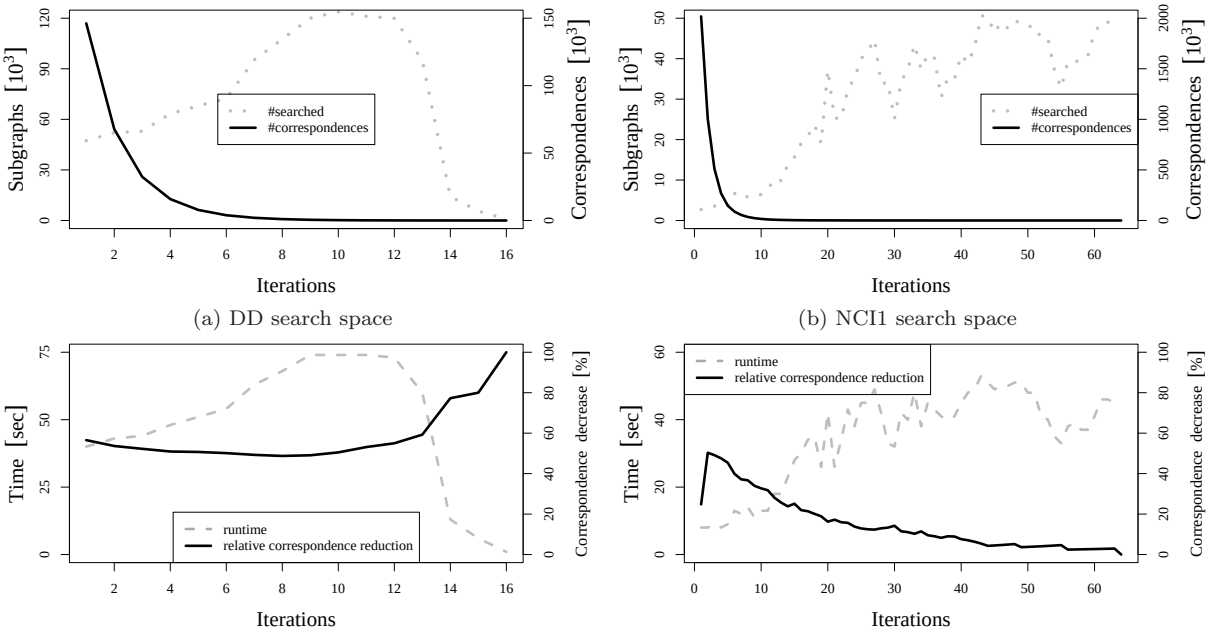
In the DD experiment (Figures 4a and 4c), we observe that in the beginning, we achieve a steep decrease in the number of correspondences, whilst enumerating a comparable number of subgraphs for each of the first 10 iterations and thus maintain an almost constant runtime per iteration. In the end, CORK 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.

This attractive behaviour can be observed if there exists a (small) subset of subgraph features that eliminates all correspondences. In the other, inseparable 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 Figures 4b and 4d, where the search space cannot be completely resolved, with 33 correspondences remaining.

A way out of this problem is to allow CORK to terminate even if not all correspondences have been resolved, i.e. to introduce a *tolerance threshold* on the number of remaining correspondences.

4.5 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 4.4, in later iterations on inseparable datasets, expensive subgraph mining results in relatively few resolved correspondences. 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 4.2. For showing the effect of the



(a) DD search space

(b) NCI1 search space

(c) DD runtimes

(d) NCI1 runtimes

Figure 4: Nested feature mining experiments on the complete datasets DD and NCI1 (σ is set to 10%): each iteration corresponds to one selected feature. **Upper plots:** number of subgraphs (in 10^3) enumerated for the selection of one feature (dotted-grey, left scale) and number of correspondences (in 10^3) present at each iteration (black, right scale). **Lower plots:** runtime per iteration (dashed-grey, left scale) percentaged decrease in the number of correspondences due to the current feature (in black, right scale).

tolerance threshold, we also compare the runtimes of the nested selection approach gSpan$_{\text{CORK}}$ to the un-nested variant offline_select$_{\text{CORK}}$ and the naïve approach of applying CORK as a common forward feature selection criterion on a pre-mined subgraph set without additional pruning. All CORK selection runs are stopped as soon as they result in less than t correspondences. The results are displayed in Table 4.

For the DD dataset (4a) 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 and offline experiments in comparison to the naïve approach demonstrate the pruning power of MAX$_{\text{CORK}}$ over the conventional un-nested variants.

Note that in Table 4a the runtimes of the nested approach are not only better than those of naïve forward selection, but they are also competitive to the quick offline variant, since the naïve and offline approaches omit the time necessary to first enumerate the set of frequent subgraphs. When thus counting the enumeration times, gSpan$_{\text{CORK}}$ is the fastest selection approach.

This effect is due to the large number of 110,131 frequent subgraphs for the DD dataset. For datasets which contain fewer frequent subgraphs, like the 2893 subgraphs for the NCI33 molecule collection in Table 4b, the offline approach and even naïve forward selection can be faster. We also point out the difference in the AUC value between the Tables 4b and 3a: The CORK evaluation of Table 3a was achieved by testing offline_select$_{\text{CORK}}$ on a pre-mined set of frequent subgraphs for the *complete* dataset. Of course, we separated the training instances from the test instances in the selection and training phase, however, the frequency bound for the mining step can cause variation in the number of frequent subgraphs between the *complete* and the *training* graphs only (gSpan$_{\text{CORK}}$) and can thus influence the classification performance.

In our experiments, the offline approach has always been faster than the naïve variant. We thus conclude that this algorithm is a useful example of how the gSpan pruning structure can be exploited even after mining has been completed.

4.6 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

		Filter		Wrapper				
		CORK			M^bT AUC values		LAR SVM	
Dataset	$\lvert\mathcal{S}^\dagger\rvert$	AUC	Std	$\lvert\mathcal{S}^\dagger\rvert$ 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 5: 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 [13], LAR-SVM = features selected (the same number $\left|\mathcal{S}^\dagger\right|$ as CORK) by LAR-LASSO evaluated via SVM). The frequency threshold σ is 10%.

DD Screening				time [min, s]		
t	$\lvert\mathcal{S}^\dagger\rvert$	AUC	Std	nested	naïve	offline
10000	5	0.745	0.036	3'27"	9'28"	23"
1000	8	0.761	0.039	6'01"	15'23"	39"
100	11	0.772	0.039	8'57"	18'41"	56"
10	13	0.776	0.037	10'09"	19'20"	1'01"
0	15	**0.778**	0.037	10'36"	19'28"	1'01"

(a) DD dataset

NCI33 Screening				time [min, s]		
t	$\lvert\mathcal{S}^\dagger\rvert$	AUC	Std	nested	naïve	offline
10000	10	0.679	0.032	1'21"	1'27"	3"
1000	18	0.707	0.031	3'43"	2'10"	7"
100	31	0.738	0.028	10'06"	2'34"	16"
10	54	0.765	0.023	21'19"	2'48"	30"
0	54	0.765	0.023	23'33"	2'48"	30"

(b) NCI33 dataset

Table 4: Nested CORK versus the two variants of un-nested CORK feature selection ("naïve": no pruning structure, "offline": the pruning approach of Algorithm 2.3) with varying tolerance thresholds t. The un-nested runtimes are omitting the time needed for the initial enumeration of frequent subgraphs (20 minutes for DD, one minute for NCI33).

mining [33], hence we wanted to get a feeling for the difference in performance. We used the same experimental setup as in Section 4.2 and compare CORK to LAR-LASSO and the decision-tree based classifiers of [13] (Table 5).

In [13], a query is classified by either directly using the feature tree formed by the subgraph mining routine (M^bT), or by building a decision tree on the selected features (DT M^bT). We compare the published experiments on the NCI screenings to ours in Table 5. Note, however, that the experiments of [13] have been conducted on the complete graph sets, while ours are resulting from balanced subsets of the whole dataset. CORK usually scores better than the model-based search tree approaches M^bT and DT M^bT, even though these employ by far more subgraphs than CORK. Let us note, that on average those two feature selectors perform slightly better than the simple ranker approach also employing Information Gain (cf. Tables 3a and 5). Information Gain can be submodular, given certain preconditions [20]. 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 [33]. The nested variant $\text{gSpan}_{\text{CORK}}$, however, seems to be more successful in matters of runtime 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. In addition, the selection runtimes of $\text{offline\_select}_{\text{CORK}}$

(between 30 and 60 seconds) are constantly below the runtime of LAR-LASSO (1 to 15 minutes). Furthermore, CORK as a filter method is useful when searching for features irrespective of a specific classifier.

5 Discussion and Outlook

In this article we have proposed a supervised feature selection approach for multi-class classification problems using frequent subgraphs. Since we use a submodular selection criterion, we can provide optimality guarantees for the set of selected features obtained by greedy forward selection. Additionally, we have explained how to integrate this criterion directly into the subgraph mining process by exploiting an upper bound for pattern-growth extension miners like gSpan. Moreover, we show how to use this bound on a set of pre-mined subgraphs, allowing for more flexibility in the choice of the type of subgraph used.

Similar to information theoretic criteria used for decision trees, CORK measures the quality of a set of features by means of its ability to separate target classes. In our experiments on classification benchmark datasets, the features selected by CORK reach the best accuracies among the filter methods. Among the wrapper methods, CORK outperforms M^bT and DT M^bT in all but one cases. The LAR-LASSO method still achieves a more accurate classification, however, CORK has runtime advantages on pre-mined patterns and large subgraphs.

A strategy to further improve the runtime of our approach is to store the DFS search tree for a set of previously mined frequent subgraphs [33]. When restricting the mining procedure to a fixed minimum support value, this entails much shorter mining times, since gSpan effectively 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.

One goal in our future research is to find optimality guarantees for the horizontal leap search strategy for pattern mining proposed in [38], and to speed up CORK by employing this search strategy while maintaining its attractive theoretical properties. Another exciting question is whether our results on the optimality of supervised feature selection can be transferred to techniques for unsupervised feature selection on frequent subgraphs [6] (S. Nijssen, personal communication (2008, 2009)).

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 *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, Santiago de Chile, Chile, 1994.

[2] A. Bairoch. The enzyme database in 2000. *Nucleic Acids Research*, 28(1):304–305, 2000.

[3] C. Borgelt and M. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 211–218, Maebashi City, Japan, 2002.

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

[5] E. Boros, T. Horiyama, T. Ibaraki, K. Makino, and M. Yagiura. Finding essential attributes from binary data. *Annals of Mathematics and Artificial Intelligence*, 39(3):223–257, 2003.

[6] B. Bringmann and A. Zimmermann. One in a million: picking the right patterns. *Knowledge and Information Systems*, 18:61–81, 2008.

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

[8] H. Cheng, X. Yan, J. Han, and C. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE'07)*, pages 716–725, Istanbul, Turkey, 2007.

[9] M. Dash, H. Liu, and H. Motoda. Consistency based feature selection. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications (PADKK'00)*, pages 98–109, London, UK, 2000. Springer.

[10] 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.

[11] P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, Jul 2003.

[12] L. Ein-Dor, O. Zuk, and E. Domany. Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer. *Proceedings of the National Academy of Sciences of the United States of America*, 103(15):5923–5928, Apr 2006.

[13] 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, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, pages 230–238, Las Vegas, NV, USA, 2008. ACM.

[14] C. Guestrin, A. Krause, and A. Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*, pages 265–272, Bonn, Germany, 2005.

[15] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, pages 549–552, Melbourne, FL, USA, 2003.

[16] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*, pages 13–23, Lyon, France, 2000.

[17] N. Jin, C. Young, and W. Wang. Graph classification based on pattern co-occurrence. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management (CIKM'09)*, pages 573–582, Hong Kong, China, 2009. ACM.

[18] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pages 321–328, Washington, DC, USA, 2003.

[19] S. Kramer, L. Raedt, and C. Helma. Molecular feature mining in HIV data. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 136–143, San Francisco, CA, USA, 2001.

[20] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI'05)*, pages 324–331, Endinburgh, Scotland, 2005.

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

[22] 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, Canada, Dec. 2004.

[23] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM'01)*, pages 313–320, San Jose, CA, USA, 2001.

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

[25] S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 647–652, Seattle, WA, USA, 2004.

[26] N. Przulj. Biological network comparison using graphlet degree distribution. In *Proceedings of the 5th European Conference on Computational Biology (ECCB'06)*, Eilat, Israel, September 2006.

[27] P. Radivojac, Z. Obradovic, A. K. Dunker, and S. Vucetic. Feature selection filters based on the permutation test. In *Proceedings of the 15th European Conference on Machine Learning (ECML'04), Pisa, Italy*, pages 334–346, Porto, Portugal, 2004. Springer.

[28] H. Saigo, N. Krämer, and K. Tsuda. Partial least squares regression for graph mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, pages 578–586, Las Vegas, NV, USA, 2008. ACM.

[29] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. gBoost: a mathematical programming approach to graph classification and regression. *Machine Learning*, 75(1):69–89, 2009.

[30] N. Shervashidze and K. M. Borgwardt. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems 22 (NIPS'09)*, pages 1660–1668, Vancouver, BC, Canada, 2009.

[31] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, Clearwater Beach, FL, USA, 2009.

[32] M. Thoma, H. Cheng, A. Gretton, J. Han, H.-P. Kriegel, A. Smola, L. Song, P. S. Yu, X. Yan, and K. Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *Proceedings of the 9th SIAM International Conference on Data Mining (SDM'09)*, pages 1075–1087, Sparks, NV, USA, 2009.

[33] K. Tsuda. Entire regularization paths for graph data. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*, pages 919–926, Oregon, OR, USA, 2007.

[34] 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.

[35] N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph patterns from semistructured data. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 458–465, Maebashi City, Japan, 2002.

[36] N. Wale and G. Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM'06)*, pages 678–689, Hong Kong, China, 2006.

[37] S. Wernicke. A faster algorithm for detecting network motifs. In *Proceedings of the 5th Workshop on Algo-*

rithms in Bioinformatics (WABI'05), pages 165–177, Palma de Mallorca, Mallorca, Spain, 2005.

[38] X. Yan, H. Cheng, J. Han, and P. S. Yu. Mining significant graph patterns by leap search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*, pages 433–444, Vancouver, BC, Canada, 2008.

[39] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 721–724, Maebashi City, Japan, 2002.

[40] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pages 412–420, Nashville, TN, USA, 1997. Morgan Kaufmann Publishers Inc.

[41] A. Zimmermann and B. Bringmann. CTC - correlating tree patterns for classification. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*, pages 833–836, Houston, TX, USA, 2005.