Revista Cubana de Ciencias Informáticas
Vol. 12, No. 3, Julio-Septiembre, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-16
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Reconocimiento de patrones
Recibido: 24/11/2017 | Aceptado: 06/06/2018

A New Multi-graph Transformation Method for Frequent Approximate Subgraph Mining

Un nuevo método basado en transformaciones de multigrafos para la minería de subgrafos frecuentes aproximados

Niusvel Acosta Mendoza

Centro de Aplicaciones de Tecnología de Avanzada (CENATAV). nacosta@cenatav.co.cu

* **Autor para correspondencia**: nacosta@cenatav.co.cu

Abstract

Frequent approximate subgraph (FAS) mining has been successfully applied in several science domains, because in many applications, approximate approaches have achieved better results than exact approaches. However, there are real applications based on multi-graphs where traditional FAS miners cannot be applied because they were not designed to deal with this type of graph. Only one method based on graph transformation, which allows the use of traditional simple-graph FAS miners on multi-graph problems was reported, but it has high computational cost. This paper aims at accelerating the mining process, thus a more efficient method is proposed for transforming multi-graphs into simple graphs and vice versa without losing topological or semantic information, that allows using traditional FAS mining algorithms and returning the mined patterns to the multi-graph space. Finally, we analyze the performance of the proposed method over synthetic multi-graph collections and additionally we show the effectiveness of the proposal in image classification tasks where images are represented as multi-graphs.

Keywords: approximate mining, frequent approximate subgraphs, graph-based classification, multi-graph mining.

Resumen

La minería de subgrafos frecuentes aproximados ha sido satisfactoriamente aplicada en varios dominios de la ciencia, debido a que los enfoques aproximados han alcanzado mejores resultados que los exactos en muchas aplicaciones. Sin embargo, existen aplicaciones basadas en multi-grafos donde los algoritmos tradicionales de minería no pueden ser aplicados porque no están diseñados para trabajar con este tipo de grafos. Solo se ha reportado un método basado en

transformaciones de grafos que permite aplicar los algoritmos tradicionales para la minería de subgrafos frecuentes aproximados en problemas representados como multi-grafos, pero tiene la limitante de un alto costo computacional. En este trabajo, con el objetivo de acelerar el proceso de minería, se propone un método más eficiente para transformar los multi-grafos en grafos simples y vice versa. Este proceso se realiza sin perder información topológica o semántica, lo cual permite el uso de los algoritmos tradicionales de minería de grafos y los patrones minados se pueden retornar al contexto de multi-grafos. Finalmente se analiza el comportamiento del método propuesto sobre colecciones de multi-grafos sintéticas y adicionalmente se muestra la utilidad de la propuesta en tareas de clasificación de imágenes, donde dichas imágenes son representadas como multi-grafos.

Palabras claves: *clasificación basada en grafos, minería aproximada, minería de multi-grafos, subgrafos frecuentes aproximados.*

Introduction

Frequent approximate subgraph (FAS) mining has become an outstanding technique in data mining with several applications such as: genetic networks and biochemical structures analysis, image classification, and circuits, cites, social networks and links analysis, among others (Flores-Garrido et al., 2015; Jia et al., 2011; Morales-González et al., 2014). In this research, FAS mining algorithms achieve better results than the ones reported by exact frequent subgraph mining algorithms. This is because the inexact matching between patterns is common in the data of a real-life application (Cook and Holder, 1994; González et al., 2001; Ketkar, 2005). However, the exact mining algorithms compute frequent patterns based on isomorphism (Yan and Huan, 2002; Zhu et al., 2007; Wang et al., 2016).

All the aforementioned algorithms process only simple graph collections, where a simple graph is a graph with a single edge between a pair of vertices and without edges connecting a vertex with itself (loops). However, there are some applications such as pathfinder on game maps, RNA molecule analysis, dynamic network with time information, image processing, and event detection from Web sites, among others (Boneva et al., 2007; Björnsson and Halldórsson, 2006; Cazabet et al., 2015; Morales-González and García-Reyes, 2013; Terroso-Saez et al., 2015; Youssef et al., 2015) in which the authors highlight that using multi-graphs allow them modeling data in a better way than using simple graphs. A multi-graph is a graph that may contain loops and multiple edges between a pair of vertices. In these applications, traditional FAS miners cannot be applied because they have not been designed to work on multi-graphs. In all of these

applications, using multi-graphs and finding interesting patterns from multi-graphs would allow to get information potentially useful to solve problems that are more complex.

As mentioned before, several researchers have focused their efforts on developing algorithms for mining FASs in simple graph collections and, and it has only been found one work which reports a solution, based on graph transformations, for using this FAS miners on multi-graph collections (Acosta-Mendoza et al., 2015). However, this method highly increases the size of each graph in the collection and therefore the runtime of the FAS mining process. For this reason, with the aim of speeding up the mining process, an alternative method is proposed, based on graph transformation, for mining a subset of FASs from a multi-graph collection. The proposal of this paper guarantees returning the mined FASs to the multi-graph space faster than the method reported in the state-of-the-art.

Computational Methodology

As we focus on working over a collection of undirected labeled multi-graphs, the first concepts to be defined are labeled graph, simple graph and multi-graph. It is important to highlight that several of the concepts presented in this section were obtained from (Acosta-Mendoza et al., 2012; Morales-González et al., 2014).

Definition 1 (Labeled graph): *Let L_V and L_E be two label sets for vertices and edges, respectively, a labeled graph G is a 5-tuple $(V_G, E_G, \varphi_G, I_G, J_G)$ where: V_G is a set of vertices; E_G is a set of edges; $\varphi_G : E_G \rightarrow V_G^{\bullet}$ is a function that returns the pair of vertices of V_G which are connected by a given edge, where $V_G^{\bullet} = \{\{u, v\} | u, v \in V_G\}$; $I_G : V_G \rightarrow L_V$ is a labeling function for assigning labels to vertices in V_G; and $J_G : E_G \rightarrow L_E$ is a labeling function for assigning labels to edges in E_G.*

Multi-edges are different edges connecting the same pair of vertices (i.e. e and $e'$are multi-edges if $e \neq e'$ and $\varphi_G(e) = \varphi_G(e') = \{u, v\}$ such that $u, v \in V_G, u \neq v$) (Acosta-Mendoza et al., 2015). A loop is an edge connecting a vertex to itself (i.e., when $\varphi_G(e) = \{u\}$ since $\varphi_G(e) = \{u, v\}$ with $v = u$; in a loop $|\varphi_G(e)| = 1$) (Acosta-Mendoza et al., 2015). Then, the concepts of simple graph and multi-graph are defined as follows:

Definition 2 (Simple-graph and multi-graph (Acosta-Mendoza et al., 2015)): *A graph G is a* simple graph *if it has no loops and no multi-edges; otherwise, G is a* multi-graph.

Definition 3 (subgraph and supergraph): *Given two graphs $G_1 = (V_{G_1}, E_{G_1}, \varphi_{G_1}, I_{G_1}, J_{G_1})$ and $G_2 = (V_{G_2}, E_{G_2}, \varphi_{G_2}, I_{G_2}, J_{G_2})$*, G_1 is a subgraph of G_2 if $V_{G_1} \subseteq V_{G_2}, E_{G_1} \subseteq E_{G_2}, \forall u \in V_{G_1}, I_{G_1}(u) = I_{G_2}(u), \forall e \in E_{G_1}, J_{G_1}(e) =$

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 3, Julio-Septiembre, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-16
http://rcci.uci.cu

$J_{G_2}(e)$, *and* $\forall e \in E_{G_1}, \varphi_{G_1}(e) = \varphi_{G_2}(e)$. *In this case, we use the notation* $G_1 \subseteq G_2$ *and we say that* G_2 *is a supergraph of* G_1.

In exact graph mining, graph matching is performed by means of graph isomorphism. For both, simple graphs and multi-graphs, isomorphism and sub-isomorphism between two graphs are defined as follow:

Definition 4 (Isomorphism and sub-isomorphism): Given two graphs $G_1 = (V_{G_1}, E_{G_1}, \varphi_{G_1}, I_{G_1}, J_{G_1})$ and $G_2 = (V_{G_2}, E_{G_2}, \varphi_{G_2}, I_{G_2}, J_{G_2})$, the pair of functions (f, g) is an isomorphism between these graphs *iff* $f: V_{G_1} \to V_{G_2}$ and $g: E_{G_1} \to E_{G_2}$ are bijective functions, such that: $\forall u \in V_{G_1}: f(u) \in V_{G_2}$ *and* $I_{G_1}(u) = I_{G_2}(f(u)); \forall e_1 \in E_{G_1}$, where $\varphi_{G_1}(e_1) = \{u, v\}: e_2 = g(e_1) \in E_{G_2}$, and $\varphi_{G_2}(e_2) = \{f(u), f(v)\}$ *and* $J_{G_1}(e_1) = J_{G_2}(e_2)$; and $\forall e_1 \in E_{G_1}$, where $\varphi_{G_1}(e_1) = \{v\}: e_2 = g(e_1) \in E_{G_2}$, and $\varphi_{G_2}(e_2) = \{f(v)\}$ and $J_{G_1}(e_1) = J_{G_2}(e_2)$. If there is an isomorphism between G_1 and G_2, then we say that G_1 and G_2 are isomorphic. Besides, if G_1 is isomorphic to a subgraph of G_2, then there is a sub-isomorphism between G_1 and G_2; in this case, we say that G_1 and G_2 are sub-isomorphic.

In almost all inexact-based graph mining approaches, the authors firstly define a function for comparing graphs, according to the application context (Cook and Holder, 1994; Jia et al., 2011; Acosta-Mendoza et al., 2012; Flores-Garrido et al., 2015). This function is known as similarity function between two graphs, denoted by $sim(G_1, G_2)$. Later, using a specific $sim(G_1, G_2)$ function, the approximate sub-isomorphism between two graphs and the maximum inclusion degree for a graph G_1 in another G_2 are defined (see the definitions 5 and 6).

Definition 5 (Approximate isomorphism and approximate sub-isomorphism): Let G_1, G_2 and G_3 be three labeled multi-graphs, let $\mathrm{sim}(G_1, G_2)$ be a similarity function, and let $\tau \in [0,1]$ 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 is a subgraph of G_3, then there is an approximate sub-isomorphism between G_1 and G_3, denoted as $G_1 \subseteq_A G_3$.

Between two multi-graphs, more than one approximate similarity with different values can be computed. Thus, in order to have only one similarity value between two graphs, the following definition is used.

Definition 6 (Maximum inclusion degree): Let G_1 and G_2 be two labeled multi-graphs, let $\mathrm{sim}(G_1, G_2)$ be a similarity function; the maximum inclusion degree of G_1 in G_2 is defined as:

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

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

Grupo Editorial "Ediciones Futuro"
Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 3, Julio-Septiembre, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-16
http://rcci.uci.cu

With Definition 7, it is possible to compute the approximate support of a subgraph in a graph collection.

Definition 7 (Approximate support): Let $D = \{G_1, \ldots, G_{|D|}\}$ be a multi-graph collection, let $sim(G_1, G_2)$ be a similarity function among graphs, let τ be a similarity threshold, and let G be a labeled multi-graph. Thus, the approximate support (denoted by appSupp) of G in D is obtained through Equation (2):

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

By using the equation (2), frequent approximate subgraphs can be defined as follows.

Definition 8 (Frequent approximate subgraph (FAS)): Let D be a multi-graph collection, let G be a multi-graph and let δ be a support threshold, G is a frequent approximate subgraph in D iff $appSupp(G, D) \geq \delta$.

Taking into account the FAS definition, frequent approximate subgraph mining in a multi-graph collection consists in, given a support threshold, a similarity function between multi-graphs, and a similarity threshold, computing all the FASs in the multi-graph collection.

Related work

There are three methods reported in the literature where multi-graphs are transformed into simple graphs, the simple graphs are analyzed and a subset of them are returned as result to the context of multi-graphs (Acosta-Mendoza et al., 2015; Boneva et al., 2007; Whalen and Kenney, 1990). The transformation method introduced in (Boneva et al., 2007) is applied for solving a problem in production systems. In (Whalen and Kenney, 1990) a transformation method for finding maximal link-disjoint paths in a multi-graph is proposed. In (Acosta-Mendoza et al., 2015), a method that allows applying FAS miner was introduced and applied on image classification tasks.

All the aforementioned methods use the same basic trick of modifying edges (i.e. replacing edges by a vertex with two incident edges to the end vertices of the original edge). This transformation process is applied over all the edges of the multi-graphs and in this way, a multi-graph G' is transformed into a simple graph G.

The transformation approaches reported in (Boneva et al., 2007; Whalen and Kenney, 1990) have some drawbacks that make them infeasible in the context of FAS mining. In (Whalen and Kenney, 1990), the method does not transform graphs with loops; however, loops could be important in some applications and they should be preserved and treated in

a special way for FAS mining in multi-graphs. Furthermore, in (Whalen and Kenney, 1990), the authors do not provide a reverse transformation from directed simple graphs to directed multi-graphs. This reverse process is trivial when the transformation is applied on a directed multi-graph, where every vertex should be connected with at least two vertices. Nevertheless, other kind of multi-graphs do not have a deterministic reverse transformation, and this kind of multi-graphs are also very common in FAS mining applications. On the other hand, the method proposed in (Boneva et al., 2007) maintains multi-edges after transforming a multi-graph with loops. Therefore, the application of a traditional FAS miner over the transformed graphs is infeasible.

The method (allEdges) proposed in (Acosta-Mendoza et al., 2015), for allowing the application of traditional pattern miners over multi-graph collections, transforms multi-graphs into simple graphs. First, the multi-graph collection is transformed into a simple graph collection. For doing that, each loop that connects a vertex v by a new vertex w and a simple edge (an edge $e_1 \in E_G$ of a graph $G = (V_G, E_G, \varphi_G, I_G, J_G)$ is a simple edge if $\forall e_2 \in E_G, \varphi_G(e_1) \neq \varphi_G(e_2)$) with the label of the loop, connecting v to w. Later, each non-loop edge (i.e. simple edges or multi-edges) e that connects a pair of vertices (u, v where $u \neq v$) is transformed into a new vertex w' and two edges (e_1 and e_2) both with the label of e, connecting u and v, respectively, to w'.

Once the multi-graph collection is transformed into a simple graph collection, a traditional pattern miner is applied on the simple graph collection, and then, the patterns identified by the pattern miner are transformed into multi-graphs. Through some special labels, it is possible to perform the reverse process without losing structural or semantic information of the multi-graph collection. In allEdges, the simple edges and the multi-edges are transformed because the authors consider that a simple edge must have occurrences in the multi-edges and vice versa. However, during this transformation process, several vertices and edges are added. A new vertex for each edge is added and the number of edges is duplicated, increasing the size of each graph, and therefore, the cost of FAS mining.

Both proposals reported in (Boneva et al., 2007; Whalen and Kenney, 1990) are focused on directed multi-graphs. The strategies followed by these methods require the vertex and edge label sets to be disjoint. Thus, traditional FAS miners cannot be used if these transformation methods are applied. On the other hand, the method proposed in (Acosta-Mendoza et al., 2015), although it allows to apply traditional FAS miners, it builds simple graphs with the double of vertices and edges than those in the multi-graph collections, which increases the cost of FAS mining. Therefore, in this paper, we present a new reversible method for transforming an undirected multi-graph collection into an undirected simple graph collection considering loops. Finally, complex simple graph collections are obtained when the method

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 3, Julio-Septiembre, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-16
http://rcci.uci.cu

proposed in (Acosta-Mendoza et al., 2015) is applied, because the number of vertices and edges are duplicated in the transformation process. In this way, the performance of the miners is negatively affected.

Proposed method

In this section, we propose a solution (called *onlyMulti*) for mining a FAS subset from multi-graph collections taking advantage of the FAS miners reported in the literature. The solution proposed in this section, as we illustrate in Figure 1, consists in transforming a multi-graph collection into a simple graph collection, mining a FAS subset from the simple graph collection by applying a FAS miner, and transforming the FASs into multi-graphs.

The idea illustrated in Figure 1 has also been followed by the method (allEdges) reported in (Acosta-Mendoza et al., 2015), but for mining all FASs from multi-graph collections, while onlyMulti is an alternative for mining a reduced number of FASs.

The proposed alternative for transforming multi-graph collections into simple graph collections consists in only transforming loops and multi-edges while simple edges are kept without changes. In this way, less edges and vertices are added during the transformation process, and the FAS miner is applied over simple graph collections smaller graphs than those obtained by the allEdges method proposed in (Acosta-Mendoza et al., 2015). After the FAS miner is applied, the mined FASs are returned to the multi-graphs through the same reversing process used in allEdges. Thus, the process for transforming a multi-graph into a simple graph of allEdges and onlyMulti are different.

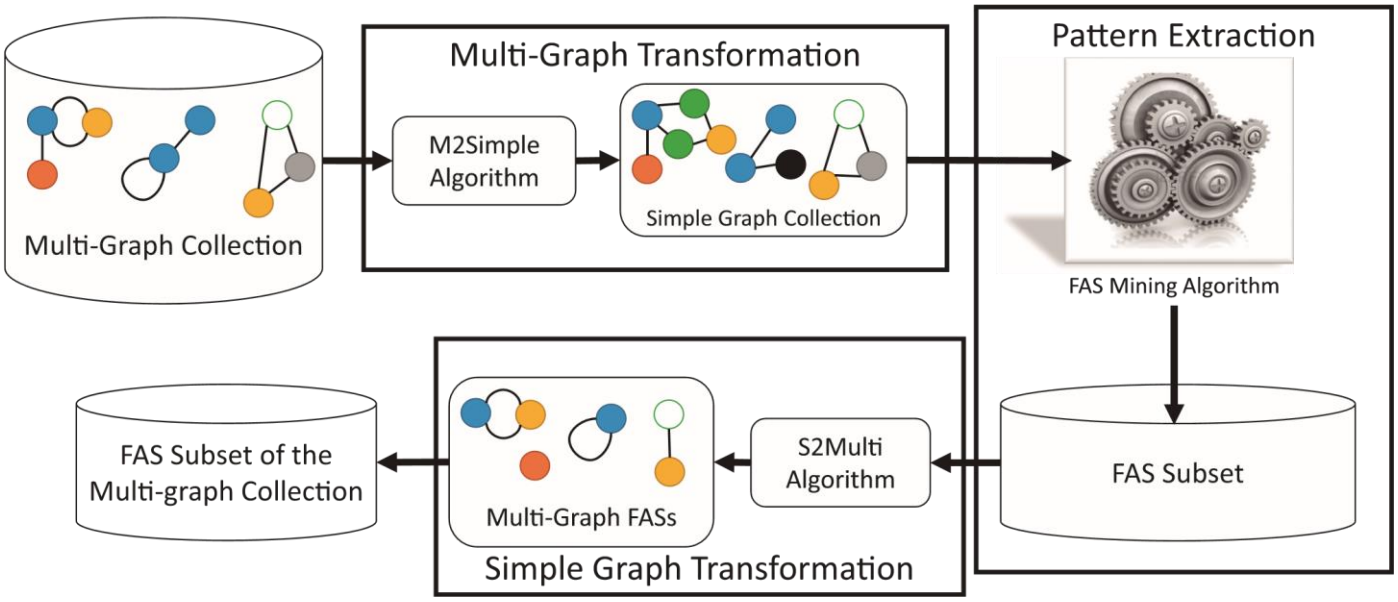

Figure 1. Workflow for FAS mining by applying the proposed graph transformation method.

Grupo Editorial "Ediciones Futuro"
Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 3, Julio-Septiembre, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-16
http://rcci.uci.cu

Following the proposed alternative, the process for transforming a multi-graph G' into a simple graph G consists in replacing each loop and each multi-edge by new vertices and simple edges likewise in allEdges; however, unlike in allEdges, the simple edges are kept without changes. In this way, a simple edge does not have occurrences in multi-edges and vice versa and this is an important characteristic of the solution proposed to be taken into account when it is applied. Each loop, connecting a vertex v of G', is replaced by a simple edge with the label of the loop; connecting v to a new vertex with a special label (k). Later, each multi-edge e in G', with $\varphi_{G'}(e) = \{u, v\}$ and $u \neq v$, is replaced by two simple edges (e_1 and e_2) both with the label of e; connecting u and v, respectively, to a new vertex with a special label (p). This process is shown in Figure 2 where each loop in G' is transformed into a new vertex and a simple edge in G, and each multi-edge in G' is transformed into a new vertex and two simple edges in G, obtaining the simple graph G from the multi-graph G'. The special label p, in the same way as k, cannot be used as label in the multi-graph collection and during the mining process, any other label, except by itself, cannot replace it. In this way, a non-loop edge will only match with other non-loop edge with the same label as well as a multi-edge will only match with other multi-edge with the same label.

Once discussed how a loop and a multi-edge is transformed into simple edges, we can introduce the algorithm (*M2Simple*) for transforming a multi-graph into a simple graph. This algorithm traverses the edges in the input multi-graph searching the loops and multi-edges. The identified loops and multi-edges are replaced by simple edges following the ideas above discussed. Applying this transformation process over each graph in a given multi-graph collection, we can transform it into a simple graph collection. The computational complexity of this process is *O*(*qd*), where *q* is the average number of edges in the multi-graphs of the collection, and *d* is the number of multi-graphs in the collection. This complexity is obtained considering that, for each multi-graph, all its edges should be visited.

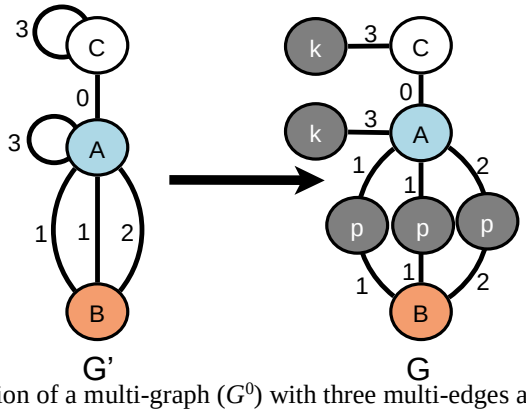

Figure 2. Example of the transformation of a multi-graph (G^0) with three multi-edges and two loops into a simple graph (G).

Grupo Editorial "Ediciones Futuro"
Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

Given a multi-graph collection, through the process above described, a transformed simple graph collection is obtained. Then, a conventional FAS miner can be applied, and the same process introduced in (AcostaMendoza et al., 2015) can be used for transforming the returnable FASs into multi-graphs. Notice that, for obtaining the FASs from the multi-graph collection, this reverse transformation process is required.

For transforming a FAS G (a simple graph) into a multi-graph G', each edge $e \in E_G$ with $\varphi_G(e) = \{u, v\}$ that has a vertex v with label k is transformed into a loop $\varphi_{G'}(e') = \{u\}$ keeping the label of e. Each pair of edges e_1 and e_2 with $\varphi_G(e_1) = \{u, w\}$ and $\varphi_G(e_2) = \{v, w\}$ that have a common vertex w with label p are replaced by an edge e' with $\varphi_{G'}(e') = \{u, v\}$ keeping the label of e_1 and e_2, which have the same label.

Following the aforementioned idea, by traversing the edges of a FAS G and replacing those edges that contain vertices with label p or k by multi-edges or loops, respectively, we can transform a simple graph into a multigraph. Notice that only vertices with label k or p are removed from the simple graph, together with the simple edges connecting those vertices. However, as discussed in (Acosta-Mendoza et al., 2015), not all the mined FASs should be transformed into multi-graphs because some of them do not represent subgraphs in the original multi-graphs. Then, with the aim of identifying the FASs from the original multi-graph collection, some conditions that the mined simple graph FASs must fulfill for being susceptible to be transformed into a multi-graph (i.e. to be a returnable FAS) were introduced in (Acosta-Mendoza et al., 2015). In Definition 9, the aforementioned conditions are presented.

Definition 9 (Returnable graph) Let k and p be the special labels used for representing loops and multi-edges, respectively. A simple graph G is returnable to a multi-graph if it fulfills the following conditions:

1. *Each vertex* $v \in V_G$ *with* $I_G(v) = p$ *has exactly two incident edges* e_1 *and* e_2*, such that* $J_G(e_1) = J_G(e_2)$
2. *Each vertex* $v \in V_G$ *with* $I_G(v) = k$ *has exactly one incident edge*.

The process of transforming a simple graph FAS into a multi-graph (*S2Multi*) has a computational complexity $O(r)$, where r is the number of edges of the input FAS. When this process is applied over a FAS set C, it has a computational complexity $O(sc)$, where c is the number of FASs in C and s is the average number of edges in the FAS in C.

Results and Discussions

With the purpose of studying the performance of the proposed method as well as its effectiveness, in this section, two experiments are presented. First, the performance of the proposed method over synthetic and real collections is evaluated. Later, the usefulness of the FASs computed by our proposed transformation method from real images for

image classification is shown. All experiments were carried out on a personal computer with an Intel(R) Core(TM) i7-3820 CPU @ 3.60 GHz with 64 GB of RAM. The algorithms S2Multi and M2Simple were implemented in ANSI-C.

In the following experiments, several synthetic multi-graph collections are used for evaluating the performance of the proposed method. These synthetic collections were generated using the PyGen[1] graph emulation library.

In addition, two real image collections were used: COIL (Nene et al., 2008) and ETH (Leibe and Schiele, 2003), which contain images of real objects taken from different viewpoints. In these cases, each image is represented as a multi-graph following the approaches described in (Morales-González and García-Reyes, 2013) and (Morales-González and García-Reyes, 2010), respectively. In COIL, we use the same 25 objects used by Morales-González and García-Reyes (Morales-González and García-Reyes, 2013). This collection is split into 198 (11%) images for training and 1602 (89%) for testing, as in (Morales-González and García-Reyes, 2013). This collection has 144 as average graph size, 19 as average of multi-edges per graphs and 25 classes. In ETH, we use the same 6 categories employed in (Morales-González and García-Reyes, 2010) (*apples*, *cars*, *cows*, *cups*, *horses* and *tomatoes*). This collection is split into 615 (25%) images for training and 1845 (75%) for testing, as in (Morales-González and García-Reyes, 2010). This collection has 179 as average graph size, 25 as average of multi-edges per graphs and 6 classes.

Performance evaluation over synthetic collections

Three kinds of synthetic multi-graph collections were used for evaluating the performance of both algorithms. In this case, we use multi-graph collections generated varying only one parameter at a time. First, we fix $|D| = 1000$ and $|E| = 200$, varying $|V|$ from 200 to 1000, with increments of 200. Next, we fix $|V| = 200$, maintaining $|D| = 1000$ and varying $|E|$ from 200 to 1000, with increments of 200. Finally, we vary $|D|$ from 1000 to 5000, with increments of 1000, keeping $|V| = |E| = 200$. Then, we assign a descriptive name for each synthetic collection, for example, *D1kV1kE200* means that the collection has $|D| = 1000$, $|V| = 1000$ and $|E| = 200$.

In Table 1, the performance results, in terms of runtime, and the average of vertices and edges obtained by the transformation algorithms (M2Simple and S2Multi) are shown. It is important to highlight that we denoted by M2Simple' the algorithm for transforming multi-graphs into simple graph proposed in (Acosta-Mendoza et al., 2015). In this table, the runtime for mining the frequent approximate subgraphs (FASs) from the transformed simple graph collections is also shown. These results were achieved by transforming each multi-graph collection into a simple graph

[1] PyGen is a graph emulation library (available in http://pywebgraph.sourceforge.net). It can be used to simulate, generate, and store different types of graphs and data structures.

collection using M2Simple or M2Simple'. The average of vertices and edges for the simple graph collections obtained are shown, as well as the runtime required for computing the FASs from these transformed simple graph collections. Finally, each pattern obtained from the simple graph collection was transformed into a multi-graph using S2Multi.

Table 1 is split into three sub-tables according to the collection type. In these sub-tables, the first column shows the collection. The other two consecutive blocks, with five columns each one, show the results obtained by applying the transformation method specified on top. The first three columns of each block show the runtime in seconds of M2Simple, the FASs mining process, and S2Multi applied over the mined FASs. The other two columns specify the average number of the vertices and edges of each collection after the transformation from multi-graphs into simple graphs.

According to the results shown in Table 1, the runtime of the transformation process grows with the increment of $|D|$, $|V|$ and $|E|$, however, when the amount of edges increases, this process grows faster than by increasing the number of vertices and the number of graphs. The number of graphs in the collection is an important variable to take into account, because it affects the performance of M2Simple (M2Simple') and S2Multi when it grows. Furthermore, S2Multi receives many more vertices and edges than M2Simple (M2Simple') for the same multi-graph collection, since M2Simple (M2Simple') creates an additional vertex and an additional edge for each transformed edge or loop. In this sense, as M2Simpleóf the method proposed in (Acosta-Mendoza et al., 2015) adds more vertices and edges in these collections than M2Simple of onlyMulti, then allEdges required more time over the same collections than onlyMulti, for both, mining patterns and returning the patterns to multi-graphs. Finally, as it can be seen in Table 1, onlyMulti allows mining patterns in less time than the method reported in (Acosta-Mendoza et al., 2015).

Table 1. Performance of the method proposed in this paper and the method proposed in (Acosta-Mendoza et al., 2015), in terms of runtime, over different synthetic multi-graph collections. The symbol "*" means that the runtime required for the process was more than 48 hours, and the symbol "–" specifies that the transformation cannot be performed because patterns were not mined.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 3, Julio-Septiembre, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-16
http://rcci.uci.cu

(a) Varying $|V|$ from 200 to 1000 with $|D| = 1000$ and $|E| = 200$.

Collection	Proposed method					Method proposed in (Acosta-Mendoza et al., 2015)				
	Runtime (s)			Collection size		Runtime (s)			Collection size	
	M2Simple	VEAM	S2Multi	$\|V\|$	$\|E\|$	M2Simple'	VEAM	S2Multi	$\|V\|$	$\|E\|$
D1kV200E200	1.130	399	0.004	218	206	0.880	1995	0.012	400	388
D1kV400E200	1.300	135	0.003	410	202	1.030	578	0.005	600	391
D1kV600E200	1.450	155	0.003	607	201	1.300	448	0.005	800	393
D1kV800E200	1.680	152	0.003	806	200	1.330	392	0.005	1000	394
D1kV1kE200	1.680	145	0.002	1006	200	1.480	358	0.005	1200	394

(b) Varying $|E|$ from 200 to 1000 with $|D| = 1000$ and $|V| = 200$.

Collection	Proposed method					Method proposed in (Acosta-Mendoza et al., 2015)				
	Runtime (s)			Collection size		Runtime (s)			Collection size	
	M2Simple	VEAM	S2Multi	$\|V\|$	$\|E\|$	M2Simple'	VEAM	S2Multi	$\|V\|$	$\|E\|$
D1kV200E200	1.130	399	0.004	218	206	0.880	1995	0.012	400	388
D1kV200E400	2.560	4070	0.029	249	425	1.610	34094	0.177	600	776
D1kV200E600	4.580	32259	0.098	290	655	2.340	75152	0.250	800	1164
D1kV200E800	7.200	43005	0.207	342	895	3.090	*	–	1000	1552
D1kV200E1k	10.710	93484	1.592	402	1143	3.810	*	–	1200	1941

(c) Varying $|D|$ from 1000 to 5000 with $|V| = 200$ and $|E| = 200$.

Collection	Proposed method					Method proposed in (Acosta-Mendoza et al., 2015)				
	Runtime (s)			Collection size		Runtime (s)			Collection size	
	M2Simple	VEAM	S2Multi	$\|V\|$	$\|E\|$	M2Simple'	VEAM	S2Multi	$\|V\|$	$\|E\|$
D1kV200E200	1.130	399	0.004			0.880	1995	0.012		
D2kV200E200	2.180	1688	0.006			1.800	8548	0.013		
D3kV200E200	3.270	3156	0.006	218	206	2.630	22102	0.014	400	388
D4kV200E200	4.350	6377	0.045			3.690	34469	0.332		
D5kV200E200	5.700	9660	1.017			4.360	51073	1.739		

Performance evaluation over real-world collections

In Table 2, the performance of onlyMulti over the two real image collections (COIL and ETH) is shown, represented as multi-graphs. In this experiment, the performance of M2Simple over the whole multi-graph collection was evaluated while the performance of S2Multi was evaluated over the simple graph subsets generated after applying VEAM to the results of M2Simple, by testing different values for the support threshold.

Table 2. Performance of the proposed method, in terms of runtime and number of identified patterns, over two real image collections.

Collection	Support (δ)	M2Simple	VEAM	No. Identified Patterns	S2Multi	No. Returnable Patterns
COIL	0.5	1.17s	79s	265	0.002s	91
	0.4		300s	867	0.008s	256
	0.3		1143s	4156	0.041s	2163
	0.2		15768s	82551	1.027s	62476
ETH	0.8	5.05s	1332s	138	0.001s	90
	0.7		5508s	883	0.008s	479
	0.6		44208s	4717	0.040s	3824

Grupo Editorial "Ediciones Futuro"
Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

	0.5	153288s	22723	0.234s	15912

The first column of Table 2 shows the collection name. The second column contains the support threshold value (δ) used by VEAM to get a subset of patterns when it is applied to the result of M2Simple. The third column shows the runtime of M2Simple when it is applied over the whole collection specified in column one. The time spent by VEAM is shown in the fourth column, while the amount of patterns computed by VEAM appears in the fifth column. The sixth column contains the runtime of S2Multi for transforming the mined patterns (simple graphs) to multi-graphs. In the last column, we show the amount of returnable patterns. As it can be seen in this Table, the number of returnable patterns identified by VEAM grows as the support threshold decreases. However, it is important to highlight that the runtime of the proposed transformation algorithms is too small regarding the runtime required by VEAM for mining FASs.

Classification results

For showing the usefulness of the patterns identified by using onlyMulti, the results obtained by it in the context of image classification using COIL and ETH collections are shown A new classifier in not being proposed, so the design of a specialized classifier for images represented as multi-graphs is out of the scope of this paper. In this experiment, the idea of the proposed method for the FAS mining in multigraph collections is followed, where all the FASs computed by the proposed method are used for building an attribute vector for each image. In the same way as (Acosta-Mendoza et al., 2012), the vectors are described through the bag-of-word technique using the patters computed after applying the proposed transformation algorithm M2Simple. Finally, once we have the vector representation of the images, a conventional classifier is applied. As in previous experiments, the FAS mining algorithm used for this experiment was VEAM, but fixing to 0,66 the similarity threshold, as recommended in (Acosta-Mendoza et al., 2012).

One of the most recent works reported in literature based on FAS for image classification is the one reported in (Morales-González et al., 2014). Thus, the onlyMulti image classification results are contrasted against those obtained by this method. Since in (Morales-González et al., 2014), the best image classification results were achieved with SVM classifier, we used this classifier for this experiment. The SVM classifier was taken from Weka v3.6.6 (Hall et al., 2009) with the default parameters.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 3, Julio-Septiembre, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-16
http://rcci.uci.cu

Table 3. Classification results (%) using the SVM classifier over multi-graph collections representing images, which are represented as vectors by means of the FASs computed using several support (δ) values.

	COIL			**ETH**	
	Accuracy	F-measure		Accuracy	F-measure
δ	**94.13**	**94.00**	δ	**67.48**	67.50
0.2	90.32		0.5	65.63	66.10
0.3	92.80 83.58		0.6	64.93	**80.37**
0.4	84.85		0.7	64.39	69.76
0.5	74.72	58.70	0.8		

The classification results (accuracy and F-measure results) are shown in Table 3. This table is split into two subtables showing the results obtained over COIL and ETH, respectively. The first column of each table shows the support threshold values used in each experiment. In this case, we use δ = 0,2, 0,3, 0,4 and 0,5 for COIL and δ = 0,5, 0,6, 0,7 and 0,8 for ETH because, in both collections, if greater or smaller values of δ are used, useful patterns could not be identified. The second and third columns show the classification results (accuracy or F-measure), using all FASs computed by VEAM.

In (Morales-González and García-Reyes, 2013), an image classification method, not based on FASs, which uses the same image collections (COIL and ETH) represented as multi-graphs in a similar way as in the current paper, was introduced. Comparing onlyMulti against the method reported in (Morales-González and García-Reyes, 2013), the proposal, using a simple pattern based classifier, obtained better results over the COIL collection, since in (Morales-González and García-Reyes, 2013) an accuracy of 91,60 was reported while onlyMulti scored 94,13. In the case of the ETH collection, we did not improve upon the results reported in (Morales-González and García-Reyes, 2013) where the authors reported an accuracy of 88,0; while the onlyMulti best result was 67,48. In spite of these results, this experiment shows the usefulness of onlyMulti, which allows transforming a multi-graph collection into a simple graph collection for applying traditional FAS miners. Although onlyMulti can be applied in different contexts where data are represented as multi-graphs in order to find out interesting patterns which could be useful for solving different problems.

Conclusions

In this paper, a new method (onlyMulti) for frequent approximate subgraph (FAS) mining in multi-graph collections by transforming multi-graphs into simple graphs and vice versa is proposed. OnlyMulti, as a first step, transforms a multi-

Grupo Editorial "Ediciones Futuro"
Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

graph collection into a simple graph collection, then over this collection a FASs mining algorithm is applied and onlyMulti transforms the patterns found to multi-graphs.

From the experiments reported in this paper, we can conclude that onlyMulti is able to mine FASs from multi-graph collections in a shorter time, producing smaller simple graphs than the only alternative option reported in literature. This is very important in order to reduce the cost of the FAS mining step. Based on the experiments we can conclude that the time required for mining multi-graphs using onlyMulti is smaller than applying the closest state-of-the-art transformation method. In addition, the usefulness of the FASs computed over multi-graph collections by applying onlyMulti in an image classification problem was shown, where in some cases the results obtained by the patterns computed by using the proposed method outperform the results obtained by state-of-the-art classifiers non-based on FASs.

Acknowledgment

This work was partly supported by the National Council of Science and Technology of Mexico (CONACyT) through the scholarship grant 287045.

References

N. Acosta-Mendoza, A. Gago-Alonso, and J.E. Medina-Pagola. Frequent approximate subgraphs as features for graph-based image classification. *Knowledge-Based Systems*, 27:381–392, 2012.

N. Acosta-Mendoza, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad, A. Gago-Alonso, and J.E. Medina-Pagola. A New Method Based on Graph Transformation for FAS Mining in Multi-graph Collections. Acepted. In *Pattern Recognition*, pages 13–22. Springer, 2015.

Y. Björnsson and K. Halldórsson. Improved Heuristics for Optimal Pathfinding on Game Maps. In *American Association for Artificial Intelligence*, page 9, 2006.

I. Boneva, F. Hermann, H. Kastenberg, and A. Rensink. Simulating Multigraph Transformations Using Simple Graphs. *Electronic Comunications of the EASST*, 6, 2007.

R. Cazabet, H. Takeda, and M. Hamasaki. Characterizing the nature of interactions for cooperative creation in online social networks. *Social Network Analysis and Mining*, 5(1):1–17, 2015.

D.J. Cook and L.B. Holder. Substructure discovery using minimum description length and background knowledge. Journal of Artificial Intelligence Research, 1:231-255, 1994.

M. Flores-Garrido, J.A. Carrasco-Ochoa, and J.Fco. Martínez-Trinidad. AGraP: an algorithm for mining frequent patterns in a single graph using inexact matching. Knowledge and Information Systems, 42(2): 1–22, 2015. doi: 10.1007/s10115-014-0747- x.

J.A. González, L.B. Holder, and D.J. Cook. Graph-Based Concept Learning. In Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference, pp. 377-381, Key West, Florida, USA, 2001. AAAI Press.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA Data Mining Software: An Update. *Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations*, 11:10–18, 2009.

Y. Jia, J. Zhang, and J. Huan. An efficient graph-mining method for complicated and noisy data with real-world applications. *Knowledge Information Systems*, 28(2):423–447, 2011.

N.S. Ketkar. Subdue: compression-based frequent pattern discovery in graph data. In OSDM'05: Proceedings of the 1st international workshop on open source data mining, pp. 71-76. ACM Press, 2005.

B. Leibe and B. Schiele. Analyzing Appearance and Contour Based Methods for Object Categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, pages 409–415. Madison, WI, USA, June 16–22 2003.

A. Morales-González and E. B. García-Reyes. Assessing the Role of Spatial Relations for the Object Recognition Task. In *The 15th Iberoamerican Congress on Pattern Recognition (CIARP'10)*, volume 6419 of *Lecture Notes in Computer Science*, pages 549–556. Springer, Heidelberg, 2010.

A. Morales-González and E. B. García-Reyes. Simple object recognition based on spatial relations and visual features represented using irregular pyramids. *Multimedia tools and applications*, 63(3):875–897, 2013.

A. Morales-González, N. Acosta-Mendoza, A. Gago-Alonso, E.B. García-Reyes, and J.E. Medina-Pagola. A new proposal for graph-based image classification using frequent approximate subgraphs. *Pattern Recognition*, 47(1):169–177, 2014. ISSN 0031-3203.

S. Nene, S. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR & SPR 2008*, 2008.

F. Terroso-Saez, M. Valdés-Vela, and A.F.. Skarmeta-Gómez. Online Urban Mobility Detection Based on Velocity Features. In *Proceeding of The 17th International Conference of Big Data Analytics and Knowledge Discovery*, volume LNCS 9263, pages 351–362. Valencia, Spain, 2015.

K. Wang, X. Xie, H. Jin, P. Yuan, F. Lu, and X. Ke. Frequent Subgraph Mining in Graph Databases Based on MapReduce. In G. Wang, Y. Han, and G. Martínez, editors, Advances in Services Computing - 10th Asia-Pacific Services Computing Conference, APSCC 2016, Zhangjiajie, China, November 16-18, 2016, Proceedings, pp. 464-476, Cham, 2016. Springer International Publishing.

J.S. Whalen and J. Kenney. Finding maximal link disjoint paths in a multigraph. *Global Telecommunications Conference and Exhibition. 'Communications: Connecting the Future'*, 1:470–474, 1990.

X. Yan and J. Huan. gSpan: Graph-Based Substructure Pattern Mining. In *International Conference on Data Mining*, Japan, 2002. Maebashi.

R. Youssef, A. Kacem, S. Sevestre-Ghalila, and C. Chappard. Graph Structuring of Skeleton Object for Its HighLevel Exploitation. *Image Analysis and Recognition*, LNCS 9164:419–426, 2015.

F. Zhu, X. Yan, J. Han, and P.S. Yu. gPrune: A Constraint Pushing Framework for Graph Pattern Mining. In Advances in Knowledge Discovery and Data Mining, 11th Pacific-Asia Conference, PAKDD 2007, Nanjing, China, May 22-25, Proceedings, volume 4426 of LNCS, pp. 388-400. Springer, 2007.