# MEJORA PARA LA MINERÍA DE SUBGRAFOS FRECUENTES APROXIMADOS MEDIANTE LA REDUCCIÓN DEL ESPACIO DE BÚSQUEDA

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

[a]{nacosta, agago, jmedina}@cenatav.co.cu
*Centro de Aplicación de Tecnologías de Avanzada, Cuba*
*La Habana*
*(Minería de Datos y Textos)*

**Resumen**

La minería de subgrafos frecuentes aproximados (MSFA) se ha convertido en una interesante tarea aplicable en varios dominios de la ciencia. Gran parte de los estudios previos se han enfocado en la reducción del espacio de búsqueda o en la optimización de las pruebas de formas canónicas (FC). En este trabajo, se proponen dos podas que permiten disminuir la cantidad de subgrafos candidatos en la MSFA. Estas permiten reducir el número de pruebas de FC, las cuales han sido utilizadas para la detección de duplicados, sin embargo, estas pruebas afectan la eficiencia del proceso de minería debido a su alta complejidad computacional. Las podas propuestas se introducen en el proceso de minería de un algoritmo del estado del arte para realizar de manera eficiente este proceso. El comportamiento de este algoritmo con las podas es comparado con su original utilizando varias colecciones artificiales de imágenes.

*Palabras claves:* Minería de grafos aproximados, cotejo de grafos aproximados, subgrafos frecuentes aproximados, grafos etiquetados.

**Abstract**

Frequent approximate sugbraph mining (FASM) has become an interesting task with wide applications in several domains of the science. Most of the previous studies have been focused on reduction of search space or optimizing the canonical form (CF) tests. In this paper, two prunes are proposed, which allow decrease the number of candidate subgraphs in FASM. These prunes allow reduce the number of CF tests that has been used for the duplicates detection, however, these tests affects the efficiency of mining process because have high computational complexity. The prunes proposed are inserted in mining process of an algorithm of related work to perform efficiently this process. The performance of this algorithm is compared against the original over several artificial collections of images.

*Keywords:* Approximate graph mining, approximate graph matching, frequent approximate subgraphs, labeled graphs.

## 1. Introducción

En los últimos años ha aumentado la necesidad del uso de cotejo aproximado en tareas de minería de subgrafos frecuentes [3, 7, 10, 11]. Esta necesidad se debe a que existen problemas concretos donde los subgrafos de interés tienen ligeras diferencias en los datos [6]. Un ejemplo de estas diferencias se puede observar en el análisis de proteínas dadas las mutaciones de estructuras similares por su evolución en varias proteínas. Otro ejemplo está en el procesamiento de imágenes, donde estas diferencias se deben a ruidos y distorsiones, o simplemente se observan diferencias espaciales

*Autor correspondiente

entre instancias de un mismo objeto. Esto significa que es necesario tolerar desajustes de vértices o aristas, cierto nivel de distorsiones geométricas o ligeras variaciones semánticas en la búsqueda de subgrafos patrones frecuentes.

Por esta razón, se ha convertido en una necesidad el evaluar la similitud entre grafos permitiendo diferencias estructurales, es decir, técnicas de cotejo aproximado. El cotejo aproximado consiste en encontrar una correspondencia entre vértices o aristas de dos grafos para determinar su similitud permitiendo diferencias entre su estructura o etiquetas.

Varios autores han expresado la necesidad del uso de cotejo aproximado de grafos en la minería de subgrafos frecuentes sobre colecciones de grafos [3, 7, 10, 11]. Estos autores defienden la idea de que se pudieran encontrar subgrafos frecuentes y de mayor interés para aplicaciones o usuarios. Angryk y Hossain recomendaron la detección de subgrafos frecuentes usando cotejo aproximado en tareas de agrupamiento de documentos [7]. Estos autores consideran que es posible obtener un mejor agrupamiento mediante la identificación de patrones que permitan algunas variaciones topológicas o semánticas. Estos tipos de patrones frecuentes han sido recomendados en tareas de procesamiento de datos químicos [3], en tareas de análisis de vínculos [10], y han sido presentados como un problema abierto en tareas de procesamiento de datos moleculares y redes sociales [11].

En respuesta a esta necesidad, varios algoritmos han sido desarrollados para la minería de subgrafos frecuentes utilizando cotejo de grafo aproximado en diferentes dominios de la ciencia, tales como: clasificación de imágenes [1], análisis de estructuras bioquímicas [4, 9, 14, 17, 18], redes genéticas [12]; análisis de circuitos, redes sociales y vínculos [6].

La aparición de candidatos duplicados durante el proceso de minería es uno de los mayores problemas en la mayoría de los enfoques recientes. Un candidato duplicado es un subgrafo que fue considerado en pasos previos, pero aparece nuevamente a partir de varios subgrafos frecuentes durante la búsqueda. El problema de los duplicados se trata representando el subgrafo con un código único conocido como forma canónica (FC) y realizando pruebas de FC, sin embargo, estas pruebas de FC tienen una gran complejidad computacional [2].

En este trabajo, se introducen varias podas en un esquema aproximado para realizar la MFSA eficientemente. Mediante estas podas se logra reducir el espacio de búsqueda y la cantidad de pruebas de FC.

Este trabajo está organizado de la siguiente manera. En la sección 2 se presentan algunos conceptos básicos; en este también se encuentra el estado del arte, la descripción de un método aproximado y se define el problema de la MSFA. Las optimizaciones se proponen en la sección 3; además, en esta sección se describen las mejoras mediante el pseudo-código del algoritmo. Seguidamente, en la sección 4 se presentan los resultados experimentales de este trabajo, donde se puede observar que con el uso de las podas propuestas se logra una mejorar en eficiencia el proceso de MSFA. Finalmente, las conclusiones de esta investigación y algunas ideas de trabajo futuros son expuestas en la sección 5.

## 2. Marco teórico

En esta sección se comenzará con la explicación de los conceptos básicos y las notaciones utilizadas a lo largo de este trabajo. Se presentan los algoritmos más relevantes del estado del arte y se describen de manera general los enfoques cercanos al nuestro. Luego, se muestra la función de similitud del algoritmo para la MSFA utilizado como base en este trabajo. Finalmente, se plantea el problema de la MSFA.

### *2.1. Conceptos básicos*

Este trabajo es enfocado en grafos etiquetados simples y no dirigidos. En lo adelante cuando se hable de grafo se suponen todas estas características y en otro caso se especificará explícitamente. Antes de presentar su definición formalmente, se define el dominio de etiquetas.

Sean $L_V$ y $L_E$ conjuntos de etiquetas, donde $L_V$ es un conjunto de etiquetas de vértices y $L_E$ es un conjunto de etiquetas de aristas, el dominio de todas las posibles etiquetas es denotado por $L = L_V \cup L_E$.

Un *grafo etiquetado* en $L$ es una 4-tupla, $G = (V, E, I, J)$, donde $V$ es un conjunto en el que sus elementos son conocidos como *vértices*, $E \subset \{\{u, v\} \mid u, v \in V, u \neq v\}$ es un conjunto en el que sus elementos son conocidos como *aristas* (la arista $\{u, v\}$ conecta el vértice $u$ con el vértice $v$), $I : V \rightarrow L_V$ es una *función etiquetadora* que asigna etiquetas a los vértices y $J : E \rightarrow L_E$ es una *función etiquetadora* que asigna etiquetas a las aristas.

Sean $G_1 = (V_1, E_1, I_1, J_1)$ y $G_2 = (V_2, E_2, I_2, J_2)$ dos grafos etiquetados en $L$, se dice que $G_1$ es un *subgrafo* de $G_2$ si $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, $\forall u \in V_1, I_1(u) = I_2(u)$ y $\forall e \in E_1, J_1(e) = J_2(e)$. En este caso, se usa la notación $G_1 \subseteq G_2$ y se dice que $G_2$ es un *supergrafo* de $G_1$.

Dados dos grafos $G_1 = (V_1, E_1, I_1, J_1)$ y $G_2 = (V_2, E_2, I_2, J_2)$ etiquetados en $L$, donde $u, v \in V_2$, $u \in V_1$ y $\{u, v\} \in E_2$, se dice que $G_2$ es un hijo de $G_1$ si:

*) $V_2 = V_1 \cup \{v\}$,
*) $E_1 = E_2 \setminus \{\{u, v\}\}$,
*) $\forall u \in V_1, I_2(u) = I_1(u)$,
*) $\forall e \in E_1, J_2(e) = J_1(e)$.

En este caso, se dice que $G_1$ es padre de $G_2$ y $e = \{u, v\}$ es una *extensión* de $G_1$. Este hecho puede ser denotado por $G_2 = G_1 \diamond e$. Por tanto, el *conjunto de extensiones* de $G_1$ es denotado por $ExtSet(G_1) = \{e \mid G_1 \diamond e \text{ es un hijo de } G_1\}$. Se dice que $e$ es una *extensión hacia atrás*, denotada por $G_2 = G_1 \diamond_b e$, si $v \in V_1$, en otro caso se dice que es una *extensión hacia delante* (esta extiende el conjunto de vértices de $G_1$), denotado por $G_2 = G_1 \diamond_f e$.

En la minería de grafos sobre colecciones de grafos etiquetados, los candidatos duplicados son detectados realizando pruebas de isomorfismo. Se dice que $f$ es un *isomorfismo* entre $G_1$ y $G_2$ si $f : V_1 \to V_2$ es una función biyectiva y:

*) $\forall u \in V_1, f(u) \in V_2 \wedge I_1(u) = I_2(f(u))$,
*) $\forall \{u, v\} \in E_1, \{f(u), f(v)\} \in E_2 \wedge J_1(\{u, v\}) = J_2(\{f(u), f(v)\})$.

Cuando existe un isomorfismo entre $G_1$ y $G_2$, se dice que $G_1$ y $G_2$ son *isomorfos*. Una manera de enfocar las pruebas de isomorfismo es utilizando FC para representar los grafos [2].

Sea $\Omega$ el conjunto de todos los posibles grafos etiquetados en $L$, la *similitud* entre dos elementos $G_1, G_2 \in \Omega$ es definida como una función $sim : \Omega \times \Omega \to R$, donde $R \subset \mathbb{R}$ es el conjunto de números reales no negativos. Se dice que los elementos son muy diferentes si $sim(G_1, G_2) = 0$ y mientras mayor sea el valor de $sim(G_1, G_2)$ mucho más semejantes son.

Sea $D = \{G_1, \ldots, G_{|D|}\}$ una colección de grafos y $G$ un grafo etiquetado en $L$, el valor de *soporte* de $G$ en $D$ se obtiene mediante la siguiente ecuación:

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

Cuando $sup_G(G, D) \geq \delta$, entonces el grafo $G$ ocurre frecuentemente en la colección $D$, siendo $G$ un *subgrafo frecuente aproximado* en $D$. El valor del umbral de soporte $\delta$ está en $(0, 1]$ asumiendo que la similitud se normaliza a 1. La *minería de subgrafos frecuentes* consiste en encontrar todos los subgrafos conexos frecuentes aproximados en una colección de grafos $D$, utilizando una función de similitud *sim* y un umbral de soporte $\delta$.

Existen varias funciones de similitud usadas por diferentes algoritmos en el proceso de cotejo de grafos [5]. En la siguiente sección (2.2), se presentan los algoritmos más relevantes del estado del arte los cuales usan técnicas de cotejo de grafos en la minería de subgrafos frecuentes. Todos estos algoritmos utilizan la definición anterior (soporte) implementando una función específica de similitud.

### *2.2. Estado del arte*

Existen varios algoritmos para la MFSA en colecciones de grafos que utilizan diferentes funciones de similitud en el cotejo de grafos. La minería de subgrafos aproximados puede dividirse en cinco según el enfoque del cotejo: SUBDUE [6] y RNGV [12] están basados en distancia de edición de grafos, Monkey [16, 17] se basa en $\beta$-arista sub-isomorfismo; CSMiner [14, 15] utiliza el sub-homeomorfismo con vértice/arista disjuntas; MUSE [18, 19] se basa en sub-isomorfismo en grafos inciertos; gApprox [4], APGM [9] y VEAM [1] están basados en probabilidades de sustitución. Estos últimos especifican cuáles vértices, aristas o etiquetas pueden reemplazar otras. De este modo, se defiende la idea de que no siempre una etiqueta de vértice o una etiqueta de arista puede ser sustituida por cualquier otra.

En el algoritmo *gApprox* se realiza la MFSA en un solo grafo y es de interés en este trabajo el procesamiento de colecciones de grafos. Los algoritmos *APGM* y *VEAM* usan matrices de sustitución para realizar la MSFA en

colecciones de grafos. APGM solamente trata las variaciones entre el conjunto de etiquetas de los vértices. Por otro lado, VEAM realiza el proceso de minería con los conjuntos de etiquetas de vértices y aristas. Este algoritmo cumple con la completitud en el espacio de búsqueda, por esta razón, procesa una gran cantidad de candidatos y lleva a cabo muchas pruebas de FC.

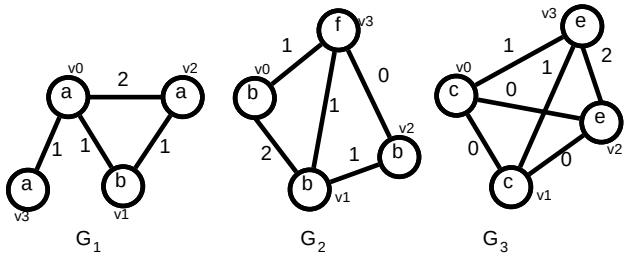
En este trabajo, se proponen mejoras para las soluciones del último de los grupos de algoritmos anteriores. Esto se debe a la necesidad de un algoritmo eficiente que permita algunas variaciones en los datos utilizando probabilidades de sustitución manteniendo la topología de los grafos.

### *2.3. Un método aproximado*

Antes de presentar el método aproximado del algoritmo VEAM [1], donde el cotejo aproximado se basa en los conjuntos de etiquetas de los vértices y aristas, se muestra la definición de matriz de sustitución. Esta matriz puede tener una interpretación probabilística, con la cual se ofrece un esquema probabilístico para esta tarea de minería de subgrafos frecuentes.

Una *matriz de sustitución* $M = (m_{i,j})$ es una $|L| \times |L|$ matriz indizada por el conjunto de etiquetas $L$. Una celda $m_{i,j}$ $(0 \leq m_{i,j} \leq 1, \sum_j m_{i,j} = 1)$ en $M$ es la probabilidad de que la etiqueta $i$ sea reemplazada por la etiqueta $j$.

Se dice que $M$ es *estable* si es diagonal dominante (i.e. $M_{i,i} > M_{i,j}, \forall j \neq i$). En lo adelante, cuando se hable de matriz de sustitución se asume este tipo de matriz. En la figura 1 se muestra una colección de grafos y sus matrices de sustitución $MV$ y $ME$, donde $MV$ está indizada por $L_V$ y $ME$ por $L_E$.




| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | 0.6 | 0 | 0 | 0.4 | 0 | 0 |
| b | 0 | 1 | 0 | 0 | 0 | 0 |
| c | 0 | 0 | 0.6 | 0.4 | 0 | 0 |
| d | 0 | 0 | 0 | 1 | 0 | 0 |
| e | 0 | 0 | 0 | 0 | 1 | 0 |
| f | 0 | 0 | 0 | 0 | 0 | 1 |

MV

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0.6 | 0.4 |
| 2 | 0 | 0 | 1 |

ME

Fig. 1: **Una colección de grafos y sus matrices de sustitución** $MV$ **y** $ME$**.**

Sean $G_1 = (V_1, E_1, I_1, J_1)$ y $G_2 = (V_2, E_2, I_2, J_2)$ dos grafos etiquetados en $L$, $MV$ una matriz de sustitución indizada por $L_V$, $ME$ una matriz de sustitución indizada por $L_E$, y $\tau$ el umbral de isomorfismo. Se dice que $G_1$ es *sub-isomorfo aproximado* a $G_2$, denotado por $G_1 \subseteq_A G_2$, si existe una función inyectiva $h : V_1 \to V_2$ tal que:

*) $\forall v \in V_1 \Rightarrow h(v) \in V_2$,

*) $\forall \{u, v\} \in E_1 \Leftrightarrow \{h(u), h(v)\} \in E_2$,

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

La función $h$ es un *sub-isomorfismo aproximado* entre $G_1$ y $G_2$, y $S_h(G_1, G_2)$ es el producto de las probabilidades normalizadas conocido como *grado del sub-isomorfismo aproximado* de $h$.

Análogamente, el *grado del cotejo aproximado* entre dos grafos, denotado por $S_{max}(G_1, G_2)$, es el mayor de los grados de sub-isomorfismos aproximados:

$$S_{max}(G_1, G_2) = max_h\{S_h(G_1, G_2)\} \tag{2}$$

Sean $G_1, G_2, T$ tres grafos etiquetados en $L$, donde $T \subseteq G_2$. Utilizando un umbral de isomorfismo $\tau$, se dice que $T$ es una *ocurrencia* de $G_1$ en $G_2$ si $S_{max}(G_1, T) \geq \tau$. De igual manera, el *conjunto de ocurrencias* de $G_1$ en $G_2$ es denotado por $O(G_1, G_2)$.

### *2.4. Formulación del problema de la MSFA*

La tarea de *minería de subgrafos frecuentes aproximados* propuesta en este trabajo consiste en encontrar, de manera eficiente, todos los subgrafos conexos frecuentes en una colección de grafos $D$ utilizando un umbral de soporte $\delta$, un umbral de isomorfismo $\tau$ y en (1) la función de similitud propuesta por Acosta-Mendoza *et al.* [1] (ver ecuación (2)).

## 3. Optimizaciones para la MSFA

En esta sección, se explica en qué consisten las tres podas propuestas para el proceso de MSFA. Estas podas son incluidas en el VEAM [1] y este proceso se describe mediante la modificación de parte del pseudo-código de dicho algoritmo.

### *3.1. Eliminando datos innecesarios de la colección*

Generalmente, al comienzo del proceso de minería sobre una colección $D$ se obtienen los subgrafos frecuentes aproximados (SFAs) con un solo vértice. Por lo tanto, es posible identificar el conjunto de aristas candidatas de cada grafo $G_i \in D$ que pudieran generar subgrafos frecuentes o no al extender un patrón. Esto se conoce solamente si el algoritmo utiliza la propiedad de clausura-descendente, como es el caso de los algoritmos (gApprox, APGM y VEAM) en los que se basa esta investigación. Esta propiedad plantea que si $P$ es un subgrafo de $T$ ($P \subseteq T$), entonces $sup_G(T, D) \leq sup_G(P, D)$ [13].

El espacio de búsqueda de un grafo $G_i$ es exponencial según el conjunto de aristas $E_i$ del mismo. Por esta razón, es necesaria una heurística que disminuya dicho espacio. Cuando se extiende un subgrafo patrón utilizando aristas que contengan vértices infrecuentes no se obtendrán subgrafos frecuentes. Este proceso se puede evitar al eliminar el conjunto de vértices infrecuentes de $D$. De esta manera se generarán menos candidatos para ser procesados y se disminuirá la cantidad de pruebas de FC sin alterar los patrones resultantes.

En la MSFA no se puede aplicar de esta manera la poda, ya que puede existir un vértice $u$ que no sea frecuente en $D$ pero que sea utilizado como alguna ocurrencia de otro $v$ que sí lo es. En este caso, si se elimina a $u$ del grafo que lo contiene, $v$ podría dejar de ser frecuente al perder esta ocurrencia en $u$.

Por otro lado, los métodos aproximados analizados se basan en las etiquetas y no en los vértices o aristas. Por estas razones, se modificó esta heurística para que se base en las características de estos métodos. Se define la primera poda de este trabajo como sigue: se eliminan de cada grafo de $D$ los vértices (con sus correspondientes aristas) que su etiqueta no sea utilizada como ocurrencia de ningún vértice frecuente aproximado. Este conjunto de vértices a eliminar se denotará como $V_D^-$.

**Ejemplo 1.** Sean $MV$ y $ME$ (ver figura 1) dos matrices de sustitución indizadas por $L_V$ y $L_E$ respectivamente, $SFA_1$ y $SFA_2$ los SFAs de un vértice de $D = \{G_1, G_2, G_3\}$ (ver figura 2) con $\tau = 0{,}6$ y $\delta = 0{,}4$, se eliminan los vértices con las etiquetas $e$ y $f$ de los grafos de $D$. Los vértices con las etiquetas $a$ y $c$ no se eliminan de $D$ porque son utilizados como ocurrencias de $SFA_2$ con una similitud de 0,667. Nótese que el espacio de búsqueda de los grafos $G_2, G_3 \in D$ se reduce considerablemente al aplicar sobre $D$ la poda descrita en esta sección.

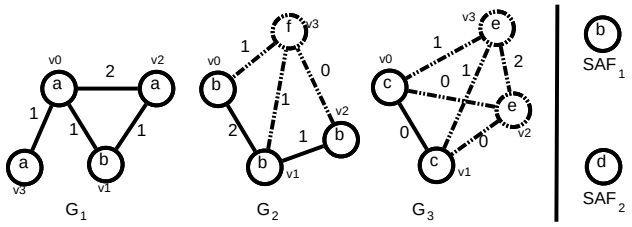



Fig. 2: **Colección de grafos** $D = \{G_1, G_2, G_3\}$ **y sus SFAs de un vértice** $SFA_1$ **y** $SFA_2$**.**

### *3.2. Usando etiquetas útiles*

En el cotejo entre grafos que realizan los algoritmos que utilizan las matrices de sustitución se incorporan todas las etiquetas que indizan estas matrices. Nótese que si existen etiquetas que no aparecen en ningún grafo de la colección y están en las matrices, entonces serán usadas en el proceso de minería identificándose SFAs con dichas etiquetas (un ejemplo de esto es el $SFA_2$ de la colección $D$ en la figura 2).

Como se mencionó anteriormente, en el cotejo aproximado se utilizan todas las etiquetas de las matrices de sustitución. En este conjunto de etiquetas existen algunas de ellas que sustituyen a otras con una probabilidad menor que el umbral $\tau$. Utilizando estas etiquetas en la creación de los subgrafos no se generan candidatos, sin embargo, se pierde tiempo en la verificación de estas extensiones.

Teniendo en cuenta esto, se presenta la siguiente poda del espacio de búsqueda para las etiquetas en el cotejo aproximado; desde el inicio se tienen las matrices de sustitución y se conoce el valor de $\tau$, por tanto, es posible obtener las etiquetas que pueden sustituir a cada etiqueta de las matrices con una probabilidad mayor o igual a $\tau$.

Esto puede obtenerse mediante dos funciones $L_V^\tau(l_v)$ para los vértices y $L_E^\tau(l_{uv})$ para las aristas. Estas funciones obtienen los conjuntos de etiquetas que sustituyen a las etiquetas especificadas $l_v$ y $l_{uv}$ con valores mayores o igual a $\tau$. De esta manera se evita iterar por etiquetas que no cumplen con $\tau$ en cada cotejo entre etiquetas, proceso que se realiza para cada candidato. Con el uso de esta poda se procesarán todas las etiquetas presentes en las matrices solamente en el peor de los casos donde estas puedan sustituir a cualquier otra con valores mayores o iguales que $\tau$.

**Ejemplo 2.** Sean $MV$ y $ME$ (ver figura 1) dos matrices de sustitución indizadas por $L_V$ y $L_E$ respectivamente para la colección $D = \{G_1, G_2, G_3\}$ (ver figura 2) con $\tau = 0{,}6$ y $\delta = 0{,}4$, el conjunto de etiquetas que cumplen con $\tau$ para $L_V$ son: $L_V^\tau(a) = \{a, d\}$, $L_V^\tau(b) = \{b\}$, $L_V^\tau(c) = \{c, d\}$, $L_V^\tau(d) = \{d\}$, $L_V^\tau(e) = \{e\}$ y $L_V^\tau(f) = \{f\}$. De igual manera, el conjunto de etiquetas que cumplen con $\tau$ para $L_E$ son: $L_E^\tau(0) = \{0\}$, $L_E^\tau(1) = \{1, 2\}$ y $L_E^\tau(2) = \{2\}$.

### *3.3. Diseño del algoritmo*

En esta sección, se presenta un pseudo-código tomando a VEAM como base, donde se muestra la inclusión de las podas propuestas. Este algoritmo se denotará como *FastVEAM* para diferenciarlo del original.

**Algorithm 1**: *FastVEAM*

**Input**: $D$ : Una colección de grafos, $MV$ : Matriz de sustitución indizada por $L_V$, $ME$ : Matriz de sustitución indizada por $L_E$, $\tau$ : Umbral de isomorfismo, $\delta$ : Umbral de soporte.

**Output**: $F$ : conjunto de SFAs.

1 $F \leftarrow C \leftarrow$ {*los vértices etiquetados en* $L_V$ *que son SFAs en* $D$};
2 Se eliminan los vértices $u$ de cada grafo de $D$ tales que: $u \in V_D^-$;
3 **foreach** $T \in C$ **do**
4 `Search(`$T, D, MV, ME, \tau, \delta, F$`);`

FastVEAM es presentado mediante tres pseudo-códigos, donde el algoritmo inicial consiste en detectar todos los vértices frecuentes aproximados y almacenarlos en los conjuntos $F$ y $C$ (ver algoritmo 1). Con esta información, se realiza la primera poda al eliminar de la colección $D$ los vértices que pertenecen al conjunto $V_D^-$ (ver línea 2). Luego,

**Algorithm 2**: *Search*

**Input**: $T = (V_t, E_t, I_t, J_t)$ : Un SFA, $D$ : Colección de grafos, $MV$ : Matriz de sustitución indizada por $L_V$, $ME$ : Matriz de sustitución indizada por $L_E$, $\tau$ : Umbral de isomorfismo, $\delta$ : Umbral de soporte.

**Output**: $F$ : Conjunto de SFA.

1 **foreach** $e = \{u, v\} \in ExtSet(o_j)$, *donde* $o_j \in O(T, G_i)$ *y* $G_i \in D$ **do**
2  $CL \leftarrow$ `appLSet(`$T, MV, ME, G_i, o_j, e, \tau$`)`;
3  **foreach** $(elabel, vlabel) \in CL$ **do**
4  Se construye el candidato $X$ utilizando la tupla $(elabel, vlabel)$;
5  Se calcula el código CAM de $X$ y se almacena en $codeCAM(X)$;
6  $C \leftarrow C \cup \{(X, codeCAM(X), score)\}$;
7 **foreach** $T_1 \in C$ **do**
8  **if** $sup_G(T_1, D) \geq \delta$ ***and*** $codeCAM(T_1) \notin F$ **then**
9  Insert $T_1$ in $F$;
10  `Search(`$T_1, D, MV, ME, \tau, \delta, F$`)`;

**Algorithm 3**: *appLSet*

**Input**: $T = (V_T, E_T, I_T, J_T)$ : Un grafo candidato, $MV$ : Matriz de sustitución indizada por $L_V$, $ME$ : Matriz de sustitución indizada por $L_E$, $\tau$ : Umbral de isomorfismo, $G = (V, E, I, J)$ : Un grafo de la colección, $G'$ : Una ocurrencia de $T$ en $G$, $e = \{u, v\}$ : Una extensión de $G'$.

**Output**: $CL$ : El conjunto de 2-tuplas candidatas $(elabel, vlabel)$.

1 **foreach** $j \in L_E^\tau(J(e))$ **do**
2  $scoreE \leftarrow S_{max}(T, G') * \frac{ME_{j,J(e)}}{ME_{j,j}}$;
3  **if** *e es una extensión hacia delante de* $G'$ **then**
4  **foreach** $i \in L_V^\tau(I(v))$ **do**
5  $score \leftarrow scoreE * \frac{MV_{i,I(v)}}{MV_{i,i}}$;
6  **if** $score \geq \tau$ **then** $CL \leftarrow CL \cup \{(j, i)\}$;
7  **else if** $scoreE \geq \tau$ **then** $CL \leftarrow CL \cup \{(j, \emptyset)\}$;

para cada elemento de $C$ se invoca el algoritmo "Search". Cuando todos los vértices frecuentes aproximados hayan sido extendidos, entonces se tiene almacenado el conjunto de todos los SFAs de $D$ en el conjunto respuesta $F$.

En el algoritmo 2 se realiza la extensión de los subgrafos patrones en una arista. Los candidatos se crean como en VEAM a partir del conjunto de etiquetas obtenidas mediante la llamada del algoritmo "appLSet" para cada extensión de un SFA (ver línea 2). De todos esos candidatos solamente se almacenan en $F$ los que son frecuentes y para cada uno de ellos se realiza la llamada recursiva al algoritmo Search. El conjunto de SFAs $F$ es almacenado como una función hash para obtener eficientemente los subgrafos procesados. Las llaves de esta hash son los códigos canónicos de los subgrafos patrones. Estos códigos representan de manera única los subgrafos mediante una cadena. En este trabajo se utilizan las matrices canónicas de adyacencia (CAM, de sus siglas en inglés: **C**anonical **A**djacency **M**atrix) y los códigos CAM. Estos códigos CAM se calculan para cada grafo como en [8]. Nótese que un grafo $G$ con $|V|$ vértices tiene $|V|!$ matrices de adyacencia diferentes, puesto que existen $|V|!$ posibilidades de ordenar los vértices de $G$. Para obtener el código canónico de $G$ se requieren pruebas de FC (ver línea 5). Estas pruebas tienen alta complejidad computacional [2] como se ha mencionado anteriormente.

El algoritmo 3 busca el conjunto de posibles etiquetas de la nueva arista $e$ la cual es una extensión hacia delante de un SFA $T$ (ver líneas 1, 2 y 7), o busca para cada posible etiqueta de la extensión hacia delante $e$ el conjunto de las posibles etiquetas del nuevo vértice que $e$ conecta con algún vértice existente en $T$ (ver líneas 1 – 6). En este algoritmo se incluye la segunda poda propuesta en este trabajo (ver líneas 1 y 4), donde se utilizan solamente las etiquetas que

cumplen con el umbral de isomorfismo al cotejarlas con las existentes en el grafo $G$ de $D$ y no todas las etiquetas de $L$. Esto evita la realización de cotejos que se conoce de antemano que sus valores son menores que el umbral $\tau$.

## 4. Resultados experimentales

En esta sección, se muestan los resultados experimentales que validan la utilidad de las podas propuestas en este trabajo. Se comparan los comportamientos de los algoritmos FastVEAM y VEAM sobre varias colecciones de grafos utilizadas por Acosta-Mendoza *et al.* [1].

Todos los experimentos se realizaron utilizando una computadora personal (64 bits) Intel(R) Core(TM)2 Duo CPU E7300 @ 2.66GHz con 2 Gb de RAM. El algoritmo FastVEAM para la MSFA fue implementado haciendo uso del lenguaje ANSI C y compilado utilizando el compilador gcc de GNU/Linux con optimización -O0.

### *4.1. Colección de grafos*

En este trabajo, se realiza el proceso de minería sobre las colecciones utilizadas por Acosta-Mendoza *et al.* [1]. Estas colecciones se confeccionaron con imágenes obtenidas mediante el Generador de imágenes aleatorias[1] de Coenen. Cada colección se generó con cantidades de grafos diferentes, la colección más pequeña está compuesta por 200 imágenes y se fueron incrementando en 100 hasta obtener 6 colecciones. Cada imagen está representada en forma de grafo utilizando el proceso de representación de las imágenes propuesto en [1].

Estas colecciones se identifican por la letra $D$ seguida por la cantidad de grafos que la componen (ie. $D200$). El mayor tamaño promedio de los grafos de una colección (en términos de cantidad de aristas) es de 48, la cantidad de etiquetas de vértices para todas las colecciones 18 y la cantidad de etiquetas de aristas de todas las colecciones es 6.

### *4.2. Comparación entre VEAM y FastVEAM*

La cantidad de pruebas de FC es directamente proporcional al número de candidatos procesados y estos delimitan el espacio de búsqueda. En esta sección se realiza una comparación (mediante la tabla 1) de los algoritmos VEAM y FastVEAM utilizando como base la cantidad de pruebas de FC y los tiempos de ejecución de ambos algoritmos.

Mediante la subtabla (*a*) de la tabla 1 se especifica la cantidad de pruebas de FC realizadas por cada algoritmo. Como se puede observar en esta tabla, el algoritmo VEAM realiza el doble de FastVEAM de estas pruebas en la mayoría de los casos. Estas pruebas van aumentando su complejidad computacional a medida que van creciendo los candidatos y el hecho de disminuir la cantidad de candidatos a procesar influye positivamente en el comportamiento de FastVEAM.

En la subtabla (*b*) de la tabla 1 se presenta el comportamiento en tiempo de VEAM y FastVEAM, donde se puede apreciar una mejoría por parte de nuestra propuesta. En esta tabla se muestra el resultado de que estos algoritmos presenten diferencias en las cantidades de pruebas de FC, ya que estas influyen en el tiempo de ejecución de ambos algoritmos.

En general los resultados presentados en esta sección reafirman la utilidad de las podas prepuestas en este trabajo. Mediante estas se logra reducir el tiempo de procesamiento y el espacio de búsqueda del algoritmo VEAM. De esta manera se realiza la MSFA eficientemente.

## 5. Conclusiones y trabajo futuro

En este trabajo, se proponen varias podas como mejoras para el proceso de MSFA en colecciones de grafos, donde los grafos son etiquetados y no dirigidos. Estas podas se introducen en el algoritmo VEAM logrando una mejoría en la eficiencia del mismo. Los resultados experimentales muestran la efectividad del uso de estas podas en la MSFA. Se logra la disminución a la mitad de las pruebas de FC por la reducción considerable de los candidatos generados en el proceso de la minería.

Como trabajo futuro, se desarrollarán nuevas estrategias de podas donde se involucren las extensiones de los patrones para obtener algoritmos más eficientes en grandes colecciones. Para estas nuevas estrategias se tomará ventaja de las propiedades de las CAM. Además, desarrollaremos enfoques utilizando las ideas de FastVEAM para la detección de otros SFAs basados en patrones cerrados y maximales.

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

Tabla 1: Comparación entre FastVEAM y VEAM con $\tau = 0{,}4$ utilizando varias colecciones de grafos.

($a$) Cantidad de pruebas de FC

| **Soporte** ($\delta$) | **Colección** | | | |
|---|---|---|---|---|
| | **D700** | | **D600** | |
| | FastVEAM | VEAM | FastVEAM | VEAM |
| 60 % | 649 | 1965 | 542 | 1657 |
| 55 % | 984 | 2913 | 748 | 2247 |
| 50 % | 1380 | 4096 | 1365 | 4076 |
| 45 % | 2287 | 6731 | 2244 | 6674 |
| 40 % | 3715 | 10926 | 3390 | 10051 |
| 35 % | 8127 | 23451 | 7449 | 21747 |
| 30 % | 22336 | 54195 | 21160 | 51527 |
| 25 % | 101250 | 145214 | 83000 | 130838 |
| 20 % | 269396 | 379660 | 262239 | 369220 |
| **Soporte** ($\delta$) | **D500** | | **D400** | |
| | FastVEAM | VEAM | FastVEAM | VEAM |
| 60 % | 640 | 1918 | 521 | 1544 |
| 55 % | 746 | 2208 | 872 | 2504 |
| 50 % | 1361 | 4028 | 1235 | 3608 |
| 45 % | 2236 | 6594 | 2176 | 6324 |
| 40 % | 3337 | 9819 | 3082 | 8869 |
| 35 % | 7562 | 21722 | 7472 | 20882 |
| 30 % | 17214 | 48475 | 17954 | 48506 |
| 25 % | 82148 | 127712 | 75529 | 117390 |
| 20 % | 249452 | 347190 | 233309 | 322253 |
| **Soporte** ($\delta$) | **D300** | | **D200** | |
| | FastVEAM | VEAM | FastVEAM | VEAM |
| 60 % | 346 | 1237 | 333 | 1217 |
| 55 % | 856 | 2459 | 809 | 2385 |
| 50 % | 1030 | 2982 | 1081 | 3176 |
| 45 % | 2111 | 6121 | 2262 | 6533 |
| 40 % | 3020 | 8640 | 2912 | 8288 |
| 35 % | 6588 | 18193 | 6588 | 17843 |
| 30 % | 14284 | 38296 | 14463 | 37655 |
| 25 % | 60750 | 92258 | 62033 | 84167 |
| 20 % | 171510 | 233902 | 145778 | 192997 |

($b$) Tiempos de ejecución (s)

| **Soporte** ($\delta$) | **Colección** | | | |
|---|---|---|---|---|
| | **D700** | | **D600** | |
| | FastVEAM | VEAM | FastVEAM | VEAM |
| 60 % | 1.141 | 1.288 | 0.724 | 0.815 |
| 55 % | 1.726 | 1.986 | 1.052 | 1.239 |
| 50 % | 2.134 | 2.473 | 1.629 | 1.892 |
| 45 % | 3.208 | 3.615 | 2.433 | 2.822 |
| 40 % | 4.826 | 5.367 | 3.420 | 3.932 |
| 35 % | 10.294 | 11.619 | 7.636 | 8.724 |
| 30 % | 23.833 | 26.259 | 16.439 | 18.595 |
| 25 % | 60.806 | 63.832 | 44.740 | 48.313 |
| 20 % | 149.442 | 160.281 | 125.939 | 136.740 |
| **Soporte** ($\delta$) | **D500** | | **D400** | |
| | FastVEAM | VEAM | FastVEAM | VEAM |
| 60 % | 0.636 | 0.742 | 0.394 | 0.479 |
| 55 % | 0.777 | 0.959 | 0.666 | 0.822 |
| 50 % | 1.239 | 1.491 | 0.844 | 1.027 |
| 45 % | 1.839 | 2.167 | 1.331 | 1.614 |
| 40 % | 2.554 | 3.067 | 1.823 | 2.167 |
| 35 % | 5.903 | 6.898 | 4.415 | 5.297 |
| 30 % | 12.369 | 14.362 | 11.643 | 13.820 |
| 25 % | 36.423 | 39.522 | 28.304 | 30.948 |
| 20 % | 102.555 | 111.913 | 83.31 | 91.089 |
| **Soporte** ($\delta$) | **D300** | | **D200** | |
| | FastVEAM | VEAM | FastVEAM | VEAM |
| 60 % | 0.114 | 0.166 | 0.192 | 0.272 |
| 55 % | 0.266 | 0.372 | 0.439 | 0.550 |
| 50 % | 0.321 | 0.411 | 0.499 | 0.622 |
| 45 % | 0.593 | 0.750 | 0.877 | 1.130 |
| 40 % | 0.736 | 0.929 | 1.206 | 1.466 |
| 35 % | 1.802 | 2.299 | 2.779 | 3.420 |
| 30 % | 4.768 | 5.903 | 5.822 | 7.102 |
| 25 % | 11.034 | 12.061 | 16.049 | 18.072 |
| 20 % | 26.018 | 28.694 | 42.406 | 46.598 |

## Referencias bibliográficas


[1] N. Acosta-Mendoza, A. Gago-Alonso, J.E. Medina-Pagola. Frequent approximate subgraphs as features for graph-based image classification. To appear: Knowledge-Based Systems (2011).

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

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

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

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

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

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

[8] J. Huan, W. Wang, J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. in: Proceedings of the 3rd IEEE International Conference on Data Mining, IEEE Computer Society, 2003, pp. 549–552.

[9] Y. Jia, J. Huan, V. Buhr, J. Zhang, L. Carayannopoulos. Towards comprehensive structural motif mining for better fold annotation in the "twilight zone.ºf sequence dissimilarity. BMC Bioinformatics 10 (2009).

[10] N. Ketkar, L. Holder, D. Cook. Mining in the proximity of subgraphs. Analysis and Group Detection KDD Workshop on Link Analysis: Dynamics and Statics of Large Networks (2006).

[11] M. Koyutürk, A. Grama, W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. Bioinformatics (2004) 200–207.

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

[13] N. Vanetik, S.E. Shimony, E. Gudes. Support measures for graph data. Data Mining and Knowledge Discovery 13(2) (2006) 243–260.

[14] Y. Xiao, W. Wang, W. Wu. Mining conserved topological structures from large protein-protein interaction networks. in: Proceedings of the 18th IEICE data engineering workshop / 5th DBSJ annual meeting, DEWS'2007, Hiroshima, Japan, 2007.

[15] Y. Xiao, W. Wu, W. Wang, Z. He. Efficient algorithms for node disjoint subgraph homeomorphism determination. in: Proceedings of the 13th international conference on Database systems for advanced applications, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 452–460.
[16] S. Zhang, J. Yang. Ram: Randomized approximate graph mining. in: Proceedings of the 20th International Conference on Scientific and Statistical Database Management, ICSSDM, 2008, pp. 187–203.
[17] S. Zhang, J. Yang, V. Cheedella. Monkey: Approximate graph mining based on spanning trees. in: International Conference on Data Engineering, IEEE ICDE, Los Alamitos, CA, USA, 2007, pp. 1247–1249.
[18] Z. Zou, J. Li, H. Gao, S. Zhang. Frequent subgraph pattern mining on uncertain graph data. in: Proceeding of the 18th ACM conference on Information and knowledge management, ACM, New York, NY, USA, 2009, pp. 583–592.
[19] Z. Zou, J. Li, H. Gao, S. Zhang. Mining frequent subgraph patterns from uncertain graph data. IEEE Transactions on Knowledge and Data Engineering 22 (2010) 1203–1218.