LA COMPLEJIDAD PARAMÉTRICA DE MINAR GRAFOS 1, RESULTADOS NEGATIVOS

J. ANDRÉS MONTOYA.

Abstract. In this paper we analyze the parameterized complexity of graph mining tasks, specifically we analyze the parameterized complexity of the listing problem consistent in: Given an input graph G, list the frequent subgraphs of G of a given size. In this paper some lower bounds for this problem are proved.

RESUMEN. En este artículo analizamos la complejidad paramétrica de algunos problemas típicos en minería de grafos, específicamente nosotros analizamos la complejidad paramétrica del problema de listado consistente en: Dado G un grafo-input, liste todos los subgrafos de G de un tamaño dado. En el artículo se prueban algunas cotas inferiores para este problema.

A Mamadimitriou e Hijodimitriou

1. Introducción

Este escrito es resultado de un proyecto de investigación que se ha fijado dos objetivos fundamentales, los cuales aunque opuestos son complementarios. El primer objetivo consiste en analizar la dificultad intrínseca de minar grafos, entendiendo esta tarea computacional como la tarea consistente en listar todos los patrones frecuentes y todos los patrones excepcionales de un grafo dado. En el escrito se confirman, o mejor se demuestran, las conjeturas iniciales del autor, a saber, que el problema de minar grafos es un problema paramétrico intratable. El segundo objetivo del proyecto, que no es tratado en este escrito, consiste en identificar restricciones del problema general que admiten soluciones eficientes en el sentido de la complejidad paramétrica.

1.1. Complejidad paramétrica.

La complejidad paramétrica intenta analizar la complejidad de lenguajes que admiten una parametrización natural. Considere por ejemplo el problema del *Clique.*

Problema 1 (*Clique*)**.** .

- *Input:* (G, k)*, donde* G *es un grafo y* $k \in \mathbb{N}$*.*
- *Problema: Decida si* G *contiene un grafo completo de tamaño* k*, (i.e. un* k*-clique).*

Es un hecho bien conocido que el problema del *Clique* es NP-completo (ver [10]). Quien revise cuidadosamente la prueba de NP-completez del problema del *Clique*, notará que en ella es necesario considerar instancias (G, k), en las que k es aproximadamente igual al tamaño de G. En las aplicaciones del problema, como

Key words and phrases. Maquinas de Turing, Clases de complejidad, Complejidad paramétrica, Algoritmos eficientes, Costos computacionales.

por ejemplo búsqueda en una base de datos de k-personas relacionadas dos a dos, el grafo, i.e. la base de datos, es inmenso mientras que k es pequeño. Es natural pensar que el análisis clásico del problema del *Clique*, que lleva a la prueba de su NP-completez y por lo tanto de su intratabilidad, es un análisis inadecuado porque ignora la naturaleza bidimensional de las instancias del problema, compuestas usualmente por una componente inmensa, el input propiamente dicho, y una componente relativamente pequeña, el parámetro. Considere la siguiente parametrización del problema del clique.

Problema 2 ($p-clique$)**.** .

- *Input:* (G,k)*, donde* G *es un grafo y* $k \in \mathbb{N}$*.*
- *Parámetro:* k*.*
- *Problema: Decida si* G *contiene un grafo completo de tamaño* k*, (i.e. un* k*-clique).*

Dado L un problema paramétrico, como por ejemplo $p-clique$, diremos que L es computable en tiempo fpt si y solo si existe un algoritmo $\mathcal{M}$ que decide L y tal que el tiempo de cómputo de $\mathcal{M}$ en una instancia (x,k) esta acotado por $f(k)\,p(|x|)$, donde f es una función computable y p es un polinomio.

La complejidad paramétrica intenta realizar un análisis paramétrico de los problemas parametrizables proponiendo para ello una noción relajada de tratabilidad. Mientras que la noción canónica, (clásica), de tratabilidad es computabilidad en tiempo polinomial, la noción de tratabilidad en el contexto paramétrico es computabilidad en tiempo fpt. Por otro lado la complejidad paramétrica propone estudiar una noción paramétrica de reducción, a la que llamaremos p-Turing reducibilidad. Esta nueva noción de reducción permite a su vez definir nuevas clases de complejidad, las cuales permiten medir la complejidad (paramétrica) de los lenguajes parametrizables.

Nota 1. *El problema del clique sigue siendo intratable desde el punto de vista paramétrica. La manera estándar de clasificar a un problema* L *como intratable consiste en probar que dicho problema es duro para una clase de complejidad, que se sospeche no está contenida en la clase de los problemas tratables. Así por ejemplo en el contexto clásico, cuando se quiere probar que un cierto problema* L *es intratable, lo que se suele intentar probar acerca de* L *es que dicho problema es* NP *duro. Una clase paramétrica análoga a* NP *es la clase* $W[1]$ *(ver* [6]*).* $p-Clique$ *es* $W[1]$ *completo, esto es,* $p-Clique$ *es, muy probablemente, intratable en el sentido paramétrico. Esto no quiere decir que todo problema intratable en el sentido clásico sigue siendo intratable en el sentido paramétrico, (si este fuera el caso ¿Tendría sentido la complejidad paramétrica?), así por ejemplo el problema del cubrimiento de vértices, (este problema parametrizable es usualmente llamado Vertex Cover (ver* [10]*)), es* NP *completo pero computable en tiempo* fpt*, esto es, tratable desde el punto de vista paramétrico.*

1.2. **Minería de grafos.** Intentaremos definir que es minería de datos en términos del tipo de problemas que se estudian en esta área de las ciencias de la computación. Sea X un objeto combinatorio como por ejemplo una base de datos o un grafo. Minar X significa extraer tanta información acerca de X como sea posible. El tipo de

información que se quiere extraer puede ser de múltiples tipos, puede ser información estructural, (por ejemplo, una lista de axiomas que describen completamente la estructura), o puede ser información estadística, (que tipos de subestructura aparecen frecuentemente en X). En la práctica la minería de datos se ha concentrado en la extracción de información estadística y de manera mas específica se ha concentrado en el siguiente tipo de problema.

Problema 3 (Minería de datos, genérico)**.** .

- *Input: Una estructura X.*
- *Problema: Liste todas las subestructuras frecuentes y todos las estructuras excepcionales en X.*

El problema genérico anteriormente descrito no admite mayores análisis, dado que su definición involucra varios términos que requieren ser precisados. Por ahora es suficiente notar que, en la práctica, el objeto X suele ser inmenso mientras que las subestructuras que se desea listar suelen ser relativamente pequeñas. El parágrafo anterior indica que, las diferentes variaciones del problema genérico de mineria de datos que ocurren en la práctica, son problemas que admiten una parametrización natural. Es por ello que consideramos que la teoría de la complejidad paramétrica es la teoría que debemos usar si pretendemos analizar la complejidad computacional de este tipo de problemas.

La minería de grafos es una subárea de la mineria de datos en la que el tipo de objetos combinatorios a minar, (el X del parágrafo anterior), son grafos. Esto, (que en principio puede parecer una restricción demasiado fuerte), es solo una pequeña restricción, dado que la riqueza expresiva de los grafos nos permite codificar una amplia variedad de estructuras combinatorias como grafos.

1.3. **Organización del escrito.** El escrito consta de cuatro secciones incluyendo la introducción. En la sección 2 se introducen los conceptos básicos de la complejidad paramétrica, se define el problema que deseamos analizar y se establecen las metas que intentamos alcanzar (y esperamos haber alcanzado). En la sección 3, que es el núcleo del artículo, se prueban los siguientes hechos.

(1) El problema de minar grafos de manera exacta es equivalente al problema de conteo exacto en $\#W$ [1] y por lo tanto muy difícil.
(2) El problema de minar grafos de manera aproximada es equivalente al problema de aproximar un problema $\#W$ [1] completo y por lo tanto intratable.
(3) El problema de minar grafos de manera aproximada es probabilísticamente reducible a un problema W [1] completo.

En la cuarta y última sección de este escrito se proponen algunos problemas y se presentan algunas conclusiones.

2. Preliminares

En esta sección se presentan los conceptos técnicos básicos que serán utilizados a lo largo del escrito. Adicionalmente se introduce un problema paramétrico que corresponde en buena medida al problema de minar grafos.

2.1. **Complejidad paramétrica.**

En esta sección introduciremos los conceptos básicos de la complejidad paramétrica.

Notación 1. $\{0,1\}^*$ *es el conjunto de las palabras de longitud finita en el vocabulario* $\{0,1\}$.

Definición 1. *Un lenguaje o problema paramétrico es un subconjunto de* $\{0,1\}^* \times \mathbb{N}$.

Ejemplo 1 (Evaluación de bases de datos)**.** .

- *Input:* (D,α)*, donde* D *es una base de datos y* α *una consulta.*
- *Parámetro:* $|\alpha|$*, donde* $|\alpha|$ *es el tamaño de la consulta, esto es, la longitud de la fórmula que expresa la consulta* α *en un lenguaje apropiado.*
- *Problema: Decida si* D *tiene la propiedad expresada por* α.

Nota 2. *Las instancias de todos los problemas que se consideraran en este escrito, así como las instancias del problema antes definido, son parejas constituidas por un número natural y un objeto combinatorio, como por ejemplo un grafo. Esto parece contradecir la definición de problema paramétrico que exige que las instancias de un tal problema estén constituidas por una palabra binaria y un número natural. Lo que sucede es que todo objeto combinatorio, (grafos, números, álgebras finitas, redes, nudos), puede ser codificado eficientemente como una palabra binaria. Nosotros usaremos una convención usual en teoría de la computación y en complejidad computacional, a saber, consideraremos las instancias de nuestros problemas como lo que son, (esto es, si son grafos como grafos si son matrices como matrices), y no nos preocuparemos por hacer explícita la codificación como palabras binarias que sabemos que existe.*

Definición 2. *Dado* L *un problema paramétrico, diremos que* L *es decidible en tiempo* fpt *si y solo si existe un algoritmo* $\mathbb{M}$ *tal que con input* (x,k)*:*

1. $\mathbb{M}$ *decide correctamente si* (x,k) *pertenece a* L.
2. *El tiempo de cómputo de* $\mathbb{M}$ *esta acotado por* $f(k)\,p(|x|)$*, donde* f *es una función computable,* p *es un polinomio y* $|x|$ *denota el tamaño de* x.

La clase $\mathcal{FPT}$ es la clase de todos los problemas decidibles en tiempo fpt. La clase $\mathcal{FPT}$ es el análogo paramétrico de la clase P, esto es, la clase $\mathcal{FPT}$ esta constituida por los problemas tratables en el sentido paramétrico.

Definición 3 (p-Turing reducibilidad)**.** *Dados* L *y* Σ *dos lenguajes paramétricos, diremos que* L *es* p*-Turing reducible a* Σ*, (en símbolos* $L \prec_T \Sigma$*), si y solo si existe un algoritmo* $\mathbb{M}$ *con acceso a un oráculo para* Σ *y tal que, con input* (x,k) :

(1) $\mathbb{M}$ *decide correctamente si el input* (x,k) *pertenece o no pertenece al lenguaje* L.
(2) *El tiempo de cómputo de* $\mathbb{M}$ *esta acotado por* $f(k)\,p(|x|)$, *donde* f *es una función computable y* p *es un polinomio.*
(3) *Si durante la computación* $\mathbb{M}$ *consulta al oráculo acerca de la instancia* (y,r), *entonces* $r \leq f(k)$.

La noción de p-Turing reduciblidad es el equivalente paramétrico de la noción clásica de Turing reducibilidad y al igual que esta nos permitirá: comparar diferentes problemas de acuerdo a su dificultad intrínseca; definir nuevas clases de complejidad, (paramétricas); clasificar los problemas paramétricos de acuerdo con su complejidad.

Definición 4. *Dados* L *y* Σ *dos lenguajes paramétricos, diremos que* L *es fpt reducible a* Σ, *(en símbolos* $L \prec_{fpt} \Sigma$*), si y solo si existe una p-Turing reducción de* L *en* Σ *tal que el oráculo es consultado una única vez en cada computación.*

La clase paramétrica que introduciremos a continuación es el análogo paramétrico de NP.

Definición 5. $W[1] = \{L : L \text{ es un lenguaje paramétrico y } L \prec_{fpt} p - Clique\}$.
(1) L *es* $W[1]$ *duro si y solo si* $p - Clique \prec_T L$.
(2) L *es* $W[1]$ *completo si y solo si* L *es* $W[1]$ *duro y* $L \in W[1]$.

En lo que sigue introduciremos una caracterización en términos de máquinas de la clase $W[1]$.

Definición 6 ($W[1]$-PRAM)**.** *Una* $W[1]$*-PRAM* $\mathbb{M}$ *es una máquina no determinística de acceso aleatorio tal que, con input* (x,k).
(1) *Realiza a lo más* $f(k)\,p(|x|)$ *pasos. A lo más* $f(k)\log(|x|)$ *de ellos no determinísticos, donde* f *es una función computable y* p *un polinomio.*
(2) *Usa a lo más* $f(k)\,p(|x|)$ *registros.*
(3) *En todo momento de la computación los números guardados en cada uno de los registros son menores que* $f(k)\,p(|x|)$.
(4) *Los pasos no determinísticos están entre los últimos* $h(k)\log(|x|)$ *pasos de la computación, donde* h *es una función computable.*

Teorema 1. $L \in W[1]$ *si y solo si existe una* $W[1]$*-PRAM que decide* L.

Para mas información acerca del teorema el lector puede consultar [6]. A continuación introduciremos un tipo adicional de reducciones que llamaremos reducciones probabilísticas

Definición 7 (Reducciones probabilísticas)**.** *Dados* L *y* Σ *dos lenguajes paramétricos, diremos que* L *probabilísticamente reducible a* Σ, *(en símbolos* $L \prec_r \Sigma$*), si y solo si existe un algoritmo probabilístico* $\mathbb{M}$ *con acceso a un oráculo para* Σ *y tal que, con input* (x,k) :

(1) *El número de bits aleatorios utilizados por* $\mathbb{M}$ *esta acotado por* $f(k)\log(|x|)$, *donde* f *es una función computable.*
(2) *El tiempo de cómputo de* $\mathbb{M}$ *esta acotado por* $f(k)\,p(|x|)$, *donde* p *es un polinomio.*
(3) *Si durante la computación* $\mathbb{M}$ *consulta al oráculo acerca de la instancia* (y,r), *entonces* $r \leq f(k)$.
(4) *Si* $(x,k) \in L$, *entonces* $\Pr_r[\mathbb{M} \text{ acepte } (x,k)] \geq \frac{3}{4}$, *la probabilidad es calculada sobre las escogencias aleatorias de* $\mathbb{M}$.
(5) *Si* $(x,k) \notin L$, *entonces* $\Pr_r[\mathbb{M} \text{ acepte } (x,k)] \leq \frac{1}{4}$.

Para finalizar introduciremos la noción de problema paramétrico de conteo y algunas nociones propias de la Complejidad Paramétrica de problemas de conteo.

Definición 8. *Un problema paramétrico de conteo es una función computable* $f : \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$.

Ejemplo 2 $(p - \#Clique)$**.** .

- *Input:* (G,k), *donde* G *es un grafo y* $k \in \mathbb{N}$.
- *Parámetro:* k.
- *Problema: Calcule el número de* k*-Cliques en* G.

La noción de p-Turing reducibilidad entre problemas de conteo es la adaptación natural de la noción de p-Turing reducibilidad al contexto de problemas de conteo.

Definición 9 (p-Turing reducibilidad entre problemas de conteo)**.** *Dados* χ *y* ψ *dos problemas paramétricos de conteo, diremos que* χ *es p-Turing reducible a* ψ, *(en símbolos* $\chi \prec_T \psi$*), si y solo si existe un algoritmo* $\mathbb{M}$ *con acceso a un oráculo para* ψ *y tal que, con input* (x,k) :

(1) $\mathbb{M}$ *calcula correctamente* $\chi(x,k)$.
(2) *El tiempo de cómputo de* $\mathbb{M}$ *está acotado por* $f(k)\,p(|x|)$, *donde* f *es una función computable y* p *es un polinomio.*
(3) *Si durante la computación* $\mathbb{M}$ *consulta al oráculo acerca de la instancia* (y,r), *entonces* $r \leq f(k)$.

La noción de reducción entre problemas de conteo nos permite definir clases de problemas de conteo. A continuación introduciremos la clase $\#W[1]$ la cual es el análogo paramétrico de $\#P$ (ver [5]).

Definición 10. $\#W[1] = \{\psi : \psi \text{ es un problema paramétrico de conteo y } \psi \prec_T p - \#Clique\}$.

Mucha mas información acerca de la teoría de la Complejidad Paramétrica puede ser encontrada en [4] y [6].

2.2. **El problema de minar grafos, primeras observaciones y definición del problema.** El objetivo de esta sección es presentar una definición formal del problema que queremos analizar.

Notación 2. .

(1) *Dado S un conjunto* $[S]^2 = \{A \subset S : |A| = 2\}$.
(2) *Dado* $n \in \mathbb{N}$, $[n]$ *denotará el conjunto* $\{1, 2, ..., n\}$.
(3) *Dado* $k \in \mathbb{N}$, K_k *es el grafo completo con* k *vértices.*

Recuerde primero que un grafo G es un par $(V(G), E(G))$, donde $V(G)$ es un conjunto, el conjunto de vértices de G, y $E(G) \subset [V(G)]^2$, $E(G)$ será llamado el conjunto de aristas del grafo G. En lo que sigue, dado G un grafo, si $|V(G)| = n$, asumiremos que $V(G) = [n]$. El tamaño de G será igual a $|V(G)|$, en ocasiones usaremos el símbolo $|G|$ para denotar el tamaño de G.

Dados G y H dos grafos, diremos que G y H son *isomorfos* si y solo si existe una biyección $g : V(G) \to V(H)$ tal que para todo par de vértices $v, w \in V(G)$, $\{v, w\} \in E(G)$ si y solo si $\{g(v), g(w)\} \in E(H)$. Dados G y H, usaremos el símbolo $G \simeq H$ para indicar que G y H son isomorfos.

Dados G y H dos grafos, *el número de ocurrrencias de H como subgrafo de G* es igual a $\left|\left\{(W, F) : W \subset V(G) \ \& \ F \subset E \cap [W]^2 \ \& \ H \simeq (W, F)\right\}\right|$

Dado $c \in \mathbb{N}$, diremos que H es *c-frecuente* en G si y solo si el número de ocurrencias de H como subgrafo de G es por lo menos $2c$. Por otro lado diremos que H es *c-excepcional* si y solo si el número de ocurrencias de H como subgrafo de G es a lo más c.

Problema 4 (Minería exacta)**.** .

- *Input: Un grafo G y dos números naturales c y k.*
- *Parámetro: k.*
- *Problema: Liste todos los subgrafos c-frecuentes y todos los subgrafos c-excepcionales de G.*

El problema anteriormente definido es precisamente el problema que quisieramos analizar. Lo primero que podemos notar es que el problema de Minería Exacta es al menos tan difícil como el problema $p - \#Clique$, el cual es bien sabido es $\#W[1]$ completo (ver [5]). Para verificar la anterior observación es suficiente considerar el siguiente argumento.

Sea (G, k) una instancia de $p - \#Clique$, para contar el número de k-cliques en G es suficiente calcular $c \in \mathbb{N}$ tal que, K_k es c-excepcional pero no es $(c-1)$-excepcional. Note simplemente que $c - 1$ es igual al número de $k - cliques$ en G. Para terminar es suficiente anotar que es posible calcular tal c, en tiempo polinomial en $|G|$, usando búsqueda binaria.

El parágrafo anterior indica que el problema de minería exacta es duro para $\#W[1]$. Un problema duro para una clase de problemas de conteo como $\#W[1]$ es un problema muy difícil. En el contexto clásico se sabe que $\#P$ contiene la jerarquia polinomial, este es el famoso teorema de Toda (ver [11]), esto quiere

decir que si L es duro para $\#P$, entonces L es mas difícil que todo problema en la jerarquía polinomial y por lo tanto ω veces más difícil que todo problema en NP, (si la jerarquía polinomial no colapsa). Aunque no se conoce un análogo del teorema de Toda en el contexto paramétrico, se conjetura que $\#W[1]$ contiene toda la jerarquía W (ver [5]). Así pues, si la conjetura es cierta, el problema de minería exacta es ω veces mas difícil que todo problema en $W[1]$, (si la jerarquía W no colapsa), lo que lo convierte en un problema extremadamente difícil, más allá de nuestras posibilidades de análisis.

El problema de minería exacta es difícil porque en el fondo el es un problema de conteo. Cuando uno tiene que lidiar con un problema de conteo tiene dos alternativas, intentar resolverlo de manera exacta, lo cual no suele ser posible, o intentar resolverlo de manera aproximada. En algunos casos es posible resolver eficientemente la versión aproximada de un problema de conteo aunque la versión exacta sea intratable (ver [1]). Es por ello que proponemos considerar la siguiente versión aproximada del problema de minar un grafo.

Problema 5 (Minería Aproximada)**.** .

- *Input: Un grafo G y dos números naturales c y k.*
- *Parámetro: k.*
- *Problema: Calcule dos listas disyuntas L_F y L_E de manera tal que L_F contenga todos los grafos c-frecuentes de tamaño k en G y L_E contenga todos los grafos c-excepcionales.*

Un algoritmo que resuelva el problema es un algoritmo que con input (G, k, c) produce dos listas disyuntas. La primera lista es una lista que contiene todos los subgrafos c-frecuentes de G de tamaño k y que puede contener otros grafos además de los c-frecuentes. La segunda lista, por su parte, ha de contener todos los subgrafos c-excepcionales de $\mathcal{G}$ de tamaño k y, al igual que la primera lista, puede contener errores, esto es grafos que no son c-excepcionales. Sea $\mathcal{GRAPHS}(k)$ una lista que contiene todos los grafos de tamaño k salvo isomorfismo. Lo primero que podemos notar es que el tamaño de $\mathcal{GRAPHS}(k)$ depende únicamente de k. Es por ello que, el problema de listado *Minería Aproximada*, es equivalente al siguiente $Gap - problem$.

Problema 6 $(Gap - GM)$**.** .

- *Input: (G, H, k, c). G y H son grafos, $|H| = k$ y $k, c \in \mathbb{N}$.*
- *Parámetro: k*
- *Instancias SI: (G, H, k, c) es una instancia SI si y solo si H es c-frecuente.*
- *Instancias NO: (G, H, k, c) es una instancia NO si y solo si H es c-excepcional.*

Un algoritmo $\mathbb{M}$ que resuelva el problema $Gap - GM$. es un algoritmo que:

(1) A toda instancia SI, $\mathbb{M}$ la clasifica como un SI.
(2) A toda instancia NO, $\mathbb{M}$ la clasifica como un NO.
(3) Puede errar en toda otra instancia.

El que los problemas $Gap - GM$ y *Minería Aproximada* sean equivalentes nos permite analizar $Gap - GM$ en lugar de *Minería Aproximada*.

3. La dureza de minar grafos

En esta sección analizaremos la complejidad paramétrica del problema $Gap - GM$. Probaremos lo siguiente:

(1) El problema $Gap - GM$ es equivalente al problema de aproximar un problema $\#W[1]$ completo dentro del rango 2
(2) $Gap - GM$ es probabilísticamente reducible a un problema en $W[1]$.
(3) Todo problema en $W[1]$ es reducible al problema de conteo aproximado en $\#W[1]$

Dado L un problema $W[1]$ completo, el numeral 1 indica que los problemas $Gap - GM$ y conteo aproximado en $\#W[1]$ son equivalentes, (Turing-equivalentes, i.e. equivalentes desde el punto de vista de las reducciones de Turing). Los numerales 1 al 3 indican que los problemas $Gap - GM$ y L son equivalentes desde el punto de vista de las reducciones probabilísticas.

3.1. La dureza de $Gap - GM$.

En esta subsección probaremos que el problema $Gap - GM$ es tan difícil como el problema de aproximar una función en $\#W[1]$.

Definición 11. *Dado χ un problema de conteo parametrizado y dado $c \gneqq 1$, un $fpt-$algoritmo $\mathbb{M}$ aproxima χ dentro del rango c si y solo si para toda instancia (x,k) deχ*

$$\chi(x,k)\frac{1}{c} \leq \mathbb{M}(x,k) \leq \chi(x,k)\, c$$

donde $\mathbb{M}(x,k)$ es el output de $\mathbb{M}$ en el input (x,k).

En lo que sigue analizaremos la complejidad de aproximar, dentro de un rango dado, un problema $\chi \in \#W[1]$. Para empezar probaremos que para poder aproximar χ dentro de un rango $c \gneqq 1$, es suficiente tener la capacidad de aproximar χ dentro del rango 2. Para ello necesitaremos el siguiente lema.

Lema 1. *Dado $\chi \in \#W[1]$ y dado $c \in \mathbb{N}$, la función $\chi_c : \Sigma^* \times \mathbb{N} \to \mathbb{N}$ definida por $\chi_c(x,k) = (\chi(x,k))^c$ pertenece a $\#W[1]$.*

Proof. Si $\chi \in \#W[1]$, sabemos que existen una $W[1]-$PRAM, llamémosla $\mathcal{M}$, dos funciones computables g y h y un polinomio p tales que

(1) Toda computación de $\mathcal{M}$ con input (x,k), consta de tres fases. En la primera fase $\mathcal{M}$ realiza una computación determinística cuyo tiempo de cómputo está acotado por $h(k)\, p(|x|)$. En la segunda fase $\mathcal{M}$ realiza $g(k)\log(|x|)$ elecciones no deterministas, adivinando con ello un elemento r de $\{0,1\}^g$. Finalmente, en la tercera fase $\mathcal{M}$ realiza una computación determinística con input (x,r,k) cuyo tiempo de cómputo está acotado por $h(k)\log(|x|)$.
(2) $\chi(x,k) = \#acc(\mathbb{M}(x,k))$.
Sea $\mathcal{M}_c$ la $W[1]$-máquina de Turing tal que con input (x,k)

- En toda computación, con input (x,k), $\mathcal{M}_c$ inicialmente simula la primera fase de la computación de $\mathcal{M}$. Esto es, $\mathcal{M}_c$ simula la computación de $\mathcal{M}$ hasta que esta realiza la primera elección no determinística.
- Tras la primera fase, i.e. la simulación de la fase determinísitica de $\mathcal{M}$, $\mathcal{M}_c$ con input (x,k), adivina de manera no determinística $r_1, ..., r_c \in \{0,1\}^g$.

- Para todo $i \leq c$, $\mathcal{M}_c$ simula la tercera fase de la computación de $\mathcal{M}$ usando como input la tripla (x, r_i, k).
- $\mathbb{M}_c$ acepta (x, k) si y solo si para todo $i \leq c$, la simulación de la tercera fase de la computación de $\mathcal{M}$ en el input (x, r_i, c) finaliza en un estado de aceptación.

Es claro que $\chi_c(x,k) = (\#acc(M(x,k)))^c$. □

Lema 2. *Dado χ un problema $\#W[1]$ completo, si existe un $fpt-$algoritmo $\mathcal{M}$ que aproxima χ dentro del rango2, entonces para todo $c \gneqq 1$, existe un algoritmo $\mathcal{M}_c$ que aproxima χ dentro del rango c.*

Proof. Sea d un número natural tal que $(2)^{\frac{1}{d}} \leq c$. El algoritmo $\mathcal{M}_c$ está definido por la siguiente lista de instrucciones:

Con input (x, k)

(1) $\mathcal{M}_c$ calcula un par (x^*, k^*) tal que $\chi(x^*, k^*) = (\chi(x,k))^d$.
(2) $\mathcal{M}_c$ simula la computación de M en el input (x^*, k^*).
(3) Dado t el output de $\mathcal{M}$, en el input $\left(x^*, k^*\right)$, $\mathcal{M}_c$ calcula $(t)^{\frac{1}{d}}$.

La computación en el paso 1 puede ser realizada en tiempo fpt, dado que $\mathcal{M}_c$ solo tiene que calcular un par (x^*, k^*) tal que $\chi(x^*, k^*) = \chi_c(x,k)$. Recuerde que $\chi_c \in \#W[1]$ y que χ es $\#W[1]$ completo.

En el paso 2, $\mathcal{M}_c$ calcula un número t tal que

$$\Pr\left[\frac{1}{2}(\chi(x,k))^d \leq t \leq 2(\chi(x,k))^d\right] \geq \frac{2}{3}$$

se tiene entonces que

$$\frac{2}{3} \leq \Pr\left[\left(\frac{1}{2}\right)^{\frac{1}{d}} \chi(x,k) \leq (t)^{\frac{1}{d}} \leq (2)^{\frac{1}{d}} \chi(x,k)\right] \leq \Pr\left[\frac{1}{c}\chi(x,k) \leq (t)^{\frac{1}{d}} \leq c\chi(x,k)\right]$$

Ahora, dado que $(t)^{\frac{1}{d}}$ es el output de $\mathcal{M}_c$, podemos concluir que

$$\Pr\left[\frac{1}{c}\chi(x,k) \leq \mathcal{M}_c(x,k) \leq c\chi(x,k)\right]$$

□

En lo que sigue mostraremos que, el problema de aproximar una problema $\chi \in \#W[1]$ dentro del rango 2 es reducible a $Gap-GM$. Note primero que, si existe un problema $\chi_0 \in \#W[1]$ tal que: χ_0 es $\#W[1]$ completo y el problema de aproximar χ_0 dentro del rango 2 es reducible a $Gap - GM$, entonces dado $\chi \in \#W[1]$, el problema de aproximar χ dentro del rango 2 es reducible a $Gap - GM$.

Considere el siguiente problema

Problema 7 $(p - \#EMB)$**.** .

- *Instancia:* (G, H, k), *G y H son grafos,* $|H| = k$ *y* $k \in \mathbb{N}$.
- *Parámetro: k.*
- *Problema: Calcule el número de ocurrencias de H en G.*

Lema 3. *El problema $p - \#EMB$ es $\#W[1]$ completo.*

Proof. Note simplemente que $p-\#Clique$ está contenido en $p-\#EMB$ □

Ahora, para cumplir con lo que prometimos al comienzo de la sección, probaremos que el problema de aproximar $p-\#EMB$ dentro del rango 2 es reducible a $Gap-GM$.

Dada (G,H,k) una instancia del problema $p-\#EMB$, existe un número natural $m \leq \log\left(|G|^k 2^{\binom{k}{2}}\right)$ tal que

$$2^m \leq p-\#EMB\,(G,H,k) \leq 2^{m+1}$$

Note que, de la definición de m tenemos que

(1) $\frac{1}{2}\left(p-\#EMB\,(G,H,k)\right) \leq 2^m \leq 2\left(p-\#EMB\,(G,H,k)\right)$.
(2) $\frac{1}{2}\left(p-\#EMB\,(G,H,k)\right) \leq 2^{m+1} \leq 2\left(p-\#EMB\,(G,H,k)\right)$.

De lo anterior tenemos que, para aproximar $p-\#EMB\,(G,H,k)$ dentro del rango 2 es suficiente calcular un número $n \in \{m, m+1\}$. La prueba del teorema a continuación se basa en este hecho.

Teorema 2. *Existe un fpt-algoritmo $\mathbb{M}$, con acceso al oráculo $Gap-GM$ y tal que*

(1) *$\mathbb{M}$ aproxima $p-\#EMB$ dentro del rango 2.*
(2) *Con input (G,H,k), $\mathbb{M}$ consulta el oráculo a lo mas $g(k)\log(|G|)$ veces, siendo g una función computable apropiada.*

Proof. Con input (G,H,k), el algoritmo $\mathbb{M}$ calcula un número $n \in \mathbb{N}$ tal que $n \in \{m, m+1\}$ y el output de $\mathbb{M}$ es 2^n.

Primero note que

- Si $i \leq m-1$, entonces $\left(G,H,k,2^i\right)$ es una instancia SI de $Gap-GM$.
- Si $i \geq m+1$, entonces $\left(G,H,k,2^i\right)$ es una instancia NO de $Gap-GM$.

$\mathbb{M}$ es el siguiente algoritmo

Con input (x,k)

(1) Para todo $i \leq \log\left(|G|^k 2^{\binom{k}{2}}\right)$, $\mathbb{M}$ calcula $v_i \in \{0,1\}$, donde v_i esta definido por:
$v_i = 0$ si y solo si la respuesta del oráculo a la consulta $\left(G,H,k,2^i\right)$ es NO, en otro caso $v_i = 1$.
(2) M calcula $n := \min\left\{i \leq \log\left(|G|^k 2^{\binom{k}{2}}\right) : v_i = 0\right\}$.
(3) M calcula e imprime 2^n.

Hecho: $n \in \{m, m+1\}$.

Se sigue inmediatamente de las siguientes observaciones:

(1) Si $i \leq m-1$, entonces $v_i = 1$.
(2) Si $i \geq m+1$, entonces $v_i = 0$.

Tenemos entonces que

- $2^n \in \left\{2^m, 2^{m+1}\right\}$.
- $\frac{1}{2}\left(p-\#EMB\,(G,H,k)\right) \leq 2^n \leq 2\left(p-\#EMB\,(G,H,k)\right)$.

De todo lo anterior tenemos que, el output de $\mathbb{M}$ es una aproximación a $\chi(x,k)$ dentro del rango 2 □

Hemos probado que el conteo aproximado de problemas en $\#W[1]$ es reducible a $Gap-GM$. Por otro lado podemos probar que $Gap-GM$ es reducible al conteo aproximado de problemas en $\#W[1]$.

Teorema 3. *$Gap-GM$ es reducible a $p-\#EMB$.*

Proof. Sea (G,H,k,c) una instancia de $Gap-GM$. Sea d un número racional tal que $1 \lneqq d \lneqq \sqrt{2}$ y sea $\mathbb{M}$ un algoritmo que aproxima $p-\#EMB$ dentro del rango d. Sea $\mathbb{A}$ el algoritmo dado por:

Con input (G,H,k,c)

(1) $\mathbb{A}$ simula la computación de $\mathbb{M}$ con input (G,H,k).
(2) Dado t el output de $\mathbb{M}$ con input (G,H,k), Si $t \geq \frac{2c}{d}$ $\mathbb{A}$ acepta en caso contrario rechaza.

Note que, si (G,H,k,c) es una instancia SI, $t \geq \frac{2c}{d}$, por el contrario si (G,H,k,c) es una instancia NO, $t \leq cd \lneqq \frac{2c}{d}$, esto es $\mathbb{A}$ acepta las instancias SI y rechaza las instancias NO. □

Lo anterior indica que, el problema $Gap-GM$ y el problema de aproximar una función en $\#W[1]$ son Turing-equivalentes. En lo que queda del escrito probaremos dos cosas, a saber:

(1) $Gap-GM$ es probabilísticamente reducible a todo problema $W[1]$ completo.
(2) Todo problema $W[1]$ completo es reducible al problema de aproximar un problema $\#W[1]$ completo dentro del rango 3, (y por lo tanto dentro del rango 2). Esto es, hemos probado que

$$p-\#EMB \equiv_T Gap-PR$$

Y por otro lado probaremos que

$$Gap-PR \prec_R p-EMB \prec_T p-\#EMB$$

Lo que nos permitirá concluir que $Gap-GM$ es equivalente a $p-EMB$ desde el punto de vista de las reducciones probabilísticas.

3.2. **$Gap-GM$ y problemas en $W[1]$.** En esta sección probaremos que el problema $Gap-GM$ es reducible, usando reducciones probabilísticas, a un cierto problema en $W[1]$. Tal teorema implica que $Gap-GM$ es reducible, usando reducciones probabilísticas, a todo problema $W[1]$ completo.

La técnica que usaremos para probar el teorema es una típica aplicación de *Hashing*. A este tipo de aplicación de *Hashing* lo llamaremos reducciones de Stockmeyer. Las reducciones de Stockmeyer nos permiten transformar una dicotomía de la forma

O existen muchos certificados (más que $2c$) o existen pocos certificados (menos que c).

En una dicotomía de la forma

La probabilidad de que exista al menos un certificado es o muy alta (mayor que $\frac{3}{4}$) o muy pequeña (menor que $\frac{1}{4}$).

Note que este es precisamente el tipo de reducción (transformación) que necesitamos en la prueba. La prueba del teorema es similar a algunos apartes de la prueba de que $no-graph-isomorphism \in BP \cdot \exists \cdot P$ (ver [9]).

Notación 3. *Dados A y B dos conjuntos, B^A denotará el conjunto de las funciones de A en B.*

Definición 12. *Dados A, B dos conjuntos, con $|B| \leq |A|$, un conjunto $H_{A,B} \subset B^A$ es un conjunto U_2-Hashing si y solo si para todo $a, b \in A$, con $a \neq b$, y para todo $c, d \in B$*

$$\Pr_{h \in H_{A,B}} [h(a) = c \ \& \ h(b) = d] = \frac{1}{|B|^2}$$

Dados $n, r \in \mathbb{N}$ con $n \geq r$ y dado $z \in \{0,1\}^n$, $(z) \restriction_r$ es el vector booleano cuyas entradas son las r primeras entradas de y. El conjunto

$$H^*_{n,r} := \{h_{a,b} : a, b \in \{0,1\}^n \ \& \ h_{a,b}(z) := (az + b) \restriction_r\}$$

Es un conjunto U_2-Hashing (ver [7]), (las operaciones aritméticas son calculadas en el campo de Galois $\mathbb{GF}(2^n)$).

Nota 3. *Es importante destacar que, el número de bits necesarios para especificar una función $h \in H^*_{n,r}$, es $O(n)$, dado que para especificar una función $h \in H^*_{n,r}$ es suficiente especificar un par $(a, b) \in \{0,1\}^{2n}$.*

Dado $S \subseteq GF(2^n)$ consideraremos la variable aleatoria Y_i definida por

$$Y_r(h) := \left|S \cap h^{-1}(0^r)\right|$$

donde 0^r es el vector cero en $\mathbb{GF}(2^r)$.

Para $E(Y_r)$, el valor esperado de Y_r, tenemos que

Hecho 1. $E(Y_r) = \frac{|S|}{p^r}$.

Para una prueba el lector puede consultar [7]. En lo que sigue denotaremos al valor esperado $E(Y_r)$ de Y_r con el símbolo ρ_r.

Lema 4 (Lema del resto)**.** *Dado $S \subset \mathbb{GF}(2^n)$ y $\epsilon \gneqq 0$*

$$\Pr[|Y_r(h) - \rho_r| \geq \epsilon\rho_r] \leq \frac{1}{\epsilon^2 \rho_r}$$

La prueba del Lema del resto puede ser consultada en [7].

Hecho todo lo anterior podemos empezar con la prueba del teorema. Note que todo subgrafo de G de tamaño k puede ser adecuadamanete codificado usando a lo más $2k^2 \log(|G|)$ bits. En lo que sigue identificaremos al conjunto de los subgrafos de G de tamaño k con un subconjunto de $\{0,1\}^{2k^2 \log(|G|)}$. Sean $n = 2k^2 \log(|G|)$ y $m = \log\left(4c^6\right)$.

Problema 8 $(p - HashingEMB)$**.** .

- *Input:* (G,H,r). *G y H son grafos y $r \in H^*_{n,m}$, ($n = 2k^2 \log(|G|)$ y $m = \log\left(4c^6\right)$).*

- *Parámetro:* $|H|$.
- *Problema: Decida si existen* $H_1, ..., H_6$ *subgrafos de* G *tales que, cada uno de ellos es isomorfo a* H *y* $r(H_1, ..., H_6) = 0^m$, *donde* r *es aplicada a los códigos de* $H_1, ..., H_6$ *los cuales por hipótesis son elementos de* $\{0,1\}^{f(k)\log(|G|)}$.

Es fácil verificar que el problema $p - HashingEMB$ pertenece a $W[1]$. En lo que sigue probaremos que $Gap - GM$ es probabilísticamente reducible a $p - HashingEMB$.

Dada (G, H, k) una instancia de $Gap-GM$, $\Omega^6((G, H, k))$ es el conjunto definido por $\{(H_1, ..., H_6) : H_1, ..., H_6$ son subgrafos de G isomorfos a $H\}$

Hecho 2. *Dada* (G, H, k) *una instancia de* $Gap - GM$

(1) *Si* (G, H, k) *es una instancia SI de* $Gap - GM$, *entonces*

$$|\Omega^6((G, H, k))| \geq 2^6 c^6$$

(2) *Si* $\Omega^6((G, H, k))$ *es una instancia NO de* $Gap - GM$, *entonces*

$$|\Omega^6((G, H, k))| \leq c^6$$

Lema 5. *Si* (G, H, k) *es una instancia SI de* $Gap - GM$, *entonces*

$$\Pr_{r \in H^*_{n,m}}\left[\exists y \left(y \in \Omega^6((G, H, k)) \ \& \ r(y) = 0^m\right)\right] \geq \frac{3}{4}$$

Proof. Dado (G, H, k) una instancia SI de $Gap - GM$, Y_m es la variable aleatoria definida por

$$Y_m(r) := \left|\Omega^6((G, H, k)) \cap r^{-1}(0^m)\right|$$

donde $r \in H^*_{n,m}$. Para el valor esperado ρ_m de Y_m tenemos que $\rho_m \geq 16$. Si usamos el lema del resto, eligiendo $\epsilon = \frac{1}{2}$, obtenemos lo siguiente

$$\Pr_{r \in H^*_{n,m}}[Y_m(r) = 0] \leq \Pr_{r \in H^*_{n,m}}\left[|Y_m(r) - \rho_m| \gneqq \frac{1}{2}\rho_m\right] \leq \frac{1}{4}$$

□

Lema 6. *Si* (G, H, k) (x, k, c) *es una instancia NO de* $Gap - GM$, *entonces*

$$\Pr_{r \in H^*_{n,m}}\left[\exists y \left(y \in \Omega^6((G, H, k)) \ \& \ r(y) = 0^m\right)\right] \leq \frac{1}{4}$$

Proof. Primero note que, para todo $y \in \{0,1\}^{6\cdot h}$, $\Pr_{r \in H^*_{n,m}}[r(y) = 0^m] \leq 2^{-m}$

De lo anterior tenemos que

$$\begin{aligned}
\Pr_{r \in H^*_{n,m}}\left[\exists y \in \Omega^6((G, H, k)) \ (\ r(y) = 0^m)\right] &\leq \\
\sum_{y \in \Omega^6((G,H,k))} \Pr_{r \in H^*_{n,m}}[r(y) = 0^m] &\leq \\
\left|\Omega^6((G, H, k))\right| 2^{-m} &\leq \\
c^6 2^{-m} &\leq \frac{1}{4}
\end{aligned}$$

□

Si la noción de reducción considerada es la de reducibilidad aleatoria, de lo anterior obtenemos el siguiente corolario.

Corolario 1. .

(1) $Gap - GM$ *es reducible a* $p - HashingEMB$.
(2) $Gap - GM$ *es reducible a todo problema* $W[1]$*-completo.*
(3) $Gap - GM$ *es reducible a* $p - EMB$.

3.2.1. $W[1]$ *y conteo aproximado.* En esta corta subsección probaremos que todo problema en $W[1]$ es reducible al problema de aproximar $p - \#EMB$ dentro del rango 3. Como un corolario de este teorema y de todo lo anterior obtendremos que la complejidad paramétrica de $Gap - GM$ es equivalente a la complejidad paramétrica de $p - EMB$, (desde el punto de vista de las reducciones aleatorias).

Problema 9 ($p - EMB$)**.** .

- *Instancia:* (G, H, k), G *y* H *son grafos,* $|H| = k$ *y* $k \in \mathbb{N}$.
- *Parámetro:* k.
- *Problema: Decida si* $|\{R \subset G : R \simeq H\}| \neq 0$

Teorema 4. *Dado* L *un problema en* $W[1]$*,* L *es reducible al problema de aproximar* $p - \#EMB$ *dentro del rango* 3.

Proof. Dada (x, k) una instancia de L es posible calcular en tiempo fpt una instancia (G, H, k) de $p - EMB$, tal que $(x, k) \in L$ si y solo si $(G, H, k) \in p - EMB$, (dado que $p - EMB$ es $W[1]-$completo). Suponga que $\mathbb{M}$ es un algoritmo que aproxima la $\#W[1]$-función $p - \#EMB$ dentro del rango 3. Sea $\mathbb{A}$ el algoritmo definido por:

Con input (x, k)

(1) Calcule (G, H, k).
(2) Simule la computación de $\mathbb{M}$ en el input (G, H, k).
(3) Si $\mathbb{M}((G, H, k))$ es el output de $\mathbb{M}$ y $\mathbb{M}((G, H, k)) \geq \frac{2}{3}$ imprima $(x, k) \in L$. En caso contrario imprima $(G, H, k) \notin L$.

Es claro que, el tiempo de cómputo de $\mathbb{A}$ está acotado por $f(k)\, p(|x|)$ para alguna función computable f y algún polinomio p. Para terminar note simplemente que $\mathbb{M}((G, H, k)) \geq \frac{2}{3}$ si y solo si $(G, H, k) \in p - EMB$ si y solo si $(x, k) \in L$ □

Corolario 2. *Dado* L *un problema en* $W[1]$*,* L *es reducible al problema de aproximar* $p - \#EMB$ *dentro de un rango* $c \gneqq 1$.

Corolario 3. $Gap - GM \equiv_R p - EMB$.

Proof. Hasta el momento hemos probado que:

(1) El problema de aproximar $p - \#EMB$ dentro de un rango constante dado es reducible a $Gap - GM$.
(2) $Gap - GM \prec_R p - EMB$.
(3) $p - EMB$ es reducible al problema de aproximar $p - \#EMB$ dentro de un rango dado.

Dado que la relación de reducibilidad es transitiva tenemos que: $Gap - GM \prec_R p - EMB$ y $p - EMB \prec_R Gap - GM$. Esto es, hemos probado que $Gap - GM \equiv_R p - EMB$ □

4. Conclusiones y problemas

En este artículo hemos probado que, el problema $Gap - GM$, (el cual es una abstracción del problema de Minería de Grafos, consistente en listar los patrones frecuentes y los patrones excepcionales de un grafo dado), es exactamente tan difícil como el problema de aproximar $p-\#EMB$ dentro del rango 2. Como el problema $p-\#EMB$ es $\#W$ [1]-completo, podemos concluir que $Gap-GM$ es un problema intratable.

Del problema $Gap - GM$ podemos considerar diferentes restricciones. En algunas aplicaciones podemos suponer que el grafo input G no es un grafo arbitrario sino que pertenece a una clase especial de grafos. En otras aplicaciones podemos suponer que el grafo parámetro H no es un grafo arbitrario sino que pertenece a alguna clase especial de grafos. Dada $\mathcal{K}$ una clase de grafos, podemos considerar dos restricciones de $Gap - GM$ determinadas por $\mathcal{K}$.

Problema 10 ($Gap - GM(\mathcal{K})$)**.** .

- (G, H, k, c). *G y H son grafos, $|H| = k$, $G \in \mathcal{K}$ y $c \in \mathbb{N}$.*
- *Parámetro: k.*
- *Instancias SI: (G, H, k, c) es una instancia SI si y solo si H es c-frecuente.*
- *Instancias NO: (G, H, k, c) es una instancia NO si y solo si H es c-excepcional.*

Problema 11 ($Gap - GM[\mathcal{K}]$)**.** .

- (G, H, k, c). *G y H son grafos, $|H| = k$, $H \in \mathcal{K}$ y $c \in \mathbb{N}$.*
- *Parámetro: k.*
- *Instancias SI: (G, H, k, c) es una instancia SI si y solo si H es c-frecuente.*
- *Instancias NO: (G, H, k, c) es una instancia NO si y solo si H es c-excepcional.*

Un problema de importancia en la práctica, y aún por resolver, es el problema de *cartografiar* la intratabilidad del problema $Gap - GM$, esto es:

(1) Caracterizar las clases $\mathcal{K}$ para las cuales $Gap - GM(\mathcal{K})$ es tratable.
(2) Caracterizar las clases $\mathcal{K}$ para las cuales $Gap - GM[\mathcal{K}]$ es tratable.

Una manera, que consideramos viable, de enfrentar este problema es explotar la relación existente entre $Gap - GM$ y $p - \#EMB$. Dada $\mathcal{K}$ una clase de grafos, podemos definir dos restricciones de $p - \#EMB$ determinadas por $\mathcal{K}$.

Problema 12 ($p - \#_2 EMB(\mathcal{K})$)**.** .

- *Instancia: (G, H, k), G y H son grafos, $|H| = k$ y $G \in \mathcal{K}$.*
- *Parámetro: k.*
- *Problema: Aproxime $|\{R \subset G : R \simeq H\}|$ dentro del rango 2.*

Problema 13 ($p - EMB[\mathcal{K}]$)**.** .

- *Instancia: (G, H, k), G y H son grafos, $|H| = k$ y $H \in \mathcal{K}$.*
- *Parámetro: k.*
- *Problema: Aproxime $|\{R \subset G : R \simeq H\}|$ dentro del rango 2.*

Note que en las reducciones presentadas en las secciones anteriores, los grafos input y grafos parámetro no se transforman, esto es, en las reducciones presentadas si se parte, por ejemplo, de una instancia $((G, H, k, c))$ de $Gap - GM$ y $G, H \in \mathcal{K}$, entonces todas las instancias calculadas en la reducción de $Gap - GM$ en $p - EMB$ son de la forma (G^*, H^*, k^*) con $G^*, H^* \in \mathcal{K}$. Esto nos permite afirmar que:

(1) Para toda clase $\mathcal{K}$, $Gap - GM(\mathcal{K})$ es tratable si y solo si $p - \#_2 EMB(\mathcal{K})$ es tratable.
(2) Para toda clase $\mathcal{K}$, $Gap - GM[\mathcal{K}]$ es tratable si y solo si $p - \#_2 EMB[K]$ es tratable.

Lo cual a su vez nos permite reducir el problema de cartografiar la tratabilidad de $Gap - GM$ a cartografiar la tratabilidad de aproximar $p - \#EMB$ dentro del rango 2. Lo interesante del cuento es que, aunque no se ha cartografiado la tratabilidad de $p - \#EMB$, existe trabajo (ver [8] y [2]) en esta dirección. Cerraremos este escrito proponiendo una conjetura, la cual si fuera resuelta de manera positiva, nos permitiría resolver el problema antes propuesto.

Conjetura 1. .

(1) *$p - \#_2 EMB(\mathcal{K})$ es tratable si y solo si $\mathcal{K}$ es una clase de arboricidad, (treewidth (ver [3])), acotada.*
(2) *$p - \#_2 EMB[\mathcal{K}]$ es tratable si y solo si $\mathcal{K}$ es una clase de arboricidad acotada.*

Agradecimientos 1. *Dedicado a Mamadimitriou e Hijodimitriou. El autor agradece a la Universidad Industrial de Santander por brindarle las facilidades para hacer posible la realización de este escrito. Esta investigación fue realizada con el auspicio de la VIE-UIS, proyecto número 5153-2007.*

Referencias

[1] V. Arvind and V. Raman. Approximation algorithms for some parameterized counting problems. In P. Bose and P. Morin, editors, *Proceedings of the 13th Annual International Symposium on Algorithms and Computation,* Volume 2518 of *Lecture Notes in Computer Science,* pages 453-464. Springer-Verlag, 2002.
[2] V. Dalmau and P. Jonsson. The complexity of counting homomorphism seen from the other side. *Theoretical Computer Science,* 329:315-323, 2004.
[3] R. Diestel. Graph theory. Springer Verlag 2005.
[4] R.G. Downey and M.R. Fellows. Parameterized Complexity. Springer-Verlag, 1999.
[5] J. Flum and M. Grohe. The parameterized complexity of counting problems. *SIAM Journal of Computing,* 33(4):892-922, 2004.
[6] J. Flum and M. Grohe. Parameterized Complexity Theory. Springer-Verlag, 2006.
[7] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, Boston MA, 1996.
[8] M. Grohe. The complexity of homomorphism and constraint satisfaction problem seen from the other side, *Journal of the ACM*, 54(1), —,2007.
[9] J. Kobler; U Schoning and J Toran. The Graph isomorphism problem, its structural complexity. Springer Verlag, 1993.
[10] C.H. Papadimitriou. Computational Complexity. Addison-Wesley, 1994.
[11] S. Toda. *PP* is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing,* 20(5):865-877, 1991.

Universidad Industrial de Santander
E-mail address: juamonto@uis.edu.co, amontoyaa@googlemail.edu.co,