Frequent approximate subgraphs as features for graph-based image classification

Niusvel Acosta-Mendoza[a,*], Andrés Gago-Alonso[a], José E. Medina-Pagola[a]

[a] *Advanced Technologies Application Center, Havana, Cuba*

Abstract

The use of approximate graph matching for frequent subgraph mining has been identified in different applications as a need. To meet this need, several algorithms have been developed, but there are applications where it has not been used yet, for example image classification. In this paper, a new algorithm for mining frequent connected subgraphs over undirected and labeled graph collections VEAM (**V**ertex and **E**dge **A**pproximate graph **M**iner) is presented. Slight variations of the data, keeping the topology of the graphs, are allowed in this algorithm. Approximate matching in existing algorithm (APGM) is only performed on vertex label set. In VEAM, the approximate matching between edge label set in frequent subgraph mining is included in the mining process. Also, a framework for graph-based image classification is introduced. The approximate method of VEAM was tested on an artificial image collection using a graph-based image representation proposed in this paper. The experimentation on this collection shows that our proposal gets better results than graph-based image classification using some algorithms reported in related work.

Keywords: Approximate graph mining, approximate graph matching, image representation, image classification, feature selection.

1. Introduction

In recent years, the need to convert large volumes of data into useful information has increased. The objects in many of these datasets are or may be represented as graphs. In Fig. 1, an example of graphs for representing images are shown.

As a result of this need, several authors have developed techniques and methods to process these datasets [1, 22]. An example of such techniques is the frequent pattern discovery [13, 30, 34].

The discovery of frequent patterns, especially the detection of frequent subgraphs in graph collections is an important problem in graph mining tasks [11, 27, 33]. In frequent subgraph mining, there are two approaches for evaluating the similarity of graphs, known as graph matching: *exact matching* and *approximate matching*.

*Corresponding author

Email address: nacosta@cenatav.co.cu (Niusvel Acosta-Mendoza)

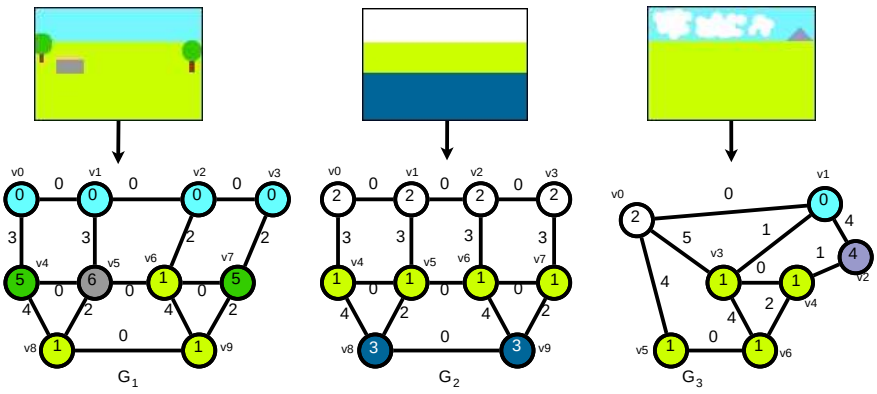

Figure 1: **Collection of graph** $D = \{G_1, G_2, G_3\}$**.**

The exact matching consists in determining whether the labels and the structure of two graphs are identical. This matching has been successfully used in many applications [7, 12, 16, 20, 23]; however, there are concrete problems where exact matching could not be applicable with positive outcome [14]. Sometimes, the interesting subgraphs show slight differences throughout the data. An example of these differences can be seen in protein analysis, where these may be due to accumulated mutations in the evolution of similar structures in several proteins. Other examples can be seen on image processing, where these differences may be due to noise and distortion, or may just illustrate slight spatial differences between instances of the same objects. This means that we should tolerate certain level of geometric distortion, slight semantic variations or vertices and/or edges mismatch in frequent subgraph pattern search.

For this reason, it has become necessary to evaluate the similarity between graphs allowing some structural differences, i.e. approximate matching techniques. The approximate matching consists in finding the match between vertices and/or edges of two graphs in order to determine their similarity, allowing structure and/or label differences. On this basis, the need to perform frequent subgraph mining using approximate graph matching has been raised [4, 15, 23, 24].

Several authors have expressed the necessity to use approximate graph matching for frequent subgraph mining on graph collection. These authors defend the idea that frequent and more interesting subgraphs for applications and users could be found. Angryk and Hossain recommended frequent subgraph detection using approximate matching on document clustering tasks [15]. They believe that is possible to obtain a better clustering through the identification of patterns which allow some topological or semantic variations. These kinds of frequent approximate patterns have been recommended in tasks of chemical data processing [4], in tasks of link analysis [23], and have been presented as an open problem in tasks of data processing in molecules and social networks [24].

In response to this need, several algorithms have been developed for frequent subgraph mining which use approximate graph matching in different domains of science like: analysis of biochemical structures [5, 17, 31, 36, 37], genetic regulatory networks [29]; circuit analysis,

social networks, and link analysis [14].

In this paper, a new algorithm for frequent subgraph mining using an approximate matching method is proposed. This method extends the mining process presented by Jia *et al.* [17, 18], by allowing approximation over edge label set.

On the other hand, frequent subgraph mining has been successfully used in image classification [2, 8, 9, 19, 21, 28]. However, all of these works use exact algorithms in the frequent subgraph mining. They do not consider some substructures because rarely occur in the exact same form and orientation throughout the image collection. We believe that with the substructures identified by an approximate algorithm, the images could be better described and at the same time relevant features could be provided for image classification (see Section 5.2).

In this paper, a graph-based image representation and a framework for graph-based image classification are proposed. This framework uses the frequent subgraph as features obtained by frequent subgraph mining algorithms on an artificial image collection. Using this framework our proposed algorithm is evaluated through image classification.

The basic outline of this paper is as follows. Section 2 provides some basic concepts; it also contains an approximate method and the related work. The new approximate method for frequent approximate subgraph mining is provided in Section 3; this section also introduces the VEAM algorithm. The framework for image classification is introduced in Section 4 as a case study to evaluate the approximate algorithm proposed. The experimental results in six collections of images generated by a Random images generator are presented in Section 5; in this section we also present the discussion and in addition, we present a computational performance comparison between VEAM and APGM algorithms. 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; also, the frequent subgraph mining using a similarity function is defined. Next, the most relevant algorithms of related work are presented and we give an overview of the approaches closest to ours. Finally, some definitions of an approximate method used by one of the algorithms analyzed in this paper are presented, and an example of a reported frequent subgraph mining task proposed is described.

2.1. Basic concepts

This work is focused on simple undirected labeled graphs. Henceforth, when we refer to graph we assume this kind. Before presenting their formal definition, we define 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.

Given two graphs $G_1 = (V_1, E_1, I_1, J_1)$ and $G_2 = (V_2, E_2, I_2, J_2)$, where $G_1 \subseteq G_2$, we say that $e = \{u, v\} \in E_2$ is an *extension* of G_1 if: $V_2 = V_1 \cup \{v\}$ and $E_1 = E_2 \setminus \{e\}$. This fact can be denoted by $G_2 = G_1 \diamond e$. We say that e is a *backward extension* if $v \in V_1$, otherwise we say that it is a *forward extension* (it extends the vertex set of G_1).

In graph mining over collections of labeled graphs, a large number of candidates subgraph are processed. Most of these candidates have already been considered in a previous step, but they appear again through several frequent subgraphs during the search. These candidates are known as duplicate candidates. The duplicates are detected using isomorphism tests. 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*. One way to approach the isomorphism test is using canonical forms (CF) for representing graphs [3].

Let Ω be the set of all possible labeled graphs in L, the *similarity* between two elements $G_1, G_2 \in \Omega$ is defined as a function $sim : \Omega \times \Omega \to \mathbb{R}^+$, where $\mathbb{R}^+ \subset \mathbb{R}$ is the set of non-negative real numbers. We say that the elements are very different if $sim(G_1, G_2) = 0$ and the higher the value of $sim(G_1, G_2)$ the more similar the elements are.

Let $D = \{G_1, \ldots, G_{|D|}\}$ be a graph collection and G be a labeled graph in L, the *support* value of G in D is obtained through the following equation:

$$supp(G, D) = \sum_{G_i \in D} sim(G, G_i)/|D| \tag{1}$$

If $supp(G, D) \geq \delta$, then the graph G occurs approximately frequent in the collection D, saying that G is a *frequent approximate subgraph* in D. The value of the support threshold δ is in $(0,\ 1]$ assuming that the similarity is normalized to 1. The *frequent subgraph mining* consists in finding all the connected frequent approximate subgraphs in a collection of graphs D, using a similarity function sim and a support threshold δ.

There are several similarity functions used by different algorithms in the graph matching process [6]. In the Section 2.2, we present the most relevant algorithms in the related work which use approximate graph matching techniques in frequent subgraph mining. All of these algorithms use the definitions presented above, implementing a specific similarity function.

2.2. Related work

There are several algorithms for frequent approximate subgraph mining in graph collections which use different similarity functions for graph matching. The approximate subgraph mining can be divided into five kinds according to the matching approach:

1. Based on graph edit distance: the algorithms SUBDUE [14] and RNGV [29] explore possible edit paths of a graph for keeping the one expected as a candidate. These algorithms do not claim completeness.
2. Based on β-edge sub-isomorphism: the Monkey algorithm [35, 36] handles only missing edge and edge label mismatch, where β is the maximum number of edge differences allowed between subgraphs.
3. Based on node/edge disjoint sub-homeomorphism: the CSMiner algorithm [31, 32] finds approximate structures that share the same topology.
4. Based on sub-isomorphism on uncertain graphs: the MUSE algorithm [37, 38] computes the expected support of each candidate through a given interval. the graphs that this algorithm processes are uncertain because they contain a non-occurrence probability.
5. Based on substitution probabilities: the algorithms gApprox [5] and APGM [17, 18] specifies which vertices, edges or labels can replace others. Thus, the idea that not always a vertex label or an edge label can be replaced by any other is defended.

In *gApprox* algorithm each vertex contains a list of vertices that can replace it with the same probability; this means that each vertex can only be replaced by those in this list and by any of them. However, this algorithm performs frequent approximate subgraph mining on a single graph and we are interested in mining on graph collections. On the other hand, *APGM* algorithm uses a substitution matrix (see section 2.3) to perform the frequent subgraph mining on a graph collection. Although the authors suggested that it can be extensible to the edge labels, this algorithm only deals with the variations between the vertex labels.

In this paper, we propose a solution based on the idea of the last group, because we are looking for an algorithm which allows some variations of the data through the substitution probability, keeping the topology of the graphs.

2.3. An approximate method

In this section, we present the approximate method of APGM algorithm [17, 18], where the approximate matching is based on vertex label. This method uses the substitution matrix that can have a probabilistic interpretation and it offers a framework for this frequent subgraph mining task.

A *substitution matrix* $M = (m_{i,j})$ is an $|L| \times |L|$ matrix indexed by a label set L. An entry $m_{i,j}$ $(0 \leq m_{i,j} \leq 1, \sum_j m_{i,j} = 1)$ in M is the probability that the label i is replaced by the label j.

We say that M is *stable* if it is diagonal dominant (i.e. $M_{i,i} > M_{i,j},\ \forall j \neq i$). Henceforth, when we refer to substitution matrix we assume this kind of matrix.

Example 1. In Fig. 2, we show a substitution matrix MV of collection D showed in Fig. 1, where the vertex label set is $L_V = \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6\}$, and MV is indexed by L_V. The probability that the vertex label 0 is substituted by 2 is $m_{0,2} = 0.4$.

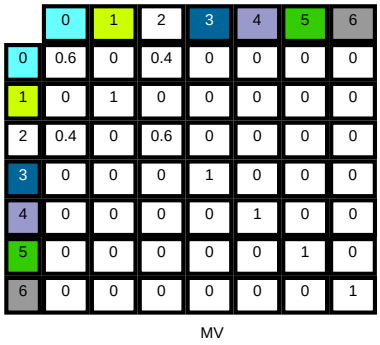

Figure 2: **Substitution matrix MV indexed by L_V.**

Definition 1 (Vertex approximate sub-isomorphism[1]). Given two labeled graphs $G_1 = (V_1, E_1, I_1, J_1)$, $G_2 = (V_2, E_2, I_2, J_2)$ in $L = L_E \cup L_V$, a substitution matrix MV indexed by L_V, and the isomorphism threshold τ whose value is in $[0,\ 1]$, G_1 is vertex approximately sub-isomorphic to G_2 if there exists an injection $f : V_1 \to V_2$ such that:

- $S_f(G_1, G_2) = \prod_{u \in V_1} \frac{MV_{I_1(u), I_2(f(u))}}{MV_{I_1(u), I_1(u)}} \geq \tau$,
- $\forall \{u, v\} \in E_1,\ \{f(u), f(v)\} \in E_2$,
- $\forall \{u, v\} \in E_1, J_1(\{u, v\}) = J_2(\{f(u), f(v)\})$,

Where $S_f(G_1, G_2)$ is the product of normalized probabilities called *vertex approximate sub-isomorphism score* of f.

Given a pair of graphs there are different ways of mapping vertices from one graph to another and hence they may be different vertex approximate sub-isomorphism score. The *vertex approximate matching score* between two graphs, denoted by $S(G_1, G_2)$, is the largest approximate sub-isomorphism score:

$$S(G_1, G_2) = max_f\{S_f(G_1, G_2)\} \tag{2}$$

The *frequent subgraph mining* task proposed by Jia *et al.* [17, 18] consists in finding all the connected frequent subgraphs in a collection of graphs D, using the Definition 1 as vertex approximate sub-isomorphism, δ as support threshold, and τ as isomorphism threshold.

In section 3, we present a proposal based on this method. Our proposal is more complex than the APGM approach, where the approximate matching between edge label set is included in this frequent subgraph mining task.

3. Frequent approximate subgraph mining

In this section, we present a new algorithm for frequent approximate subgraph mining. Before we proceed to the algorithmic details, we introduce the following definitions to facilitate the demonstration of our approximate method. Also in this section, the differences between VEAM and APGM through comparative examples and comments will be identified.

Definition 2 (Approximate sub-isomorphism). Let $G_1 = (V_1, E_1, I_1, J_1)$ and $G_2 = (V_2, E_2, I_2, J_2)$ be two labeled graphs in L, MV be a substitution matrix indexed by L_V, ME be a substitution matrix indexed by L_E, and τ be the isomorphism threshold. We say that G_1 is *approximate sub-isomorphic* to G_2, denoted by $G_1 \subseteq_A G_2$, if there exists an injection $h : V_1 \to V_2$ such that:

- $\forall\{u, v\} \in E_1,\ \{h(u), h(v)\} \in E_2$,
- $S_h(G_1, G_2) = \prod_{u \in V_1} \frac{MV_{I_1(u),I_2(h(u))}}{MV_{I_1(u),I_1(u)}} * \prod_{e=\{u,v\} \in E_1} \frac{ME_{J_1(e),J_2(\{h(u),h(v)\})}}{ME_{J_1(e),J_1(e)}} \geq \tau$.

The injection h is an *approximate sub-isomorphism* between G_1 and G_2, and $S_h(G_1, G_2)$ is the product of normalized probabilities called *approximate sub-isomorphism score* of h. Notice that the previous definition is an extension of the *vertex approximate sub-isomorphism* (see Definition 1), originally presented by Jia *et al.* [17, 18] for APGM algorithm, since it allows both, probabilistic vertex label substitution (as in Definition 1) and probabilistic substitutions on the edge label set.

Analogously, the *approximate matching score* between two graphs, denoted by $S_{max}(G_1, G_2)$, is the largest approximate sub-isomorphism score:

$$S_{max}(G_1, G_2) = max_h\{S_h(G_1, G_2)\} \tag{3}$$

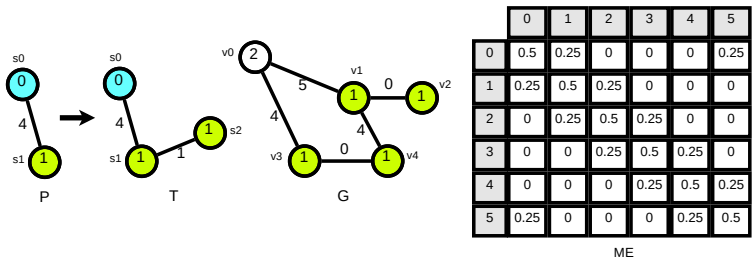

Figure 3: **Example of three labeled graphs in L and a substitution matrix ME indexed by L_E.**

Example 2. In Fig. 3, we show three labeled graphs in L and a substitution matrix ME for the graph collection showed in Fig. 1, where ME is a matrix indexed by L_E and $T \subseteq_A G$. We use the substitution matrix MV showed in Fig. 2 and matrix ME for computing the approximate matching score between $T = (V_T, E_T, I_T, J_T)$, $P = (V_P, E_P, I_P, J_P)$ and $G = (V_G, E_G, I_G, J_G)$. We have $P \subseteq_A G$ and $T = P \diamond \{s_1, s_2\} \subseteq_A G$, where the isomorphism

threshold $\tau = 0.15$, with the approximate matching score equal to 0.667 and 0.333 respectively. There are several ways for mapping vertices and edges of T to those of G, but only two of them satisfy the constraints of Definition 2. These ways are $h_1 : \{s_0, s_1, s_2\} \rightarrow \{v_0, v_1, v_2\}$ and $h_2 : \{s_0, s_1, s_2\} \rightarrow \{v_0, v_3, v_4\}$ where $h_1(s_0) = v_0$, $h_1(s_1) = v_1$, $h_1(s_2) = v_2$, $h_2(s_0) = v_0$, $h_2(s_1) = v_3$ and $h_2(s_2) = v_4$. These two ways fulfill the first and second constraints of Definition 2 and τ is less than its values of approximate sub-isomorphism score (see Fig. 4). The values of approximate sub-isomorphism of h_1 and h_2 are $S_{h_1}(T, G) = 0.167$ and $S_{h_2}(T, G) = 0.333$ respectively. Thus, the value of approximate matching score is $S_{max}(T, G) = 0.333$, because it is the highest value of $S_{h_1}(T, G)$ and $S_{h_2}(T, G)$.

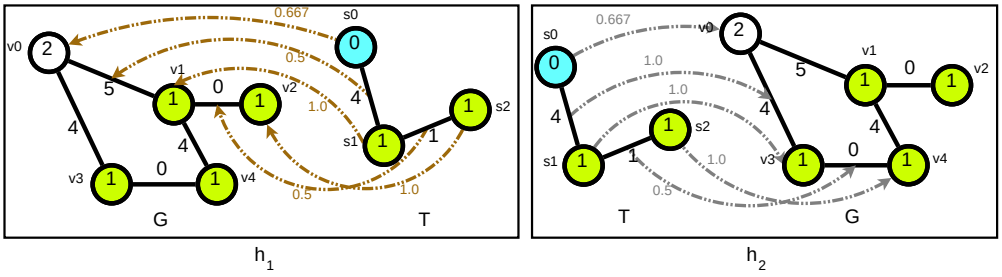

Figure 4: **Example of two ways to map vertices and edges of the labeled graph T to those of the labeled graph G.**

It is important to notice that the subgraph T is one of the patterns identified by our proposal, which is not identified by APGM. In fact, we use the Definition 2 in VEAM, while APGM uses the Definition 1. As it can be seen, the subgraph T does not satisfy the 4th constraint of Definition 1, where the same edge labels are required. Such constraint is missing in Definition 2 allowing us the identification of T through of an extension of P. This is where the label of the new edge $\{s_1, s_2\}$, which is identified as possible extension of P, do not exactly match with any edge labels of G (see Fig. 3).

Definition 3 (Embedding and embedding set). Let $G_1 = (V_1, E_1, I_1, J_1), G_2 = (V_2, E_2, I_2, J_2), T = (V_T, E_T, I_T, J_T)$ be three labeled graphs in L, where $T \subseteq G_2$. Using an isomorphism threshold τ, we say that T is an *embedding* of G_1 in G_2 if $G_1 \subseteq_A T$, $|V_1| = |V_T|$ and $|E_1| = |E_T|$. Thus, the *embedding set* of G_1 in G_2 is denoted by $O(G_1, G_2)$.

Example 3. In Fig. 5, we show a pattern P and one of its embeddings $S = (\{v_0, v_3\}, \{\{v_0, v_3\}\}, I, J), S \subset G_3$, where $\tau = 0.3$. The approximate matching score of the embedding is computed using the matrices MV showed in Fig. 2 and ME showed in Fig. 3, and its value is $S_{max}(P, S) = \frac{MV_{(0,2)}}{MV_{(0,0)}} * \frac{ME_{(4,5)}}{ME_{(4,4)}} * \frac{MV_{(1,1)}}{MV_{(1,1)}} = \frac{MV_{(0,2)}}{MV_{(0,0)}} * \frac{ME_{(4,5)}}{ME_{(4,4)}} = 0.333$. The embedding S is stored in $O(P, G_3)$ if $\tau \leq 0.3$. APGM does not identify S as an embedding of P in G_3 because it uses the Definition 1 and the edge label does not satisfy the constraints of this definition.

Definition 4 (Extension set). Let T an embedding of G_1 in $G_2 = (V_2, E_2, I_2, J_2)$, using an isomorphism threshold τ. Thus, the *extension set* of T is denoted by $ExtSet(T) = \{e \in E_2 \,|e$ is an extension of $T\}$.

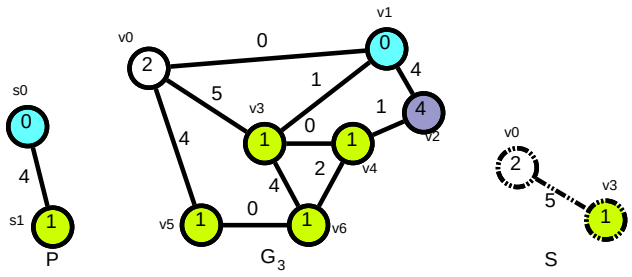

Figure 5: **Example of an embedding S of P in G_3 with an isomorphism threshold $\tau = 0.3$.**

The *frequent subgraph mining* task proposed in this paper consists in finding all the connected frequent subgraphs in a collection of graphs D, using the Definition 2 as approximate sub-isomorphism, δ as support threshold, and τ as isomorphism threshold.

3.1. Algorithm design

In this section, we propose a new algorithm for frequent approximate subgraphs mining called VEAM (**V**ertex and **E**dge **A**pproximate graph **M**iner). This algorithm incorporates in the approximate graph matching all labels that appear in the substitution matrices. Notice that if there are labels that do not appear in any graph of the collection and appear in the matrices, then these labels will be used in the mining process, and frequent approximate subgraphs with those labels can be obtained.

VEAM computes the frequency of the vertices, and starts mining from the frequent ones. At a subsequent step, it adds an edge to an existing pattern, in all possible ways, to create new candidate subgraphs and identify their support value. In this algorithm depth-first search (DFS) is used to find candidates by extending each current subgraphs pattern using a new edge. VEAM stops when no more patterns can be extended.

VEAM is shown through three pseudo-codes where the main algorithm consists in finding all frequent vertices, then these vertices are stored in sets F and C (see Algorithm 1). Later, for each $T \in C$ the algorithm $VEAMSearch$ is invoked. When all frequent vertices in C have been extended, then the set F of all frequent approximate subgraphs in D is returned.

Algorithm 1: $VEAM$

Input: D : A graph collection, MV : Substitution matrix indexed by L_V, ME : Substitution matrix indexed by L_E, τ : Isomorphism threshold, δ : Support threshold.

Output: F : Frequent approximate subgraph set.

```
1 F ← C ← {the single vertices labeled in L_V, which are frequent approximate
  subgraphs in D};
2 foreach T ∈ C do
3     VEAMSearch(T, D, MV, ME, τ, δ, F);
```

Algorithm 2 shows the pseudo-code that performs the extension of subgraph patterns on one edge. Thus, all candidate subgraphs are created using the label set obtained through the algorithm invoked in line 4. These candidates are created in two different forms: one is to add a backward extension to the subgraph pattern T (see line 7), and the other is to add a forward extension to T (see line 9). In order to make this pseudo-code behave like APGM is enough to replace *appLSet* call with *APGMappLSet* at line 4.

The set of frequent approximate subgraphs F is stored as a hash function to obtain efficiently the processed subgraphs. The hash key of the function is a canonical code of the subgraph pattern, which is a unique string representation of a graph. We use CAM and CAM code to compute the canonical code of a graph as in [12, 16]. Notice that a single graph G with $|V|$ vertices have $|V|!$ different adjacency matrices, since there are $|V|!$ possibilities to order the vertices of G. To obtain canonical code of G the CF tests are demanded (see line 10). These tests have very high computational complexity [3] as is mentioned above.

Algorithm 2: $VEAMSearch$

Input: $T = (V_t, E_t, I_t, J_t)$: A frequent approximate subgraph, D : Graph collection, MV : Substitution matrix indexed by L_V, ME : Substitution matrix indexed by L_E, τ : Isomorphism threshold, δ : Support threshold.

Output: F : Frequent approximate subgraph set.

```
1  foreach G_i = (V_i, E_i, I_i, J_i) ∈ D do
2      foreach o_j ∈ O(T, G_i) do
3          foreach e = {u, v} and e ∈ ExtSet(o_j) do
4              CL ← appLSet(T, MV, ME, G_i, o_j, e, τ);
5              foreach (elabel, vlabel) ∈ CL do
6                  if e is a backward extension of T then
7                      X = (V_t, E_t ∪ {e}, I_t, J_t ∪ {(e, elabel)});
8                  else
9                      X = (V_t ∪ {v}, E_t ∪ {e}, I_t ∪ {(v, vlabel)}, J_t ∪ {(e, elabel)});
10                 The code CAM of X is compute and store in codeCAM(X);
11                 C ← C ∪ {(X, codeCAM(X), score)};
12 foreach T_1 ∈ C do
13     if supp(T_1, D) ≥ δ and codeCAM(T_1) ∉ F then
14         Insert T_1 in F;
15         VEAMSearch(T_1, D, MV, ME, τ, δ, F);
```

Algorithm 3 seeks the possible approximate label set for the new edge e which is a backward extension of subgraph pattern T (see lines 1, 2 and 7), or seeks for each possible label of forward edge e, the possible label set of the new vertex that e connects with the existing vertex in T (see lines $1-6$).

Algorithm 3: *appLSet*

Input: T : A candidate graph, MV : Substitution matrix indexed by L_V, ME : Substitution matrix indexed by L_E, $G = (V, E, I, J)$: A graph of the collection, G' : Embedding of T in G, $e = \{u, v\}$: An extension of G', τ : Isomorphism threshold.

Output: CL : A set of candidate 2-tuple $(elabel, vlabel)$.

```
foreach j ∈ L_E do
    scoreE ← S_max(T, G') * ME_{j,J(e)} / ME_{j,j};
    if e is a forward extension of G' then
        foreach i ∈ L_V do
            score ← scoreE * MV_{i,I(v)} / MV_{i,i};
            if score ≥ τ then CL ← CL ∪ {(j, i)};
    else if scoreE ≥ τ then CL ← CL ∪ {(j, ∅)};
```

Algorithm 4: *APGMappLSet*

Input: T : A candidate graph, MV : Substitution matrix indexed by L_V, $G = (V, E, I, J)$: A graph of the collection, G' : Embedding of T in G, $e = \{u, v\}$: An extension of G', τ : Isomorphism threshold.

Output: CL : A set of candidate 2-tuple $(elabel, vlabel)$.

```
l_uv = J({u, v});
if e is a forward extension of G' then
    foreach i ∈ L_V do
        score ← S_max(T, G') * MV_{i,I(v)} / MV_{i,i};
        if score ≥ τ then CL ← CL ∪ {(l_uv, i)};
else CL ← CL ∪ {(l_uv, ∅)};
```

Algorithm 4 is presented in addition to show the differences between APGM and VEAM by making the least possible changes to adapt the VEAM algorithm. *APGMappLSet* obtains the possible approximate label set for the new vertex that the forward extension e connects with the existing vertex in T and always keeps the edge label of e in G. This pseudo-code is less expensive than Algorithm 3, because the computational cost of Algorithm 3 is $O(|L_E| * |L_V|)$ and Algorithm 4 is $O(|L_V|)$.

4. Scheme for image classification

In this section we start by introducing the graph-based image representation for frequent subgraph mining implementation. After that, we present a framework used to show the utility of our approximate method in image classification tasks. Details of image classification

are introduced with the proposed framework.

4.1. Graph-based image representation

Several techniques have been developed to represent images in graph forms [10, 25, 26]. This kind of representation has been a great help for image processing since graphs can describe the structural and topological information of images. The main idea of graph-based image representation is that the regions of the image, which contain similar properties, are denoted by graph vertices, and the relations between different regions are denoted by graph edges. The vertex and edge attributes usually describe the characteristics of that region and the relation between regions respectively. A simple approach to keep the structural and topological information of an image is to use digital image representation techniques; for instance, quad-trees [10], etc. By modeling images as graphs, the task of image classification becomes one of classifying graphs.

Quad-trees [10] have been used for representing images in the form of trees. A quad-tree consists in splitting an image in four equal-sized quadrants. Each of these quadrants can be further split into four sub-quadrants (NW, NE, SW and SE), and so on. The global process of the quad-tree consists in dividing the sub-images recursively until the imposed limit for the number of divisions is met. An example of quad-tree with 4 as depth limit of divisions over an image is showed in Fig. 6. When the depth limit of divisions is achieved and any sub-quadrant contains several properties to obtain, then the predominant property in this sub-quadrant is taken as attribute of a tree node. The property taken as an attribute of the nodes in the example of Fig. 6 is the color.

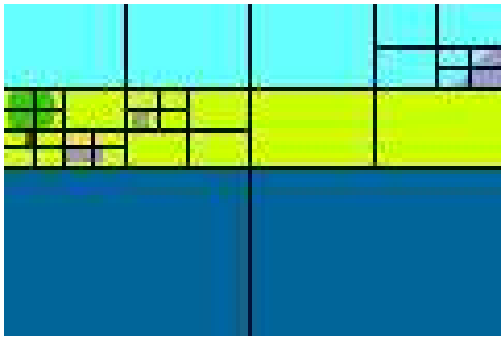

Figure 6: **Example image with the quadrants of quad-tree identified.**

After obtaining the quad-tree of an image, we generate a graph to represent the image with the goal of maintaining the structural information such as the relations between objects. So, the graph (image) mining can be performed to discover implicit patterns among graph collection. The process of graph generation proposed consists in three steps:

1. The vertices of the graph are made with the quad-tree leaves (sub-quadrants) with its attributes. The attributes of the leaf nodes of the quad-tree is the midpoint of the quadrant and the predominant attributes of the image is their labels (i.e. texture, color, etc.).

2. Iteratively for each level from top to bottom of the quad-tree:
   a) If the quadrant is a NW (NE) leaf, then it is joined by an edge to each of the most west (east) leaf in the NE (NW) sibling non-leaf quadrant, or by an edge to the NE (NW) sibling leaf in case it is so; doing the same with a SW (SE) leaf relative to the leaves of the SE (SW) sibling quadrant.
   b) If the quadrant is a NW (SW) leaf, then it is joined by an edge to each of the most north (south) leaf in the SW (NW) sibling non-leaf quadrant, or by an edge to the SW (NW) sibling leaf in case it is so; doing the same with a NE (SE) leaf relative to the leaves of the SE (NE) sibling quadrant.

   The most west leaves of a non-leaf quadrant at certain level are the leaves labeled as NW and SW in the following down level, or those leaves labeled as NW and SW but being recursively children of NW and SW sub-quadrants. The same condition is applied to the most east leaves with labels NE and SE, the most north leaves with labels NW and NE, and the most south leaves with labels SW and SE.

   Example 4. Let a quadrant divisions showed in Fig. 7 be the quad-tree of an image. We say that the vertex denoted by the number eight and each vertex of the set $\{5,\ 7,\ 9,\ 13,\ 14,\ 17\}$ are connected by an edge.

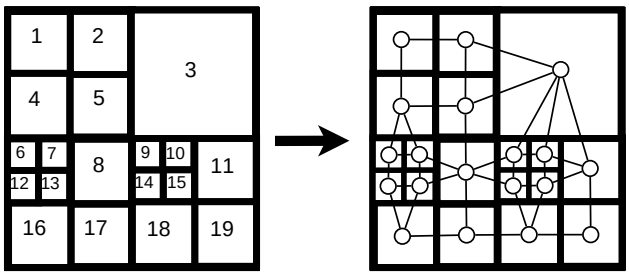

Figure 7: **Example of quadrant divisions of an image.**

3. The label of each edge e, denoted by $label(e)$, is an index obtained through the following function over the angle α between this edge and the horizontal line. This function depends on a value of the number of classes n that categorize the possible angles. The possible angles cover 180^o supposing that these edges are undirected.

$$label(e) = \begin{cases} \lfloor \frac{\alpha * n}{\pi} \rfloor & \text{if } 0 \leq \alpha < \pi \\ \\ 0 & \text{if } \alpha = \pi \end{cases} \tag{4}$$

In this paper, the Random image generator[2] of Coenen is used to obtain the collection of images. Later, we use the quad-tree to represent each image in a tree form. Then, a graph to represent each image in the collection is constructed from each respective tree. In Fig. 8, we show the graph generated from the quad-tree of Fig. 6 using $n = 6$.

[2]http://www.csc.liv.ac.uk/∼frans/KDD/Software/ImageGenerator/imageGenerator.html

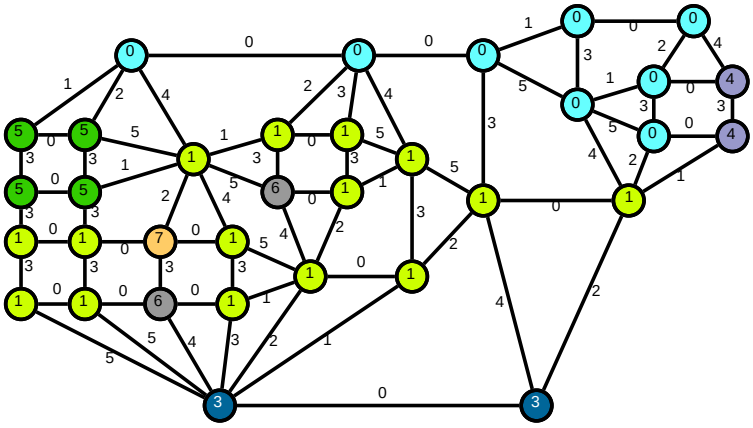

Figure 8: **Graph generated from the quad-tree shown in Figure 6 using $n = 6$.**

The matrices are built using the semantic of the data from images. The vertex labels of graphs indicate the predominant attributes (colors in our examples) of sub-quadrants from each represented image.

The collection that we used in our experiments contains 18 colors, which make up the set L_V. Only to illustrate the correspondences between the index number and the color we describe the colors as a 2-tuple (index, color) as follows: (0, white), (1, sea blue), (2, light yelow), (3, jungle green), (4, dark green), (5, dark brown), (6, brown), (7, light green), (8, light gray), (9, green), (10, light brown), (11, lilac), (12, gray), (13, light blue), (14, red), (15, orange), (16, black), (17, yellow).

With the previous colors, we create the substitution matrix MV used in our experimentations. In any row of this matrix, a color c has a 0.5 probability of being replaced by itself; the remaining 0.5 is equally distributed among the colors which describe the same group of objects as described by c. If a group of objects or an object are described by only one color, then this color can only be replaced by himself with a probability of 1.0. We distribute the substitution probabilities in MV according to the following groups of colors: $\{0,\ 8,\ 13\}$ are used to describe the sky; $\{1\}$ for the sea; $\{2,\ 17\}$ for the cabins and boat sails; $\{3,\ 4,\ 9,\ 10\}$ for the treetops; $\{7\}$ for the land; $\{12\}$ for the houses; $\{14\}$ for the boats; $\{15\}$ for the roofs; $\{16\}$ for the ships, house doors and windows. The label 5 can be substituted by 6 with a probability of 0.5 because are used for the tree trunks. The label 6 can be substituted by $\{10,\ 11\}$, because are used for the tree trunks and mountains. The label 11 can be substituted by $\{6,\ 10\}$ because are used for the mountains.

We create the substitution matrix ME used in our experimentations which allows some position variations of the objects in the images. These allowed variations of the angles formed by the edges and the horizontal lines is $\frac{\pi}{n}$. In the collection used as example in this paper we use several values of n. With the goal to allow slight position variations mentioned we define that the label 1 can be substituted by 0 or 2 with a probability of 0.5 and the same occurs for the rest of labels. So, the matrix ME showed in Fig. 3 is the matrix that we used to obtain our results.

4.2. Proposed framework

Given a set of pre-labeled images generated by the Random image generator, we obtain the quad-trees of these images; with these quad-trees the graph collection that represents these images are generated. After that, the algorithms for frequent subgraph mining are implemented with the goal of obtaining all frequent subgraphs over a graph collection. Later, these subgraphs are used as features and the feature vectors of the original images are made. Finally, we employ a classifier generator using these vectors as data to produce an image classifier. The complete flowchart of our experiment procedure is shown in Fig. 9.

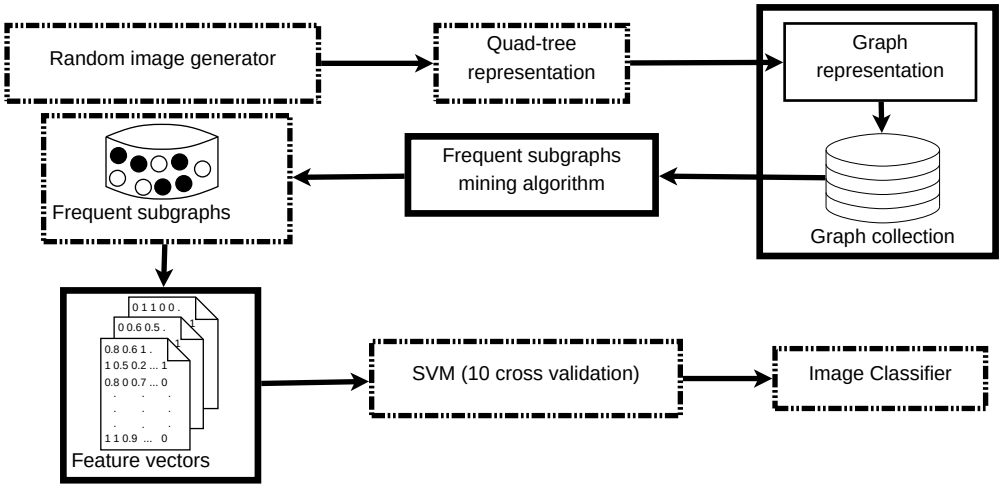

Figure 9: **Framework of graph-based image classification.**

This framework is similar to that proposed by Jiang and Coenen [19]. The phases represented by dashed lines in Fig. 9 are identical to the phases proposed by Coenen. We modified the three phases represented by solid lines showed in the same figure: (1) the graph representation used to obtain the graph collection (see section 4.1), (2) frequent subgraph mining algorithm (see section 3) and (3) feature vector built (see section 4.3).

4.3. Image classification

In order to evaluate the quality of the patterns identified we use the frequent subgraphs detected on classification tests. For that, the package libSVM[3] is used for image classification through *Support Vector Machine* (SVM) classifier. Given these frequent subgraphs, we build feature vectors upon them, then an image is represented as a feature vector $V = (v_1, \ldots, v_x)$, where x is the total number of subgraphs identified. Thus, we build a matrix where the row number ($1 \leq i \leq |D|$) corresponds to the number of graph (images) in the collection, and the number of columns ($1 \leq j \leq x$) corresponds to the number of frequent subgraphs (features). Each feature value can be assigned using the *binary setting* or *similarity setting*.

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

In the binary setting, an entry of the matrix $v_{i,j}$ is $v_{i,j} = 1$ if the feature j occurs in the image i of the collection and otherwise $v_{i,j} = 0$. An entry of matrix $v_{i,j}$ in the similarity setting is the highest similarityvalue of the occurrences of feature j in the image i of the collection and $v_{i,j} = 0$ in cases where feature j does not occur in image i. The similarity value of each feature is obtained through (1) using the similarity function that corresponds to each particular algorithm. In the APGM case, the matrix MV described in section 4.1 is used. In our case, we use the matrix ME described in the same section 4.1 and the same matrix MV used by APGM.

5. Experimental results

In this section, we show the experimental results that validate the efficacy of the proposed algorithm of frequent approximate subgraph mining for image classification. The results of classification using VEAM, gSpan [33] and APGM[4] algorithms are compared. These two last ones are chosen as the representation of the algorithms for frequent subgraph mining using exact matching (gSpan), and for the comparison between algorithms that use approximate matching (APGM). For the experiments we use a collection of 700 images obtained through a random image generator, and each image is represented as a labeled graph.

All our experiments were carried out using a personal computer (64 bits) Intel (R) Core (TM)2 Duo CPU E7300 @ 2.66 GHz with 2 Gb main memory. The algorithm VEAM for frequent approximate subgraph mining was implemented in ANSI C language and compiled using gcc compiler of GNU/Linux with -O0 optimization.

5.1. Graph collection

The collection consists of 700 images obtained by the Random image generator. These images are classified as either "landscape" or "seascape" according to content as shown in Fig. 10. Each of these images is represented as quad-tree with 4 as depth limit of divisions. From these trees the graphs that represent each image of the collection were built using several values of n. The process of building the graphs is explained in detail on Section 4.1.

The collection was divided into six sub-collections with different sizes to be used in our experiments. These sub-collections are described in table 1 where the first column is the given name of the sub-collection, P_D is the number of graphs in the collection, P_T is the average size of graphs (in terms of the number of edges), P_V is the number of vertex labels, and P_E is the number of edge labels.

5.2. Frequent approximate subgraphs identified

The frequent approximate subgraphs are identified according to the definition of two arguments which are made pruning the search space in the mining process. The first argument is the support threshold ($0 < \delta \leq 1$) and the second is the isomorphism threshold

[4]The APGM algorithm used in our experimentation is implemented by us to detect all frequent vertex approximate subgraphs and not just the clique subgraphs as proposed by its authors [17, 18].

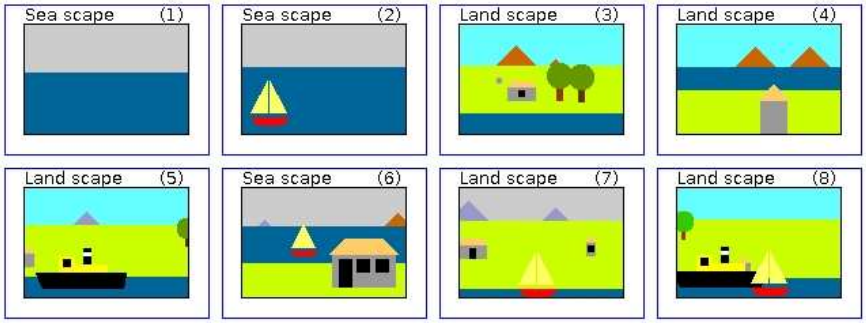

Figure 10: **Sample images by Random image generator.**

Table 1: Graph collections used in the experiments, where $n = 24$.

Data	P_D	P_V	P_T	P_E
D_1	200	18	45	24
D_2	300	18	43	24
D_3	400	18	45	24
D_4	500	18	46	24
D_5	600	18	47	24
D_6	700	18	47	24

$(0 < \tau \leq 1)$. The identified frequent approximate subgraphs are shown in table 2. Notice that the value of τ is set to 0.4 in APGM and VEAM algorithms.

Table 2: Number of patterns identified by gSpan, APGM($\tau = 0.4$) and VEAM($\tau = 0.4$) on several graph collections, which were built using $n = 24$.

Data	**Algorithm**	**Support (δ)**								
		20%	**25%**	**30%**	**35%**	**40%**	**45%**	**50%**	**55%**	**60%**
D_6	gSpan	711	243	72	26	11	8	8	7	6
	APGM	746	266	81	30	14	11	11	10	9
	VEAM	864	321	116	44	23	13	12	11	10
D_5	gSpan	347	150	55	19	9	8	8	7	6
	APGM	385	172	64	23	12	11	11	10	9
	VEAM	498	257	99	35	20	13	12	11	10
D_4	gSpan	287	145	54	21	9	8	8	6	6
	APGM	321	168	63	24	12	11	11	9	9
	VEAM	453	238	93	35	19	13	12	11	10
D_3	gSpan	286	99	38	19	8	8	8	6	5
	APGM	325	121	47	23	11	11	11	9	8
	VEAM	433	203	79	31	18	13	12	11	9
D_2	gSpan	227	77	32	17	8	8	6	6	3
	APGM	258	99	42	24	11	11	9	9	6
	VEAM	374	154	69	31	17	13	11	10	7
D_1	gSpan	197	60	30	20	9	8	6	6	3
	APGM	227	82	41	26	12	11	9	9	6
	VEAM	340	143	72	35	18	13	11	10	6

The isomorphism threshold (τ) was selected according to the substitution matrices used in the experimentation (described in section 4.1). Given the features of the matrix used in

the approximate matching between edge labels (see Figure 3), it is necessary that τ must be less than or equal to 0.5 in order to get an approximation between these labels. About the matrix indexed by the vertex labels, it is necessary that τ must be less than or equal to 0.66. For these reasons, we choose τ as 0.4 to satisfy the above considerations.

The matrices used in APGM and VEAM on frequent subgraphs detection are built using the semantic of the data from images as described in section 4.1. In order to illustrate how different is our approximate algorithm (in terms of the number of identified patterns), Fig. 11 shows the subgraphs identified by VEAM algorithm over the collection of Fig. 1 using the matrices of figures 2 and 3 with $\delta = 0.5$ and $\tau = 0.4$. In case of using an exact algorithm over the same collection only subgraphs which are inside the box in Fig. 11 had been identified. On the other hand, if we use the APGM algorithm the subgraphs represented by dashed lines are identified.

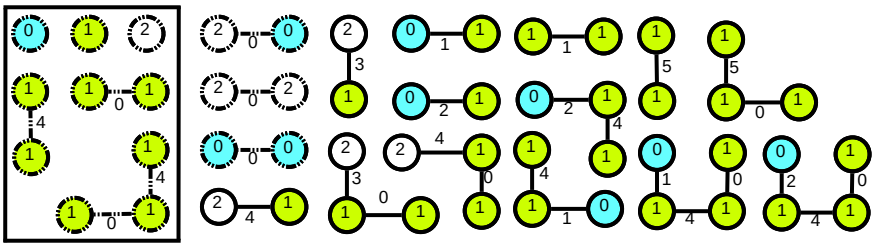

Figure 11: **A set of frequent subgraphs extracted by VEAM algorithm over the collection shown in figure 1, where $\delta = 0.5$ and $\tau = 0.4$.**

Our proposed algorithm provides more patterns than any exact algorithm and APGM algorithm because some concepts or reasonably similar images are grouped. The fact that the patterns identifies by VEAM provide more information is because our algorithm identifies interesting patterns with comprehensible or acceptable semantic variations in the edges and vertices which gSpan cannot detect. The slight variations about the rotation of the edges allow us to detect patterns which APGM cannot identify. These variations are very common in processing image collections, because the colors may not be exactly the same and the objects may appear with some difference of position on several images that illustrate the same context.

5.3. Computational performance

As we explained in section 3 and it was argued in section 5.2, our proposed algorithm extends more candidate subgraphs than APGM. The number of CF tests performed by an algorithm is directly proportional to the candidate subgraphs processed.

In order to compare the performance of VEAM and APGM algorithms we included the following tables. In table 3 we specify the quantities of CF tests performed by each algorithm on several graph collections built with $n = 24$ (see section 4.1). We can observe that VEAM requires twice (and more) the amount of test APGM should do in most of the cases. These tests become much more complex to great extent with the increase of the candidate size. The values showed in this table are due to the extensive search space of VEAM to include edge variations into the graph matching process.

Table 3: Number of CF test performed by APGM and VEAM with $\tau = 0.4$ on several graph collections, which were built using $n = 24$.

Data	**Algorithm**	**Support (δ)**								
		20%	**25%**	**30%**	**35%**	**40%**	**45%**	**50%**	**55%**	**60%**
D_6	APGM	245986	79972	20993	6981	3084	2413	2413	2077	1782
	VEAM	605103	208285	64894	23089	11585	6367	5779	4999	3795
D_5	APGM	98916	41886	14900	5034	2544	2336	2336	2030	1738
	VEAM	287765	147907	51706	17761	9977	6195	5631	4881	3704
D_4	APGM	76711	39654	13994	5036	2466	2260	2260	1685	1685
	VEAM	240419	127856	45049	16700	9184	5946	5403	4693	3564
D_3	APGM	70379	24733	9038	4508	2147	2147	2147	1602	1320
	VEAM	205322	96673	34190	13915	8337	5663	5145	4461	3376
D_2	APGM	50075	18072	7469	4287	2075	2075	1561	1561	923
	VEAM	157470	62541	27491	12527	7345	5459	4338	3765	1902
D_1	APGM	37618	12964	6710	4378	2087	1905	1457	1457	875
	VEAM	115926	49648	24915	12661	7165	5032	4030	3507	1793

In this paper, we do not present comparisons of runtimes because the patterns which process the algorithms are very different (see Table 2). There is a great difference between the number of CF tests performed by APGM and VEAM algorithms (see Table 3).

5.4. Classification results

In order to evaluate our approximate algorithm the SVM is used with two kernels: *Linear* and *Radial Basis Function* (RBF). As described in Section 4.3, the classification is performed on feature vectors, where their values are obtained using the binary setting or similarity settings. The binary setting is used for gSpan results, while the similarity setting is used for APGM and VEAM results.

The classification results are summarized in table 4. Subtable (a) of this table illustrates the results obtained using the RBF kernel and subtable (b) illustrates the results using the linear kernel. The first column of these subtables indicates the collection used, the next column shows the support threshold used in the frequent subgraphs mining task and the last three columns correspond to the accuracies achieved with kernels taken in the SVM classification algorithm. The classification accuracy is tested on 10 cross-validation.

The results of Table 4 are due to the patterns identified by the algorithms providing useful information for classification. Moreover, in order that the APGM and VEAM algorithms identify these patterns, we consider that δ must be greater than or equal to 20% and less than or equal to 60%. On the other hand, these results (see Table 4) show that the patterns obtained by VEAM are better than those found by APGM and gSpan in image classification tasks. Thus, VEAM obtains the best accuracies in most of the cases.

5.5. Discussion

The use of approximate algorithms is necessary when enough features for a good classification cannot be identified by exact methods. This is shown through the results presented in Section 5.4 where $\delta = 55\%$ and $\delta = 50\%$. However, APGM and VEAM algorithms report in a few of the cases the same number of subgraphs with largest size when $\delta = 55\%$ and

Table 4: Accuracies achieved using SVM on different sets of graphs (images), which were built using $n = 24$.

(a) Using RBF kernel

Data	δ	gSpan	APGM	VEAM
	60%	**78.7143**%	**78.7143**%	**78.7143**%
	55%	78.5714%	78.5714%	**84.1429**%
	50%	78.5714%	78.5714%	**83.5714**%
	45%	78.5714%	78.5714%	**83.8571**%
D_6	40%	83.7143%	83.7143%	**94.5714**%
	35%	91.7143%	91.7143%	**95.4286**%
	30%	92.7143%	93%	**95.5714**%
	25%	90.5714%	90.2857%	**95.5714**%
	20%	90.7143%	90.2857%	**95.2857**%
	60%	77.6667%	77.6667%	**78.3333**%
	55%	77%	77.6667%	**85.6667**%
	50%	77.1667%	77.3333%	**85.5**%
	45%	77.1667%	77.3333%	**85.5**%
D_5	40%	84.6667%	84.6667%	**93.5**%
	35%	92.5%	92%	**93.5**%
	30%	92.8333%	92.8333%	**95.3333**%
	25%	90.3333%	90%	**95**%
	20%	90.6667%	90.5%	**95.6667**%
	60%	79.2%	79.4%	**79.8**%
	55%	79.2%	79.4%	**83.4**%
	50%	79.2%	79.8%	**81.8**%
	45%	79.2%	79.8%	**83.8**%
D_4	40%	85%	85%	**93**%
	35%	92.4%	92.4%	**93.6**%
	30%	93%	92.2%	**95.4**%
	25%	91.2%	91.2%	**95.4**%
	20%	90.4%	90%	**94.8**%
	60%	81.75%	82%	**82.25**%
	55%	81.75%	81.75%	**82.5**%
	50%	81.75%	81.75%	**82**%
	45%	81.75%	81.75%	**82.5**%
D_3	40%	81.75%	81.75%	**95**%
	35%	92.5%	92.5%	**93**%
	30%	92%	91.25%	**96.75**%
	25%	90.75%	90.75%	**94.75**%
	20%	91.25%	90.75%	**93.25**%
	60%	**82.6667**%	**82.6667**%	**82.6667**%
	55%	79.6667%	81.3333%	**91.3333**%
	50%	79.6667%	81.3333%	**88.3333**%
	45%	80%	80.3333%	**87**%
D_2	40%	80%	80.3333%	**94.3333**%
	35%	92%	92.3333%	**94**%
	30%	92%	91%	**94.3333**%
	25%	91%	91%	**95.3333**%
	20%	92%	**92.6667**%	**92.6667**%
	60%	**81.5**%	**81.5**%	**81.5**%
	55%	80.5%	81.5%	**85**%
	50%	80.5%	81.5%	**83.5**%
	45%	78%	78%	**82**%
D_1	40%	87.5%	88.5%	**92**%
	35%	**92**%	**92**%	**92**%
	30%	**92.5**%	92%	**92.5**%
	25%	91.5%	91%	**93.5**%
	20%	**93**%	92%	92.5%

(b) Using Linear kernel

Data	δ	gSpan	APGM	VEAM
	60%	78.1429%	78.1429%	**78.7143**%
	55%	79.4286%	79.4286%	**83.5714**%
	50%	79%	79.4286%	**83.4286**%
	45%	79%	79.4286%	**83.5714**%
D_6	40%	83.5714%	83.5714%	**95.4286**%
	35%	91.7143%	92%	**95.7143**%
	30%	92%	92%	**95.5714**%
	25%	92.4286%	93%	**95.5714**%
	20%	95.2857%	93.7143%	**95.8571**%
	60%	76.6667%	76.8333%	**77.6667**%
	55%	76%	76.5%	**85.6667**%
	50%	76.5%	76.5%	**85.6667**%
	45%	76.5%	76.5%	**85.6667**%
D_5	40%	84.5%	84.5%	**95.3333**%
	35%	92.1667%	92.5%	**94.5**%
	30%	90.8333%	91.3333%	**95.6667**%
	25%	92.6667%	93.1667%	**95.3333**%
	20%	94.8333%	93%	**95.8333**%
	60%	76%	75.6%	**77**%
	55%	76%	75.6%	**85.6**%
	50%	75.8%	75.8%	**85.6**%
	45%	75.8%	75.8%	**85.6**%
D_4	40%	84.6%	84.6%	**95.6**%
	35%	93%	93%	**96**%
	30%	92.2%	92.6%	**97**%
	25%	92.8%	94%	**97.2**%
	20%	94.2%	95.2%	**96.6**%
	60%	81.75%	82%	**84.25**%
	55%	81.75%	83.75%	**87.75**%
	50%	81.5%	83.75%	**87.75**%
	45%	81.5%	83.75%	**87.25**%
D_3	40%	81.5%	83.75%	**95**%
	35%	93%	93%	**96**%
	30%	92.75%	92.75%	**97**%
	25%	92.75%	93.75%	**97.75**%
	20%	94.25%	93.25%	**96.75**%
	60%	82%	82%	**82.3333**%
	55%	80.3333%	87%	**90.6667**%
	50%	80.3333%	87%	**91**%
	45%	80%	87%	**91**%
D_2	40%	80%	87%	**95**%
	35%	93%	92%	**95.6667**%
	30%	91.6667%	91.6667%	**94.3333**%
	25%	92.3333%	92.6667%	**96.3333**%
	20%	94.3333%	94.3333%	**97.3333**%
	60%	**82.5**%	**82.5**%	**82.5**%
	55%	81.5%	82.5%	**87**%
	50%	81.5%	82.5%	**87**%
	45%	81.5%	82.5%	**86**%
D_1	40%	87.5%	87%	**92.5**%
	35%	92%	92.5%	**95.5**%
	30%	93.5%	92.5%	**95.5**%
	25%	91.5%	95%	**96**%
	20%	93%	92%	**97.5**%

$\delta = 50\%$. Besides that, Table 2 shows that VEAM generally detects more subgraphs than gSpan and APGM.

Table 5: Number of frequent subgraphs identified by gSpan, APGM ($\tau = 0.4$) and VEAM ($\tau = 0.4$) on several graph collections, which were built using $n = 24$.

Data	**Algorithm**	$\delta = 55\%$ **Size of subgraphs (number of edges)**				$\delta = 50\%$ **Size of subgraphs (number of edges)**			
		0	1	2	3	0	1	2	3
D_6	gSpan	2	2	2	1	2	2	2	2
	APGM	5	2	2	1	5	2	2	2
	VEAM	5	3	2	1	5	3	2	2
D_5	gSpan	2	2	2	1	2	2	2	2
	APGM	5	2	2	1	5	2	2	2
	VEAM	5	3	2	1	5	3	2	2
D_4	gSpan	2	2	1	1	2	2	2	2
	APGM	5	2	1	1	5	2	2	2
	VEAM	5	3	2	1	5	3	2	2
D_3	gSpan	2	2	1	1	2	2	2	2
	APGM	5	2	1	1	5	2	2	2
	VEAM	5	3	2	1	5	3	2	2
D_2	gSpan	2	2	1	1	2	2	1	1
	APGM	5	2	1	1	5	2	1	1
	VEAM	5	3	1	1	5	3	2	1
D_1	gSpan	2	2	1	1	2	2	1	1
	APGM	5	2	1	1	5	2	1	1
	VEAM	5	3	1	1	5	3	2	1

As mentioned in Section 5.2, it is common that objects in image representing the same context varies in position. That is why VEAM allows slight angular differences between the positions of image segments. This makes that the patterns identified describe permissible variations in images for those collections. Table 5 shows the number of subgraphs reported for each size where $\delta = 55\%$ and $\delta = 50\%$ on a graph collections which were built using $n = 24$. As it can be seen in more than 50% of the cases, VEAM detects more frequent subgraphs than gSpan and APGM for each number of edges. We consider that more frequent subgraphs semantically related to the model can provide information necessary to a better characterization of it, specially when there are very few graphs or none obtained by other models. So, VEAM algorithm achieves better accuracies for these frequency thresholds in particular, and in general.

On the other hand, a high number of edge labels were used. This allowed us to show the effectiveness of our approximate approach when there is a high diversity of edge labels in a graph collection. In this paper, the function $label(e)$ is used to obtain the edge label of any edge e in the graphs of a collection (see Section 4.1). The edge labels of the graph collection are clustered according to n because the function $label(e)$ depend on that parameter. For this reason, with the increasing of n, the number of edge labels increases and the range of angular differences between the positions of image segments allowed in the matching process of all algorithms decreases. This range can be reduced to zero if $n = 180$, then there are as many edge labels in the collection as angles between the edges and the horizontal

line. VEAM, even if $n = 180$, allows a slight angular differences between the positions of segments in images. These differences are specified by the substitution matrix indexed by L_E and they allow VEAM to keep a stable behavior unlike APGM and gSpan algorithms. To show the effect of increasing n on the images, we use a series of results obtained using n in the interval $[12, 30]$. As an example, we use the first collection (D_6) as the data set where the RBF kernel and $\delta = 20\%$. The figure 12 show that the VEAM algorithm obtains results with the increasing of the number of edge labels.

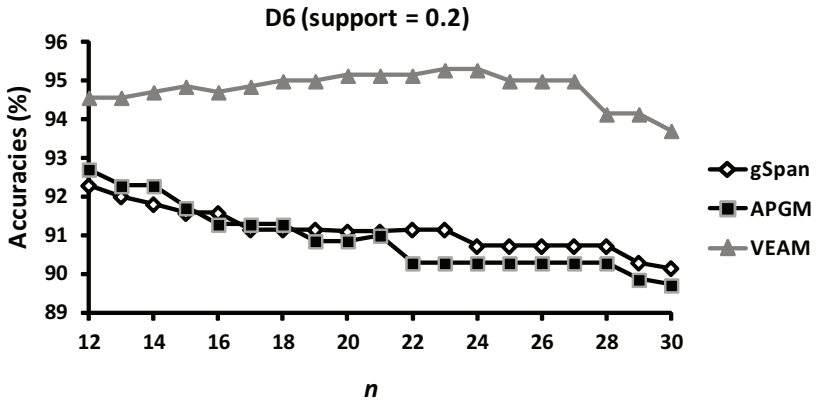

Figure 12: **Example of** *accuracies*$/n$ **obtained by gSpan, APGM and VEAM on several graph collections.**

The overall results show that with the use of approximate algorithms we can achieve better results than with exact ones for image classification. Also, the results show that our approach achieves a better classification using RBF kernel function for the data types used in the experiments.

6. Conclusions and future work

In this paper, we propose a new algorithm called VEAM for frequent approximate subgraphs mining on graph collection where the graphs are labeled and undirected. VEAM uses a new approximate method considering approximation in vertex and edge label set. This approach identifies the frequent patterns in collections of images allowing slight angular differences between the positions of image segments. We also present a graph-based image representation using the information provided by quad-tree technique.

The experimental results show the efficacy of our approximate algorithm versus exact algorithms and APGM algorithm on image classification. The patterns detected by VEAM allow semantic variations which are not considered on exact approaches. Moreover, our proposal also allows some object position variations which cannot be treated by APGM; since APGM only considers semantic variations on vertex labels. Semantic variations on edges can be used for modeling many real datasets. As an example of the application of our

approach, we propose in this paper a solution for image collections. Therefore, the accuracy results of classification obtained through the exact algorithms and APGM algorithm are smaller than the obtained by VEAM in almost all cases. Thus, we conclude that the patterns identified by our proposal are more representative, in accordance with these results, than those detected by APGM and gSpan. Finally, we show the usefulness of using frequent approximate patterns to improve image classification.

As future work, we will propose a new interest measures to reduce the number of patterns identified which keep or increase the classification accuracy. We will develop novel prune strategies to obtain algorithms more efficient for larger collections.

References

[1] R. Alves, D.S. Rodríguez-Baena, J.S. Aguilar-Ruiz. Gene association analysis: a survey of frequent pattern mining from gene expression data. Briefings in Bioinformatics 11 (2010) 210–224.

[2] O. Bahadir, A. Selim. Image classification using subgraph histogram representation. in: Proceedings of the 20th International Conference on Pattern Recognition, IEEE Computer Society, Washington, DC, USA, 2010, pp. 1112–1115.

[3] C. Borgelt. Canonical forms for frequent graph mining. in: Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation *e.V.*, Universitat Berlin, 2006, pp. 8–10.

[4] C. Borgelt, M. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. in: Proceedings of 2002 International Conference on Data Mining (ICDM'02), Maebashi, Japan, 2002, pp. 211–218.

[5] C. Chen, X. Yan, F. Zhu, J. Han. gapprox: Mining frequent approximate patterns from a massive network. in: International Conference on Data Mining, IEEE Computer Society, 2007, pp. 445–450.

[6] D. Conte, P. Foggia, C. Sansone, M. Vento. Thirty years of graph matching in pattern recognition. International Journal of Pattern Recognition and Artificial Intelligence (2004).

[7] F. Eichinger, K. Böhm. Software-Bug Localization with Graph Mining. in: C.C. Aggarwal, H. Wang (Eds.), Managing and Mining Graph Data, volume 40, Springer-Verlag New York, 2010.

[8] A. Elsayed, F. Coenen, C. Jiang, M. García-Fiñana, V. Sluming. Region of interest based image categorization. in: Proceedings of the 12th international conference on Data warehousing and knowledge discovery, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 239–250.

[9] A. Elsayed, F. Coenen, C. Jiang, M. García-Fiñana, V. Sluming. Corpus callosum mr image classification. Knowledge-Based Systems 23 (2010) 330–336.

[10] R.A. Finkel, J.L. Bentley. Quad trees: A data structure for retrieval on composite keys. Acta Informatica 4 (1974) 1–9.

[11] A. Gago-Alonso, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, J.F.M. Trinidad. Duplicate candidate elimination and fast support calculation for frequent subgraph mining. in: Proceedings of the 10th international conference on Intelligent data engineering and automated learning, Springer-Verlag, Berlin Heidelberg, 2009, pp. 292–299.

[12] A. Gago-Alonso, A. Puentes-Luberta, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, J.F. Martínez-Trinidad. A new algorithm for mining frequent connected subgraphs based on adjacency matrices. Intelligence Data Analysis 14 (2010) 385–403.

[13] J. Han, H. Cheng, D. Xin, X. Yan. Frequent pattern mining: Current status and future directions. Data Mining and Knowledge Discovery, 10th Anniversary Issue 15 (2007) 55–86.

[14] L.B. Holder, D.J. Cook, H. Bunke. Fuzzy substructure discovery. in: Proceedings of the ninth international workshop on Machine learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992, pp. 218–223.

[15] M.S. Hossain, R.A. Angryk. Gdclust: A graph-based document clustering technique. in: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops, IEEE Computer Society, Washington, DC, USA, 2007, pp. 417–422.

[16] J. Huan, W. Wang, J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. in: Proceedings of the 3rd IEEE International Conference on Data Mining, IEEE Computer Society, 2003, pp. 549–552.
[17] Y. Jia, J. Huan, V. Buhr, J. Zhang, L. Carayannopoulos. Towards comprehensive structural motif mining for better fold annotation in the "twilight zone" of sequence dissimilarity. BMC Bioinformatics 10 (2009).
[18] Y. Jia, J. Zhang, J. Huan. An efficient graph-mining method for complicated and noisy data with real-world applications. Knowl. Inf. Syst. 28 (2011) 423–447.
[19] C. Jiang, F. Coenen. Graph-based image classification by weighting scheme. in: Proceedings of the Artificial Intelligence, Springer, Heidelberg, 2008, pp. 63–76.
[20] C. Jiang, F. Coenen, R. Sanderson, M. Zito. Text classification using graph mining-based feature extraction. Knowledge-Based Systems 23 (2010) 302–308.
[21] C. Jiang, F. Coenen, M. Zito. Frequent sub-graph mining on edge weighted graphs. in: Proceedings of the 12th international conference on Data warehousing and knowledge discovery, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 77–88.
[22] A. Jiménez, F. Berzal, J.C.C. Talavera. Frequent tree pattern mining: A survey. Intelligence Data Analysis 14 (2010) 603–622.
[23] N. Ketkar, L. Holder, D. Cook. Mining in the proximity of subgraphs. Analysis and Group Detection KDD Workshop on Link Analysis: Dynamics and Statics of Large Networks (2006).
[24] M. Koyutürk, A. Grama, W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. Bioinformatics (2004) 200–207.
[25] R. Marfil, F. Sandoval. Energy-based perceptual segmentation using an irregular pyramid. in: Proceedings of the 10th International Work-Conference on Artificial Neural Networks, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 424–431.
[26] A. Morales-González, E.B.G. Reyes. Assessing the role of spatial relations for the object recognition task. The 15th Iberoamerican Congress on Pattern Recognition (2010) 549–556.
[27] S. Nijssen, J.N. Kok. A quickstart in frequent structure mining can make a difference. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (2004) 647–652.
[28] S. Nowozin, K. Tsuda, T. Uno, T. Kudo, G. BakIr. Weighted substructure mining for image analysis. in: Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, 2007, pp. 1–8.
[29] Y. Song, S.S. Chen. Item sets based graph mining algorithm and application in genetic regulatory networks. Data Mining, IEEE International Conference on Volume, Issue (2006) 337–340.
[30] C.H. Weng. Mining fuzzy specific rare itemsets for education data. Knowledge-Based Systems (2011) 697–708.
[31] Y. Xiao, W. Wang, W. Wu. Mining conserved topological structures from large protein-protein interaction networks. in: Proceedings of the 18th IEICE data engineering workshop / 5th DBSJ annual meeting, DEWS'2007, Hiroshima, Japan, 2007.
[32] Y. Xiao, W. Wu, W. Wang, Z. He. Efficient algorithms for node disjoint subgraph homeomorphism determination. in: Proceedings of the 13th international conference on Database systems for advanced applications, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 452–460.
[33] X. Yan, J. Huan. gspan: Graph-based substructure pattern mining. in: Proceedings International Conference on Data Mining, Maebashi, Japan, 2002, pp. 721–724.
[34] U. Yun, K.H. Ryu. Approximate weighted frequent pattern mining with/without noisy environments. Knowledge-Based Systems 24 (2011) 73–82.
[35] S. Zhang, J. Yang. Ram: Randomized approximate graph mining. in: Proceedings of the 20th International Conference on Scientific and Statistical Database Management, ICSSDM, 2008, pp. 187–203.
[36] S. Zhang, J. Yang, V. Cheedella. Monkey: Approximate graph mining based on spanning trees. in: International Conference on Data Engineering, IEEE ICDE, Los Alamitos, CA, USA, 2007, pp. 1247–1249.

[37] Z. Zou, J. Li, H. Gao, S. Zhang. Frequent subgraph pattern mining on uncertain graph data. in: Proceeding of the 18th ACM conference on Information and knowledge management, ACM, New York, NY, USA, 2009, pp. 583–592.
[38] Z. Zou, J. Li, H. Gao, S. Zhang. Mining frequent subgraph patterns from uncertain graph data. IEEE Transactions on Knowledge and Data Engineering 22 (2010) 1203–1218.