Frequent Subgraph Retrieval in Geometric Graph Databases

Sebastian Nowozin
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
sebastian.nowozin@tuebingen.mpg.de

Koji Tsuda
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
koji.tsuda@tuebingen.mpg.de

Abstract

Discovery of knowledge from geometric graph databases is of particular importance in chemistry and biology, because chemical compounds and proteins are represented as graphs with 3D geometric coordinates. In such applications, scientists are not interested in the statistics of the whole database. Instead they need information about a novel drug candidate or protein at hand, represented as a query graph. We propose a polynomial-delay algorithm for geometric frequent subgraph retrieval. It enumerates all subgraphs of a single given query graph which are frequent geometric ϵ-subgraphs under the entire class of rigid geometric transformations in a database. By using geometric ϵ-subgraphs, we achieve tolerance against variations in geometry. We compare the proposed algorithm to gSpan on chemical compound data, and we show that for a given minimum support the total number of frequent patterns is substantially limited by requiring geometric matching. Although the computation time per pattern is larger than for non-geometric graph mining, the total time is within a reasonable level even for small minimum support.

1. Introduction

Frequent subgraph mining algorithms (FSM) such as gSpan [21] and extensions have been successfully applied to different applications such as chemical compound classification [10] and molecular QSAR analysis [16], graph clustering [20] and computer vision [13]. In FSM, we deal with a large database of attributed graphs, where nodes and edges have discrete labels. The purpose of FSM is to enumerate all frequently appearing subgraphs in the database. Typically, all subgraphs whose frequency is above a minimum support threshold are enumerated. FSM helps users to analyze the database and can be used to create a feature space for subsequent machine learning algorithms as well.

One domain of graph mining is *geometric graph mining*, where each node of a graph has 2D or 3D coordinates and subgraph patterns have to match geometrically under a given transformation class [11, 4, 1]. To deal with real world data, a geometric pattern is matched with a certain tolerance ϵ, because a perfect match does not happen in real data. Due to the tolerance, geometric mining is significantly more difficult than ordinary graph mining. There are many different geometric representations of the same pattern, all of which are ϵ-isomorphic to each other. To enumerate all patterns satisfying minimum support, we need to define a canonical pattern for these equivalent patterns. We call this problem the *pattern ambiguity problem*. To the best of our knowledge, there is no effective solution to this problem.

Extracting such geometric patterns from molecular 3D structures is one of the central topic in computational biology, and numerous approaches have been proposed. Most of them are optimization methods, which detect one pattern at a time by minimizing a loss function (e.g., [14, 15, 6]). They are different from our approach enumerating all patterns satisfying a certain geometric criterion. In particular, they do not have a minimum support constraint. Instead they try to find a motif that matches all graphs.

Kuramochi et al. [11] used a heuristic way to extend a geometric pattern, so there is no explicit description of the set they enumerated. Deshpande et al. [4] used non-geometric graph patterns from chemical molecules augmented with average inter-atomic distances. Their patterns contain geometric information, but only in a summarized form. Arimura et al. [1] proposed a method to enumerate all maximum geometric patterns, but they assumed zero tolerance and therefore small variations of the vertex coordinates are treated as different patterns. Huan et al. [8] applied an enumerative approach to detect structural motifs, but proteins are represented as graphs with discrete labels in advance by thresholding the distances among atoms.

We deal with the problem of *frequent subgraph retrieval* (FSR), which is similar to but different from frequent subgraph mining. In mining, frequently appearing subgraph patterns in the graph database are enumerated. In retrieval, the user has a *query graph* apart from the database. FSR is frequent subgraph mining, where the set of patterns is re-

stricted to the set of subgraphs of the query graph. In biology and chemistry applications, this setting makes sense, because users access databases to gain some knowledge about a newly discovered chemical compound or protein at hand. In such a case, geometric patterns that do not match the query graph are not useful. Furthermore, the statistics or characteristics of the whole database are not of interest here, which is strikingly different from the market basket analysis setting addressed by the usual frequent mining algorithms.

Our FSR algorithm is based on reverse search [2], and the pattern set to be enumerated is unambiguously described as an exact subgraph of the query graph, thereby avoiding the pattern ambiguity problem. Straightforwardly, one can create a retrieval version of the popular gSpan frequent subgraph mining algorithm [21], which we call gSpan-retrieval. It does not use geometric information and the search tree is pruned as soon as the pattern is not included in the given query graph. However, gSpan-retrieval is exponential-delay, due to the minimum DFS code checks in the gSpan algorithm. While our proposed method is slower than gSpan-retrieval in practice, we prove that our method has polynomial-delay [9].

In experiments, we used four chemical datasets with 3D atomic coordinates. In efficiency comparison with gSpan-retrieval, we will show that the geometric information reduces the number of frequent patterns found significantly. While the computational time per pattern of our method is much larger than that of gSpan-retrieval, when the minimum support threshold is decreased, the number of patterns of gSpan explodes super-exponentially to an intractable level, whereas our method can keep the number of patterns small due to the discriminative geometry required for matching. To prove that our small set of geometric patterns are still informative, naive Bayes prediction of chemical activities is performed based on our patterns and the patterns by gSpan-retrieval. On the four data sets examined, the prediction accuracy is improved significantly on three sets.

2. Method

In order to describe our method we first define the notion of geometric graph and geometric ϵ-subgraph.

Definition 1 (Geometric graph) *A labeled, connected, undirected geometric graph $G = (V, E, \mathcal{L}^V, \mathcal{L}^E, C^V)$ consists of a vertex set $V \subset \mathbb{N}$, an edge set $E \subseteq V \times V$, a vertex labeling $\mathcal{L}^V : V \to \mathbb{N}$, an edge labeling $\mathcal{L}^E : E \to \mathbb{N}$ and vertex coordinates $C^V : V \to \mathbb{R}^d$, assigning each vertex a vector in $\mathbb{R}^d$. Let $\mathcal{G}$ denote the set of all possible geometric graphs and let $\emptyset \in \mathcal{G}$ denote the empty graph.*

Definition 2 (Geometric ϵ-subgraph relation) *Given two geometric graphs $g_1 = (V_1, E_1, \mathcal{L}^{V_1}, \mathcal{L}^{E_1}, C^{V_1})$, $g_2 = (V_2, E_2, \mathcal{L}^{V_2}, \mathcal{L}^{E_2}, C^{V_2})$, $g_1, g_2 \in \mathcal{G}$, and a geometric tolerance $\epsilon \geq 0$, we define $g_1 \subseteq_\epsilon g_2$ to be true if and only if $\exists(m, T)$, $m : V_1 \to V_2$ injective and complete on V_1, $T : \mathbb{R}^d \to \mathbb{R}^d$ with T being a rigid transformation, such that*

1. *$\forall i \in V_1 : \mathcal{L}^{V_1}(i) = \mathcal{L}^{V_2}(m(i))$, the vertex labels are matched, and*
2. *$\forall (i,j) \in E_1 : (m(i), m(j)) \in E_2, \mathcal{L}^{E_1}(i,j) = \mathcal{L}^{E_2}(m(i), m(j))$, all edges of g_1 also exist in g_2 with the correct label, and*
3. *$\forall i \in V_1 : \|C^{V_1}(i) - T(C^{V_2}(m(i)))\| \leq \epsilon$, the geometry matches under transformation up to the required tolerance.*

We write $g_2 \supseteq_\epsilon g_1$ iff $g_1 \subseteq_\epsilon g_2$. Also, we write $g_1 \subset_\epsilon g_2$ iff $g_1 \subseteq_\epsilon g_2$ and $|V_1| < |V_2|$.

Problem 1 (Frequent geometric subgraph retrieval) *For a given geometric* query graph $q \in \mathcal{G}$, *a geometric graph database $G = \{g_1, \ldots, g_{|G|}\}$, $g_i \in \mathcal{G}$, a minimum support threshold $s > 0$, and a geometric tolerance $\epsilon \geq 0$, find all $g \in \mathcal{G}$ with $g \subseteq_0 q$ such that g is frequent enough in G, i.e. $|\{i = 1, \ldots, |G| : g \subseteq_\epsilon g_i\}| \geq s$.*

Comparing Problem 1 with a general frequent mining problem we note that it allows tolerant matching into the database for counting the support, but requires exact matching in the query graph. This simplification with regards to the query graph makes it tractable to solve the problem efficiently, as the set $\{g \in \mathcal{G} : g \subseteq_0 q\}$ is finite. This also overcomes the pattern ambiguity problem by making the query graph the canonical reference.

Our proposed method solves Problem 1 and is composed of two parts, i) the enumeration of all $g \subseteq_0 q$ by means of reverse search and, ii) the geometric matching $g \subseteq_\epsilon g_i$ within the graph database. By integrating these two steps, we can prune large parts of the search space efficiently without losing any frequent ϵ-subgraph. We now discuss the two parts separately.

2.1. Reverse Search Enumeration

We generate candidate frequent geometric subgraphs by recursively enumerating the set of all exact geometric subgraphs of the query graph, i.e. $\{g \in \mathcal{G} : g \subseteq_0 q\}$ and explicitly checking the frequency in the geometric graph database.

Enumerating all subgraphs of a given graph is itself a non-trivial problem. To solve the subgraph enumeration problem efficiently, we use the *reverse search* algorithm [2]. We establish *polynomial delay* and *output-polynomial* time complexity for our algorithm.

In the general reverse search framework, we enumerate elements from a set $\mathcal{X}$ by means of a *reduction mapping* $f : \mathcal{X} \to \mathcal{X}$. The reduction mapping reduces any element from $\mathcal{X}$ to a "simpler" one in the neighborhood of

the input element. The mapping is chosen such that when it is applied repeatedly, we eventually reduce it to some *solution elements* in the set $\mathcal{S} \subset \mathcal{X}$. Formally, we write $\forall x \in \mathcal{X} : \exists k \geq 0 : f^k(x) \in \mathcal{S}$. In our subgraph enumeration problem, we identify $\mathcal{X}$ with the set of all possible connected labeled graphs $\mathcal{G}$. This setting is similar to the enumeration of all connected induced subgraphs in Avis and Fukuda [2]; but here we distinguish connected subgraphs also on the presence of edges. To this end, the mapping $f : \mathcal{G} \to \mathcal{G}$ removes either one edge or one vertex-edge. By evaluating the mapping repeatedly the graph is shrunk to the empty graph. Thus, here we have $\mathcal{S} = \{\emptyset\}$. Precisely, we define the reduction mapping as follows.

Definition 3 (Reduction Mapping f_q) *Given a geometric query graph $q \in \mathcal{G}$, a geometric graph $g \in \mathcal{G}$ such that $g \subseteq_0 q$, i.e. g is an exact geometric subgraph of the query graph, and let $U^{E_q} : E_q \to \mathbb{N}$ and $U^{V_q} : V_q \to \mathbb{N}$ be an index labeling assigning unique indices to edges and vertices of q, respectively. We then define the mapping $f_q : \mathcal{G} \to \mathcal{G}$ which reduces g to $f_q(g)$, with $g \supset_0 f_q(g)$ by performing the following actions.*

1. *If g contains one or more edges forming a cycle, the graph $f_q(g)$ is the same as g, with the edge corresponding to the highest-index cycle-edge in q removed.*
2. *If g contains no edges forming a cycle, the graph $f_q(g)$ is the same as g, with the leaf vertex corresponding to the highest-index vertex in q and its adjacent edge removed, if it exist. A vertex is said to be a* leaf*, if it only has one adjacent edge.*

Clearly the above mapping is well-defined because each possible graph $g \subseteq_0 q$ is successively reduced to the empty graph in a unique way. By considering $f_q(g)$ for all possible $g \in \mathcal{G}$, $g \subseteq_0 q$, a *reduction tree* is defined, with $\emptyset \in \mathcal{G}$ being the root node. Reverse search performs subgraph enumeration by inverting the reduction mapping such that the tree is explored from the root node towards the leaves.

The inverse mapping $f_q^{-1} : \mathcal{G} \to \mathcal{G}^*$ generates for a given graph $g \subseteq_0 q$ a set X of enlarged graphs $g' \in X$, $g' \supset_0 g$, $g' \subseteq_0 q$ such that $f_q(g') = g$. The inverse mapping follows uniquely from definition 3.

By recursively evaluating f_q^{-1} we can traverse the tree and thus efficiently enumerate all subgraphs of q without generating duplicate subgraphs. We consider the second subproblem of geometrically matching the enumerated graphs in the graph database.

2.2. Matching with tolerances

Checking the geometric ϵ-subgraph relation requires one to test whether $g \subseteq_\epsilon g_i$ is satisfied. In this paper we limit ourselves to the practically relevant case of 3D vertex coordinates, i.e. $C^V : V \to \mathbb{R}^3$ and rigid transformations. However, our algorithm works for arbitrary dimensions and transformations.

For the 3D case, we parametrize the general rigid transformation matrix using homogeneous coordinates by a translation vector (t_x, t_y, t_z) and three rotation angles (α, β, γ) around the canonical axes. The resulting matrix is shown in Figure 1. Consider the question whether there exist a set $(\alpha, \beta, \gamma, t_x, t_y, t_z)$ of parameters such that we can *align* two given point sets $X = \{x_1, \ldots, x_N\}$, $Y = \{y_1, \ldots, y_N\}$, $x_i, y_i \in \mathbb{R}^3$ such that the norm between each original point and the matched point under the transformation satisfies a given tolerance $\epsilon \geq 0$, i.e. whether $\exists\ (\alpha, \beta, \gamma, t_x, t_y, t_z) : \forall i = 1, \ldots, N : \|x_i^h - T(\alpha, \beta, \gamma, t_x, t_y, t_z) y_i^h\| \leq \epsilon$, where x^h and y^h are the homogeneous extensions of x and y, respectively.[1] The "no-effect" transformation is simply $T_0 := T(0, 0, 0, 0, 0, 0)$. We directly minimize the squared error $\sum_{i=1}^N \|x_i^h - T(\alpha, \beta, \gamma, t_x, t_y, t_z) y_i^h\|^2$ using BFGS [3]. This six-dimensional minimization problem is non-convex, however for the case of $\mathbb{R}^3$ we consider here we are guaranteed to obtain the optimal solution, essentially because we add one vertex at a time and all six parameters can be uniquely recovered from three or more non-collinear vertex correspondences. By solving the above optimization problem we can determine whether a good alignment exists. Now we have the two ingredients – subgraph enumeration and geometric matching – necessary to state the whole algorithm.

2.3. Algorithm

The algorithm uses location sets $L_G(g)$ which map the occurrence of a geometric subgraph g into the geometric graph database G. Let $L_G(g) = \{(i, m, T) : g \subseteq_\epsilon g_i$ with node-matching m under rigid transformation $T\}$. Algorithm 1 consists of three functions, FREQGEO, MINE and FILTERLOCATION. The main function is MINE,

Algorithm 1 Frequent Geometric Subgraph Retrieval

Input: database $G = (g_1, \ldots, g_{|G|}), g_i \in \mathcal{G}$,
query $q \in \mathcal{G}$, minsup $s \geq 1$, tolerance $\epsilon \geq 0$.

Algorithm:

```
1: function FREQGEO(G, q, s, ε)
2:     MINE(G, ∅, q, s, ε, {(i, ∅, T_0) : i = 1, ..., |G|})
3: end function
4: function MINE(G, g, q, s, ε, L_G(g))
5:     if |L_G(g)| < s return                ▷ support satisfied?
6:     REPORT(g, L_G(g), g is visited for the first time)
7:     X ← f_q^{-1}(g)                       ▷ All valid extensions of g in q
8:     for g' ∈ X do                         ▷ For all extensions, recurse
9:         MINE(G, g', s, τ,
10:             FILTERLOCATION(G, L_G(g), g', g, ε))
11:    end for
12: end function
```

[1] $x^h = [x', 1]'$.

$$T(\alpha,\beta,\gamma,t_x,t_y,t_z) \quad := \quad \begin{bmatrix} \cos(\alpha)\cos(\beta), & \cos(\alpha)\sin(\beta)\sin(\gamma)-\sin(\alpha)\cos(\gamma), & \cos(\alpha)\sin(\beta)\cos(\gamma)+\sin(\alpha)\sin(\gamma), & t_x \\ \sin(\alpha)\cos(\beta), & \sin(\alpha)\sin(\beta)\sin(\gamma)+\cos(\alpha)\cos(\gamma), & \sin(\alpha)\sin(\beta)\cos(\gamma)-\cos(\alpha)\sin(\gamma), & t_y \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma), & t_z \\ 0, & 0, & 0, & 1 \end{bmatrix}$$

Figure 1. General 3D rigid transformation matrix used.

which is called once for each node visited in the reverse search tree. The main loop in line 8 calls MINE for all child nodes of the current node, effectively walking the reverse search tree in a depth-first fashion. While this tree walk is performed, a *location list* of occurrences of the current geometric subgraph is kept and updated by means of the FILTERLOCATION function. The FILTERLOCATION function is called for each $g' \supset_0 g$ extension where all matches of g are recorded in a location list. It filters the location list such that only those matches in $g_i \in G$ are kept which fulfill $g' \supseteq_\epsilon g_i$.

The predicate "g is visited for the first time" in line 6 of the algorithm tests whether it is globally the first visit of g as a ϵ-subgraph. Without limit of generality we can assume $q \in G$, then this check can be performed in $O(1)$, as then all occurrences of $g \subseteq_\epsilon q$ are in the location list $L_G(g)$ in a unique order; therefore we simply have to check if g is the first element in q's location list.[2] Based on the value of this predicate, the REPORT function can decide whether to report only the first occurrence of $g \subseteq_\epsilon q$, immediately returning whenever the predicate is false, or to report every occurrence of $g \subseteq_0 q$, $g' \subseteq_0 q$, even if both $g \subseteq_\epsilon g'$ and $g' \subseteq_\epsilon g$ hold. For all experiments we use the first option.

Complexity Analysis. We now state that Algorithm 1 for the important case of coordinates in $\mathbb{R}^3$ enumerates all frequent geometric subgraphs of a given query graph with *polynomial delay* and therefore in *output polynomial time*.

Theorem 1 (Polynomial delay) *For a geometric graph database G of N geometric graphs, a query graph $q \in \mathcal{G}$, a minimum support $s \geq 1$ and matching tolerance $\epsilon \geq 0$, the time between two successive calls to* REPORT *in line 6 is bounded by a polynomial in the dimensions of input data.*

Proof sketch. (ommitted in the short version of the paper.)

For pattern mining algorithms, output polynomial time follows from polynomial delay.

3. Experiments and Results

We will now evaluate the proposed algorithm in three experiments. As data sets we use chemical compound data sets for structure-activity relationship (SAR) prediction of Sutherland, O'Brien and Weaver [19]. The data sets contain specific molecules; the BZR data set contains ligands for the benzodiazepine receptor, the COX data set contains cyclooxygenase-2 inhibitors, the DHFR data set contains inhibitors of dihydrofolate reductase and the ER data set contains estrogen receptor ligands. Besides containing related molecules, the data set additionally comes with activity labels. Not all of the samples present in the data sets are labeled; the labeled samples have a binary label, being either -1 or 1. Table 1 shows the number of molecules in the different data sets as well as the number of labeled samples in the provided training and testing splits. In the first experiment we assess empirically the runtime and number of retrieved subgraphs of our algorithm and compare it to gSpan-retrieval, a modified version of the popular gSpan graph mining algorithm [21]. The second experiment is concerned with the discriminativeness of the geometric subgraph features. In the third experiment we show a retrieval application in which similar compounds are retrieved using geometric subgraph features, where similarity is measured by means of the χ^2-distance on histograms of geometric subgraph features.

Dataset name	Number of Chemicals	Training samples	Test samples
BZR 3D	405	181	125
COX2 3D	467	178	125
DHFR 3D	756	233	160
ER 3D	1009	266	180

Table 1. Datasets used in the experiments.

3.1. Experiment 1: Runtime

To assess the empirical runtime of our proposed algorithm we conduct the following experiment. We select ten random chemicals from the 405 molecules of the BZR 3D database. For minimum supports in the range 20 to 405 we perform subgraph retrieval using FreqGeo and gSpan-retrieval, a modified version of the open-source gSpan implementation available in the gboost toolbox[3]. We modify gSpan such that at each extension step an additional pruning condition is checked for: the current frequent subgraph must appear also in a given query graph. Thus, the two algorithms perform the same task, except that FREQGEO requires the 3D geometry to match, while the modified gSpan retrieval algorithm does not use 3D geometry at all. The runtime over all ten query molecules is averaged. Regarding the runtime, we expect a tradeoff between the following two effects in FREQGEO geometric subgraph retrieval, i)

[2] If $q \notin G$, we define $G' = G \cup \{q\}$ and perform mining on G' with minimum support increased by one.

[3] `http://www.kyb.tuebingen.mpg.de/bs/people/nowozin/gboost/`

more expensive extension and matching due to the 3D geometry check, ii) more discriminative matching for larger subgraphs due to 3D geometry.

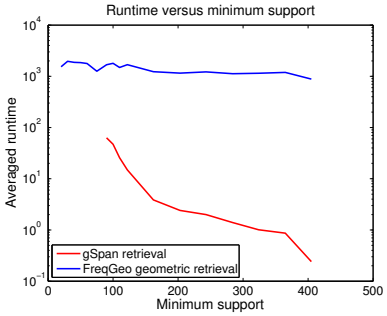

Figure 2. Runtime: gSpan-retrieval, FreqGeo.

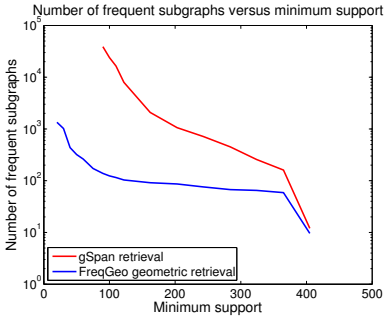

Figure 3. Avg. count of frequent subgraphs.

Figure 2 shows the averaged runtime for both gSpan-retrieval and FREQGEO subgraph retrieval, Figure 3 the number of frequent subgraphs found. We see that gSpan is faster by roughly two orders of magnitude for high values of minimum support and by roughly one order to magnitude for low minimum support. For a minimum support below 90, the used implementation of gSpan-retrieval starts to exhausts the main memory of our system, whereas we use FREQGEO to mine down to a minimum support of 20.

The smaller number of patterns returned by FREQGEO provides a better interpretability: a small set of geometric subgraphs returned by our algorithm (around 1000 for a minimum support of 20) is more interpretable than a large one (around 30,000 for a minimum support of 90) of non-geometric subgraphs returned by gSpan-retrieval. Thus, geometry aids interpretability.

3.2. Experiment 2: Geometric Features

We have shown that the number of patterns is reduced dramatically by introducing the geometric constraints. However, it could be the case that important information is lost by pattern set reduction. To check this, we perform simple supervised classification to compare our geometric patterns with non-geometric ones.

In principle, retrieval methods are not directly amenable to supervised learning, because the feature space created by patterns depends on a query graph. It would be possible to create a feature space by collecting patterns retrieved from several query graphs, but then the problem is how to select query graphs optimally. For supervised classification based on non-geometric patterns, please refer to [17, 13] and references therein. For 3D QSAR analysis, several prediction methods such as CoMFA have been proposed [7, 18]. In this experiment, we would like to measure the remaining information in our geometric patterns, instead of confronting state-of-the-art methods in accuracy.

For each of the chemical compound database used in the previous experiment, the training and testing subsets are combined. On these four geometric graph sets we perform geometric subgraph retrieval on ten randomly selected chemical compounds from the training set of each database. For each graph set, we use the class labels of only the training subset to train a naive Bayes classifier on the retrieved features. The classifier is used to predict the labels of the test sets. The same experiment is performed with the retrieval version of gSpan with identical settings but not making use of geometric information. The subgraph retrieval, both for FREQGEO and gSpan-retrieval is performed using a minimum support of 75. For the naive Bayes classifier, we use the class conditional m-estimate of the probability of a graph s_j appearing as subgraph of a given test graph g as $p(s_j \subseteq_\epsilon g|y=1) \approx \frac{\sum_{n=1}^{N} I(s_j \subseteq_\epsilon g_i)+mp}{\sum_{n=1}^{N} I(y_n=1)+m}$, where m is the *equivalent sample size* and p is the prior probability, see e.g. [12]. We use $m = 1$ and $p = \frac{1}{2}$ for all experiments. The probability estimate for $p(s_j \subseteq_\epsilon g|y=-1)$ is used analogously. With these per-feature probability estimates we consider the naive Bayes classifier log-odds $\phi(g) = \log \frac{p(y=1|g)}{p(y=-1|g)} = \log \frac{p(y=1)}{p(y=-1)} + \sum_j \log \frac{p(s_j \subseteq_\epsilon g|y=1)}{p(s_j \subseteq_\epsilon g|y=-1)}$. If the log-odd is positive, g is predicted to be in the positive class, otherwise its negative. To evaluate the predictive performance for the test set, we consider the ROC Area Under Curve (AUC) and ROC Equal Error Rate (EER) [5]. The results shown in Table 2 demonstrate that for some of the data sets using geometry improves the performance of our simple classifier. It implies that the information required in activity prediction is well preserved in our small geometric pattern set.

	FREQGEO		gSpan retrieval	
Dataset	ROC AUC	ROC EER	ROC AUC	ROC EER
BZR 3D	**0.6585±0.026**	**0.6559±0.029**	0.6096±0.025	0.6125±0.024
COX2 3D	**0.7109±0.010**	**0.6325±0.012**	0.6952±0.029	0.6159±0.019
DHFR 3D	**0.7447±0.030**	**0.6732±0.030**	0.6868±0.051	0.6311±0.060
ER 3D	0.7241±0.065	0.6743±0.061	0.7222±0.064	0.6741±0.050

Table 2. Averaged test set classifications.

3.3. Experiment 3: Retrieval by Similarity

As last experiment we assess the retrieval application. In this setting, a query graph is given to the algorithm and

similar graphs should be retrieved from the database. In order to rank the graphs from a database G by similarity to a query graph q, we perform frequent geometric subgraph retrieval on q in the database G and consider the histogram h^q and h^g of occurrences of the retrieved frequent subgraphs in q and $g \in G$. The χ^2-distance is a natural metric for these histogram representation, and we simply rank all database graphs by the χ^2-distance as $\chi^2(h^q, h^g) = \sum_{\{n:h_n^q+h_n^g>0\}} \frac{(h_n^q-h_n^g)^2}{h_n^q+h_n^g}$, where h_n is the number of occurrences of the n'th frequent geometric subgraph found. In Table 3 the top five matching graphs for a set of query graphs in the BZR 3D database are shown. We used a minimum support of 20. While subjective, the order demonstrates the ability of retrieving similar chemical compounds, by graph structure and geometry.

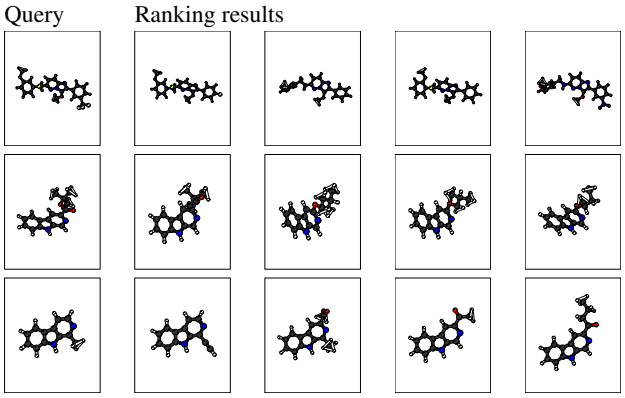

Table 3. Rankings on the BZR dataset.

4. Conclusion

We have proposed and experimentally evaluated an algorithm able to retrieve frequent geometric subgraph patterns. The retrieved subgraphs are allowed to match up to a prespecified geometric tolerance. We believe our contribution is a novel and clean method to robustly use geometry information for subgraph retrieval and will be especially useful in computational biology. Previous approaches require either exact matches and thus are not robust, or discretize or heuristically summarize geometric information into edge or vertex attributes, the effects of which on are hard to understand. The proposed algorithm overcomes the *pattern ambiguity problem* of these previous approaches. Instead of asking for globally frequent geometric subgraphs in a graph database, we remove all ambiguities by establishing a single query graph as reference for geometric matching. This simplifies the problem, but it also makes sense when statistics such as similarity to the query graph are the object of interest, for example in classification applications where mining is used to generate meaningful features. Our implementation is available at `http://www.kyb.mpg.de/bs/people/nowozin/freqgeo/`.

Acknowledgments. This work is funded in part by the EU CLASS project, IST 027978 and the PASCAL Network of Excellence.

References

[1] H. Arimura, T. Uno, and S. Shimozono. Time and space efficient discovery of maximal geometric subgraphs. In *Discovery Science 2007*, pages 42–55, 2007.
[2] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65:21–46, 1996.
[3] D. P. Bertsekas. *Nonlinear Programming*. Athena, 1999.
[4] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. Knowledge and Data Engineering*, 17(8):1036–1050, 2005.
[5] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, July 2001.
[6] L. Holm and C. Sander. Dali: a network tool for protein structure comparison. *Trends Biochem Sci*, 20(11):478–480, 1995.
[7] H. Hong, H. Fang, Q. Xie, R. Perkins, D. Sheehan, and W. Tong. Comparative molecular field analysis (CoMFA) model using a large diverse set of natural, synthetic and environmental chemicals for binding to the androgen receptor. *SAR and QSAR in Environmental Research*, 14(5-6):373–388, 2003.
[8] J. Huan, D. Bandyopadhyay, W. Wang, J. Snoeyink, J. Prins, and A. Tropsha. Comparing graph representations of protein structure for mining family-specific residue-based packing motifs. *J Comput Biol*, 12(6):657–671, 2005.
[9] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–123, Mar. 1988.
[10] T. Kudo, E. Maeda, and Y. Matsumoto. An application of boosting to graph classification. In *NIPS*, 2004.
[11] M. Kuramochi and G. Karypis. Discovering frequent geometric subgraphs. In *ICDM*, pages 258–265, 2002.
[12] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
[13] S. Nowozin, K. Tsuda, T. Uno, T. Kudo, and G. H. Bakır. Weighted substructure mining for image analysis. In *CVPR*, 2007.
[14] B. J. Polacco and P. C. Babbitt. Automated discovery of 3d motifs for protein function annotation. *Bioinformatics*, 22(6):723–730, 2006.
[15] R. B. Russell. Detection of protein three-dimensional side-chain patterns: new examples of convergent evolution. *J Mol Biol*, 279(5):1211–1227, 1998.
[16] H. Saigo, T. Kadowaki, and K. Tsuda. A linear programming approach for molecular QSAR analysis. In *MLG*, 2006.
[17] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. A mathematical programming approach to graph classification and regression. *Machine Learning*, 2008.
[18] L. Shi, H. Fang, W. Tong, J. Wu, R. Perkins, and R. Blair. QSAR models using a large diverse set of estrogens. *J. Chem. Inf. Comput. Sci.*, 41:186–195, 2001.
[19] J. J. Sutherland, L. A. O'Brien, and D. F. Weaver. Spline-fitting with a genetic algorithm: A method for developing classification structure-activity relationships. *Journal of Chemical Information and Computer Sciences*, 43(6):1906–1915, 2003.
[20] K. Tsuda and T. Kudo. Clustering graphs by weighted substructure mining. In *ICML*, 2006.
[21] X. Yan and J. Han. gspan: graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.