Representative Frequent Approximate Subgraph Mining in Multi-Graph Collections

Niusvel Acosta Mendoza, Jesús Ariel Carrasco-Ochoa, Andrés Gago-Alonso,
José Fco. Martínez-Trinidad, José Eladio Medina-Pagola

Technical Report No. CCC-15-001
February 2015

Computational Sciences Coordination
Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla, Mexico

Representative Frequent Approximate Subgraph Mining in Multi-Graph Collections

Niusvel Acosta-Mendoza[a,b,*] Jesús Ariel Carrasco-Ochoa[a] Andrés Gago-Alonso[b]
José Fco. Martínez-Trinidad[a] José Eladio Medina-Pagola[b]

[a]Computer Science Department at Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, CP. 72840, Mexico.
[b]Data Mining Department at Advanced Technologies Application Center (CENATAV)
7ma A# 21406 e/ 214 y 216, Siboney, Playa. C.P. 12200. Havana, Cuba.
E-mail: nacosta@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. There are some applications where multi-graphs are used for modeling data; however, most FAS miners have been designed to work just with simple-graphs. Furthermore, the large number of mined patterns is one of the fundamental drawbacks of FAS mining, causing a negative impact in the computational performance of the methods where they will be used. Therefore, in this PhD research proposal, in order to reduce the amount of mined patterns, as well as to deal with multi-graph structures, we propose to develop FAS mining algorithms capable of retrieving a subset of representative patterns in multi-graph collections. The representative patterns should be identified during the mining process, ideally alleviating the computational performance, since it shall not be necessary to retrieve all possible patterns but subsequently picking only those considered representative.

Keywords: *Approximate multi-graph mining, representative frequent subgraphs, approximate graph matching*.

Contents

1 Introduction **2**

2 Background **4**

3 Frequent subgraph mining **9**

3.1 Algorithms based on sub-isomorphism 9

3.2 Algorithms based on edit distance 9

3.3 Algorithms based on heuristics for uncertain graphs 10

3.4 Algorithms based on semantic substitution 11

3.5 Critical evaluation 12

3.6 VEAM algorithm overview 13

4 Justification and motivation **15**

5 Research problem **16**

6 Research proposal **17**

6.1 Research questions 17

6.2 General Aim 17

6.3 Specific Aims 17

6.4 Expected contributions 18

6.5 Methodology 18

6.5.1 Schedule 21

7 Preliminary Results **21**

7.1 Proposed transformation Method 22

7.1.1 Processing multi-graphs 22

7.1.2 Experimental Results 24

7.2 Proposed Extension of CAM for Representing Multi-Graphs 28

7.2.1 A Miner Based on the New CAM for Multi-Graph Collections 29

7.2.2 Experimental Results 31

8 Conclusions **34**

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 and linguistics, among others [25]. This topic includes different techniques for pattern extraction, where frequent subgraph mining techniques have been highlighted. Frequent subgraph mining techniques use graphs as basic structure. A graph is used to model data due to its representation expressiveness and its suitability for applications where some kind of entities and their relationships must be encoded [40]. Graphs are general and powerful structures that allow us naturally modeling data in multiple domains [34, 6].

Several algorithms have been developed for finding all frequent subgraphs in a graph database. Starting by *AGM* [22], followed by *FSG* [29], *ISG* [47], *FFSM* [21], and *grCAM* [17], these algorithms identify and compute frequent subgraphs using a breadth-first search approach growing the frequent subgraphs one vertex or one edge at time. However, these algorithms have considerable overhead into the process for generating candidate graph patterns. In order to avoid this overhead, other algorithms have been developed, most of which adopt a pattern-growth strategy. These algorithms extend a frequent subgraph by adding a new edge, in every possible position. However, a potential problem with this candidate generation process is that the same subgraph can be obtained many times. Thus, to reduce the generation of duplicate graphs, each frequent graph should be extended as conservatively as possible. Some of the algorithms that follow this approach are *gSpan* [58], which is based on canonical representations for graphs; *ADI-Mine* [49] proposed for incrementing the processing capacity of gSpan; *gPrune* [60] that improves gSpan introducing pruning properties for complicated structural constraints in graph mining; *gRed* [16] and *gdFil* [15] that improve gSpan based on canonical form and right-most extension properties. Other algorithms are *MoFa* [9], which introduces the breadth-first search (*BFS*) code for representing candidate subgraphs; and *Gaston* [37] which first considers fragments that are paths or trees, processing general graphs with cycles at the end.

The aforementioned algorithms use exact matching strategies for computing the frequent subgraphs, but there are several practical problems where arises the need to allow some variations in the data. This issue is because there are specific problems where exact matching is not applicable with a positive outcome [20, 10, 3, 31]. Therefore, it is important to tolerate certain level of variability: semantic variations, vertices or edges mismatching during frequent pattern search. It is required to evaluate the similarity between graphs considering approximate matching [12]. In this sense, several algorithms have been developed for frequent approximate subgraph (*FAS*) mining, which use different approximate graph matching techniques allowing the detection of frequent subgraphs with some variations in the data [20, 18, 42, 59, 24, 3, 4]. A pioneer in this inexact approach is *SUBDUE* [20], which is based on the edit distance using the minimum description length (*MDL*) principle for computing the FASs in a large graph. This algorithm was extended by *subdueCL* [18] for processing and classifying graph collections. Different heuristics and graph matching approaches are used as basis for FAS mining algorithms, for example: edit distance [42], vertex/edge disjoint homeomorphism [55], β-edge isomorphism [59], heuristic over uncertain graphs [31], and heuristics based on semantic substitutions [11, 24, 3].

FAS mining algorithms have been successfully used for supervised classification, where FASs are used as features for representing objects. This approach has been used in several domains such as: analysis of biochemical structures [11, 55, 24]; genetic networks analysis [42]; circuits, links and social networks analysis [20]; and image classification [3]. These FAS mining algorithms have become important tools for real applications in several domains. There exist some applications where complex data structures for modeling the problem are required to achieve a good representation [52, 7, 34, 35]. An example of these complex data structures are multi-graphs. In this case, most existing FAS mining algorithms cannot directly be applied because they were not originally designed to deal with this kind of structures, but for working with simple-graphs. Thus, in this PhD research proposal, we are interested for proposing FAS mining algorithms for processing multi-graph collections.

In most of the aforementioned FAS mining applications, usually a large number of frequent subgraphs is retrieved [24, 2]. Discovering interesting patterns in this collection of patterns is still a challenge [25]. Several techniques have been proposed for identifying interesting subgraphs, reducing the dimensionality of the identified pattern set, such as: identifying only maximal, cliques, and closed subgraphs, among others [57, 61, 24, 38, 45, 14]. So, this PhD research proposal is focused on proposing FAS miners for identifying only representative patterns on multi-graph collections.

Using only maximal frequent subgraphs instead of using all the patterns is one of the techniques used to avoid redundancy among the computed patterns and consequently for reducing the dimensionality of this set of patterns. A maximal frequent subgraph is a pattern which is not a subgraph of any other frequent subgraph [27, 14]. From the frequent maximal subgraphs it is possible to reconstruct the whole set of frequent subgraphs because all of them are summarized into the maximal patterns. However, from the maximal patterns, the information about the support of non-maximal patterns cannot be retrieved. To face this problem, in several applications, closed frequent subgraphs are used. A closed frequent subgraph is a pattern that does not have any supergraph with the same frequency [57, 42, 45]. Thus, from the closed frequent subgraphs, it is possible to reconstruct the whole set of frequent subgraphs including the information about their support.

In real applications such as biochemical compounds, frequent clique subgraphs [32, 26, 39, 24, 48, 56] have been used for reducing the amount of mined patterns. A frequent clique subgraph is a pattern where every two vertices are connected by an edge. Using this kind of patterns, specially when the graph collection contains many clique graphs, the resulting number of patterns is too high. Furthermore, other kind and generalizations of clique patterns such as maximal clique, quasi-clique and near-clique patterns [32, 26, 48, 56] are proposed for solving some specific application according to their context.

Finally, based on the aforementioned techniques, in this PhD research proposal, we focus on develop algorithms for retrieving only representative (closed, clique, or maximal) FASs from multi-graph collections. Therefore, a comparative study for identifying the adequate representative patterns for some specific applications (supervised graph classification, or information retrieved, amount other) will be performed.

This research proposal is structured as follow: in Section 2, some basic concepts are presented. In Section 3, the related work of frequent subgraph mining algorithms in graph collections is presented. Justification and motivation of this research is included in Section 4. In Section 5, the research problem is presented. In Section 6, the PhD research proposal, where the research questions, general aim, specific aims, expected contributions, methodology and schedule are described. Next, in Section 7, the preliminary results are summarized. Finally, preliminary conclusions and future work directions are discussed in Section 8.

2 Background

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

Definition 2.1 (Graph)**.** A *graph* is a triplet, $G = (V_G, E_G, \phi_G)$, where V_G is a set whose elements are called *vertices*, E_G is a set whose elements are called *edges*, $\phi_G : E_G \to \{\{u, v\}|u, v \in V_G\}$ is the *incidence function* (the edge e, through the function $\phi_G(e)$, connects the vertices u and v if $\phi_G(e) = \{u, v\}$).

Henceforth, when we use the notations V_G, E_G and ϕ_G we refer to the elements of the graph G.

Definition 2.2 (Subgraph and supergraph)**.** Let G_1 and G_2 be two graphs, G_1 is a *subgraph* of G_2 if $V_{G_1} \subseteq V_{G_2}$, $E_{G_1} \subseteq E_{G_2}$, ϕ_{G_1} is a restriction[1] of ϕ_{G_2} to V_{G_1}. In this case, $G_1 \subseteq G_2$ denotes that G_1 is a *subgraph* of G_2 or that G_2 is a *supergraph* of G_1.

Definition 2.3 (Multi-edge and loop)**.** Let G be a graph. An edge $e \in E_G$, where $\phi_G(e) = \{u, v\}$ and $u \neq v$, is a *multi-edge* if there is $e' \in E_G$ such that $e \neq e'$ and $\phi_G(e) = \phi_G(e')$; otherwise, e is a *simple-edge*. If $|\phi_G(e)| = 1$, e is called a *loop*, i.e. $\phi_G(e) = \{u\} = \{v\}$.

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

Definition 2.4 (Simple-graph and multi-graph)**.** A graph G is a *simple-graph* if it has no loops and no multi-edges, i.e. to has only simple-edges, $E_G = E_G^{(s)}$, being $E_G^{(m)} = E_G^{(l)} = \emptyset$; otherwise, it is a *multi-graph*.

Definition 2.5 (Induced subgraphs)**.** Let G, G_1 and G_2 be three graphs and $V' \subseteq V_G$ and $E' \subseteq E_G$. G_1 is known as the *v-induced subgraph* of G regarding V', denoted by $G[V']$, if $V_{G_1} = V'$ where $E_{G_1} = \{e \in E_G|\phi_G(e) \subseteq V'\}$. In a similar way, G_2 is the *e-induced subgraph* of G regarding E', denoted by $G[E']$, if $E_{G_2} = E'$ where $V_{G_2} = \bigcup_{e \in E'} \phi_G(e)$.

For both, simple-graphs and multi-graphs, the isomorphism between two graphs is defined as follows.

Definition 2.6 (Isomorphism)**.** Given two graphs G_1 and G_2, a pair of functions (f, g) is an *isomorphism* between these graphs (denoted by $G_1 \simeq G_2$) if $f : V_{G_1} \to V_{G_2}$ and $g : E_{G_1} \to E_{G_2}$ are bijective functions and for each edge $e \in E_{G_1}$, $\phi_{G_2}(g(e)) = \{f(v)|v \in \phi_{G_1}(e)\}$. If there is an isomorphism between G_1 and G_2, then G_1 and G_2 are *isomorphic*.

[1]A restriction of a function is the result of trimming its domain

It is common that between two graphs more than one isomorphism can be found, then we can define the isomorphism set as $\Upsilon(G_1, G_2) = \{(f, g)|(f, g)$ is an isomorphism between G_1 and $G_2\}$.

Definition 2.7 (Sub-isomorphism)**.** Given three graphs G_1, G_2 and G_3. If $G_1 \simeq G_3$ and $G_3 \subseteq G_2$, then there is a *sub-isomorphism* between G_1 and G_2, denoted by $G_1 \subseteq_s G_2$, and G_1 is *sub-isomorphic* to G_2.

This work is focused on undirected labeled graphs in the domain of all possible labels $L = L_V \cup L_E$, where L_V and L_E are the label sets for vertices and edges, respectively.

Definition 2.8 (Labeled graphs)**.** A *labeled graph*, is a 5-tuple, $G = (V_G, E_G, \phi_G, I_G, J_G)$, where $G' = (V_G, E_G, \phi_G)$ is a graph, $I : V_G \to L_V$ is a *labeling function* for assigning labels to vertices and $J : E_G \to L_E$ is a *labeling function* for assigning labels to edges.

Definition 2.9 (Labeled subgraph)**.** Let G_1 and G_2 be two labeled graphs, and let G'_1 and G'_2 their corresponding unlabeled graphs, respectively. G_1 is a *labeled subgraph*, or simply *subgraph*, of G_2 if G'_1 is a subgraph of G'_2 and each labeling function of G_1 is a restriction of its corresponding one in G_2.

Definition 2.10 (Forward, backward extensions and child)**.** Let G_1 and G_2 be two labeled graphs, where $G_1 \subseteq G_2$, the edge $e \in E_{G_2}$ is an *extension* of G_1 if $E_{G_2} = E_{G_1} \cup \{e\}$, $V_{G_1} \cap \phi_{G_2}(e) \neq \emptyset$, $V_{G_2} = V_{G_1} \cup \phi_{G_2}(e)$. This fact will be denoted by $G_2 = G_1 \diamond e$. The edge e is a *backward extension* if $\phi_{G_2}(e) \subseteq V_{G_1}$, otherwise e is a *forward extension* (it extends the vertex set of G_1). When e is an extension of G_1 and $G_2 = G_1 \diamond e$, then we will refer to G_2 is as a child of G_1.

The following definitions state an useful framework for describing a specific inexact-based graph mining process. In almost all graph mining approaches [11, 55, 59, 24, 31, 3, 13], the authors firstly define the criteria for comparing graphs, according to the application context. In [11, 24, 3], such criterion is defined following the idea of the Definition 2.11.

Definition 2.11 (Similarity between labeled graphs preserving the topology)**.** Let G_1 and G_2 be two labeled graphs, where $V_{G_1} = \{v_1, v_2, \ldots, v_{|V_{G_1}|}\}$ and $E_{G_1} = \{e_1, e_2, \ldots, e_{|E_{G_1}|}\}$. The *similarity* between G_1 and G_2, preserving the topology, is defined as:

$$sim(G_1, G_2) = \begin{cases} \max\limits_{(f,g)\in\Upsilon(G_1,G_2)} \Theta_{(f,g)}(G_1, G_2) & if\ \Upsilon(G_1, G_2) \neq \emptyset \\ 0 & otherwise \end{cases} \tag{1}$$

where $\Theta_{(f,g)}(G_1, G_2)$ is a similarity function for comparing the label information between G_1 and G_2, according to the isomorphism (f, g).

Notice that the similarity function $\Theta_{(f,g)}$ can be defined by means of different operations between the vertex and edge labels, preserving the graph topologies. In this paper, an example of similarity function for an specific context is shown in Definition 3.1 described in Section 7.1.

Definition 2.12 (Approximate isomorphism and approximate sub-isomorphism)**.** Let G_1, G_2 and G_3 be three labeled graphs, let $sim(G_1, G_2)$ be a similarity function, preserving the topology, and let $\tau = [0, 1]$ 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 $G_2 \subseteq G_3$, then there is an *approximate sub-isomorphism* between G_1 and G_3, denoted as $G_1 \subseteq_A G_3$.

Definition 2.13 (Occurrence)**.** Let G_1, G_2 and T be three labeled graphs, where T is a subgraph of G_2, then T is an occurrence of G_1 in G_2, using a specific similarity threshold τ, if $sim(G_1, T) \geq \tau$.

Definition 2.14 (Maximum inclusion degree)**.** Let G_1 and G_2 be two labeled graphs, let $sim(G_1, G_2)$ be a similarity function, preserving the topology; 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:

$$maxID(G_1, G_2) = \max_{G \subseteq G_2} sim(G_1, G), \tag{2}$$

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.

Using Definition 2.14, a support definition allowing the computation of inexact matching between graphs can be introduced.

Definition 2.15 (Approximate support)**.** Let $D = \{G_1, \ldots, G_{|D|}\}$ be a labeled graph collection, let $sim(G_1, G_2)$ be a similarity, let τ be a similarity threshold, and let G be a labeled graph. Thus, the *approximate support* (denoted by *appSupp*) of G in D, is obtained through equation (3):

$$appSupp(G, D) = \frac{\sum_{G_i \in \{G_j | G_j \in D, G \subseteq_A G_j\}} maxID(G, G_i)}{|D|} \tag{3}$$

Using (3), G is a *FAS* in D if $appSupp(G, D) \geq \delta$, for a given support threshold $\delta = [0, 1]$, a similarity function among graphs $sim(G_1, G_2)$ and a similarity threshold τ.

FAS mining consists of, given a support threshold δ and a similarity threshold τ, finding all the FAS in a collection of graphs D, using a certain similarity function sim.

The following definitions are useful for describing the representative patterns mining process.

Definition 2.16 (Maximal frequent approximate subgraph (*maximal FAS*))**.** Let D be a graph collection, let $sim(G_1, G_2)$ be a similarity function among graphs, let δ be a support threshold and let τ be a similarity threshold. A *maximal FAS* (*maximal FAS*) in D is a FAS with no frequent approximate supergraphs.

Maximal FAS mining consists in, given a support threshold δ and a similarity threshold τ, finding all the maximal FASs in a collection of graphs D, using a certain similarity function sim.

Definition 2.17 (Closed frequent approximate subgraph)**.** Let D be a graph collection, let $sim(G_1, G_2)$ be a similarity function among graphs, let δ be a support threshold and let τ be a similarity threshold. A *closed FAS* in D is a FAS which does not have any supergraph with the same frequency.

Closed FAS mining consists in, given a support threshold δ and a similarity threshold τ, finding all the closed FASs in a collection of graphs D, using a certain similarity function sim.

Definition 2.18 (Clique frequent approximate subgraph)**.** Let D be a graph collection, let $sim(G_1, G_2)$ be a similarity function among graphs, let δ be a support threshold and let τ be a similarity threshold. A *clique FAS* in D is a FAS where every two vertices are connected by an edge.

A generalization of Definition 2.18 is defined in Definition 2.19, which is one of the most used in the literature where clique patterns are treated.

Definition 2.19 (γ-Quasi-Clique frequent approximate subgraph)**.** Let D be a graph collection, let $sim(G_1, G_2)$ be a similarity function among graphs, let δ be a support threshold and let τ be a similarity threshold. A *γ-quasi-clique (*$0 \leq \gamma \leq 1$*) FAS* in D is a FAS G where every vertex $v \in V_G$, $degree(v)\lceil\gamma \cdot (|V_G| - 1)\rceil$. Being $degree(v)$ the degree of the vertex v.

Notice that if $\gamma = 1$ for a graph G, then G is a clique pattern.

γ-quasi-clique FAS mining consists in, given a support threshold δ, a similarity threshold τ and an approximation clique threshold γ, finding all the γ-quasi-clique FASs in a collection of graphs D, using a certain similarity function sim.

3 Frequent subgraph mining

In this section, we describe some algorithms for frequent subgraph mining reported in the literature. These algorithms are the most closer of our proposal. Then, based on the matching function and heuristics used for mining patterns, these algorithms could be classified into several categories.

3.1 Algorithms based on sub-isomorphism

All exact algorithms are proposed under sub-isomorphism criterion for mining frequent subgraph in graph collections. However, only in the *CloseGraph* [57] algorithm, a way for mining patterns in multi-graph collections is proposed. The CloseGraph algorithm is an extension of the gSpan algorithm [58] for mining only the closed frequent subgraphs from a graph collection. In this work, some prunes based on equivalent occurrences and early termination concepts are introduced for improving the computational performance of gSpan.

3.2 Algorithms based on edit distance

Under this criterion, different heuristics based on edit operations (i.e. insertion, deletion and substitution) are used by the algorithms for mining frequent approximate subgraphs (FASs) in graph collections.

- *subdueCL* [18] uses a constrained beam search method and the minimum description length (*MDL*) principle [41] in their subgraph discovery strategy on a graph collection with two classes. Subgraphs, that can best compress the graphs in the positive class and covering in the lowest degree the graphs of the negative class, based on the MDL principle, are computed. subdueCL performs approximate matching allowing slight variations of substructures using an edit distance strategy by incrementally growing a single vertex. This algorithm can deals with multi-graphs because it is not based on canonical forms, and the contractions performed based on MDL are not affected by this complex structure.

- *RNGV* [42] computes all closed patterns which are approximately frequent in a graph collection. In this algorithm, a candidate closed subgraph is frequent if it appears at least $\delta * |D|$ times into the input graph collection D, where δ is the support rate. RNGV starts following the basic structure of the Apriori approach, but each identified pattern is extended by one edge using a depth first search (*DFS*) technique.

- *AGraP* [13] identifies frequent patterns in a single graph using inexact matching, combining a patterns-growth approach, a dissimilarity measure proposed by the authors and a strategy to find occurrences that could have structural differences, in edges as well as in vertices, with respect to the pattern that they represent. The dissimilarity measure proposed by Flores *et al.* is based on the graph edit distance and it does not require a bijection between the vertex sets of the graphs being compared. In fact, the main difference between AGraP and other algorithms is that the former allows differences not only in labels or edges but, also, in vertices, in such a way that pattern occurrences could have "missed" or "extra" vertices with respect to the pattern.

- *maxAGraP* [14] is an extension of the AGraP algorithm for computing only maximal patterns on a single graph. This algorithm identifies the maximal FASs following the strategy proposed in [13].

3.3 Algorithms based on heuristics for uncertain graphs

In this case, the claim which is defended by the authors is that mining uncertain graph data is widely applicable in real problems. This claim is because, in real applications, graph data is subject to uncertainties due to incompleteness and imprecision. This approach is semantically different from and computationally more complex than, mining exact graph data. Each graph, processed by an algorithm based on this approach, is uncertain because it is built taking into account the existence probability over the original graph.

- *Top-k maximal clique algorithm (TkMCA)* [61] proposes computing the top-k maximal clique subgraphs in an uncertain graph. In this approach, a combination of both, maximal and clique, are used for taking advantage from both approaches. This algorithm uses an exact approach for computing sub-isomorphism between graphs, but during the candidate generation process, each candidate is identified as a clique evaluating the probability that the candidate has of being a clique across all processed graphs.

- *MUSE* [31] is an algorithm for mining frequent subgraphs from uncertain graph data. This algorithm is proposed to find a set of approximately frequent subgraph patterns by allowing an error tolerance on the expected support of discovered subgraph patterns. In this algorithm, a method to determine whether a subgraph pattern is frequent or not, and a pruning method to reduce the complexity of examining subgraph patterns, are introduced.

3.4 Algorithms based on semantic substitution

The authors of this kind of algorithms follow the idea that not always a vertex or an edge or their labels can be replaced by any other. Thus, semantic variations are taken into account for computing FASs.

- *gApprox* [11] computes all FASs in a single graph. In this algorithm, each vertex contains a list of vertices that can replace it with the same probability; meaning that a vertex v can only be replaced by another vertex u if and only if u is in the v corresponding substitution list.

- *APGM* [24] computes all FASs that are cliques in a graph collection. This algorithm uses a DFS approach for building each clique candidate pattern by extending the edges and using the canonical adjacency matrix (CAM) code of each candidate for sub-isomorphism tests. Also, in this process, a substitution matrix, containing probabilities of substitution between vertex labels, is used. So, this algorithm only deals with variations between vertex labels.

- *VEAM* [3, 4] computes all FASs in a graph collection. VEAM is an extension of APGM, where both: vertex and edge distortions are taken into account. This algorithm uses a DFS approach for building each candidate pattern by extending them through an edge in each iteration. VEAM applies CAM codes for each candidate sub-isomorphism test. Also, in this process, two substitution matrices are used: one for vertex labels and another for edge labels.

3.5 Critical evaluation

The algorithms described in this section identify different types of patterns in a graph collection. These patterns are computed based on the matching function used to compare graphs in the mining process. In Table 1, different characteristics of these algorithms are summarized.

Table 1. Characteristics of the closer frequent subgraph mining algorithms.

Algorithm	FAS	Graph Collections	Allowing Semantic Variations	Keeping Graph Topology	Representative Patterns	Multi-Graph
CloseGraph (2003)	No	**Yes**	No	**Yes**	**Closed**	Theoretically
gApprox (2007)	**Yes**	No	Partially	**Yes**	No	No
AGraP (2014)	**Yes**	No	**Yes**	No	No	No
maxAGraP (2014)	**Yes**	No	**Yes**	No	**Maximal**	No
MUSE (2012)	**Yes**	**Yes**	No	**Yes**	No	No
TkMCA (2010)	**Yes**	**Yes**	No	**Yes**	**Maximal clique**	No
RNGV (2006)	**Yes**	**Yes**	**Yes**	No	**Closed**	No
subdueCL (2001)	**Yes**	**Yes**	**Yes**	No	No	**Yes**
APGM (2011)	**Yes**	**Yes**	Partially	**Yes**	**Clique**	No
VEAM (2012)	**Yes**	**Yes**	**Yes**	**Yes**	No	No

Analysing the reported algorithms (see Table 1), 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. In some algorithms (MUSE, and TkMCA), 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 when exact matching is used.

3. Other algorithms (CloseGraph, RNGV, TkMCA and APGM), which compute representative patterns, were not designed to deal with semantic distortions between vertex and edge label sets in graph collections. CloseGraph is an exact algorithm, RNGV was proposed for treating topological distortions through edit distance heuristic, TkMCA was designed for processing uncertain graph collections, and APGM only deals with variations between vertex labels.

4. The AGraP, maxAGraP, and gApprox algorithms were not designed for dealing with graph collections.

In this PhD research proposal, we focus on the approximate graph mining approach, which allows semantic variations in vertex and edge labels, keeping the graph topology, over multi-graph collections.

3.6 VEAM algorithm overview

As we mentioned, VEAM [3, 4] is the only algorithm that computes frequent approximate subgraphs on a simple-graph collection, keeping the graph topology. In this section, the VEAM algorithm is described.

The VEAM algorithm starts computing all frequent approximate subgraphs corresponding to single-vertex graphs and single-edge graphs. Then, following a *Depth-First Search* (DFS) approach, each frequent single-edge is extended by adding another single-edge at a time through a recursive function, called "Search". When all frequent approximate single-edges have been extended, then the set of all frequent approximate subgraphs in the multi-graph collection D is returned.

The Search function, shown in Algorithm 1, recursively performs the candidate extension of all frequent graphs. A candidate set for the given frequent graph is obtained from "GenCandidate" function invoked in line 2 of the pseudo-code. Then, the frequency of each candidate is verified, keeping only the candidates that satisfy the support constraint and which are not identified in previous steps; these candidates are stored as output patterns and are extended performing a recursive call to Search function.

Algorithm 1: *Search*

```
Input: $T$ : A frequent approximate subgraph, $D$ : Simple-graph collection, $M_V$ : Vertex label
       substitution matrix, $M_E$ : Edge label substitution matrix, $\tau$ : Similarity threshold, $\delta$ : Support
       threshold, $F$ : Frequent approximate subgraph set.
Output: $F$ : Frequent approximate subgraph set.
1 $C \leftarrow$ GenCandidate($T, D, M_V, M_E, \tau$);
2 foreach $(T_1, S_{T_1}) \in C$ do
3     if $appSupp(S_{T_1}, D) \geq \delta$ and $T_1 \notin F$ then
4         $F \leftarrow F \cup T_1$;
5         Search($T_1, D, M_V, M_E, \tau, \delta, F$);
```

The aim of the GenCandidate function, shown in Algorithm 2, is to compute all candidate extensions of a given frequent graph T. In this candidate generation phase, all child of T and its occurrences in the simple-graph collection D are searched. Later, the canonical forms based on adjacency matrices (CAM) for these subgraphs, which satisfy the similarity constraint using Definition 3.1, are computed by the "ComputeCAM" function invoked in line 3 in Algorithm 2. Finally, each subgraph G with its corresponding CAM code and its similarity value is stored as an output candidate in C.

Algorithm 2: *GenCandidate*

```
Input: $T$ : A frequent approximate subgraph, $D$ : Simple-graph collection, $M_V$ : Vertex label
       substitution matrix, $M_E$ : Edge label substitution matrix, $\tau$ : Similarity threshold.
Output: $C$ : A set of candidate graph.
1 $C \leftarrow$ An empty hash table;
2 $O \leftarrow \{(G, T_e) | G$ is a child of $T$ and $T_e$ is an occurrence of $G$ in $G_k \in D\}$;
3 foreach $(G, T_e) \in O$ do
4     $CAM_G =$ ComputeCAM($G$,$e'$) ; // see Algorithm 3
5     $sim(G, T_e)$ is inserted into $C[CAM_G]$;
```

The similarity function used by VEAM for the graph matching process is an extension of the one proposed in [24], but in this case, this function allows performing the approximate matching considering both, vertex and edge labels (see the definition 3.1).

Definition 3.1 (Similarity function $\Theta_{(f,g)}$ based on substitution matrices)**.** Let G_1 and G_2 be two labeled graphs, and let M_V and M_E be two substitution matrices in L_V and L_E, respectively. The similarity function is defined as:

$$\Theta_{(f,g)}(G_1, G_2) = \prod_{v \in V_{G_1}} \frac{MV_{I_1(v),I_2(f(v))}}{MV_{I_1(v),I_1(v)}} * \prod_{e \in E_{G_1}} \frac{ME_{J_1(e),J_2(g(e))}}{ME_{J_1(e),J_1(e)}} \tag{4}$$

where (f, g) is an isomorphism between G_1 and G_2.

The canonical form used by VEAM is known as *Canonical Adjacency Matrix* (CAM) as it is based on the graph adjacency matrix. In Definition 3.2 the adjacency matrix is formalized. Notice that, since the adjacency matrix of an undirected simple-graph is symmetric and this is the kind of graphs treated, only the upper or lower triangular adjacency matrices are needed for computing the CAM code of a graph.

Definition 3.2 (Adjacency matrix of a labeled simple-graph)**.** Let $v_i, v_j \in V_G$ be two arbitrary vertices of a given labeled simple-graph G, the adjacency matrix $M = (m_{i,j})_{|V_G| \times |V_G|}$ for G is defined by:

$$m_{i,j} = \begin{cases} I_G(v_i) & if\ i = j \\ J_G(e) & if\ i \neq j, e \in E_G, \phi_G(e) = \{v_i, v_j\} \\ - & otherwise \end{cases} \tag{5}$$

The symbol "–" is used for representing the edge labels absence.

In the VEAM algorithm, the CAM code of a graph G is obtained following the ideas proposed in [29], where the adjacency matrix representation of a simple-graph is converted into a sequence of symbols. In this way, the label vertex set, as well as the degree-based partition order are used as vertex invariant as presented in [29]. Using this vertex invariant, vertices of a graph can be partitioned into equivalence classes, where vertices of the same class have equivalent vertex invariant values. This criteria allows performing permutations of the vertices into the same cluster (partition) instead to performing permutations into the whole set of vertices. This is because two isomorphic graphs will lead to the same partitioning of the vertices and therefore the same canonical code is computed from them.

In Algorithm 3, the process of computing the CAM code of a given simple-graph is shown. First, the partition set P for the vertices of G is generated according to the vertex labels and the vertex degrees as partitioning criteria. Next, P is sorted in descending way, according to the vertex labels in first place and the degrees as secondary criterion (see lines 2). Later, in each partition an internal sorting taking into account a maximum lexicographical order obtained from its permutations (see lines $3 - 14$) is performed. Finally, the CAM code CAM_G of G is obtained, which is computed according to the final ordering of P.

4 Justification and motivation

Accordingly to the expressed into the sections 1 and 3, frequent approximate subgraph (FAS) mining is an important problem in graph mining. This type of mining has become into a remarkable technique where structural or semantic distortions are taken into account for detecting patterns. In the literature, better results have been reported using approximate approaches [55, 59, 24, 31, 5, 1] than those results reported by exact approaches. However, most FAS mining algorithms are designed to work just with simple-graphs, and, there are some real applications where multi-graphs are necessary for modeling the nature of the data [52, 7, 43, 34, 35, 28, 46]. In the literature, there are some works where multi-graphs are apparently used [50, 51, 23, 44, 53, 54]. However, in these works, the multi-graph term refers to the use of multiple simple graphs. Therefore, these works are not related to our research problem.

FAS mining algorithms compute large sets of patterns and several of these patterns include redundant information. There are some works that are focused on this problem, where the main idea is to develop methods for reducing the dimensionality of the set of patterns [2, 1]. However, in these cases, the reduction

Algorithm 3: $ComputeCAM$

```
Input: $G$ : A candidate subgraph, $e'$ : The last edge added into $G$.
Output: $CAM_G$ : The CAM code for the candidate subgraph $G$.
1  $P \leftarrow$ Partition of $V_G$ such that: $v_i, v_j \in P_k$, have the same label and degree, where $i \neq j$ and $P_k \in P$;
2  $P$ is lexicographically sorted $[(l_{i-1}, d_{i-1}) \succ (l_i, d_i)]$, being $l_k$ and $d_k$ are the label and the degree of $P_k$;
3  foreach $P_k = \{p_1, \ldots, p_{|P_k|}\} \in P$ do
4      $l = |P_k|$;
5      $l_k$ is the label for the clusters $P_k$;
6      while $l \neq 1$ do
7          $newl = 1$;
8          foreach $i = \{2, \ldots, |P_k|\}$ do
9              $X = (x_{(1,1)}, \ldots, x_{(|P_1|,1)}, \ldots, x_{(1,k)}, \ldots, x_{(i-1,k)})$ is computed based on Definition 3.2,
               being $x_{(j,w)}$ the label of the edge between the vertex in $p_{i-1}$ and the vertex in $p_j \in P_w \in P$;
10             $Y = (y_{(1,1)}, \ldots, y_{(|P_1|,1)}, \ldots, y_{(1,k)}, \ldots, y_{(i-2,k)}, y_{(i,k)})$ is computed based on
               Definition 3.2, being $y_{(j,w)}$ the label of the edge between the vertex in $p_i$ and the vertex in
               $p_j \in P_w \in P$;
11             if $X < Y$, following a lexicographical order then
12                 Swap($p_{i-1}$, $p_i$);
13                 $newl = i$;
14         $l = newl$;
15 The adjacency matrix $M$ of $G$ is built sorting its cells following the lexicographical order achieved in $P$;
16 The CAM code is obtained from $M$ and it is stored into $CAM_G$;
```

is achieved in a post-processing stage taking into account the supervised information provided by some classification tasks. Therefore, another important challenge is to identify only representative (maximal, closed, or clique, among others) patterns from multi-graph collections during the mining process. In this way, the reduction will be based on the information contained into the graph collection rather than on the application task. Finally, we will perform an analysis in some application context for identifying which kind of representative pattern is more adequate to be used.

5 Research problem

Using frequent approximate subgraph (*FAS*) mining for knowledge extraction, better results than the exact techniques are reported in some tasks like graph classification, however, it has the following shortcomings.

Limitation 1. Existing mining algorithms for identifying FASs allowing semantic variations, keeping the graph topology, cannot be applied in a context where data are modeled by using multi-graphs.

Limitation 2. The number of computed patterns during the mining process if often prohibitively large. Thus it affects the performance of the methods where they will be used.

The research problem consists in providing solutions for the above limitations, which will be faced from a data mining approach, specifically following the frequent subgraph mining approach; therefore, they should be separated from the representation and classification approaches, among others. Thus, this PhD research proposal should be evaluated from a frequent subgraph mining viewpoint.

6 Research proposal

In this section, the proposal of the current research is presented.

6.1 Research questions

Considering the aforementioned research problem, the following research questions arise:

1. How can frequent approximate subgraphs be mined in multi-graph collections?
2. What representative frequent approximate subgraphs should be considered to reduce the number of patterns computed by traditional frequent approximate subgraph mining algorithms, preserving most of the information contained in the whole set (in terms of occurrences, structural complexity or pattern size)?

6.2 General Aim

To propose algorithms for mining representative frequent approximate subgraphs in multi-graph collections, which must be competitive in time with the FASs mining algorithms for simple-graph collections.

6.3 Specific Aims

1. Propose an algorithm for frequent approximate subgraph mining in multi-graph collections.
2. Propose an algorithm for computing maximal frequent approximate subgraphs in multi-graph collections.

3. Propose an algorithm for computing closed frequent approximate subgraphs in multi-graph collections.

4. Propose an algorithm for computing clique frequent approximate subgraphs in multi-graph collections.

5. Propose a classification framework based on frequent approximate subgraphs, for evaluating the accuracy and performance of the representative subgraphs computed by our algorithms.

6.4 Expected contributions

1. An algorithm for computing frequent approximate subgraphs in multi-graph collections.

2. An algorithm for computing maximal frequent approximate subgraphs in a multi-graph collection.

3. An algorithm for computing closed frequent approximate subgraphs in a multi-graph collection.

4. An algorithm for computing clique frequent approximate subgraphs in a multi-graph collection.

5. A multi-graph classification framework based on representative frequent approximate subgraph mining.

6.5 Methodology

In order to answer the research questions of this PhD proposal and to reach the general aim, we propose the following methodology.

1. Critical study of frequent approximate subgraph mining algorithms reported in the literature for:

   (a) Identifying properties from frequent approximate subgraph mining algorithms designed to work on simple-graphs, which could be reusable on multi-graphs.

   (b) Identifying an adequate type of patterns for reducing the dimensionality of the set of patterns computed by frequent approximate subgraph mining algorithms.

2. Designing an algorithm for frequent approximate subgraph mining in multi-graph collections. To achieve this objective, two research lines will be explored.

   (a) Transforming a multi-graph collection into a simple-graph collection to allow applying traditional frequent approximate subgraph mining algorithms, and then transforming a simple-graph collection into a multi-graph collection in order to perform the analysis of the results in terms of multi-graphs.

      i. Propose a way for transforming the multi-edges from a multi-graph into simple-edges.
      ii. Propose a way for transforming the loops from a multi-graph into simple-edges.
      iii. Integrate the proposed transformation ways into a method for transforming multi-graph collections into simple-graph collections, allowing applying traditional frequent approximate subgraph miners and returning the results to the multi-graph representation.
      iv. Demonstrate the theoretical correctness of the proposed transformations.

   (b) Computing frequent approximate subgraphs in multi-graph collections, directly into the mining process. In this case, different canonical forms will be explored for determining an appropriate way to represent multi-graphs into the mining process. We will explore the next canonical forms among others:

      i. Canonical form based on depth-first search (*DFS*) code.
      ii. Canonical form based on breadth-first search (*BFS*) code.
      iii. Canonical form based on canonical adjacency matrix (*CAM*) code.

3. Propose algorithms for computing representative frequent approximate subgraphs in multi-graph collections.

   (a) Propose a new algorithm for computing maximal frequent approximate subgraphs in multi-graph collections.

      i. Define maximal frequent patterns into the approximate context.
      ii. Identify all maximal patterns from all patterns computed by a traditional frequent approximate subgraph mining algorithm.

iii. Find a strategy to compute only maximal patterns into the mining process.

iv. Integrate the maximal pattern mining strategy with the approximate approach for mining maximal frequent approximate subgraphs directly into the mining.

(b) Propose a new algorithm for computing closed frequent approximate subgraphs in multi-graph collections.

i. Define closed frequent patterns into the approximate context.

ii. Identify all closed patterns from all patterns computed by a traditional frequent approximate subgraph mining algorithm.

iii. Find a strategy to compute only closed patterns into the mining process.

iv. Propose prune strategies for the graph search space taking into account the closed properties.

v. Integrate the closed pattern mining and the prune strategies with the approximate approach for mining closed frequent approximate subgraphs directly into the mining.

(c) Propose a new algorithm for computing clique frequent approximate subgraphs in multi-graph collections.

i. Define clique frequent patterns into the approximate context.

ii. Identify all clique patterns from all patterns computed by a traditional frequent approximate subgraph mining algorithm.

iii. Find a strategy to compute only clique patterns into the mining process.

iv. Integrate the clique strategy with the approximate approach for mining clique frequent approximate subgraphs directly into the mining.

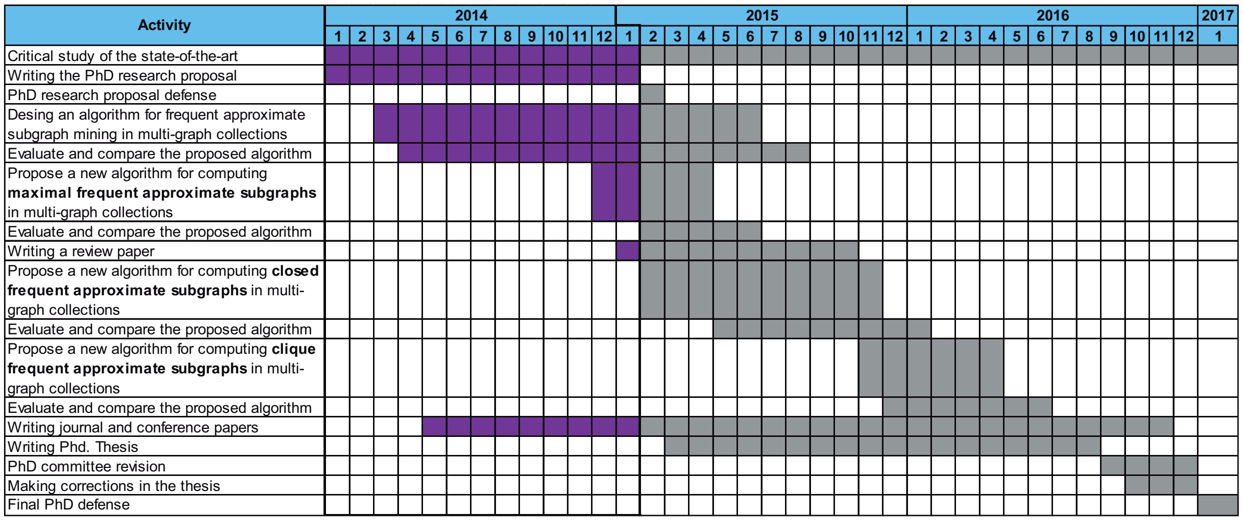
4. Experimental evaluation, on multi-graph collections, of the representative frequent approximate subgraphs.

(a) Propose a graph-based classification framework for working with representative frequent approximate subgraphs in multi-graph collections. In this framework, the frequent patterns identified by our proposed algorithms will be used for representing the original graph collection into the classification process.

(b) Use the proposed classification framework for evaluating the accuracy and performance of the representative subgraphs computed by our algorithms.

6.5.1 Schedule

To fulfill the objectives of this PhD research proposal, we propose the schedule shown in Figure 1. In this figure, each cell colored represents the time when the activity in the corresponding row will be performed. The cells colored in violet (dark) represent the activities that have already been carried out and the cells colored in gray (light) represent the activities to be held. The activities are specified in the first column of the table. The remaining columns specify the years subdivided in months.



Activity	2014												2015												2016												2017
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1
Critical study of the state-of-the-art	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Writing the PhD research proposal	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪																								
PhD research proposal defense														▨																							
Desing an algorithm for frequent approximate subgraph mining in multi-graph collections			🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	▨	▨	▨	▨	▨																			
Evaluate and compare the proposed algorithm				🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	▨	▨	▨	▨	▨	▨	▨																	
Propose a new algorithm for computing **maximal frequent approximate subgraphs** in multi-graph collections												🟪	🟪	▨	▨	▨																					
Evaluate and compare the proposed algorithm														▨	▨	▨	▨	▨																			
Writing a review paper													🟪	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Propose a new algorithm for computing **closed frequent approximate subgraphs** in multi-graph collections														▨	▨	▨	▨	▨	▨	▨	▨	▨	▨														
Evaluate and compare the proposed algorithm																	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Propose a new algorithm for computing **clique frequent approximate subgraphs** in multi-graph collections																							▨	▨	▨	▨	▨	▨									
Evaluate and compare the proposed algorithm																								▨	▨	▨	▨	▨	▨	▨							
Writing journal and conference papers					🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Writing Phd. Thesis															▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
PhD committee revision																																	▨	▨	▨	▨	
Making corrections in the thesis																																		▨	▨	▨	
Final PhD defense																																					▨

Figure 1. Activity schedule for fulfilling the objectives of this PhD proposal.

7 Preliminary Results

In this section, the preliminary results obtained so far are presented. Following the methodology above discussed, first we introduce a new method for mining frequent approximate subgraphs (FASs) in multi-graph collections, transforming a multi-graph collection into a simple-graph collection to allow applying traditional FAS mining algorithms, and then transforming a simple-graph collection into a multi-graph collection in order to get the results in terms of multi-graphs (see Section 7.1). Additionally, an extension of the canonical adjacency matrix (CAM), which is used for developing a new FAS miner for computing FASs from multi-graph collections, is also proposed (see Section 7.2).

7.1 Proposed transformation Method

Following the idea of transforming a multi-graph into a simple-graph [52, 8][2], we propose a method for transforming a multi-graph collection into a simple-graph collection without losing any topological or semantic information. This shall allow applying traditional FAS miners over the simplified graph collection. Later, the results are returned to multi-graphs; again without losing any topological or semantic information.

7.1.1 Processing multi-graphs

For transforming a multi-graph into a simple-graph, two steps are used: (1) transforming each loop into a simple-edge, and (2) transforming each multi-edge into two simple-edges. When a loop is found, it is changed by adding a new vertex with an especial label (κ) and an edge with the same loop label, which connects the new vertex with the one containing the loop. When a multi-edge e, where $\phi(e) = \{u, v\}$ and $u \neq v$, is found, it is changed by a new vertex with an especial label (ϱ) and two edges (e_1 and e_2) both with the label of e; the edge e_1 connects the new vertex with u and e_2 connects the new vertex with v. The special labels ϱ and κ, cannot be used in the multi-graph collection and cannot be replaced, during the FAS mining process, by any other, except by itself.

For transforming a multi-graph into a simple-graph, the following steps are performed by our algorithm, which we denote be $M2Simple$. This algorithm starts initializing an empty graph G', and then traverses the edges in the input multi-graph G. For each edge $e \in V_G$, the graph $G_e \leftarrow G[\{e\}]$ is calculated according to the definition 2.5, then the graph G'_e is computed and added to G', taking into account if e is a loop, or a multi-edge, or a simple-edge. Notice that for building a simple-graph only vertices with label κ and ϱ were added, including the simple-edges connecting such vertices.

In the same way that there is a transformation from a multi-graph into a simple-graph, it is desirable to obtain a multi-graph starting from their corresponding simple-graph (in our case a FAS computed by a FAS mining algorithm). In fact, this reversing process is required for doing compatible the mined FASs with the

[2]In [52, 8] the transformation is different to the one introduced in this research proposal, even more the transformation was done for finding the maximum number of link-disjoint paths and for solving a problem in production systems, i.e., these transformations were not proposed for mining frequent patterns in multi-graphs.

original multi-graph collection. The definition 7.1 expresses the conditions that a simple-graph must fulfil in order to be susceptible of being transformed to a multi-graph.

Definition 7.1 (Returnable graph)**.** Let κ and ϱ be two different vertex labels that will be used in two kind of vertices that will be used to represent all loops and all multi-edges, respectively. 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 $J'(e_1) = J'(e_2)$, ans
2. Each vertex $v \in V'$ with $I'(v) = \kappa$ has exactly one incident edge.

For transforming a returnable simple-graph into a multi-graph, the following steps are performed by our algorithm, which we denote be $S2Multi$. First of all, the ν-induced subgraph G of the input G' in the set $V' \setminus (V'_\varrho \cup V'_\kappa)$ is calculated, according to the definition 2.5. Next, the algorithm traverses the vertices with label ϱ for adding multi-edges to G. Finally, the algorithm traverses the vertices with label κ for adding loops to G. Notice that, for building generalizations, only vertices with label κ and ϱ were removed, including the simple-edges connecting such vertices.

In Figure 2, an example of the multi-graph/simple-graph transformations is shown. In this example we illustrate a special case of pattern which is not returnable. First, as it can be seen in this figure, G' and g' are multi-graphs and they could be transformed into G and g, respectively, using the previously explained M2Simple algorithm. In the same way, G and g, which are simple-graphs, could be transformed into G' and g', respectively (using S2Multi algorithm). The graph q, that appears in Figure 2, is not returnable according to the definition 7.1 because the vertex with label ϱ has only one adjacent edge.

The proposed transformation process, for multi-graph collections to simple-graph collections and vice versa, allows us applying any algorithm for traditional FAS mining.

The workflow of our method for processing multi-graph collections is shown in Figure 3. In Figure 3, first, given a multi-graph collection, by applying our proposed method the multi-graphs are transformed into simple-graphs. Later, the FASs are obtained using a FAS mining algorithm for simple-graphs. Then, by applying our method the obtained FASs are transformed into multi-graphs for obtaining the original multi-graph collection.

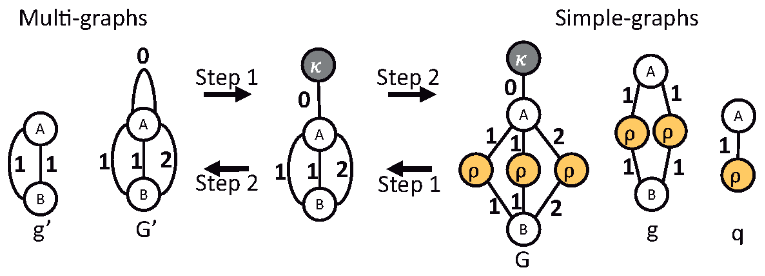


Figure 2. Examples of transformations between multi-graphs and simple-graphs, where the graph q is an example of a not returnable graph.

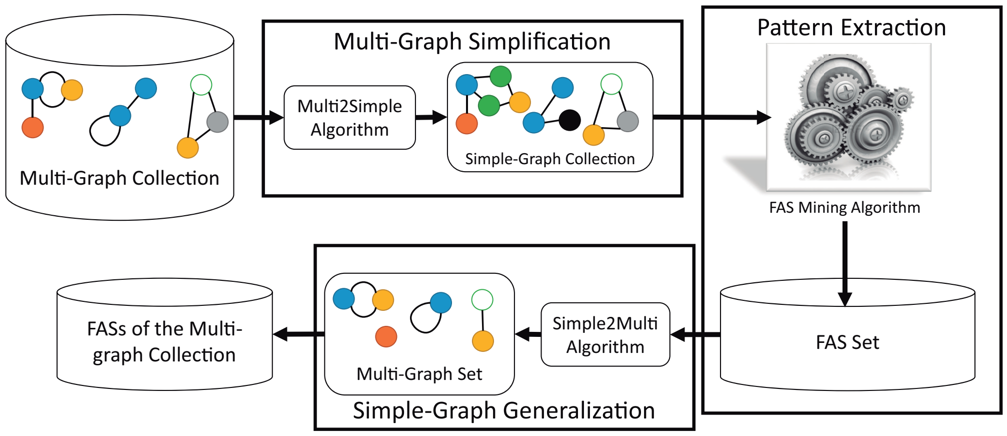


Figure 3. Proposed method for processing multi-graph collections.

7.1.2 Experimental Results

With the aim of studying the performance of the proposed method as well as its usefulness for image classification tasks, we present two experiments: first, the behaviour of our proposed method over synthetic databases is evaluated; then, the usefulness of the proposed method for image classification tasks is shown over real image databases represented as multi-graphs.

For the first experiment, several synthetic multi-graph collections are used for evaluating the performance of the proposed method. These synthetic databases were generated using the PyGen[3] graph emulation library. For building these synthetic multi-graph collections, we vary the values for three parameters: $|D|$

[3]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.

the number of multi-graphs to be generated, $|E|$ the number of edges and $|V|$ the amount of vertices into each multi-graph of the collection. Then, we assign a descriptive name for each synthetic database, for example, $D5kV1kE2k$ means that the collection has $|D| = 5000$, $|V| = 1000$ and $|E| = 2000$.

Three kinds of multi-graph collections were used for evaluating the performance of both algorithms (M2Simple and S2Multi algorithms). First, we fix $|D| = 5000$ and $|E| = 1000$, varying $|V|$ from 1000 to 5000, with increments of 1000. Next, we fix $|V| = 1000$, maintaining $|D| = 5000$ and varying $|E|$ from 1000 to 5000, with increments of 1000. Finally, we vary $|D|$ from 10000 to 50000, with increments of 10000, keeping $|V| = |E| = 1000$.

In Table 2, the performance, in terms of runtime, of both proposed transformation algorithms is shown. The results summarized in this table were achieved by transforming each multi-graph collection into a simple-graph collection (process denoted by M2Simple). Then, each obtained simple-graph collection was transformed into a multi-graph (process denoted by S2Multi). In this table, the first column shows the collection and the other two columns show the runtime in seconds of the proposed transformation algorithms.

Table 2. Runtime in seconds of our transformation method for several synthetic multi-graph collections.

Collection	M2Simple	S2Multi	Collection	M2Simple	S2Multi	Collection	M2Simple	S2Multi
D5kV1kE1k	33.43	40.05	D5kV1kE1k	33.43	40.05	D10kV1kE1k	68.36	75.21
D5kV2kE1k	45.35	51.16	D5kV1kE2k	56.86	68.07	D20kV1kE1k	134.01	146.53
D5kV3kE1k	56.20	59.58	D5kV1kE3k	80.16	87.41	D30kV1kE1k	204.57	229.09
D5kV4kE1k	66.32	73.02	D5kV1kE4k	106.08	120.00	D40kV1kE1k	269.25	285.31
D5kV5kE1k	78.06	85.00	D5kV1kE5k	133.37	148.29	D50kV1kE1k	343.59	369.06

As we can observe in Table 2, the proposed transformation algorithms are sensitive to the number of vertices and the amount of edges of the transformed graph. When a collection is processed, the number of graphs into the collection should be taken into account as another important parameter. As it can be seen in Table 2, our transformation algorithms are more sensitive to the number of edges than to the number of vertices. 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 multi-edge or loop.

For the second experiment, two real image databases (COIL [36] and ETH [30]) are used to show the usefulness of the patterns retrieved by our proposed transformation method. Both databases contain images of real objects taken from different viewpoints. In both cases, each image is represented as a multi-graph [35]. In COIL we use the same 25 objects used in [35]. This database is split into 198 (11%) images for training and 1602 (89%) for testing, as in [35] with an average graph size of 144 edges, this database contain an average of 19 multi-edges per graph. In ETH we use the same 6 categories employed in [34] (apples, cars, cows, cups, horses and tomatoes). This database is split into 615 (25%) images for training and 1845 (75%) for testing, as in [34] with an average graph size of 179 edges, this database contains an average of 25 multi-edges per graph.

In this experiment, we show the usefulness of our proposed method in the context of image classification where images are represented as multi-graphs. In this case, we use a classification framework similar of that proposed in [3], but we process multi-graph collections using our transformation method (see Figure 3). First, the multi-graphs are transformed into simple-graphs by applying M2Simple. Then, all FASs are computed by VEAM [3]. These FASs are returned to multi-graphs and they are used as attributes for building an attribute vector for each image. Each attribute vector represents an object (image) of the database, where each cell of the vector contains the similarity value of a FAS with regard to the multi-graph representing the image (using $maxID$, see Definition 2.14). Finally, once we have the vector representation of the images, a conventional classifier is applied.

In Table 3, the classification results of our method in two image databases, represented as multi-graph collections, are shown for different values for the support threshold δ. In this experiment, the value for the similarity threshold τ was fixed[4] to 0.66. In our classification experiments, several classifiers were used: Support Vector Machine (SVM), decision trees (J48graft), based on rules (Decision table), and regression (ClassificationViaRegression). All these classifiers were taken from Weka v3.6.6 [19] using the default parameters.

Table 3 includes two sub-tables with the accuracy and F-measure results, respectively. The first column of these sub-tables shows the database name and the second one shows the support threshold values used in each experiment, we used $\delta = 0.2$, 0.3, 0.4 and 0.5 for COIL and $\delta = 0.5$, 0.6, 0.7 and 0.8 for ETH.

[4]Note: We compute the mean of similarities obtained from the substitution matrices, for COIL and ETH databases obtaining 0.658932 and 0.6615642, respectively. Therefore, we rounded them to 0.66 for both databases.

Table 3. Classification results of Accuracy (a) and F-measure (b), using several classifiers and different δ values over multi-graph collections.

(a) Accuracy results (%)

Database	δ	J48graft	Decision Table	Regression	SVM
COIL	0.2	73.91	55.56	72.47	94.13
	0.3	79.03	50.25	79.46	90.32
	0.4	71.60	53.18	76.97	83.58
	0.5	73.35	58.68	73.91	74.72
ETH	0.5	61.13	60.00	58.59	67.48
	0.6	58.21	48.56	61.46	65.63
	0.7	61.95	44.28	58.86	64.93
	0.8	65.15	49.21	60.22	64.39

(b) F-measure results (%)

Database	δ	J48graft	Decision Table	Regression	SVM
COIL	0.2	72.30	55.80	71.30	94.00
	0.3	81.94	52.00	79.17	92.80
	0.4	75.18	54.90	77.85	84.85
	0.5	72.00	58.00	80.00	58.70
ETH	0.5	61.40	60.70	59.00	67.50
	0.6	58.00	48.50	62.40	66.10
	0.7	65.65	46.67	62.14	80.37
	0.8	66.28	48.70	56.82	69.76

The other four consecutive columns show the classification results (accuracy or F-measure) for the classifier specified at the top of these columns, using all FASs computed by the proposed method.

Our proposal is compared against the classification method proposed in [35], which does not use FAS mining techniques. In this case, our proposal obtained better results, since in [35], over the COIL database, an accuracy of 91.60 is reported while our method scored 94.13 with SVM using $\delta = 0.2$. Using the same method, in the case of ETH, the classification result reported is 88.00%, which is not improved by our method. In this case, we do not compare in terms of F-measure since [35] does not report results using this metric. This comparison cannot be made against the results reported in [5, 33], since they do not represent the image databases as multi-graph collections.

A paper reporting this preliminary result was sent to the Neurocomputing Journal under the title "*A New Method for FAS Mining in Multi-graph Collections*". Currently, this paper is under review.

7.2 Proposed Extension of CAM for Representing Multi-Graphs

The canonical adjacency matrix (CAM) introduced in this section allows us representing multi-graphs, including information of the multi-edges and loops into each cell. In this case, each cell of the adjacency matrix contains information of the corresponding vertex or edge or edges when there are multi-edges, or both information when the cell is in the diagonal and the vertex has loops. In this way, matrix dimensions are the same as those required for the simple-graph canonical adjacency matrix ($|V_G| \times |V_G|$). In definition 7.1, the proposed adjacency matrix for a multi-graph is introduced.

Definition 7.1 (Adjacency matrix for a labeled multi-graph)**.** Let G be a labeled multi-graph, let $v_i, v_j \in V_G$ two different vertices of G, let $LL(v_i)$ an ordered list of loop labels, where $LL(v_i) = \langle J_G(e_1), \ldots, J_G(e_{|LL(v_i)|}) \rangle$, such that $e_k \in E_G, \phi(e_k) = \{v_i\}$, and let $ML(v_i, v_j)$ be an ordered list of edge labels, where $ML(v_i, v_j) = \langle J_G(e_1), J_G(e_2), \ldots, J_G(e_{|ML(v_i,v_j)|}) \rangle$, such that $e_k \in E_G, \phi_G(e_k) = \{v_i, v_j\}$. The adjacency matrix $M = (m_{i,j})_{|V_G| \times |V_G|}$ for G is defined by:

$$m_{i,j} = \begin{cases} (I_G(v_i), LL(v_i)) & if\ i = j, LL(v_i) \neq \emptyset \\ I_G(v_i) & if\ i = j, LL(v_i) = \emptyset \\ ML(v_i, v_j) & if\ i \neq j, ML(v_i, v_j) \neq \emptyset \\ - & otherwise \end{cases} \tag{6}$$

The symbol "$-$" is used for representing edge absence (*i.e.* non-connected vertices).

In Definition 7.1, the conditions of Definition 3.2 are kept, but loops and multi-edges information is introduced for representing a multi-graph. However, for computing the CAM of a multi-graph, the comparisons concerning to the cells of the matrix cannot be between simple labels as in traditional adjacency matrices. In the case of multi-graph matrices, the comparison must be between cells as structures (vertex labels and loop label lists, and edge label lists in our case). Then, likewise a conventional adjacency matrix, but using its cells as structures, the CAM of a multi-graph can be computed.

In our case, we use the information of vertex degrees and vertex labels as vertex invariants, then the degree-based partition order is applied for computing the CAM of a multi-graph. First, the label list of each cell of the matrix is sorted following a lexicographical order. Next, we assign each cell of the matrix diagonal to a partition according to the vertex label. Then, we apply a second partitioning process, where for each partition we assign each element to a sub-partition according to the vertex degree. Later, with the aim of following an ordering criterion, we sort each partition according to a lexicographical order for vertex labels and in descendant form for the vertex degrees used in each sub-partition. Next, the constrained-partition permutations are used for obtaining the CAM.

The algorithm for computing the CAM code for a given multi-graph follows the idea of the algorithm "ComputeCAM" used in VEAM and described in Section 3.6, but in this case the loop and multi-edge information plays an important role. As it was shown in Algorithm 4, several restrictions are included for taking into account multi-edges and loops. The first restriction can be seen in line 2 of Algorithm 4, where the amount of loops of each vertex is used as a third comparison criteria. Another difference of this algorithm regarding Algorithm 3 can be found in lines 9 and 10, where the arrays X and Y, which contain lists of multi-edge labels, are built by means of Definition 7.1. Notice that, in this case, each element of X and Y are lists of multi-edge labels instead of only one edge label. In addition, the lists of loop labels for the comparison of vertices are stored in Xl and Yl. Then, the multi-edge and loop information is used following a lexicographical order for sorting the tuples into each partition of P. Finally, the CAM code CAM_G can be obtained for a given multi-graph G, taking into account the final order of P.

7.2.1 A Miner Based on the New CAM for Multi-Graph Collections

The algorithm for mining frequent approximate subgraphs (FASs) proposed in this section, called *MgVEAM*, is an extension of the VEAM algorithm. The MgVEAM algorithm is designed for computing the FASs over a multi-graph collection, following a pattern grown approach. This algorithm allows us to compute FASs directly from multi-graph collections using Definition 7.1 for representing the identified cansidate subgraphs. MgVEAM uses substitution matrices for specifying replacement criteria between vertex and edge labels, allowing variations into these labels keeping graph topology.

Once described the needed definitions (i.e., a labeled graph, similarity function between multi-graphs[5] based on substitution matrices, approximate support, and adjacency matrix of a labeled multi-graph, as well the CAM code of a multi-graph), we can describe our algorithm MgVEAM.

[5]The similarity function between multi-graphs is the same sim function (see Definition 2.11), but applied on the multi-graph context.

Algorithm 4: $ComputeMgCAM$

```
Input: $G$ : A cansidate multi-subgraph, $e'$ : The last edge added into $G$.
Output: $CAM_G$ : The CAM code for the cansidate multi-subgraph $G$.
1  $P \leftarrow$ Partition list of $v \in V_G$ such that: $v_i, v_j \in P_k$, have the same label and degree, where $i \neq j$ and
   $P_k \in P$;
2  $P$ is lexicographically sorted $[(l_{i-1}, d_{i-1}, nl_{i-1}) \succ (l_i, d_i, nl_i)]$, being $l_k, d_k$ and $nl_k$ the label and the
   degree of $P_k$, and the amount of loops for each vertex $v_j \in P_k$;
3  foreach $P_k = \{p_1, \ldots, p_{|P_k|}\} \in P$ do
4      $l = |P_k|$;
5      $l_k$ is the label for $P_k$;
6      while $l \neq 1$ do
7          $newl = 1$;
8          foreach $i = \{2, \ldots, |P_k|\}$ do
9              $X = (x_{(1,1)}, \ldots, x_{(|P_1|,1)}, \ldots, x_{(1,k)}, \ldots, x_{(i-1,k)})$ is computed based on Definition 7.1,
               being $x_{(j,w)}$ the descendant sorted multi-edge label list between the vertex in $p_{i-1}$ and the
               vertex in $p_j \in P_w \in P$;
10             $Y = (y_{(1,1)}, \ldots, y_{(|P_1|,1)}, \ldots, y_{(1,k)}, \ldots, y_{(i-2,k)}, y_{(i,k)})$ is computed based on
               Definition 7.1, being $y_{(j,w)}$ the descendant sorted multi-edge label list between the vertex
               in $p_i$ and the vertex in $p_j \in P_w \in P$;
11             $Xl$ and $Yl$ are the loop label list of vertices in $p_{i-1}$ and $p_i$, respectively;
12             if $X < Y$ or $(X = Y$ ans $Xl < Yl)$, following a lexicographical order then
13                 Swap($p_{i-1}$, $p_i$);
14                 $newl = i$;
15         $l = newl$;
16 The adjacency matrix $M$ of $G$ is built sorting its cells following the lexicographical order achieved in
   $P$;
17 The CAM code is obtained from $M$ and it is stored into $CAM_G$;
```

The “Main”, “Search”, and “GenCansidate” pseudo-codes of MgVEAM are the same used in VEAM, but, in our case, the function “ComputeMgCAM” is used. As it can be seen, we were able to maintain the same flow of VEAM for computing FASs on multi-graph collections; since only extending the canonical form representation for simple-graphs to multi-graphs was required to do this task. Then, an advantage of our proposal is that our canonical form can be used for other FAS mining algorithm without critical changes into its code to allow it to process multi-graphs.

7.2.2 Experimental Results

For evaluating the based on the extended CAM algorithm introduced in Section 7.2.1, we present two experiments: first, the performance of our proposal is evaluated on synthetic databases; then, the usefulness of the patterns identified by our proposed algorithm for image classification tasks is shown on image databases, where images are represented as multi-graphs.

For our first experiment, as in Section 7.1.2, several synthetic multi-graph collections are used for evaluating the behavior of the proposed algorithm for different values of its parameters, δ and τ (in terms of runtime and number of identified patterns). These synthetic collections were generated using the PyGen graph emulation library. For obtaining the synthetic multi-graph collections, we vary the values for the parameter $|D|$ (the number of multi-graphs to be generated) from 100 to 1000, with an increment of 300, maintaining $|V| = 20$ and $|E| = 100$. Then, we assign a descriptive name for each synthetic database, for example, $D1kV20E100$ means that the collection has $|D| = 1000$, $|V| = 20$ and $|E| = 100$.

In Table 4, the results, in terms of runtime and number of identified FASs, of the proposed algorithm are shown. These results show: the number of identified FASs and the runtime between parenthesis. This table has two sub-tables; the first one shows the runtime and number of identified FASs of MgVEAM varying the support threshold δ, and the second one shows the runtime and number of identified FASs of MgVEAM varying the similarity threshold τ. Each sub-table is split into five columns, where the first one shows the collections and the other consecutive four columns show the results for the different threshold values specified at the top of the column.

As we can see in Table 4, MgVEAM is more sensitive to the similarity threshold than to the support threshold. However, a decrement on the support threshold produces an increment of the number of identified patterns, as well as an increment in the runtime of the algorithm.

In our second experiment, an image database generated with the Ransom image generator of Coenen[6], is used to show the usefulness of the patterns identified by our proposal. We ransomly generate 1000 images with two classes. These images were ransomly divided into two sub-sets: one for training with 700 (70%) images and another for testing with 300 (30%) images. For showing the usefulness of the patterns obtained by our proposal, the MgVEAM algorithm is used in a multi-graph database (*CoenenDB-Multi*)

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

Table 4. Performance evaluation (in terms of runtime and number of identified patterns) of the MgVEAM algorithm over synthetic multi-graph collections. The number of patterns is shown followed by the runtime values between parenthesis. The terms "h", "m" and "s" means hours, minutes and seconds, respectively.

	Varying support threshold with the similarity threshold $\tau = 0.4$			
Collection	0.09	0.08	0.07	0.06
D100V20E100	190866 (10.26m)	239273 (12.38m)	303210 (16.00m)	480984 (25.24m)
D400V20E100	217240 (32.47m)	263581 (37.47m)	293389 (40.25m)	312782 (41.33m)
D700V20E100	216993 (56.24m)	262620 (1.05h)	294343 (1.13h)	313813 (1.16h)
D1kV20E100	231996 (1.50h)	271869 (2.09h)	298471 (2.15h)	316072 (2.22h)
	Varying similarity threshold with the support threshold $\delta = 0.06$			
Collection	0.7	0.6	0.5	0.4
D100V20E100	16614 (21.07s)	16614 (21.07s)	412425 (20.17m)	480984 (25.24m)
D400V20E100	5878 (12.52s)	5878 (12.52s)	284462 (33.09m)	312782 (41.33m)
D700V20E100	5858 (19.02s)	5858 (19.02s)	281776 (54.23m)	313813 (1.16h)
D1kV20E100	5868 (15.14s)	5868 (15.14s)	288008 (1.41h)	316072 (2.22h)

with multi-edges. Also, the results obtained by using the VEAM algorithm results in simple-graph databases (*CoenenDB-Angle* and *CoenenDB-Distance*) is shown. This last approach shows the solution to the problem considering just one edge of the multi-graph and applying a FAS mining algorithm for simple-graphs. It would be a way for solving the problem without our proposal. For these experiments, two well-known classifiers are used: SVM with a polynomial kernel and J48graft; taken from Weka v3.6.6 [19] using the default parameters.

From the image database we built the graph databases using the quad-tree approach described in [3], as follows:

- *CoenenDB-Angle*: the angles between vertices are used as edge labels for a simple-graph representation. This collection comprises 21 vertex labels, 24 edge labels and the average graph size is 135 edges.
- *CoenenDB-Distance*: the distance between vertices are used as edge labels for a simple-graph representation. This collection comprises 21 vertex labels, 24 edge labels and the average graph size is 135 edges.
- *CoenenDB-Multi*: both (angles and distance) values are used as labels for two multi-edge, respec-

tively, between the vertices representing each image as a multi-graph. This collection comprises 21 vertex labels, 48 edge labels and the average graph size is 270, where all edges are multi-edges.

The FASs computed by VEAM [3] on CoenenDB-Angle and CoenenDB-Distance collections, and the FASs computed by MgVEAM on CoenenDB-Multi collection are used as attributes for image classification. The classification results are shown in Table 5. In this experiment, the value of τ was fixed to 0.4 as proposed in [3] for this image collection.

Table 5. Classification results of Accuracy (a) and F-measure (b), using two classifiers and different δ values over the Coenen image collection.

(a) Accuracy results (%).

	CoenenDB-Angle		CoenenDB-Distance		CoenenDB-Multi	
Support	SVM	J48graft	SVM	J48graft	SVM	J48graft
0.8	78.33	78.33	78.33	78.33	78.33	78.33
0.7	**79.00**	77.67	78.33	76.33	78.67	**78.67**
0.6	85.00	79.67	90.33	91.33	**92.33**	**92.33**
0.5	89.00	89.00	90.33	**93.33**	**92.33**	**93.33**
0.4	90.67	89.67	89.67	91.27	**90.90**	**91.65**
Average	*84.40*	*82.87*	*85.40*	*86.12*	***86.51***	***86.86***

(b) F-measure results (%).

	CoenenDB-Angle		CoenenDB-Distance		CoenenDB-Multi	
Support	SVM	J48graft	SVM	J48graft	SVM	J48graft
0.8	68.90	68.90	68.90	68.90	68.90	68.90
0.7	71.23	74.13	68.90	72.37	**80.00**	**77.46**
0.6	84.85	77.15	89.97	90.85	**90.98**	**91.76**
0.5	87.55	88.17	89.97	92.59	**91.13**	**93.00**
0.4	89.31	89.27	89.23	90.13	**90.22**	**91.59**
Average	*80.37*	*79.52*	*81.39*	*82.97*	***84.25***	***84.54***

Table 5 is compound by two sub-tables with the results of the accuracy and F-measure, respectively. The first column of each sub-table shows the values used for the support threshold. The other three consecutive pairs of columns show the classification results using the graph collection specified at the top of these columns. These last columns show the classification results (accuracy or F-measure) obtained using the classifier specified on top of each column.

The classification results achieved (see Table 5) over the image collection show the usefulness of the patterns computed by our proposal, where in almost all the cases, the best classification results were obtained using our method for multi-graphs. Furthermore, by means of these results, we can show the usefulness of the multi-graph representation regarding the simple-graph one in this image database. As it can be seen in this table, the combined information of both kinds of edge labels (angle and distance) allows us to obtain better classification quality using the multi-graph representation than using this information individually.

Currently, we are preparing a paper for reporting this algorithm in a Journal, under the title "*An Extension of Canonical Adjacency Matrices for Frequent Subgraph Mining on Multi-graph Collections*".

8 Conclusions

Frequent approximate subgraph (FAS) mining is a widely used technique in Data Mining applications where there is some distortion into the data. However, FAS mining in multi-graph collections, remains a challenge. Moreover, usually a large number of frequent patterns is computed by the FAS mining algorithms. Using only representative patterns instead of using all the patterns is a technique that can be used to reduce the dimensionality of the set of patterns computed by a FAS mining algorithm. Therefore, the aim of this PhD research is to propose algorithms for mining FASs and representative FASs from multi-graph collections.

As preliminary results, we introduce a method for computing FASs in multi-graph collections. This method includes a way for transforming a multi-graph collection into a simple-graph collection, then the mining is performed over this collection and later, an inverse transforming process is applied to return the mining results (simple-graphs) to multi-graphs, without losing any topological or semantic information. The performance of this proposed method is evaluated over several synthetic graph databases. Based on these experiments, our proposal for transforming multi-graphs to simple-graphs and vice versa is able to process multi-graph collections in a reasonable time. Furthermore, the usefulness of our transformation method for image classification has been shown. It is important to highlight that before our proposed method the application of FAS miners in multi-graphs would be infeasible. And, as it was commented in the experimental section, in some cases the results obtained by our proposal outperform those results obtained by state-of-the-art classifiers non-based on FAS.

In addition, an extension of the canonical form based on canonical adjacency matrix (CAM) representation is proposed. Using the proposed CAM we were able to process multi-graph collections extending an state-of-the-art algorithm for mining FASs. Our proposal has as an important advantage, that the proposed extended CAM can be used for extending other FAS miner to allow them to process multi-graph collections without requiring critical changes into the mining algorithms. Finally, based on our experiments, the usefulness of the use of multi-graph representation has been shown in an image classification task. Furthermore, as it was commented in the experimental section, in most of the cases the results achieved by our proposal, which identifies FASs over multi-graph collections, outperform those results obtained by a method that computes FASs on simple-graph collections.

Based on the preliminary results we can conclude that our goals are achievable following the proposed methodology, within the time defined by the Computational Sciences Coordination for a PhD research.

References

[1] N. Acosta-Mendoza. Clasificación de imágenes basada en subconjunto de subgrafos frecuentes aproximados. Master's thesis, The National Institute of Astrophysics, Optics and Electronics of Mexico (INAOE), July 2013.

[2] N. Acosta-Mendoza, A. Gago-Alonso, J.A. Carrasco-Ochoa, J.Fco. Martínez-Trinidad, and J.E. Medina-Pagola. Feature Space Reduction for Graph-Based Image Classification. In *Proceedings of the 18th Iberoamerican Congress on Pattern Recognition (CIARP'13)*, volume Part I, LNCS 8258, pages 246–253, Havana, Cuba, november 2013. Springer-Verlag Berlin Heidelberg.

[3] 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.

[4] N. Acosta-Mendoza, A. Gago-Alonso, and J.E. Medina-Pagola. On speeding up frequent approximate subgraph mining. In *Proceedings of the 17th Iberoamerican Congress on Pattern Recognition (CIARP'12)*, volume LNCS 7441, pages 316–323. Buenos Aires, Argentina, Springer-Verlag Berlin Heidelberg, 2012.

[5] N. Acosta-Mendoza, A. Morales-González, A. Gago-Alonso, E.B. García-Reyes, and J.E. Medina-Pagola. Classification using frequent approximate subgraphs. In *Proceedings of the 17th Iberoamerican Congress on Pattern Recognition (CIARP'12)*, volume LNCS 7441, pages 292–299. Buenos Aires, Argentina, Springer-Verlag Berlin Heidelberg, 2012.

[6] N.B. Aoun, M.M., and C.B. Amar. Bag of sub-graphs for video event recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Florence, Italy,*, pages 1547–1551, 2014.

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

[8] I. Boneva, F. Hermann, H. Kastenberg, and A. Rensink. Simulating multigraph transformations using simple graphs. *Electronic Comunications of the EASST*, 6, 2007.

[9] C. Borgelt. Mining molecular fragments: Finding relevant substructures of molecules. In *Proc. IEEE International Conference on Data Mining (ICDM)*, pages 51–58, Maebashi City, Japan, 2002. IEEE Press.

[10] C. Chen, C.X. Lin, X. Yan, and J. Han. On effective presentation of graph patterns: a structural representative approach. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 299–308, 2008.

[11] C. Chen, X. Yan, F. Zhu, and J. Han. gApprox: Mining Frequent Approximate Patterns from a Massive Network. In *International Conference on Data Mining (ICDM'07)*, pages 445–450, 2007.

[12] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298, 2004.

[13] 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*, 2014.

[14] M. Flores-Garrido, J.A. Carrasco-Ochoa, and J.Fco. Martínez-Trinidad. Mining maximal frequent patterns in a single graph using inexact matching. *Knowledge-Based Systems*, 66:166–177, 2014.

[15] A. Gago-Alonso, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, and J.Fco. Martínez-Trinidad. Full Duplicate Candidate Pruning for Frequent Connected Subgraph Mining. *Integrated Computer-Aided Engineering*, 17:211–225, August 2010.

[16] A. Gago-Alonso, J.E. Medina-Pagola, J.A. Carrasco-Ochoa, and J.Fco. Martínez Trinidad. Mining Frequent Connected Subgraphs Reducing the Number of Candidates. In *Machine Learning and Knowledge Discovery in Databases, European Conference, (ECML/PKDD)*, volume 5211 of *Lecture Notes in Computer Science*, pages 365–376. Springer, 2008.

[17] A. Gago-Alonso, A. Puentes-Luberta, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, and J.Fco. Martínez-Trinidad. A new algorithm for mining frequent connected subgraphs based on adjacency matrices. *Intelligent Data Analysis*, 14:385–403, 2010.

[18] 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*, pages 377–381, Key West, Florida, USA, 2001. AAAI Press.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11, 2009.

[20] L.B. Holder, D.J. Cook, and H. Bunke. Fuzzy substructure discovery. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 218–223, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[21] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *The 3rd IEEE International Conference on Data Mining*, pages 549–552, FL, 2003. Melbourne.

[22] A. Inokuchi, T. Washio, K. Nishimura, and H. Motoda. A fast algorithm for mining frequent connected subgraphs. In *Technical Report RT0448, In IBM Research*, Tokyo Research Laboratory, 2002.

[23] J. Jia, N. Yu, X. Rui, and M. Li. Multi-Graph Similarity Reinforcement for Image Annotation Refinement. In *International Conference on Image Processing*, pages 993–996. December 29, 2008.

[24] 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.

[25] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *Knowledge Engineering Review*, 2012.

[26] D. Jiang and J. Pei. Mining frequent cross-graph quasi-cliques. *ACM Transactions Knowledge Discovery from Data*, 2(4):16:1–16:42, 2009.

[27] B. Kimelfeld and P.G. Kolaitis. The complexity of mining maximal frequent subgraphs. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013*, pages 13–24. ACM, 2013.

[28] D. Knisley, J. Knisley, C. Ross, and A. Rockney. Classifying multigraph models of secondary rna structure using graph-theoretic descriptors. *International Scholarly Research Network (ISRN) Bioinformatics*, Article ID 157135, 2012.

[29] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. Technical report, IEEE Transactions on Knowledge and Data Engineering, 2002.

[30] 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, 2003.

[31] J. Li, Z. Zou, and H. Gao. Mining frequent subgraphs over uncertain graph databases under probabilistic semantics. *VLDB J.*, 21(6):753–777, 2012.

[32] G. Liu and L. Wong. Effective Pruning Techniques for Mining Quasi-Cliques. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD, Antwerp, Belgium, Proceedings*, Part II, pages 33–49, 2008.

[33] 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.

[34] 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.

[35] 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*, pages 1–23, 2011.

[36] 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.

[37] S. Nijssen and J.N. Kok. A Quickstart in Frequent Structure Mining can make a Difference. In *The 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 647–652, New York, NY, USA, 2004. ACM.

[38] T. Ozaki and M. Etoh. Closed and maximal subgraph mining in internally and extermally weighted graph databases. In *Proceedings of the IEE Workshops of International Conference on Adbanced Information Networking and Applications.*, pages 626–631. IEEE Computer Society., 2011.

[39] P.M. Pardalos and S. Rebennack. Computational Challenges with Cliques, Quasi-cliques and Clique Partitions in Graphs. In P. Festa, editor, *SEA*, volume 6049 of *Lecture Notes in Computer Science*, pages 13–22. Springer, 2010.

[40] K. Riesen and H. Bunke. IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. pages 208–297. Orlando, USA, 2008.

[41] J. Rissanen. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1989.

[42] Y. Song and S.S. Chen. Item sets based graph mining algorithm and application in genetic regulatory networks. *Data Mining, IEEE International Conference on Volume, Issue*, pages 337–340, 2006.

[43] L. Speičys and C.S. Jensen. Enabling location-based services - multi-graph representation of transportation networks. *GeoInformatica*, 12:219–253, May 2008.

[44] M. Stikic, D. Larlus, and B. Schiele. Multi-graph based semi-supervised learning for activity recognition. *International Symposium on Wearable Computers*, pages 85–92, Jan 2009.

[45] I. Takigawa and H. Mamitsuka. Efficiently Mining δ-tolerance Closed Frequent Subgraphs. *Machine Learning*, 82(2):95–121, 2011.

[46] C. Taranto, N. Di Mauro, and F. Esposito. Uncertain (multi)graphs for personalization services in digital libraries. *In Digital Libraries and Archives - 8th Italian Research Conference (IRCDL)*, 354 of Communications in Computer and Information Science:141–152, 2012.

[47] L. Thomas, S. Valluri, and K. Karlapalem. ISG: Itemset based subgraph mining. *Technical Report*, IIIT, Hyderabad, December 2009.

[48] C.E. Tsourakakis. A Novel Approach to Finding Near-Cliques: The Triangle-Densest Subgraph Problem. *CoRR*, abs1405.1477, 2014.

[49] C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Chi. Scalable Mining of Large Disk-based Graph Databases. In *Proc. of the 2004 ACM SIGKDD of International Conference on Knowledge Discovery in databases (KDD)*, pages 316–325. Seattle, WA, 2004.

[50] M. Wang, X.S. Hua, X. Yuan, Y. Song, and L.R. Dai. Multi-graph semi-supervised learning for video semantic feature extraction. *International Conference on Multimedia and Expo*, pages 1978–1981, July 2007.

[51] M. Wang, X.S. Hua, X. Yuan, Y. Song, and L.R. Dai. Optimizing Multi-Graph Learning: Towards A Unified Video Annotation Scheme. In *ACM Multimedia*, pages 862–871. September 23, 2007.

[52] 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.

[53] J. Wu, X. Zhu, C. Zhang, and Z. Cai. Multi-instance Multi-graph Dual Embedding Learning. In *13th International Conference on Data Mining*, pages 827–836. January 1, 2013.

[54] J. Wu, X. Zhu, C. Zhang, and P.S. Yu. Bag constrained structure pattern mining for multi-graph classification. *IEEE Transactions on Knowledge and Data Engineering*, 2013.

[55] Y. Xiao, W. Wu, W. Wang, and Z. He. Efficient Algorithms for Node Disjoint Subgraph Homeomorphism Determination. In *Proceedings of the 13th international conference on Database systems for advanced applications*, pages 452–460, New Delhi, India, 2008. Springer-Verlag, Berlin, Heidelberg.

[56] Y. Xu, J. Cheng, A. Wai-Chee Fu, and Y. Bu. Distributed Maximal Clique Computation. In *2014 IEEE International Congress on Big Data*, pages 160–167. IEEE, 2014.

[57] X. Yan and J. Han. ClosedGraph: Mining Closed Frequent Graph Patterns. In *Proc. of the 9th ACM SIGKDD of International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 286–295. Washington, DC, 2003.

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

[59] S. Zhang and J. Yang. RAM: Randomized Approximate Graph Mining. In *The 20th International Conference on Scientific and Statistical Database Management*, pages 187–203, China, 2008. Hong Kong.

[60] 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'07)*, volume 4426 of *Lecture Notes in Computer Science*, pages 388–400. Springer, 2007.

[61] Z. Zou, J. Li, H. Gao, and S. Zhang. Finding top-k maximal cliques in an uncertain graph. In *IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 649–652, 2010.