A New proposal for Graph Classification using Frequent Geometric Subgraphs

Andrés Gago-Alonso[a,*], Alfredo Muñoz-Briseño[a], Niusvel Acosta-Mendoza[a]

[a] *Advanced Technologies Application Center, 7a ♯ 21812, Siboney, Playa, CP: 12200, Havana, Cuba*
E-mail: {agago, amunoz, nacosta}@cenatav.co.cu

Abstract

Geometric graph mining has bees identified as a need in many applications. This technique detect patterns with some tolerance under a geometric transformation. To meet this need, some graph miners have been developed for detecting frequent geometric subgraphs. However, there are few works for applying this kind of geometric patterns as feature for classification tasks. In this paper, a new geometric graph miner and a framework for using frequent geometric subgraphs in classification, are proposed. Our solution was tested in two real collections. The experimentation on these collections shows that our proposal gets better results than graph-based image classification using non-geometric graph miners.

Keywords: Mining methods and algorithms, classification, clustering, frequent subgraph mining

1. Introduction

In recent years, several authors have developed techniques and tools for facing tasks related with converting large volumes of data into useful information [19]. Frequent pattern discovery is an example of such techniques [37], when the objects of these datasets are represented as graphs [25]. These techniques has been successfully employed for classification tasks [1, 12, 20], using frequent subgraphs as features for representing objects. As example of this, in literature we can find classification of images [1, 8, 29, 30, 20], texts [21] and chemical compounds [6, 18].

In the graph collections used in these applications, geometric features of vertices can be considered for modelling the objects. For example, atom coordinates are included in some molecular datasets [24], the spatial coordinates of regions of interest are considered in image collections [1], among others [32]. These datasets are commonly affected by some geometric shaped distortions of similar structures in several objects. Therefore, the application of a mechanism for dealing with such distortions can help to improve classification results in geometric graph databases.

[*] Corresponding author

Distortions in data is one of the challenges for developing classifiers based on frequent subgraphs [1, 6, 13, 18]. Thus, Deshpande *et al.* [6] proposed a classification that use frequent geometric subgraphs as features for representing objects. Moreover, some tools for geometric graph mining are reported in the literature [24, 28].

Geometric graph mining has become in an interesting problem in data mining. This kind of mining is performed taking into account some tolerance under a geometric transformation. The first geometric graph miner was gFSG [23, 24], which finds frequent geometric subgraphs in collections of geometric graphs. This algorithm was followed by MaxGeo [3] and FreqGeo [28], which were conceived for detecting particular subsets of frequent geometric subgraphs.

In this paper, a new algorithm for frequent geometric subgraph mining called gsmFil, is presented. This proposal is based on a pattern-growth depth-first strategy for traversing the search space.

In literature, we can find geometric graphs whose edges are determined by some kind of geometric relation between the vertices. As example of this, we have interval and Gabriel graphs, Delaunay triangulations, and so on. All these geometric structures have some properties that could be exploited in the mining process. However, the modelling of objects by graphs not always results in a graph with some specific properties, as Delaunay triangulations for instance. Considering this, the algorithm for frequent geometric subgraph mining presented in this work, does not assumes the existence of any additional information in the structure of the processed graphs.

On the other hand, there are three main approaches in algorithms that use pattern recognition on classification tasks: the statistical, syntactical and structural approaches [5]. In statistical pattern recognition, the patterns are represented by feature vectors that can be understood as a point in the n-dimensional real space. This representation offers useful properties as the mathematical operations available in a vector space, which can be efficiently performed. The syntactical approaches, can be used to discriminate between different object classes because they are encoded as a natural language elements. The structural approach is based on symbolic data structures, such as strings, trees or graphs for pattern representation. The last one considers spatial properties and characteristics of the objects and inter-relationships between its component parts.

Taking into account these approaches, several works based on these types of representations of objects for classification tasks, have been reported [7, 31, 34, 33]. The idea of mapping the patterns identified into a dissimilarity space using embedded sets of the n--dimensional feature space vector, was presented by Duin *et al.* [7] and Pekalska*et al.* [31]. This approach was extended to map string representations into vectorial spaces [34], which later was generalized to the domain of graphs [33]. Several authors [33, 5] propose a combination of both, structural and statistical approaches. The first one provide the preservation of representational generalization of graphs. Moreover, the statistical pattern recognition allows to use many clustering and classification algorithms reported, and the high performance in mathematical computation in feature vector spaces. In these works, significant improvement in the classification results are archived.

Another improve proposed in this paper, is the definition of a novel framework for geometric graph classification that uses the fusion of both statistical and structural approaches. The main novelty of our proposal is that the described framework uses frequent geometric subgraphs identified in a geometric graph mining process, as features for representing objects and build the feature vectors. The classification results of our proposal are evaluated in a well know geometric graph dataset [32].

The basic outline of this paper is as follows. Section 2 provides some basic concepts and related works. The new geometric method for frequent geometric subgraph mining is provided in Section 3. This section also introduces the gsmFil algorithm. The framework for graph-based classification is introduced in Section 4 as a case study to evaluate the geometric algorithm proposed. The experimental results in two real collections are presented in Section 5. Finally, conclusions of the research and some ideas about future directions are exposed in Section 6.

2. Background

In order to explain the foundation of our algorithm, we start by providing the background knowledge and notation used in the following sections. Next, the most relevant related works are presented and we give an overview of some of them.

2.1. Basic concepts

In this work we used simple undirected labeled graph as a basis for geometric subgraph mining. This kind of graph is defined as follow. Before describing their formal concept, we introduce the domain of labels.

Let L_V and L_E be label sets, where L_V is a set of vertex labels and L_E is a set of edge labels, the domain of all possible labels is denoted by $L = L_V \cup L_E$.

A *labeled graph* in L is a 4-tuple, $G = (V, E, I, J)$, where V is a set whose elements are called *vertices*, $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ is a set whose elements are called *edges* (the edge $\{u, v\}$ connecting the vertex u with the vertex v), $I : V \to L_V$ is a *labeling function* for assigning labels to vertices and $J : E \to L_E$ is a *labeling function* for assigning labels to edges.

Let $G_1 = (V_1, E_1, I_1, J_1)$ and $G_2 = (V_2, E_2, I_2, J_2)$ be two graphs, we say that G_1 is a *subgraph* of G_2 if $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, $\forall u \in V_1, I_1(u) = I_2(u)$, and $\forall e \in E_1, J_1(e) = J_2(e)$. In this case, we use the notation $G_1 \subseteq G_2$ and we say that G_2 is a *supergraph* of G_1.

We say that f is an *isomorphism* between G_1 and G_2 if $f : V_1 \to V_2$ is a bijective function where:

- $\forall u \in V_1, I_1(u) = I_2(f(u))$, and
- $\forall \{u, v\} \in E_1, \{f(u), f(v)\} \in E_2 \wedge J_1(\{u, v\}) = J_2(\{f(u), f(v)\})$.

When there is an isomorphism between G_1 and G_2, we say that G_1 and G_2 are *isomorphic.*

Using the labeled graph concept we can define geometric graphs. This kind of graph plays an important role in the classification scheme presented in this paper.

A *geometric graph* in L is a 5-tuple, $G' = (V, E, I, J, K)$, where $G = (V, E, I, J)$ is a labeled graph, $K : V \to \mathbb{R}^2$ is a function for assigning coordinates to vertices, $\mathbb{R}$ is the set of real numbers, and $K(u) \neq K(v)$ for each $u \neq v$. In this case, we say that G is the *associated* labeled graph to G'.

In this paper, we use $\mathbb{R}^2$ as coordinate space for simplifying the explanation of our proposal. However, our results could be extended to a three-dimensional space $\mathbb{R}^3$.

In the geometric context, two graphs may be geometrically similar and they can still have very different vertex coordinates. An example of this fact is shown in Figure 1. This situation can occur, because one of these graphs could be scaled, rotated, or translated with respect to the other one. Therefore, for matching two geometric graphs we need to consider the best geometric transformation between these graphs. It is important to note that in some contexts, the rotation of two graphs must be the same in order to say that they are similar. This can occur when the degree of rotation have semantic implications. For instance in the representation of images of numbers or characters (9 is not the same as 6).

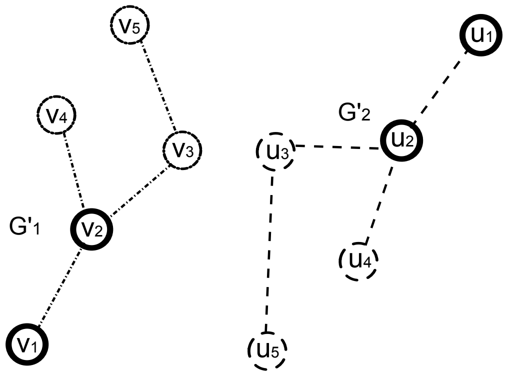


Figure 1: Two similar geometric graphs.

Let $G'_1 = (V_1, E_1, I_1, J_1, K_1)$ and $G'_2 = (V_2, E_2, I_2, J_2, K_2)$ be two geometric graphs, such that their associated labeled graphs G_1 and G_2, respectively, are isomorphic. Let f be an isomorphism between G_1 and G_2. A geometric transformation T in $\mathbb{R}^2$ can be defined as a function $T : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} \cos(\omega) & -\sin(\omega) \\ \sin(\omega) & \cos(\omega) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}.$$

where λ is the scaling factor, ω is the rotation angle, t_x and t_y are the translations in the X- and Y- axes respectively.

Let $e_1 = \{v_i, v_j\} \in E_1$ and $e_2 = \{u_i, u_j\} \in E_2$ such that $f(v_i) = u_i$ and $f(v_j) = u_j$, the *geometric transformation* T between e_1 and e_2 is calculated by choosing $\lambda = \|\eta_1\| / \|\eta_2\|$, ω the angle between the vectors η_1 and η_2, where $\eta_1 = K_1(v_i) - K_1(v_j)$, $\eta_2 = K_2(u_i) - K_2(u_j)$ and $\| \cdot \|$ is the Euclidean norm in $\mathbb{R}^2$, and the translations t_x and t_y are calculated by

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = K_1(v_i) - \lambda \begin{pmatrix} \cos(\omega) & -\sin(\omega) \\ \sin(\omega) & \cos(\omega) \end{pmatrix} K_2(u_i).$$

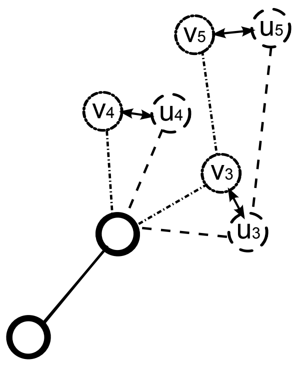


Figure 2: Transformed geometric graphs with a common reference frame. The distances between corresponding vertices are indicated with the arrows.

In our experimentations we tested the two following expressions to compute the *error* of a geometric transformation T for mapping G'_2 in G'_1

$$\epsilon_1(T) = \max_{v \in V_1} \|K_1(v) - T(K_2(f(v)))\|. \tag{1}$$

$$\epsilon_2(T) = \frac{\sum_{v \in V_1} \|K_1(v) - T(K_2(f(v)))\|}{\|V_1\|}. \tag{2}$$

For example in Figure 2, we can see the transformation applied to G'_2 using the geometric transformation for $i = 1$ and $j = 2$. In this Figure, the error of this transformation is the highest distance between vertices linked by arrows when $\epsilon_1(T)$ is used, and the average of these distances when we consider $\epsilon_2(T)$.

In this paper, we use this error concept as basis for a heuristic to calculate pseudo-best geometric transformation between two geometric graphs.

2.2. A brief of traditional graph mining

Let $D = \{G_1, G_2, \ldots, G_{|D|}\}$ be a collection of labeled graphs and let δ be a predefined threshold of frequency. The *support* of a graph G in D is defined as the set of graphs $G_i \in D$ such that there is a subgraph isomorphism from G to G_i. The notations $\Delta(G, D)$ and $\sigma(G, D) = |\Delta(G, D)|$ can also be used for referring support and frequency of G in D, respectively. A graph G frequently occurs in the collection D if $\sigma(G, D) \geq \delta$. Traditional graph mining is the process of finding connected subgraphs that frequently occur in a collection of labeled graphs. In Figure 3 we can see an example of a graph G that is subgraph of three graphs of a collection D. The occurrences of G are marked in bold. In this case, the support value is 3.

The first traditional graph miner was AGM [17], which allows us finding all frequent (connected or unconnected) subgraphs in a collection of labeled graphs. This algorithm was followed by AcGM [16] and FSG [22] for mining all frequent connected subgraphs. Both algorithms are based on the original Apriori algorithm [2] for mining frequent itemsets.

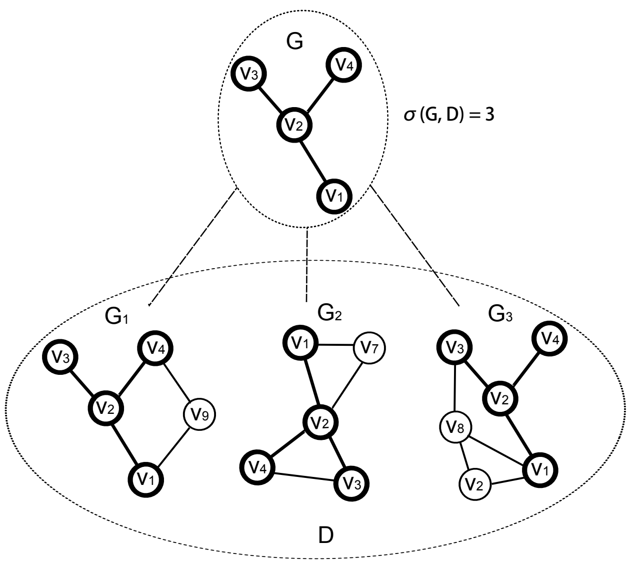


Figure 3: Support of graph in a collection D.

Later, pattern growth based algorithms such as Moss-MoFa [4], MoFa [4], gdFil [9], gRed [10], FFSM [15], Gaston [26] and gSpan [35], were developed. Previous comparative studies have shown that pattern growth based algorithms have better performance than Apriori based ones [27]. Therefore, in this paper we use pattern growth based scheme as basis for geometric graph mining. In our case, we use gdFil [9].

2.3. A brief of geometric graph mining

Geometric graph mining is an interesting problem in graph mining, where every vertex of each graph have a coordinate in a two- or three-dimensional coordinate space. This problem is more complex than the traditional graph mining [1, 9] because a non-geometric pattern can be placed in an infinite number of geometric positions.

In the last ten years, some geometric graph miners have been published. For example, the algorithm gFSG [23, 24] finds frequent geometric subgraphs in collections of geometric graphs, considering rotation, scaling and translation invariants. This method also uses an heuristic in order to extend geometric patterns and is presented as the geometric extension of a traditional graph miner called FSG [22].

Moreover, there are another two algorithms called MaxGeo [3] and FreqGeo [28], which were conceived considering some restrictions in the mining setup. MaxGeo only enumerates the maximal frequent subgraphs with vertices of two dimension coordinates. This method assumed zero tolerance, so patterns with small variations in their coordinates are considered different. Also, the authors of this work do not present any kind of experimental results in order to validate the efficiency or accuracy of its algorithm. On the other hand, in FreqGeo a retrieval approach was proposed that, given a reference graph G', it enumerate all frequent geometric graph that are subgraphs of G' [28]. This proposal is based on reverse search and has a polynomial delay. As we can see, this method only finds a subset of all the frequent geometric graphs that can be obtained in a specific collection.

Finally, there is another approach [14] that do not perform mining on geometric graphs, but uses the vertex coordinates to construct them.

2.4. A brief of classification using frequent patterns

As mentioned, frequent subgraph patterns have been successfully used for classification tasks in different domains of science, for instance, in image processing [1, 8, 29, 30], chemical compound studies [6, 18] and text mining [21]. However, only one of these solutions uses frequent geometric subgraphs in classification tasks [6]. This method perform feature (subgraph) selection using a heuristic to obtain discriminative features, purging the already mined set of frequent subgraphs (geometric and non-geometric). From the features identified using the mentioned selection strategy, the feature vectors which are used in classification are built. However, such feature vectors reflect only the occurrence or not of each subgraph in the graphs, leaving aside the degree of geometric distortion.

In our research, we propose a novel framework for geometric graph classification, considering only frequent geometric subgraphs as features for representing objets. Moreover, we take into account the similarity values between geometric graphs for building feature vectors. This framework was evaluated on an image collection and a molecular dataset.

3. Geometric subgraph detection

In this section, a new algorithm for frequent geometric subgraph mining is presented. Before explaining the algorithmic details, we introduce the following definitions to facilitate the description of our geometric method.

3.1. Geometric graph matching

Let $G_1' = (V_1, E_1, I_1, J_1, K_1)$ and $G_2' = (V_2, E_2, I_2, J_2, K_2)$ be two geometric graphs with n vertices, such that their associated labeled graphs G_1 and G_2, respectively, are isomorphic. Let f be an isomorphism between G_1 and G_2. In our work, the pseudo-best geometric transformation between G_1' and G_2' is calculated by choosing the pair of edges $\{v_i, v_j\} \in E_1$ and $\{f(v_i), f(v_j)\} \in E_2$ such that the geometric transformation between them achieves the minimum error.

Let T be the pseudo-best geometric transformation between G_1' and G_2'. We say that G_1' and G_2' are isomorphic with tolerance τ if $\epsilon(T) < \tau$. In this case, we define the similarity function between G_1' and G_2' by mean of

$$\phi(G_1', G_2') = \frac{1}{n\tau} \sum_{v \in V_1} \|K_1(v) - T(K_2(f(v)))\|. \tag{3}$$

The previously defined similarity function is normalized between 0 and 1. In Figure 2 we can see an example of two transformed graphs. The distances between their corresponding vertices are indicated with arrows. Two vertices where chosen as reference frame since the graphs are represented in $\mathbb{R}^2$. If one of these distances is greater than the tolerance threshold τ, these graph are not isomorphic with tolerance τ.

3.2. Geometric graph clustering

Let G be a labeled graph and let $H' = \{G'_1, G'_2, \ldots, G'_{|H'|}\}$ be a set of geometric graphs, such that G is the associated labeled graph of G'_i for each $1 \leq i \leq |H'|$. In this section, we describe a clustering algorithm for grouping the elements in H' according to the similarity function (3).

Clustering is the process of grouping a set of data objects into a set of subclasses, called clusters; these clusters could be disjoint or not. A cluster is a collection of objects that have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. In our work, objects refer to geometric graphs, so clusters could not be disjoint. Moreover, it is not possible to know apriori the number of clusters that will be generated. The unique two assumptions for preparing our clustering scheme is the error function (2) and the tolerance threshold τ. For these reasons, we need to use an overlapped clustering algorithm based on similarity graphs [36]. In this way, we can use any algorithm for clustering that satisfy the mentioned aspects. In our case, we use ACONS [11].

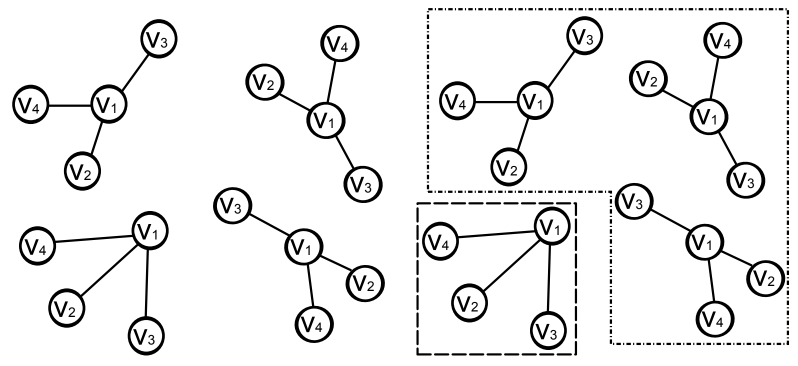
ACONS algorithm, finds condensed clusters in which the detection of centers is very easy. The first step of ACONS consists on building the similarity graph of the set of objects. In this paper, the Algorithm 1 is used for building the similarity graph of H' in the geometric graph context. The vertices of the similarity graph $\mathcal{G}$ are the geometric graphs of H' (see line 1). The edges in $\mathcal{G}$ are added between geometric graphs which are mutually isomorphic with tolerance τ (see line 5).

Algorithm 1: GetSimGraph(H', τ)

```
Input: $H' = \{G'_1, G'_2, \ldots, G'_{|H'|}\}$ - set of geometric graphs, $\tau$ - tolerance threshold
Output: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - similarity graph of $H$
1 $\mathcal{V} \leftarrow H'$ ;                    // The vertices of $\mathcal{G}$ are the elements in $H'$
2 $\mathcal{E} \leftarrow \emptyset$ ;           // The edges of $\mathcal{G}$ are added in the following lines
3 for $i \leftarrow 1$ to $|H'| - 1$ do
4     for $j \leftarrow i + 1$ to $|H'|$ do
5         if $G'_i$ and $G'_j$ are isomorphic with tolerance $\tau$ then
6             $\mathcal{E} \leftarrow \mathcal{E} \cup \{G'_i, G'_j\}$;
7 return $(\mathcal{V}, \mathcal{E})$;
```

In the next steps, ACONS creates a set of clusters for structuring H' using $\mathcal{G}$ as start point. In this way, ACONS marks every vertex of $\mathcal{G}$ as center or satellite using a graph theoretic heuristic, and guaranteeing that each object of the collection belongs at least to one group. Each one of these groups fulfil the following properties:

1. Let C_j be a group, then there is at least a element $G^* \in C_i$ such that every $G' \in C_i$ is connected by an edge with G^* in the similarity graph $\mathcal{G}$. In this case, G^* is the center of C_j.



(a) Occurrences of a frequent connected subgraph.
(b) Occurrences grouped using ACONS.

Figure 4: Clustering of occurrences from a frequent subgraph.

2. Let C_j and C_k be two distinct groups, then their corresponding centers must also be distinct.

The centers of such clusters are used for detecting frequent geometric subgraphs in section 3.3.

In Figure 4(a) we can see four occurrences of a subgraph obtained with an algorithm of mining frequent connected subgraphs. These occurrences are isomorphic to each other. Also, in Figure 4(b) we can see the occurrences clustered using ACONS. Each cluster contains the occurrences that are geometrically similar using the similarity function previously defined.

3.3. Geometric graph mining

Let $D' = \{G'_1, G'_2, \ldots, G'_{|D'|}\}$ be a collection of geometric graphs, let τ be a previously known tolerance threshold, and let δ be an also predefined threshold of frequency. The *support* of a geometric graph G' in D' is defined as the set of graphs $G'_i \in D'$ such that there is a subgraph of G'_i which is isomorphic to G' with tolerance τ. The notation $\Delta(G', D')$ is used for referring the above mentioned support and $\sigma(G', D') = |\Delta(G', D')|$ is used for indicating the frequency of G' in D'. We say that G' frequently occurs in the collection D' if $\sigma(G', D') \geq \delta$. In this way, *frequent geometric subgraph mining* is the process of finding the geometric graphs that frequently occur in a collection of geometric graphs.

The geometric graph miner proposed by us can be seen as an extension of any traditional graph miner. Traditional graph miners work with labeled graph collection without considering geometric information. Thus, we can also use these methods for processing geometric graph collections, but ignoring vertex coordinates.

Theorem 1. *Let G be a frequent connected graph in D', with $\sigma(G, D') \geq \delta$. If G has only one vertex or it has only one edge, then any geometric graph having G as associated graph, is frequent in D'.*

Proof. According to the hypotheses, G must be geometrically represented as a single point or a segment. Therefore, any two geometric graphs, having G as associated graph, are mutually isomorphic with tolerance r, since there is always a geometric transformation between two points or two segments. Thus, the frequency of these geometric graphs will be the same one of G. $\square$

Theorem 1 states that there are infinite frequent geometric subgraphs associated with a frequent labeled graph with only one vertex or only one edge. Therefore, during geometric subgraph mining we only need to report a geometric graph for each frequent connected graph G with such conditions.

Theorem 2. *Let G be a frequent connected graph in D', with $\sigma(G, D') \geq \delta$, let H' be the set of geometric graphs G' associated with G whose frequencies are at least 1, that is $\sigma(G', D') \geq 1$, and let $\mathcal{G}$ be the similarity graph of H' using τ as tolerance threshold. Suppose that we apply ACONS in order to find clusters in $\mathcal{G}$. Let C_j be a cluster outputted by ACONS, whose center is G^*; then, the support of G^* is $\Delta(G^*, D') = \{G'_i \in D' | \exists G' \in C_j, G' \subseteq G'_i\}$.*

Proof. According to the criteria for building $\mathcal{G}$, the G^* is isomorphic with tolerance τ to any geometric graph in C_j. Therefore, if $G' \in C_i$ and $G' \subseteq G'_i$ for some $G'_i \in D'$; then, there is a subgraph G'_i which is isomorphic to G^* with tolerance τ, and $G'_i \in \Delta(G^*, D')$. Thus, $\{G'_i \in D' | \exists G' \in C_j, G' \subseteq G'_i\} \subseteq \Delta(G^*, D')$.

On the other hand, if $G_i \in \Delta(G^*, D')$ then there is a G' such that $G' \subseteq G_i$, G' is isomorphic with tolerance τ to G^*. Therefore, $G' \in C_j$, since $\sigma(G', D') \geq 1$. Thus, $\Delta(G^*, D') \subseteq \{G'_i \in D' | \exists G' \in C_j, G' \subseteq G'_i\}$. Finally, $\Delta(G^*, D') = \{G'_i \in D' | \exists G' \in C_j, G' \subseteq G'_i\}$. $\square$

Algorithm 2: gsmGeomSearching (D', G, τ, δ)

```
Input: $D'$ - geometric graph collection, $G$ - frequent labeled graph with at least two
       edges, $\tau$ - tolerance threshold, $\delta$ - frequency threshold
Output: $R$ - set of frequent geometric graphs
1  $H' \leftarrow \emptyset$;
2  forall $G'_i \in \Delta(G, D')$ do
3      forall geometric graph $G' \subseteq G'_i$ whose associated graph is isomorphic to $G$ do
4          $H' \leftarrow H' \cup \{G'\}$;
5  $\mathcal{G} \leftarrow$ GetSimGraph ($H'$, $\tau$);
6  $C \leftarrow$ ACONS ($\mathcal{G}$, $\tau$);
7  $R \leftarrow \emptyset$;
8  forall cluster $C_j \in C$ do
9      $G^* \leftarrow$ the center of $C_j$;
10     if $\sigma(G^*, D') \geq \delta$ then
11         $R \leftarrow R \cup \{G^*\}$;
12 return $R$;
```

Theorem 2 states that the clusters outputted by ACONS can be used for detecting frequent geometric subgraph associated with a labeled graph G with more than one edge, see Algorithm 2.

Lines 1-4 of Algorithm 2 are dedicated for gathering the elements of H', which is the set of geometric graphs associated with G whose frequencies is at least 1. Line 5 is an invocation to the function `GetSimGraph`, see Algorithm 1, to calculates the similarity graph $\mathcal{G}$ of H' regarding the threshold τ. The graph $\mathcal{G}$ is used as input for the clustering algorithm, ACONS, in line 6. Thus, the last lines of Algorithm 2 are focused on detecting frequent geometric subgraphs, according to the Theorem 2. The frequency of each center G^*, in line 9, is efficiently calculated by storing the supergraphs $G_i \in D'$ of each geometric graph in the cluster C. Moreover, the support of the labeled graph G, and geometric subgraph of each G_i, can also be calculated using the embedding structure introduced by the traditional graph miner gdFil [9].

Algorithm 3: gsmFil(D', τ, δ, S)

```
Input: $D'$ - geometric graph collection, $\tau$ - tolerance threshold, $\delta$ - frequency
       threshold
Output: $S$ - mining results
1 Remove infrequent vertices and edges from $D'$;
2 $S \leftarrow$ all frequent vertices including the coordinates $(0, 0)$ to each vertex;
3 $S^1 \leftarrow$ all frequent 1-edge labeled graphs;
4 forall code $G \in S^1$ do
5     gsmFilMining($D'$,$G$,$\delta$,$S$);
6     $D' \leftarrow D' \setminus G$;
7     if $|D'| < \delta$ then break;
```

Algorithms 3 and 4 shown the whole pseudo-code of our gsmFil algorithm. The main method is `gsmFil` (see Algorithm 3) which starts by removing infrequent vertices and edges from D'. Afterwards, frequent geometric graphs with only one vertex are calculated. According to Theorem 1 these graphs are also frequent in the geometric context. The coordinates of the first vertex of each frequent graph found will be $(0, 0)$ in order to use the same coordinate system for all of them. Lines 4-8 work identically to the ancestor gdFil [9], traversing the set of labeled graphs with only one edge. For each labeled graph with only one edge the `gsmFilMining` algorithm is invoked. At the end of each iteration, the used edge is dropped from the collection. This means that it will not be used any more as a possible extension in the next iterations.

The Algorithm 4 recursively generates all labeled graphs that hold G, while the generated DFS codes are frequent. Lines 1-3 are used for detecting frequent geometric graphs with only one edge. The coordinates of the second vertex of each frequent geometric graph found will be $(0, 1)$. Thus, the coordinates of the next vertices found will be transformed according to the first two vertices of the same graph. Line 4-8 use the function `gsmGeomSearching` (see Algorithm 2) to detect frequent geometric graphs with more than one edge. Next, lines 9--13 work identically to its ancestor gdFil, traversing frequent geometric labeled subgraphs, removing duplicates candidates, calculating embedding structures, and so on [9].

Algorithm 4: gsmFilMining (D', G, τ, δ, S)

```
Input: $D'$ - geometric graph collection, $G$ - frequent labeled graph, $\tau$ - tolerance
       threshold, $\delta$ - frequency threshold
Output: $S$ - mining results
1  if $G$ has only one edge then
2      $G' \leftarrow$ a geometric graph whose associated graph is $G$ and the coordinates of the
       first and second vertices are $(0,0)$ and $(0,1)$, respectively;
3      $S \leftarrow S \cup \{G'\}$;
4  else
5      $R \leftarrow$ gsmGeomSearching ($G$, $\tau$, $\delta$, $S$);
6      if $R = \emptyset$ then return;
7      else $S \leftarrow S \cup R$;
8  $ME \leftarrow$ the set of non-duplicates extensions of $G$ in $D'$, using the procedures designed
   for gdFil;
9  Remove from $ME$ non-frequent extensions, according to $\delta$;
10 forall extension $e \in ME$ do
11     gsmFilMining($D$, $G \diamond e$, $\tau$, $\delta$, $S$);
```

4. Classification scheme

In order to evaluate the quality of the patterns identified, we use the frequent geometric subgraphs detected on classification tests. The package libSVM[1] is used for graph classification through Support Vector Machine (SVM) classifier.

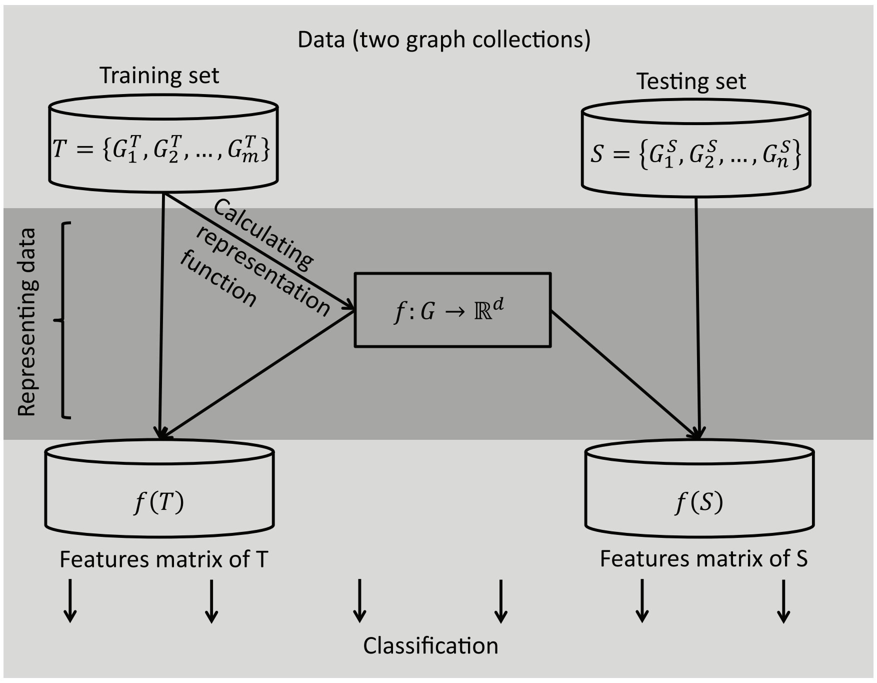
Given a training set of geometric graph $T = \{G_1^T, G_2^T, \ldots, G_m^T\}$, we obtain a representation function f that will be used to build the features matrix of T. We subsequently employ a classifier generator that uses this matrix as data to produce a classifier. Before the classification phase of a set of geometric graph $S = \{G_1^S, G_2^S, \ldots, G_n^S\}$, we applied the function f to S in order to obtain the features matrix that describe S, based on the data collected from T. With these two matrices we proceed to classify S. We can see the described method represented in Figure 5.

Our main contributions are focused on the process of construction of the feature matrices using the frequent geometric subgraphs extracted from T (see Section 3.3). The construction of the features matrices is presented in detail in section Section 4.1.

4.1. Classification task

Given the set $F = \{S_1, S_2, \ldots, S_d\}$ of frequent geometric subgraphs of T, and a set of geometric graphs $R = \{G_1^R, G_2^R, \ldots, G_m^R\}$, the function f returns a feature vector $f(G_i^R) = \{c_1, c_2, \ldots, c_d\}$ where d is the total number of geometric subgraphs identified. Thus, we build a matrix where the row number $(1 \leq i \leq m)$ corresponds to the number of graph in the

[1]http://www.csie.ntu.edu.tw/∼cjlin/libsvm



Data (two graph collections)
Training set
Testing set
$T = \{G_1^T, G_2^T, \ldots, G_m^T\}$
$S = \{G_1^S, G_2^S, \ldots, G_n^S\}$
Representing data
Calculating representation function
$f: G \to \mathbb{R}^d$
$f(T)$
$f(S)$
Features matrix of T
Features matrix of S
Classification

Figure 5: Classification scheme.

collection R, and the number of columns ($1 \leq j \leq d$) corresponds to the number of frequent geometric subgraphs in F. Each feature value can be assigned using the follow settings:

- *Binary setting*: The value of an entry $v_{i,j}$ of the matrix is $v_{i,j} = 1$ if S_j occurs in G_i^R, $v_{i,j} = 0$ otherwise.
- *Similarity setting*: The value of an entry $v_{i,j}$ of the matrix is the highest similarity value of the occurrences of S_j in G_i^R, and $v_{i,j} = 0$ in cases where S_j does not occur in graph G_i^R. The similarity value of each feature is obtained with the following expression:

$$sim(G_i^R, S_j) = \max_{G' \subseteq G_i^R} \{\phi(S_j, G')\}, \tag{4}$$

  where ϕ is the formula (3). This value can by efficiently pre-calculated during the mining and included in the output of gsmFil.

In Figure 6 we can see the steps followed in order to obtain the representation function.

5. Experimental results

In this section, we show experimental results that validate the efficacy of the patterns identified, using frequent geometric subgraph mining for classification tasks. The results of classification using our proposal and gdFil (proposed by Gago-Alonso *et al.* [9]) algorithm are compared. The last one is chosen as the representation of the algorithms for exact

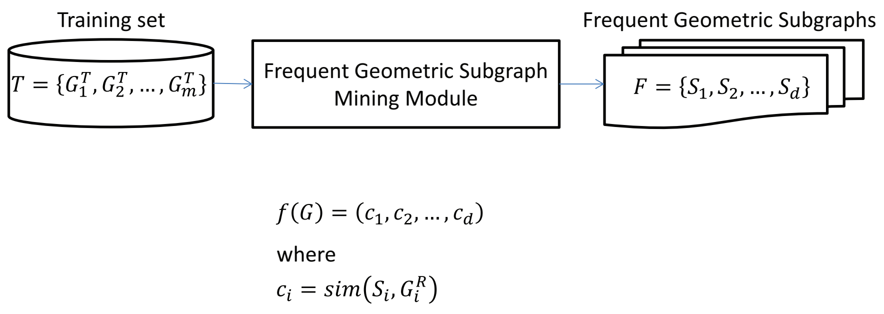

Training set
$T = \{G_1^T, G_2^T, \ldots, G_m^T\}$
Frequent Geometric Subgraph Mining Module
Frequent Geometric Subgraphs
$F = \{S_1, S_2, \ldots, S_d\}$
$f(G) = (c_1, c_2, \ldots, c_d)$
where
$c_i = sim(S_i, G_i^R)$

Figure 6: Representation function.

frequent subgraph mining; these algorithms do not consider the geometric information, only the labels of vertices. Notice that VEAM algorithm [1] is not used in our experiments because we do not have any substitution matrix for the collection employed. For the experiments we used an available geometric labeled graph collections proposed by Riesen & Bunke [32]. This collections have for each vertex a two-dimensional attribute giving its position.

All our experiments were carried out using a personal computer (64 bits) Intel (R) Core (TM) 2 Quad CPU $Q9450$ @ 2.66 GHz with 4 Gb main memory. The algorithm for frequent geometric subgraph mining was implemented in ANSI C language and compiled using gcc compiler of GNU/Linux with -O0 optimization.

5.1. AIDS graph collection

The AIDS database [32] is divided into three disjoint collections: "training", "valid" and "test". In our experiments we used the "train" collection, composed by 250 graphs, in order to train the classifier. We also used the "test" collection, formed by 1500 graphs, to test the classifier previously trained. The graphs contained in this database are divided into two classes (*active* and *inactive*). These classes represent molecules with activity against HIV or not. Each graph is a molecular compound where each node represents an atom and edges are the covalent bonds.

The average size of these graphs is 16.24 (in terms of the number of edges) with 38 vertex labels and 3 edge labels. Each vertex represents an atom and is labeled with the number of the corresponding chemical symbol. Each edge represents a covalent bond and is labeled with the valence of the linkage. In Figure 7 (provided with the collection as a sample) molecular compounds of both classes are illustrated.

5.2. Classification results

As we mentioned in Section 4.1, the SVM classifier is used in this paper. On this classifier two kernel functions are employed : *Sigmoid* and *Radial Basis Function* (RBF). The classification is performed using both the binary setting and similarity settings on feature vectors. The binary setting is used for gdFil results, while the similarity setting is used for gsmFil results. The parameters of the kernels are estimated performing a cross-validation. The classification accuracy is tested on "test" collection.

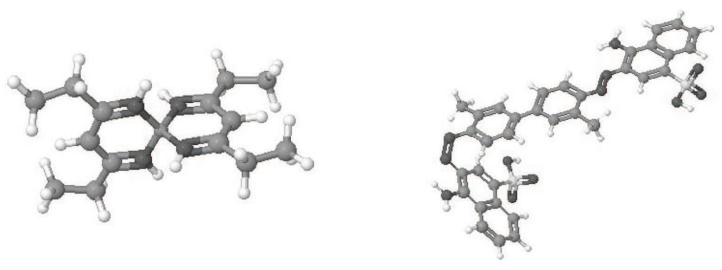


Figure 7: A molecular compound of both classes (inactive and active in the same order).

The classification results where performed with with $\tau = 4$. These values are shown in Figure 8 and Figure 9 using e_1 and in Figure 10 and Figure 11 with e_2. The value of τ was selected according to the proximity As we can see, our proposal outperform gdFil for every support value. The results obtained by our proposal, in some cases are greater than 98% of accuracy using e_1 and a sigmoid kernel. These values are grater than the reported in AIDS database [32]. The results obtained with a RBF kernel or e_2 are slightly less accurate.

Notice that one of the reasons of the use of geometric methods is that exact ones cannot identify enough features for a good classification. In Table 1 we can see that the amount of patterns identified by geometric methods is greater that non-geometric algorithm on this database. Also, the geometric information provides important information that is more representative of the collection and is wasted by exact methods like gdFil.

Table 1: Number of patterns identified by gdFil and our geometric algorithm ($\tau = 1$) on AIDS Database with several support values.

Support (δ)	**exact algorithm**	**our geometric algorithm**
10%	200	996
15%	99	520
20%	54	309
25%	33	219
30%	28	142
35%	18	77
40%	15	66

In addition, in Table 2 we show the results of the classification using several values for τ on AIDS database, with 10% as support and a sigmoid kernel. The threshold τ have a great impact on the accuracy, as evidenced in the results shown. This occurs because τ plays an important role in the process of finding geometric isomorphism between graphs. The value of τ is directly proportional to the number of found isomorphisms between the geometric graphs. As we can see in the referred figure, the optimal value is reached with $\tau = 4$.

Also, in Table 3 the execution times of the whole process of parameters adjustment, training and classification are shown with different values of τ. As we can see, τ has impact in both: accuracy and execution time, so the correct selection of this value is a very critical step in our proposal.

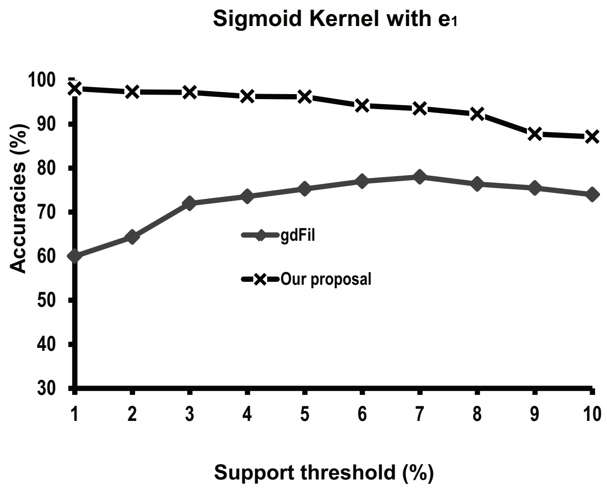


Figure 8: Accuracies achieved using SVM with a sigmoid kernel and e_1.

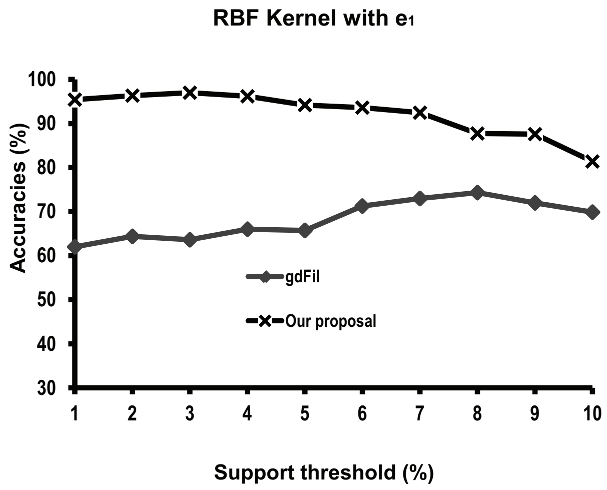


Figure 9: Accuracies achieved using SVM with a RBF kernel and e_1.

Table 2: Accuracies achieved using SVM with a Sigmoid kernel and $\delta = 10\%$ in AIDS collection, varying tolerance threshold.

Tolerance (τ)	**exact algorithm**	**our geometric algorithm**
3.5		87.33%
4.0		98.06%
4.5		97.60%
5.0	60.04%	96.80%
5.5		97.60%
6.0		97.73%

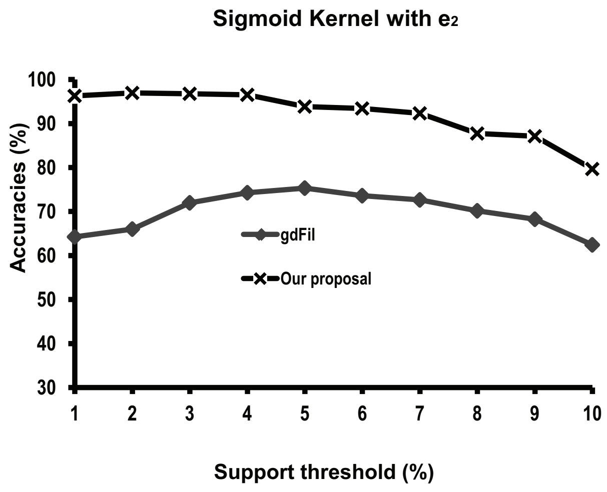


Figure 10: Accuracies achieved using SVM with a sigmoid kernel and e_2.

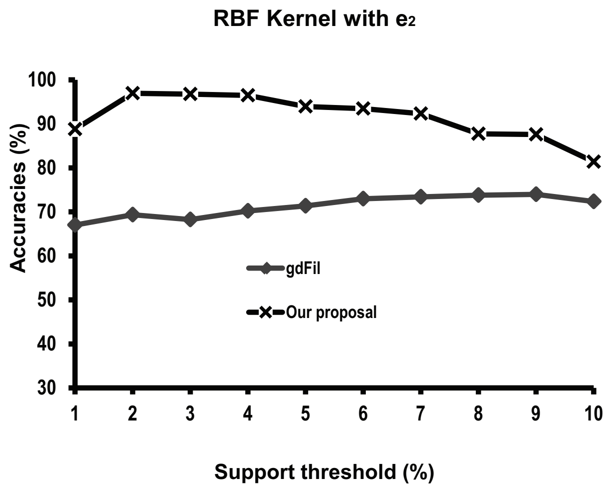


Figure 11: Accuracies achieved using SVM with a RBF kernel and e_2.

Table 3: Executing times using SVM with a Sigmoid kernel and $\delta = 10\%$ in AIDS collection, varying tolerance threshold.

Tolerance (τ)	Execution time (in seconds)
3.5	217
4.0	358
4.5	401
5.0	478
5.5	513
6.0	587

We also conducted experiments that validate the impact of the use of a similarity function in our approach. We performed two experiments in AIDS collection: the first one using a binary setting for the feature values and the second one using the similarity setting previously defined. In Figure 12 we can see the results: the use of similarity settings have a positive impact on the accuracy of the classification process. As in previously made experiments, the values of τ was 4.

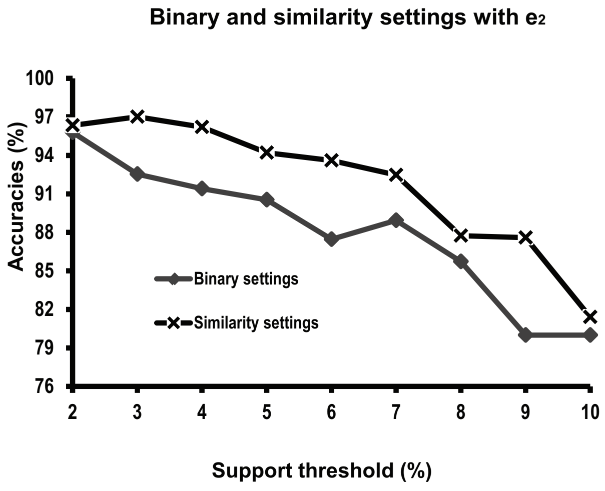


Figure 12: Accuracies achieved in AIDS collection using our proposal with binary and similarity settings.

In summary, the results shown that using geometric algorithms we can achieve better accuracy in most of the cases than with exact ones, for classification task. These results illustrate the usefulness of the patterns identified by frequent geometric subgraph mining process. These patterns provide more representative information because our algorithm takes into account the geometric vertex positions of the graphs. Moreover, a spatial tolerance in the vertices is allowed. We can see that the better classification is achieved using Sigmoid kernel function for the data types used in the experiments. Finally we have observed that for very small values of δ, the accuracy is decreased due to the poor representativeness of the encountered patterns. On the other hand, if the value of δ is to high, a very small number of patterns will be found.

6. Conclusions

In this paper, we proposed a new algorithm called gsmFil for frequent geometric subgraph mining on geometric graph collections where graphs are labeled and undirected. Using gsmFil frequent patterns are identified in graph collections allowing slight geometric transformation differences between graphs.

The experimental results shown the efficacy of our geometric algorithm versus traditional graph miners on graph-based classification. In fact, the patterns detected by gsmFil consider spatial variations which are useful for classifying and are not exploited on traditional approaches. These spatial variations can be used for modelling many real datasets. As an example of the application of our approach, we proposed in this paper a solution for real graph-based collections. In the experiments we can see that the accuracy results of classification using geometric patterns are better than the obtained with an exact mining approach. In fact, the classification result achieved in AIDS using our proposal, are better than those reported in literature, in most cases. Thus, we can conclude that the patterns identified by our proposal have a positive impact in some graph-based classification tasks.

As future work, we are going to develop new ways for taking advantage of feature selection strategies for improving geometric graph classification. These strategies in combination with our proposal could be useful for reducing dimensionality and improving the efficiency of geometric graph classifiers. Also, the patterns found may have a great impact in the construction of indices for indexing tasks.

References

[1] N. Acosta-Mendoza, A. Gago-Alonso, J.E. Medina-Pagola, Frequent Approximate Subgraphs as Features for Graph-Based Image Classification, Knowledge-Based Systems 27 (2012) 381–392.

[2] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules, in: Proceedings of the 1994 International Conference on Very Large Data Bases, Santiago, Chile, 1994, pp. 487–499.

[3] H. Arimura, T. Uno, S. Shimozono, Time and Space Efficient Discovery of Maximal Geometric Graphs, in: Proceedings of the 10th International Conference on Discovery Science, Sendai, Japan, 2007, pp. 42–55.

[4] C. Borgelt, M.R. Berthold, Mining Molecular Fragments: Finding Relevant Substructures of Molecules, in: Proceedings of the 2002 International Conference on Data Mining, Maebashi, Japan, 2002, pp. 211–218.

[5] Bunke, H., Riesen, K.: Towards the unification of structural and statistical pattern recognition. Pattern Recognition Letters, (33) 208–297, 2012.

[6] M. Deshpande, M. Kuramochi, N. Wale, G. Karypis, Frequent Substructure-Based Approaches for Classifying Chemical Compounds, IEEE Transactions on Knowledge Data Engineering 17 (2005) 1036–1050.

[7] Duin, R., Pekalska, E.: The Dissimilarity Representations for Pattern Recognition: Foundations and Applications. World Scientific, Singapore, 2005.

[8] A. Elsayed, F. Coenen, C. Jiang, M. García-Fiñana, V. Sluming, Corpus Callosum MR Image Classification, Knowledge-Based Systems 23 (2010) 330–336.

[9] A. Gago-Alonso, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, J.F. Martínez-Trinidad, Full Duplicate Candidate Pruning for Frequent Connected Subgraph Mining, Integrated Computer-Aided Engineering 17 (2010) 211–225.

[10] A. Gago-Alonso, J.E. Medina-Pagola, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad, Mining Frequent Connected Subgrahps Reducing the Number of Candidates, in: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Antwerp, Belgium, 2008, pp. 365–376.

[11] A. Gago-Alonso, A. Pérez-Suárez, J.E. Medina-Pagola, ACONS: A New Algorithm for Clustering Documents, in: Proceedings of 12th Iberoamerican Congress on Pattern Recognition, Valparaíso, Chile, 2007, pp. 664–673.

[12] R. Hernández-León, J.A. Carrasco-Ochoa, J.Fco. Martínez-Trinidad, J. Hernández-Palancar, CAR-NF: A Classifier based on Specific Rules with High Netconf, Intelligent Data Analysis 16 (2012) 49–68.

[13] L. B. Holder, D. J. Cook, H. Bunke, Fuzzy Substructure Discovery, in: Proceedings of the 9th International Workshop on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992, pp. 218–223.

[14] J. Huan, D. Bandyopadhyay, W. Wang, J. Snoeyink, J. Prins, A. Tropsha, Comparing Graph Representations of Protein Structure for Mining Family-Specific Residue-Based Packing Motifs, Journal of Computational Biology 12 (2005) 657–671.

[15] J. Huan, W. Wang, J. Prins, Efficient Mining of Frequent Subgraph in the Presence of Isomorphism, in: Proceedings of the 2003 International Conference on Data Mining, Melbourne, FL, USA, 2003, pp. 549–552.

[16] A. Inokuchi, T. Washio, K. Nishimura, H. Motoda, A Fast Algorithm for Mining Frequent Connected Subgraphs, Technical Report RT0448, IBM Research, Tokyo Research Laboratory, 2002.

[17] A. Inokuchi, T. Washio, H. Motoda, An Apriori based Algorithm for Mining Frequent Substructures from Graph Data, in: Proceedings of the 2000 European Symposium on the Principle of Data Mining and Knowledge Discovery, Lyon, France, 2000, pp. 13–23.

[18] Y. Jia, J. Zhang, J. Huan, An Efficient Graph-Mining Method for Complicated and Noisy Data with Real-World Applications, Knowledge Information Systems 28 (2011) 423–447.

[19] C. Jiang, F. Coenen, M. Zito, A Survey of Frequent Subgraph Mining Algorithm, To appear in: Knowledge Engineering Review 2012.

[20] C. Jiang, F. Coenen. Graph-based Image Classification by Weighting Scheme, in: Proceedings of the 28th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK, 2008, pp. 63–76.

[21] C. Jiang, F. Coenen, R. Sanderson, M. Zito, Text Classification using Graph Mining-based Feature Extraction, Knowledge-Based Systems 23 (2010) 302–308.

[22] M. Kuramochi, G. Karypis, Frequent Subgraph Discovery, in: Proceedings of the 2001 IEEE International Conference on Data Mining, California, USA, 2001, pp. 313–320.

[23] M. Kuramochi, G. Karypis, Discovering Frequent Geometric Subgraphs, in: Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi, Japan, 2002, pp. 258–265.

[24] M. Kuramochi, G. Karypis, Discovering Frequent Geometric Subgraphs, Information Systems 32 (2007) 1101–1120.

[25] A. Nanopoulos, Y. Manolopoulos, Mining Patterns from Graph Traversals, Data & Knowledge Engineering 37 (2001) 243–266.

[26] S. Nijssen, J. Kok, A Quickstart in Frequent Structure Mining can Make a Difference, in: Proceedings of the 2004 ACM SIGKDD International Conference on Kowledge Discovery in Databases, Seattle, WA, USA, 2004, pp. 647–352.

[27] S. Nijssen, J. Kok, Frequent Subgraph Miners: Runtimes Don't Say Everything, in: Proceedings of the Workshop Mining and Learning with Graphs, held with the 2006 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, 2006, pp. 173–180.

[28] S. Nowozin, K. Tsuda, Frequent Subgraph Retrieval in Geometric Graph Databases, in: Proceedings of the 2008 IEEE International Conference on Data Mining, Pisa, Italy, 2008, pp. 953–958.

[29] S. Nowozin, K. Tsuda, T. Uno, T. Kudo, G. BakIr, Weighted Substructure Mining for Image Analysis, in: Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 2007, pp. 1–8.

[30] B. Ozdemir, S. Aksoy, Image Classification Using Subgraph Histogram Representation, in: Proceedings of the 20th International Conference on Pattern Recognition, Istanbul, Turkey, 2010, pp. 1112–1115.

[31] Pekalska, E., Duin, R., Paclik, P.: Prototype selection for dissimilarity-based classifiers. Pattern Recognition 39(2), 189208, 2006.

[32] K. Riesen, H. Bunke, IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning, in: Proceedings of the 12th Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Orlando, FL, USA, 2008, pp. 287–297.

[33] Riesen, K., Neuhaus, M., Bunke, H.: Graph Embedding in Vector Spaces by Means of Prototype Selection. GbRPR, LNCS 4538, Springer-Verlag Berlin Heidelberg, 383–393, 2007.

[34] Spillmann, B., Neuhaus, M., Bunke, H., Pekalska, E., Duin, R.: Transforming Strings to Vector Spaces using Prototype Selection. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.)

Structural, Syntactic, and Statistical Pattern Recognition. LNCS, vol. 4109, pp. 287296. Springer, Heidelberg, 2006.

[35] X. Yan, J. Han, gSpan: Graph-Based Substructure Pattern Mining, in: Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi, Japan, 2002, pp. 721–724.

[36] L. Yen, F. Fouss, C. Decaestecker, P. Francq, M. Saerens, Graph Nodes Clustering with the Sigmoid Commute-time Kernel: A Comparative Study, Data & Knowledge Engineering 68 (2009) 338–361.

[37] U. Yun, K.H. Ryu, Approximate Weighted Frequent Pattern Mining with/without Noisy Environments, Knowledge-Based Systems 24 (2011) 73–82.