# A New Method Based on Graph Transformation for FAS Mining in Multi-graph Collections

Niusvel Acosta-Mendoza[1,2(✉)], Jesús Ariel Carrasco-Ochoa[2], José Fco. Martínez-Trinidad[2], Andrés Gago-Alonso[1], and José E. Medina-Pagola[1]

[1] Advanced Technologies Application Center (CENATAV), 7a ♯ 21406 e/ 214 and 216, Siboney, Playa, CP: 12200 Havana, Cuba
{nacosta,agago,jmedina}@cenatav.co.cu
[2] Instituto Nacional de Astrofísica, Óptica Y Electrónica (INAOE), Luis Enrique Erro No. 1, Sta. María Tonantzintla, CP: 72840 Puebla, Mexico
{nacosta,ariel,fmartine}@ccc.inaoep.mx

**Abstract.** Currently, there has been an increase in the use of frequent approximate subgraph (*FAS*) mining for different applications like graph classification. In graph classification tasks, FAS mining algorithms over graph collections have achieved good results, specially those algorithms that allow distortions between labels, keeping the graph topology. However, there are some applications where multi-graphs are used for data representation, but FAS miners have been designed to work only with simple-graphs. Therefore, in this paper, in order to deal with multi-graph structures, we propose a method based on graph transformations for FAS mining in multi-graph collections.

**Keywords:** Approximate graph mining · Approximate multi-graph mining · Graph-based classification

## 1 Introduction

In Data Mining, frequent pattern identification has become an important topic with a wide range of applications in several domains of science, such as: biology, chemistry, social science and linguistics, among others [1]. Therefore different techniques for pattern extraction, where frequent subgraph mining algorithms have been developed.

From these techniques, in the last years, the most popular strategies are the approximate approaches, because in practice there are specific problems where exact matching is not applicable with a positive outcomes [2–4]. Therefore, it is important to tolerate certain level of variability: semantic distortions, vertices or edges mismatching during frequent pattern search it implies to evaluate the similarity between graphs considering approximate matching. In this sense, several algorithms have been developed for frequent approximate subgraph (*FAS*) mining, which use different approximate graph matching techniques allowing the detection


J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2015, LNCS 9116, pp. 13–22, 2015.
DOI: 10.1007/978-3-319-19264-2_2

of frequent subgraphs when some minor (non-structural) variations in the subgraphs are permitted [3,5]. Different heuristics and graph matching approaches have been used as basis for developing FAS mining algorithms, for example: edit distance [6], heuristic over uncertain graphs [4,7], and heuristics based on semantic substitutions [3,5,8–10], being this last approach the idea followed in this paper. However, these algorithms perform FAS mining on simple-graph collections, and they were not designed to deal with other structures as multi-graphs. Nevertheless, several authors argue that the nature of the phenomenon in some application can be better modeled through multi-graphs [11–13].

Analyzing the aforementioned algorithms, some problems are detected: (1) the reported algorithms for mining FASs from graph collections, which allow semantic distortions and keep the graph topology, were not designed to deal with multi-graphs; (2) the algorithms reported in [4,7], the sub-isomorphism tests are computationally expensive because the occurrences of the candidates in the collection are not stored. These tests are more expensive when approximate matching is used than for exact matching; (3) the algorithms proposed in [5,7], which compute representative patterns, were not designed to deal with semantic distortions between vertex and edge label sets in graph collections; (4) the algorithms introduced in [8–10] were not designed for dealing with graph collections. Thus, in this paper, we focus on the approximate graph mining approach, which allows semantic distortions between vertex and edge label sets, keeping the graph topology, over multi-graph collections.

The organization of this paper is the following. In Sect. 2, some basic concepts and notations are provided. Our proposed method for processing multi-graphs is introduced in Sect. 3. Several experiments are presented in Sect. 4. Finally, our conclusions and future work directions are discussed in Sect. 5.

## 2 Background

In this section, the background and notation needed to understand the following sections, as well as the FAS mining problem are presented. Notice that most of definitions are presented in a way that they allow treating both types of undirected graphs: simple-graphs and multi-graphs.

**Definition 1 (Labeled Graph).** A *labeled graph* with the domain of labels $L = L_V \cup L_E$, where $L_V$ and $L_E$ are the label sets for vertices and edges respectively, is a 5-tuple, $G = (V, E, \phi, I, J)$, where $V$ is a set whose elements are called *vertices*, $E$ is a set whose elements are called *edges*, $\phi : E \to V \times V$ is the *incidence function* (the edge $e$, through the function $\phi(e)$, connects the vertices $u$ and $v$ if $\phi(e) = \{u, v\}$), $I : V \to L_V$ is a *labeling function* for assigning labels to vertices and $J : E \to L_E$ is a *labeling function* for assigning labels to edges.

**Definition 2 (Subgraph and Supergraph).** Let $G_1 = (V_1, E_1, \phi_1, I_1, J_1)$ and $G_2 = (V_2, E_2, \phi_2, I_2, J_2)$ be two graphs, $G_1$ is a *subgraph* of $G_2$ if $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, $\phi_1$ is a restriction of $\phi_2$ to $V_1$, $I_1$ is a restriction of $I_2$ to $V_1$, and $J_1$ is

a restriction of $J_2$ to $E_1$ (a restriction of a function is the result of trimming its domain). In this case, the notation $G_1 \subseteq G_2$ is used, and it is also said that $G_2$ is a *supergraph* of $G_1$.

**Definition 3 (Multi-edge and Loop).** An edge $e \in E$, where $\phi(e) = \{u, v\}$ and $u \neq v$, is a *multi-edge* if there is $e' \in E$ such that $e \neq e'$ and $\phi(e) = \phi(e')$; otherwise, $e$ is a *simple-edge*. If $|\phi(e)| = 1$, $e$ is called a *loop*, i.e. $\phi(e) = \{u\} = \{v\}$.

In this way, the set of edges $E$ of a graph can be partitioned into three disjoint subsets $E^{(s)}$, $E^{(m)}$, and $E^{(l)}$ containing simple-edges, multi-edges, and loops, respectively.

**Definition 4 (Simple-graph and Multi-graph).** A graph is a *simple-graph* if it has no loops and no multi-edges, i.e. it has only simple-edges, $E = E^{(s)}$, being $E^{(m)} = \emptyset$ and $E^{(l)} = \emptyset$; otherwise, it is a *multi-graph*.

**Definition 5 (The Operator $\oplus$).** Let $G_1 = (V_1, E_1, \phi_1, I_1, J_1)$ and $G_2 = (V_2, E_2, \phi_2, I_2, J_2)$ be two graphs, where for each $v \in V_1 \bigcap V_2$ $I_1(v) = I_2(v)$, and for each $e \in E_1 \bigcap E_2$ $\phi_1(e) = \phi_2(e)$ and $J_1(e) = J_2(e)$. In this case, it is said that $G_1$ and $G_2$ are *mutually compatible* graphs. Thus, the *sum* of $G_1$ and $G_2$ is a supergraph of $G_1$ and $G_2$ denoted by $G_1 \oplus G_2 = (V_3, E_3, \phi_3, I_3, J_3)$, where $V_3 = V_1 \bigcup V_2$; $E_3 = E_1 \bigcup E_2$; for each $v \in V_1$ $I_3(v) = I_1(v)$ and for each $v \in V_2$ $I_3(v) = I_2(v)$; for each $e \in E_1$ $\phi_3(e) = \phi_1(e)$ and $J_3(e) = J_1(e)$, and for each $e \in E_2$ $\phi_3(e) = \phi_2(e)$ and $J_3(e) = J_2(e)$. We will use the notation $\bigoplus_i G_i$ for denoting the successive sum of several graphs $G_i$.

**Definition 6 (Isomorphism).** Given two graphs $G_1$ and $G_2$, a pair of functions $(f, g)$ is an *isomorphism* between these graphs if $f : V_1 \to V_2$ and $g : E_1 \to E_2$ are bijective functions, where:

1. $\forall u \in V_1 : f(u) \in V_2$ and $I_1(u) = I_2(f(u))$
2. $\forall e_1 \in E_1$, where $\phi_1(e_1) = \{u, v\}$: $e_2 = g(e_1) \in E_2$, and $\phi_2(e_2) = \{f(u), f(v)\}$ and $J_1(e_1) = J_2(e_2)$.

If there is an isomorphism between $G_1$ and $G_2$, it is said that $G_1$ and $G_2$ are *isomorphic*.

**Definition 7 (Sub-isomorphism).** Given three graphs $G_1$, $G_2$ and $G_3$. If $G_1$ is isomorphic to $G_3$ and $G_3 \subseteq G_2$, then it is said that there is a *sub-isomorphism* between $G_1$ and $G_2$, denoted by $G_1 \subseteq_s G_2$, and it is also said that $G_1$ is *sub-isomorphic* to $G_2$.

**Definition 8 (Similarity).** Let $\Omega$ be the set of all possible labeled graphs in $L$, the *similarity* between two graphs $G_1, G_2 \in \Omega$ is defined as a function $sim : \Omega \times \Omega \to [0, 1]$. The higher the value of $sim(G_1, G_2)$ the more similar the graphs are, and if $sim(G_1, G_2) = 1$ then there is an isomorphism between these graphs.

**Definition 9 (Approximate Isomorphism and Approximate Sub–isomorphism).** Let $G_1$, $G_2$ and $G_3$ be three graphs, let $sim(G_1, G_2)$ be a similarity function among graphs, and let $\tau$ be a similarity threshold, there is an

*approximate isomorphism* between $G_1$ and $G_2$ if $sim(G_1, G_2) \geq \tau$. Also, if there is an approximate isomorphism between $G_1$ and $G_2$, and $G_2 \subseteq G_3$, then there is an *approximate sub-isomorphism* between $G_1$ and $G_3$, denoted as $G_1 \subseteq_A G_3$.

**Definition 10 (Maximum Inclusion Degree).** Let $G_1$ and $G_2$ be two graphs, let $sim(G_1, G_2)$ be a similarity function among graphs, since a graph $G_1$ can be enough similar to several subgraphs of another graph $G_2$, the *maximum inclusion degree* of $G_1$ in $G_2$ is defined as:

$$maxID(G_1, G_2) = \max_{G \subseteq G_2} sim(G_1, G), \tag{1}$$

where $maxID(G_1, G_2)$ means the maximum value of similarity at comparing $G_1$ with all of the subgraphs $G$ of $G_2$.

**Definition 11 (Approximate Support).** Let $D = \{G_1, \ldots, G_{|D|}\}$ be a graph collection, let $sim(G_1, G_2)$ be a similarity function among graphs, let $\tau$ be a similarity threshold, and let $G$ be a graph. Thus, the *approximate support* (denoted by *appSupp*) of $G$ in $D$ is obtained through equation (2):

$$appSupp(G, D) = \frac{\sum_{\{G_i | G_i \in D, G \subseteq_A G_i\}} maxID(G, G_i)}{|D|} \tag{2}$$

Using (2), $G$ is a *FAS* in $D$ if $appSupp(G, D) \geq \delta$, for a given support threshold $\delta$, a similarity function among graphs $sim(G_1, G_2)$ and a similarity threshold $\tau$. The values of $\delta$ and $\tau$ are in $[0, 1]$ because the similarity is defined in $[0, 1]$.

*FAS mining* consists in, given a support threshold $\delta$ and a similarity threshold $\tau$, finding all the FAS in a collection of graphs $D$, using a similarity function $sim$.

## 3 Method Based on Transformation for Processing Multi-graphs

Following the idea used in [11,14] for transforming a multi-graph into a simple-graph[1], we propose a method for transforming a multi-graph collection into a simple-graph collection without losing any topological or semantic information.

The proposed method for transforming a multi-graph into a simple-graph comprises two steps: (1) transforming each loop into a simple-edge, and (2) transforming each edge (simple-edge or multi-edge) into two simple-edges. Then, each loop is changed by adding a new vertex with an special label ($\kappa$) and a simple-edge with the loop label. Later, each edge $e$ which connects two vertices $u \neq v$ is changed by a new vertex with a special label ($\varrho$) and two simple-edges ($e_1$ and $e_2$) both with the label of $e$. The simple-edge $e_1$ connects the new vertex with $u$ and $e_2$ connects the new vertex with $v$. The transformation from a multi-graph into a simple-graph is formally defined in Definition 12.

[1] It is important to highlight that in [11,14] the transformation is different to the one introduced in this paper. Furthermore, these transformations were not proposed for mining frequent patterns in multi-graphs.

**Definition 12 (Multi-graph Simplification).** Let $G = (V, E, \phi, I, J)$ be a connected multi-graph, and let $\kappa$ and $\varrho$ be two different vertex labels that will be used in two kind of vertices for representing all loops and all edges, respectively. The *multi-graph simplification* of $G$ is a graph defined as:

$$G' = \bigoplus_{e \in E} G'_e, \tag{3}$$

where the graph $G'_e$ is defined as follows:

- If $e$ is an edge and $\phi(e) = \{u, v\}$, the graph $G'_e$ is defined as $G'_e = (V_1, E_1, \phi_1, I_1, J_1)$, where $V_1 = \{u, v, w\}$, $E_1 = \{e_1, e_2\}$, $E_1 \cap E = \emptyset$, $\phi_1(e_1) = \{u, w\}$, $\phi_1(e_2) = \{w, v\}$, $I_1$ is a restriction of $I$ to $V_1$ with $I_1(w) = \varrho$, $J_1(e_1) = J_1(e_2) = J(e)$, and $w \notin V$ is a new vertex.
- If $e$ is a loop and $\phi(e) = \{v\}$, the graph $G'_e$ is defined as $G'_e = (V_2, E_2, \phi_2, I_2, J_2)$, where $V_2 = \{v, w\}$, $E_2 = \{e'\}$, $e' \notin E$, $\phi_2(e') = \{v, w\}$, $I_2$ is a restriction of $I$ to $V_2$ with $I_2(w) = \kappa$, $J_2(e') = J(e)$, and $w \notin V$ is a new vertex.

Based on Definition 12, an algorithm for transforming a multi-graph into a simple-graph, called *M2Simple*, can be introduced. First, by traversing the edges in the input multi-graph, for each edge $e$, according to Definition 12, the graph $G'_e$ can be calculated and added to $G'$. Thus, the complexity of this algorithm is $O(k * n^2 * d)$, where $k$ is the largest number of edges between two vertices, $n$ is the number of vertices in the graph with the largest amount of vertices, and $d$ is the number of graphs. Notice that for building simplifications only vertices with label $\kappa$ and $\varrho$ were added, including the simple-edges connecting such vertices.

For doing compatible a simple-graph with the original multi-graph collection, a reversing process is required. Then, there is a transformation from a simple-graph to a multi-graph. In Definition 13, the required condition that a simple-graph must fulfill in order to be transformed to a multi-graph is introduced.

**Definition 13 (Returnable Graph).** Let $\kappa$ and $\varrho$ be the special labels used in the Definition 12. The simple-graph $G' = (V', E', \phi', I', J')$ is *returnable* to a multi-graph if it fulfills the following conditions:

1. Each vertex $v \in V'$ with $I'(v) = \varrho$ has exactly two incident edges $e_1$ and $e_2$, such that $J'(e_1) = J'(e_2)$, and
2. Each vertex $v \in V'$ with $I'(v) = \kappa$ has exactly one incident edge.

The transformation from a simple-graph into a multi-graph is formally defined in the Definition 14.

**Definition 14 (Graph Generalization).** Let $G' = (V', E', \phi', I', J')$ be a returnable connected graph and let $\kappa$ and $\varrho$ be the special labels used in the Definitions 12 and 13. Let $V'_\varrho$ be the set of all of $v \in V'$ such that $I'(v) = \varrho$, and let $V'_\kappa$ be the set of all of $v \in V'$ such that $I'(v) = \kappa$. Thus, the *generalization* of $G'$ is a graph defined as:

$$G = \bigoplus_{w \in V'_\varrho \cup V'_\kappa} G_w, \tag{4}$$

where the graph $G_w$ is defined as follows:

- If $I'(w) = \varrho$, by the first returnable condition there are exactly two incidents edges $e_1$ and $e_2$, such that $\phi'(e_1) = \{u, w\}$ and $\phi'(e_2) = \{w, v\}$, then $G_w$ is defined as $G_w = (V_1, E_1, \phi_1, I_1, J_1)$, where $V_1 = \{u, v\}$, $E_1 = \{e\}$, $e \notin E'$, $\phi_1(e) = \{u, v\}$, $I_1$ is a restriction of $I'$ to $V_1$, and $J_1(e) = J'(e_1) = J'(e_2)$.
- If $I'(w) = \kappa$, by the second returnable condition there is exactly one incident edge $e_2$, such that $\phi'(e_2) = \{v, w\}$, then $G_w$ is defined as $G_w = (V_2, E_2, \phi_2, I_2, J_2)$, where $V_2 = \{v\}$, $E_2 = \{e\}$, $e \notin E'$, $\phi_2(e) = \{v\}$, $I_2$ is a restriction of $I'$ to $V_2$, and $J_2(e) = J'(e_2)$.

An algorithm for transforming a returnable simple-graph into a multi-graph, called *S2Multi* can be introduced. Based on Definition 14, this algorithm comprises two steps: first, the algorithm traverses the vertices with label $\varrho$ for adding edges to $G$. Then, the algorithm traverses the vertices with label $\kappa$ for adding loops to $G$. Taking into account the way in which $G$ is built, through the proposed simplification process (see Definition 12), the complexity of this algorithm is $O((n + k * n^2) * d)$, where $n$, $k$ and $d$ are the same as used for complexity of the M2Simple algorithm. Notice that, for building generalizations, only vertices with label $\kappa$ and $\varrho$ were removed, including the simple-edges connecting such vertices.

The correctness of our method, which is based on transformations, is proved trough the following theorems. The proofs of these theorems were omitted due to space limitations.

**Theorem 1.** *Let $G_1 = (V_1, E_1, \phi_1, I_1, J_1)$ and $G_2 = (V_2, E_2, \phi_2, I_2, J_2)$ be two isomorphic connected (returnable) graphs. Then, the graphs, $G'_1 = (V'_1, E'_1, \phi'_1, I'_1, J'_1)$ and $G'_2 = (V'_2, E'_2, \phi'_2, I'_2, J'_2)$, obtained from the simplifications (generalizations) of $G_1$ and $G_2$ are also isomorphic.*

**Theorem 2.** *Let $G = (V, E, \phi, I, J)$ be the simplification (generalization) of the connected graph $G_1 = (V_1, E_1, \phi_1, I_1, J_1)$, and let $\kappa$ and $\varrho$ be the special labels; then $G_1$ is isomorphic to the generalization (simplification) of $G$.*

**Theorem 3.** *Let $G'_1 = (V'_1, E'_1, \phi'_1, I'_1, J'_1)$ and $G'_2 = (V'_2, E'_2, \phi'_2, I'_2, J'_2)$ be two multi-graphs, and let $G_1 = (V_1, E_1, \phi_1, I_1, J_1)$ and $G_2 = (V_2, E_2, \phi_2, I_2, J_2)$ be two simple-graphs, such that $G_1$ and $G_2$ are simplifications of $G'_1$ and $G'_2$, respectively; then $G'_2$ is sub-isomorphic to $G'_1 \Leftrightarrow G_2$ is sub-isomorphic to $G_1$.*

**Corollary 1.** *Let $D = \{G_1, \ldots, G_N\}$ be a collection of $N$ simple-graphs, let $D' = \{G'_1, \ldots, G'_N\}$ be a collection of $N$ multi-graphs, where $G_i$ is the simplification of $G'_i$, for each $1 \leq i \leq N$. Let $G$ be the simplification of a multi-graph $G'$. Thence, the support of $G$ in $D$ is the same that the support of $G'$ in $D'$.*

**Corollary 2.** *Let $D$ and $D'$ be the collections of simple-graphs and multi-graphs, respectively used in Corollary 1. Let $G$ be the simplification of a multi-graph $G'$. Thence, if $G$ is FAS in $D \Leftrightarrow G'$ is FAS in $D'$.*

Finally, it is important to highlight that the transformation process of multi-graph collections into simple-graph collections allows us applying any algorithm for traditional FAS mining.

## 4 Experiments

In this section, with the aim of studying the performance of the proposed method as well as its usefulness for image classification tasks, two experiments are performed. All our experiments were carried out using a personal computer with an Intel(R) Core(TM) i7 with 64 GB of RAM. The M2Simple and S2Multi algorithms was implemented in python and executed on GNU/Linux (UBUNTU).

In our first experiment, the performance of our transformation algorithms (M2Simple and S2Multi) is evaluated. For this evaluation, three kinds of synthetic multi-graph collections were used. These synthetic collections were generated using the PyGen[2] graph emulation library. For building these collections, first, we fix $|D| = 5000$ and $|E| = 800$, varying $|V|$ from 100 to 500, with increments of 100 (see Table 1(a)). Next, we fix $|V| = 500$ and $|D| = 5000$ varying $|E|$ from 600 to 1000, with increments of 100 (see Table 1(b)). Finally, we vary $|D|$ from 10000 to 50000, with increments of 10000, keeping $|V| = 500$ and $|E| = 800$ (see Table 1(c)). Notice that, we assign a descriptive name for each synthetic collection, for example, $D5kV600E2k$ means that the collection has $|D| = 5000$, $|V| = 600$ and $|E| = 2000$.

In Table 1, the performance results, in terms of runtime, of both proposed transformation algorithms (M2Simple and S2Multi) is shown. These results were achieved by transforming each multi-graph collection into a simple-graph collection using M2Simple. Then, each obtained simple-graph collection was transformed into a multi-graph using S2Multi. In this table, the first column shows the collection and the other two columns show the runtime in seconds of both proposed transformation algorithms.

According to the complexity of the proposed transformation processes presented in Sect. 3, remarked by the results shown in Table 1, we can conclude that our transformation algorithms are more sensitive to the number of edge variations

**Table 1.** Runtime results in seconds of our transformation algorithms over several synthetic collections.

(a) Varying $|V|$ from 1000 to 5000 with $|D| = 5000$ and $|E| = 1000$.

| Collection | M2Simple | S2Multi |
|---|---|---|
| *D5kV100E800* | 16.77 | 22.34 |
| *D5kV200E800* | 18.00 | 23.83 |
| *D5kV300E800* | 18.60 | 25.21 |
| *D5kV400E800* | 18.93 | 25.63 |
| *D5kV500E800* | 20.06 | 27.27 |

(b) Varying $|E|$ from 1000 to 5000 with $|D| = 5000$ and $|V| = 1000$.

| Collection | M2Simple | S2Multi |
|---|---|---|
| *D5kV500E600* | 15.95 | 21.18 |
| *D5kV500E700* | 17.97 | 24.64 |
| *D5kV500E800* | 20.06 | 27.27 |
| *D5kV500E900* | 22.54 | 31.27 |
| *D5kV500E1k* | 24.88 | 32.74 |

(c) Varying $|D|$ from 10000 to 50000 with $|E| = 1000$ and $|V| = 1000$.

| Collection | M2Simple | S2Multi |
|---|---|---|
| *D10kV500E800* | 41.28 | 55.64 |
| *D20kV500E800* | 81.45 | 109.33 |
| *D30kV500E800* | 122.47 | 163.48 |
| *D40kV500E800* | 157.54 | 252.94 |
| *D50kV500E800* | 199.83 | 268.52 |

[2] PyGen is available in http://pywebgraph.sourceforge.net.

than to the number of vertex variations in this kind of experiments. This is because the required runtime grows faster when the amount of edges increases than increasing the number of vertices. Furthermore, M2Simple receives many more vertices and edges than S2Multi for the same multi-graph collection, since S2Multi creates an additional vertex and an additional edge for each transformed edge or loop. On the other hand, the number of graph of the collection is an important variable to take into account, because it affects the performance of both proposed algorithms when it has high values. Finally, the most important fact to take into account is that the transformations are performed in a runtime less than 5 min over collections with high dimensions.

In addition, three graph collections representing images generated with the Random image generator of Coenen[3] are used. For obtaining the Coenen image collections, we randomly generate 1000 images with two classes. These images were randomly divided into two sub-sets: one for training with 700 (70 %) images and another for testing with 300 (30 %) images. The first two collections, denoted by *C-Angle* and *C-Distance*, are represented as simple-graph collections, using a quad-tree approach [15] and the angles and the distances between regions as edge labels, respectively. These two collections have 21 vertex labels, 24 edge labels and 135 as the average graph size. The last collection, denoted by *C-Multi*, was also built using a quad-tree approach, but it uses both (angles and distances) values as labels for two multi-edges, respectively. This collection comprises 21 vertex labels, 48 edge labels and the average graph size is 270, where all edges are multi-edges. These collections are used for assessing the performance of our method based on graph transformation and also for showing the usefulness of the identified patterns on a multi-graph collection. The algorithm for FAS mining used in our method is VEAM [3].

In order to show the usefulness of the patterns computed in multi-graphs through our proposed method, in Table 2 we show the classification results by using the patterns computed after applying our method in the multi-graph collections. For these experiments, the classifier J48graft taken from Weka v3.6.6 [16] using the default parameters, was used. This table shows the results of the accuracy and F-measure achieved over the three graph collections representing the image collection. The first column shows the used support threshold values. The other three consecutive columns show the accuracy results using the collection specified in the top of these columns, and in the other three consecutive columns the F-measure results are shown. The results achieved using VEAM over C-Angle and C-Distance collections represent the solution to the problem if our proposal were not available since for applying VEAM the images would be represented as simple-graphs using only one edge between vertices as in C-Angle or C-Distance.

The classification results achieved (see Table 2) over the image collection show the usefulness of the patterns computed by our proposal, where in most of these cases, the best classification results are obtained by using our method. Furthermore, the representation of the images as multi-graphs allows us to obtain better classification results than using simple-graphs.

[3] www.csc.liv.ac.uk/~frans/KDD/Software/ImageGenerator/imageGenerator.html.

**Table 2.** Classification results (%) achieved over the different graph collection representing the image collection using the J48graft classifier with similarity threshold $\tau = 0.4$ and different support threshold values.

| Support | Accuracy | | | F-measure | | |
|---|---|---|---|---|---|---|
| | C-Angle | C-Distance | C-Multi | C-Angle | C-Distance | C-Multi |
| 0.8 | 78.33 | 78.33 | 78.33 | 68.90 | 68.90 | 68.90 |
| 0.7 | 77.67 | 76.33 | **78.67** | 74.13 | 72.37 | **77.46** |
| 0.6 | 79.67 | 91.33 | **92.33** | 77.15 | 90.85 | **91.76** |
| 0.5 | 89.00 | **93.33** | **93.33** | 88.17 | 92.59 | **93.00** |
| 0.4 | 89.67 | 91.27 | **91.65** | 89.27 | 90.13 | **91.59** |
| *Average* | *82.87* | *86.12* | ***86.86*** | *79.52* | *82.97* | ***84.54*** |

## 5 Conclusions

In this paper, a new method based on graph transformation for FAS mining in multi-graph collections is proposed. In our method, we first transform a multi-graph collection into a simple-graph collection, then the mining is performed over this collection using a traditional FAS miner and later, an inverse transformation allow us returning the mining results (simple-graphs) to multi-graphs. This process can be performed without losing any topological or semantic information. The performance of the proposed transformation algorithms (M2Simple and S2Multi) is evaluated over different synthetic multi-graph collections. Additionally, the usefulness of the patterns identified after applying our method is also shown.

As future work, we plan to develop a FAS miner allowing us to compute FASs directly from multi-graph collections, without the transformation steps.

**Acknowledgment.** This work was partly supported by the National Council of Science and Technology of Mexico (CONACyT) through the project grant *CB*2008-106366; and the scholarship grant 287045.

## References

1. Jiang, C., Coenen, F., Zito, M.: A survey of frequent subgraph mining algorithms. Knowl. Eng. Rev. **28**(1), 75–105 (2012)
2. Holder, L., Cook, D., Bunke, H.: Fuzzy substructure discovery. In: ML92: Proceedings of the Ninth International Workshop on Machine Learning, pp. 218–223. Morgan Kaufmann Publishers Inc., San Francisco (1992)
3. Acosta-Mendoza, N., Gago-Alonso, A., Medina-Pagola, J.: Frequent approximate subgraphs as features for graph-based image classification. Knowl.-Based Syst. **27**, 381–392 (2012)
4. Li, J., Zou, Z., Gao, H.: Mining frequent subgraphs over uncertain graph databases under probabilistic semantics. VLDB J. **21**(6), 753–777 (2012)
5. Jia, Y., Zhang, J., Huan, J.: An efficient graph-mining method for complicated and noisy data with real-world applications. Knowl. Inf. Syst. **28**(2), 423–447 (2011)

6. Song, Y., Chen, S.: Item sets based graph mining algorithm and application in genetic regulatory networks. In: IEEE International Conference on Data Mining, pp. 337–340 (2006)
7. Zou, Z., Li, J., Gao, H., Zhang, S.: Finding top-k maximal cliques in an uncertain graph. In: IEEE 26th International Conference on Data Engineering (ICDE 2010), pp. 649–652 (2010)
8. Chen, C., Yan, X., Zhu, F., Han, J.: gApprox: mining frequent approximate patterns from a massive network. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), pp. 445–450 (2007)
9. Flores-Garrido, M., Carrasco-Ochoa, J., Martínez-Trinidad, J.: AGraP: an algorithm for mining frequent patterns in a single graph using inexact matching. Knowl. Inf. Syst., pp. 1–22 (2014)
10. Flores-Garrido, M., Carrasco-Ochoa, J., Martínez-Trinidad, J.: Mining maximal frequent patterns in a single graph using inexact matching. Knowl.-Based Syst. **66**, 166–177 (2014)
11. Whalen, J.S., Kenney, J.: Finding maximal link disjoint paths in a multigraph. In: IEEE Global Telecommunications Conference and Exhibition. 'Communications: Connecting the Future', GLOBECOM 1990, pp. 470–474. IEEE (1990)
12. Björnsson, Y., Halldórsson, K.: Improved heuristics for optimal pathfinding on game maps. In: American Association for Artificial Intelligence (AIIDE). pp. 9–14 (2006)
13. Morales-González, A., García-Reyes, E.B.: Simple object recognition based on spatial relations and visual features represented using irregular pyramids. Multimedia Tools Appl. **63**(3), 875–897 (2013)
14. Boneva, I., Hermann, F., Kastenberg, H., Rensink, A.: Simulating multigraph transformations using simple graphs. In: Proceedings of the Sixth International Workshop on Graph Transformation and Visual Modeling Techniques, Braga, Portugal. Electronic Communications of the EASST, vol. 6, EASST (2007)
15. Finkel, R., Bentley, J.: Quad trees: a data structure for retrieval on composite keys. Acta Informatica **4**, 1–9 (1974)
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: an update. ACM SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)