Estudio Comparativo de Algoritmos de Minería de Subgrafos Frecuentes

Santiago Bianco, Sebastian Martins, Ramón García-Martínez

Laboratorio de Investigación y Desarrollo en Ingeniería de Explotación de Información
Grupo de Investigación en Sistemas de Información. Universidad Nacional de Lanús
Remedios de Escalada, Buenos Aires, Argentina.
santiago.bianco.sb@gmail.com, smartins@gmail.com, rgm1960@yahoo.com

Resumen. Dentro las técnicas de minería de grafos se encuentran las correspondientes a búsqueda de subgrafos frecuentes. Existen varios algoritmos orientados a reconocer subestructuras comunes entre un conjunto de grafos entre los que destacan: FSG, FFSM, gSpan y GASTON. El objetivo de esta investigación es analizar el comportamiento de estos algoritmos a través de distintos experimentos diseñados para identificar si existe un algoritmo superior al resto y, en caso de que no lo haya, poder definir en qué escenarios es más recomendable la elección de cada uno.

Palabras clave. Minería de grafos, minería de subgrafos frecuentes, ingeniería de explotación de información, análisis de algoritmos.

1. Introducción

La búsqueda de subgrafos frecuentes se basa en encontrar estructuras recurrentes en un conjunto de grafos, es decir, buscar subgrafos que se repitan en una base de datos compuesta por grafos. El descubrimiento de estos patrones puede ser el propósito final del proceso o los subgrafos descubiertos pueden ser parte de otro proceso de clasificación [1]. Para determinar que un subgrafo es frecuente, tiene que superar determinado umbral, denotado como support. La definición formal es la siguiente: dado un conjunto de grafos GD y un umbral σ (o threshold en inglés), de manera que $0 < \sigma \leq 1$, el soporte de un grafo G, denotado como supG, es igual a la cantidad de grafos en GD en los cuales G es un sub-isomorfismo. Escrito como fórmula sería:

$$Sup_G = \frac{|\{G' \in GD \mid G \subseteq G'\}|}{|GD|}$$

Teniendo en cuenta esto, un grafo G es frecuente en una base de datos de grafos si $supG \geq \sigma$, siendo σ el soporte mínimo o minimum support. Por lo tanto, el problema de FSM se resume en, dado un umbral σ y un conjunto de grafos GD, encontrar todos los subgrafos frecuentes G en GD que cumplan con $sup_G \geq \sigma$.
Existen una variedad de algoritmos para realizar esta tarea, entre los que destacan: algoritmo FSG [6], algoritmo gSpan (graph-based Subestructure pattern mining) [2],

lgoritmo FFSM (Fast Frequent Subgraph Mining) [3], y algoritmo GASTON (GrAph/Sequence/Tree extractiON) [7].

Algoritmo FSG (Frecuent Sub-Graph) [6]: es el algoritmo más antiguo de los que se evalúan en este trabajo, por lo que introdujo algunas características que luego serían usados por el resto, que tienen que ver con la forma de representación de los grafos, la generación de subestructuras candidatas y detección de isomorfismos. En primer lugar, utiliza un tipo de representación para grafos dispersos que minimiza costos de procesamiento y almacenamiento. Este tipo de representación es usada para almacenar candidatos intermedios y los subgrafos que se van encontrando. Consiste en transformar las representaciones canónicas de los grafos, inicialmente representadas con matrices de adyacencia, para implementarlos como listas de adyacencia, las cuales disminuyen el uso de memoria y del procesador para estructuras dispersas. En segundo lugar, incrementa el tamaño de los subgrafos a buscar de a una arista por vez, permitiendo que la generación de candidatos sea más eficiente. Finalmente, usa algoritmos simples para implementar las representaciones canónicas de los grafos y las detecciones de isomorfismos que funcionan de manera eficiente para grafos chicos, e incorpora varias optimizaciones para el proceso de generación de candidatos y conteo (determinación de la frecuencia de un subgrafo) que permiten que el algoritmo sea escalable para grandes conjuntos de grafos.

Algoritmo gSpan (graph-based Subestructure pattern mining) [2]: busca superar los inconvenientes que tienen los algoritmos que utilizan una estrategia apriorística como el FSG: el costo de la generación de candidatos y la detección de falsos positivos a la hora de la evaluación de isomorfismos. Como el problema de búsqueda de sub-isomorfismos está catalogado como NP-completo, es muy costoso computacional-mente volver a evaluar los resultados. Lo más destacado de este algoritmo es la implementación del recorrido en profundidad para reducir el espacio de búsqueda y la introducción de dos nuevas técnicas: los códigos DFS y DFSM para generar la representación canónica de los grafos. Otros aspectos destacables incluyen la eliminación de los procesos de generación de candidatos para el descubrimiento de subgrafos, así como también el recorte de falsos positivos. Además, combina los procedimientos de crecimiento y evaluación de subestructuras en uno solo, acelerando el proceso de búsqueda.

Algoritmo FFSM (Fast Frequent Subgraph Mining) [3]: utiliza el mismo enfoque de búsqueda en profundidad del gSpan, incorporando nuevas técnicas para mejorar su eficiencia como la utilización de la representación canónica CAM y nuevos métodos para agilizar el proceso de generación de candidatos mediante la manipulación de las matrices de adyacencia. Se introduce un procedimiento para garantizar que todos las subestructuras frecuentes sean enumerada unívocamente y sin ambigüedades (suboptimal CAM tree) y se evita el testeo de sub-isomorfismos, manteniendo una lista de cada subgrafo frecuente. Esta última es quizá la más relevante de todas las características, debido al gran potencial que puede llegar a tener el algoritmo al evitar ese procedimiento tan computacionalmente complejo.

Algoritmo GASTON (GrAph/Sequence/Tree extractiON) [7]: aprovecha un principio que llamaron ‘quickstart principle’, que considera siguiente hecho: los grafos, árboles y caminos están incluidos unos en otros, por lo que se puede dividir el proceso en distintos pasos de creciente complejidad, lo que simplifica el procedimiento general.

Primero, se buscan los caminos frecuentes, luego los árboles y finalmente los subgrafos frecuentes. Cada etapa tiene un proceso distinto para representar las estructuras en su forma canónica, debido a las distintas características que presentan dichas estructuras. Sin embargo, debido al quickstart principle antes mencionado, los códigos generados en una etapa pueden ser usados para la etapa siguiente, concatenando las nuevas ramificaciones encontradas en el caso de los árboles o los ciclos en el caso de los grafos. De esta manera, se reduce la complejidad del proceso.
Para el proceso de conteo de grafos se utiliza un procedimiento similar al del algoritmo FFSM, con listas que almacenan las subestructuras ya analizadas. Debido a problemas de escalabilidad con este método, también proponen utilizar un proceso alternativo para grafos de gran tamaño, similar al utilizado en el algoritmo FSG.
En este contexto, emerge el interés de analizar el estudio comparado del comportamiento de estos algoritmos. En la sección 2 se describe el problema, se describe en la sección 3 la formulación del diseño experimental con detalle de los experimentos con grafos sintéticos (sección 3.1) y experimentos con grafos reales (sección 3.2), en la sección 4 se presenta los resultados discriminando el análisis de las pruebas con grafos sintéticos (sección 4.1) de las realizadas con grafos reales (sección 4.2), en la sección 5 se enuncian las conclusiones preliminares y las futuras lineas de trabajo.

2. Descripción del Problema

Se han desarrollado varios algoritmos de minería de grafos en los últimos años [3; 12]. Los mismos, varían en la estrategia que utilizan para recorrer los grafos, el tipo de entrada que utilizan y la información de salida que proveen. Para dar soluciones más eficientes a los problemas, es de interés compararlos y determinar su comportamiento en distintos escenarios de manera que se pueda elegir la mejor opción en base a los datos con los que se cuente. En esta investigación se evaluarán los algoritmos FFSM, FSG, gSpan y GASTON, los cuales buscan estructuras considerando una frecuencia de ocurrencia mínima determinada y como resultado generan una lista con las subestructuras que cumplan con esa condición. Los mismos fueron elegidos ya que cada uno presenta novedosos enfoques y son los más reconocidos en el estado del arte. Si bien se han hecho comparaciones entre algoritmos [6; 7], las mismas sólo comparan el tiempo de ejecución de los algoritmos en base a distintas estructuras y definiendo cuál es el más rápido. Se considera de mayor interés ampliar las investigaciones determinando bajo qué circunstancias es más eficiente uno que otro, no solo en base al tiempo que demore sino también en la cantidad de patrones que pueda encontrar.
De lo anteriormente expuesto surgen la siguiente pregunta de investigación: ¿es posible definir distintos escenarios experimentales de manera que se pueda determinar bajo qué circunstancias es más eficiente la utilización de un algoritmo por sobre otro, de manera que se pueda facilitar la elección de los mismos dependiendo de las necesidades que se tengan y los datos disponibles?

3. Formulación del Diseño Experimental

En las siguientes secciones, se provee una descripción más detallada de los experimentos que se llevan a cabo y sus diferentes variantes: con grafos sintéticos y con grafos reales. Posteriormente, se establecen las variables dependientes e independientes involucradas en cada una de las pruebas a realizar tanto para los grafos sintéticos como para los grafos reales.

3.1. Experimentos con grafos sintéticos

Se prueban los algoritmos con datos generados aleatoriamente, variando la cantidad de nodos y arcos. Se utilizan conjuntos de cien grafos en todas las pruebas. Las pruebas se repiten cien veces y los resultados finales son un promedio de los resultados parciales. Este tipo de escenario a su vez contiene otros dos, que varían de acuerdo a cómo se incrementa la cantidad de arcos experimento a experimento. Estas pruebas se dividen en: (i) experimentos con incremento fijo, y (ii) experimentos con incremento variable.
Dentro de estos dos últimos hay otras variaciones que consisten en probar los algoritmos en conjuntos de grafos en los que todos sus nodos son diferentes, así como también en estructuras con repeticiones en las etiquetas de los elementos que los componen. Teniendo en cuenta esto, cada experimento a su vez incluye otros dos:

- Experimentos sin repetición de nodos: todos los vértices de los grafos contienen etiquetas distintas, de manera que cada elemento sea distinto del resto. Un ejemplo de un grafo con diez nodos y diez arcos que podría formar parte de este tipo de pruebas puede observarse en la figura 1. En todos los experimentos con grafos sintéticos las pruebas se realizan con grafos no dirigidos y cuyas aristas pueden tener tres tipos de etiquetas: “1”, “2” o “3”.
- Experimentos con repetición de nodos: en estos experimentos se ejecutaron los algoritmos con bases de datos en las cuales los vértices de los grafos contenían etiquetas repetidas, de manera que se puedan modelar situaciones en las cuales existan dos o varios objetos iguales dentro de la red. En estas pruebas, se utiliza una variable llamada nodos únicos que es igual a |V|/2, la cual determina la cantidad máxima de vértices con distintas etiquetas que puede tener un grafo. Por ejemplo, si la variable nodos únicos tiene un valor de 5, quiere decir que los nodos tendrán etiquetas desde el 0 al 4. Con esto se garantiza que haya elementos repetidos. Un ejemplo de un grafo con diez nodos, diez arcos y cinco nodos únicos que podría formar parte de este tipo de pruebas puede observarse en la Figura 2.

Se consideran las siguientes variables independientes para cada uno de estos procesos:

Cantidad de Nodos: Indica el total de nodos que tiene cada grafo del conjunto de prueba en cada experimento. Su valor es de 10 en la primera prueba y se incrementa a 100 en la última, con incrementos de a diez por cada cinco pruebas.

Cantidad de Arcos: Indica el total de arcos que tiene cada grafo del conjunto de prueba en cada experimento. Difiere dependiendo del tipo de experimento. Para aquellos con incremento fijo, su valor es de 10 en la primera prueba y se incrementa hasta 120 en la última, con incrementos de a cinco y reajustando su

valor por cada cinco pruebas. Para aquellos con incremento variable, comienza en 10 y se incrementa hasta 300. Los incrementos se hacen según la fórmula *(Cantidad de Nodos / 2)*.

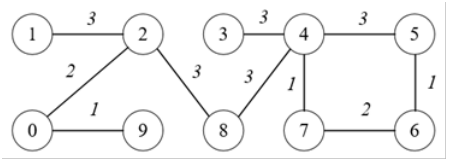

Fig. 1. Ejemplo de grafo, Pruebas con grafos sintéticos sin repetición de nodos

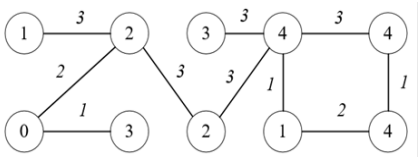

Fig. 2. Ejemplo de grafo, Pruebas con grafos sintéticos con repetición de nodos.

Nodos Únicos: Indica la cantidad máxima de nodos sin repetirse que puede haber en cada grafo. Es utilizada para las pruebas con repeticiones de nodos. Su valor se corresponde con la fórmula (*Cantidad de Nodos / 2),* garantizando que siempre haya nodos con etiquetas repetidas.

Cantidad de grafos: Indica la cantidad de grafos que tendrán los datos generados para las pruebas. Su valor para todos los experimentos de 100.

Minimum Support Threshold o Umbral de Frecuencia Mínimo: Indica la cantidad mínima de grafos que deben contener a una subestructura para considerarla de frecuente. En estos experimentos el valor se fija en 5% para todos los algoritmos.

A su vez, se busca comparar las siguientes variables dependientes para poder evaluar el comportamiento de los algoritmos en cada experimento:

Cantidad de subestructuras: Indica la cantidad de subestructuras frecuentes que los algoritmos encuentran en cada experimento. Con esto se busca identificar en qué situaciones un algoritmo puede llegar a ser más útil que el resto.

Tiempo de ejecución: Indica cuánto demora cada algoritmo en conseguir resultados. Esta variable será medida en segundos.

Cotejando ambas variables en conjunto se busca analizar el comportamiento global de cada algoritmo según las características del conjunto de datos en los que se apliquen.

3.2. Experimentos con grafos reales

En estas pruebas se ejecutan los algoritmos sobre un dataset real Compounds_422 distribuido con la implementación de gSpan. El mismo es utilizado por varios autores para testear el rendimiento de sus algoritmos [5, 7]. Sus características pueden verse en la Tabla 1. Sobre esta base de datos se ejecutan pruebas cien veces y se promedian los resultados.

Para estas pruebas se considera únicamente a la siguiente variable independiente:

Minimum Support Threshold o Umbral de Frecuencia Mínimo: Indica la cantidad mínima de grafos que deben contener a una subestructura para considerarla de frecuente. En estos experimentos el valor se fija en 25% para la primera prueba y aumenta 5% por cada prueba hasta llegar a 95%.

Tabla 1. Características del archivo a utilizar en las pruebas con grafos reales.

Compounds_422: Archivo de prueba con estructuras moleculares	
Cantidad de grafos	422
Cantidad de etiquetas de arcos distintas	4
Cantidad de etiquetas de nodos distintas	21
Promedio de arcos por grafo	42
Promedio de nodos por grafo	40
Cantidad máxima de arcos por grafo	196
Cantidad máxima de nodos por grafo	189

Además, Se busca comparar las siguientes variables dependientes para poder evaluar el comportamiento de los algoritmos en cada experimento:

Cantidad de subestructuras: Indica la cantidad de subestructuras frecuentes que los algoritmos encuentran en cada experimento.

Tiempo de ejecución: Indica cuánto demora cada algoritmo en conseguir resultados. Esta variable será medida en segundos.

Cotejando ambas variables en conjunto se busca analizar el comportamiento global de cada algoritmo en un conjunto de datos reales, incrementando en cada paso el umbral mínimo.

4. Resultados

A continuación se realiza un análisis de los resultados obtenidos por cada tipo de prueba y se presentan las observaciones desprendidas de cada uno.

4.1. Análisis de las pruebas con grafos sintéticos

Para el análisis de los resultados obtenidos a partir de las pruebas con nodos únicos, se unificaron los experimentos con incremento fijo y variable para así realizar una comparación global, generando un total de 82 distintos escenarios al combinar los dos tipos de prueba. En estos escenarios se varía el promedio de nodos y arcos del conjunto de grafos en el siguiente orden (teniendo en cuenta <aristas promedio, arcos promedio>):

(10,10), (10,15), (10,20), (10,25), (10,30), (20,20), (20,25), (20,30), (20,35), (20,40), (20,50), (20,30), (30,30), (30,35), (30,40), (30,45), (30,50), (30,60), (30,75), (30,90), (40,40), (40,45), (40,50), (40,55), (40,60), (40,80), (40,100), (40,120), (50,50), (50,55), (50,60), (50,65), (50,70), (50,75), (50,100), (50,125), (50,150), (60,60), (60,65), (60,70), (60,75), (60,80), (60,90), (60,120), (60,150), (60,180), (70,70), (70,75), (70,80), (70,85), (70,90), (70,105), (70,140), (70,175), (70,210), (80,80), (80,85), (80,90), (80,95), (80,100), (80,120), (80,160), (80,200), (80,240), (90,90), (90,95), (90,100), (90,105), (90,110), (90,135), (90,180), (90,225), (90,270), (100,100), (100,105), (100,110), (100,115), (100,120), (100,150), (100,200), (100,250), (100,300)

En la figura 3 se muestra el gráfico comparativo. En el eje de abscisas se presentan los resultados expresados en estructuras/milisegundos. De esta manera, puede observarse el desempeño general de cada algoritmo a medida que se incrementa el tamaño de las bases de datos, y como son afectados por la variación en la densidad de los grafos que las componen. Analizando el gráfico puede notarse que el algoritmo GASTON es superior al resto para los primeros experimentos, en los que las bases son relativamente chicas. Sin embargo, su rendimiento decae abruptamente en las pruebas

11 y 12, en las cuales se incrementa la cantidad de aristas. En estos escenarios, la implementación directamente no es capaz de encontrar ninguna subestructura y cesa su ejecución, mientras que el resto de los algoritmos sí lo hace. Este comportamiento es recurrente cada vez que se aumenta la densidad de los grafos (pruebas 18, 25, 35, 42, 52, 60, 70 y 78). En el resto de las pruebas, cuando es capaz de encontrar estructuras, sus resultados son mejores que los que producen los otros algoritmos. Los otros dos algoritmos, el gSpan y el FSG son más lentos pero son capaces de encontrar resultados en todos los escenarios, incluso en las últimas pruebas. También se puede observar que sus comportamientos son más lineales que los del resto.

Comparando al gSpan y FSG, éste último obtiene mejores resultados en gran parte de las pruebas, pero esta tendencia se revierte en los experimentos finales, a partir del número 48. Desde allí, el gSpan es ligeramente superior, y luego de la prueba 78 el gSpan supera no sólo al FSG sino a los demás algoritmos también.

En el caso de las pruebas con repetición de nodos, para el análisis de los algoritmos también se combinarán los resultados obtenidos de las pruebas con incremento fijo e incremento variable al igual que en los experimentos anteriores, siempre recordando que la cantidad máxima de etiquetas distintas va a ser igual a la mitad de la cantidad de vértices (|V|/2). El gráfico resumen de los resultados obtenidos se puede observar en la figura 4.

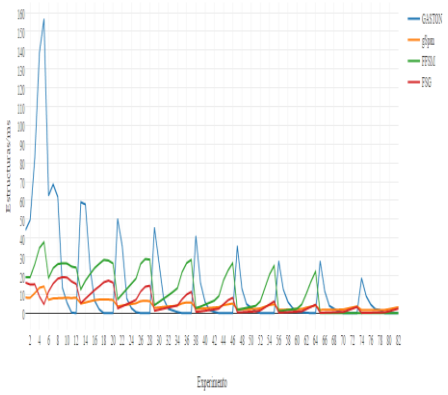

Fig. 3. Gráfico resumen de los resultados de los experimentos con grafos sintéticos con nodos únicos

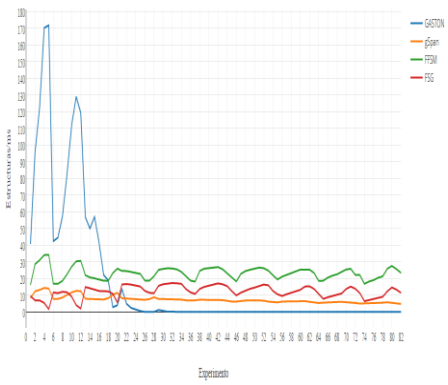

Fig. 4. Gráfico resumen de los resultados de los experimentos con grafos sintéticos con nodos repetidos.

Analizando el gráfico generado a partir de los resultados, a simple vista se puede observar un cambio en el comportamiento de los algoritmos con respecto a los presentados en las pruebas anteriores. El GASTON sigue siendo muy superior para las primeras pruebas pero hasta la prueba 17, luego su rendimiento decae y a partir de la prueba 24 no es capaz de encontrar subestructuras, exceptuando la prueba 29. Se siguen observando dificultades a medida que aumenta la densidad de la base de datos, aunque ahora este problema se ve más acentuado. El resto de los algoritmos se comporta de manera más constante que en las pruebas anteriores y todos son capaces de encontrar resultados hasta el final, siendo el FFSM el que encuentra más resultados en menor tiempo de comienzo a fin. En las primeras pruebas el gSpan y el FSG arrojan

resultados cambiantes hasta la prueba 20, en la cual ambos se estabilizan y el FSG consigue tener un mejor rendimiento.

4.2. Análisis de las pruebas con grafos reales

Mirando la conformación de la base (Tabla 1) puede notarse que la estructura de los grafos es muy variada, teniendo en cuenta que el grafo con mayor cantidad de vértices posee 189 y el promedio es de 40. Considerando sólo los promedios y haciendo una analogía con las pruebas con grafos sintéticos, esta prueba podría clasificarse dentro de los experimentos con nodos repetidos, aproximadamente entre las pruebas 13 y 21 analizadas anteriormente. En la figura 5 se presentan los resultados obtenidos, contrastando el minimum support de cada prueba con la cantidad de estructuras encontradas por milisegundo. A simple vista puede verse como el algoritmo GASTON supera ampliamente al resto de los algoritmos en las pruebas. Esto se debe a que las estructuras que conforman la base de datos no son tan densas y tal como ocurrió en las pruebas con grafos sintéticos, es estos escenarios el GASTON es el que se comporta de manera más eficiente. Para analizar lo ocurrido con el resto de los algoritmos con más detalle se presenta la figura 6. En la misma se observa que el algoritmo gSpan es capaz de encontrar resultados en todas las pruebas y más eficientemente que el FSG. Este último deja de encontrar resultados cuando el support es de 95%. El algoritmo FFSM, a pesar de haber conseguido obtener más resultados por unidad de tiempo que el resto con excepción del GASTON en las pruebas con grafos sintéticos, no pudo encontrar resultados con esta base de datos. Tal vez se deba al tamaño de la base en relación al support utilizado ya que en las pruebas con grafos sintéticos, el umbral mínimo era del 5% y las bases estaban compuestas por 100 grafos, en lugar de los 400 que se utilizaron ahora. Hay menor probabilidad de hallar grafos frecuentes con un support elevado a medida que aumenta el tamaño de la base.

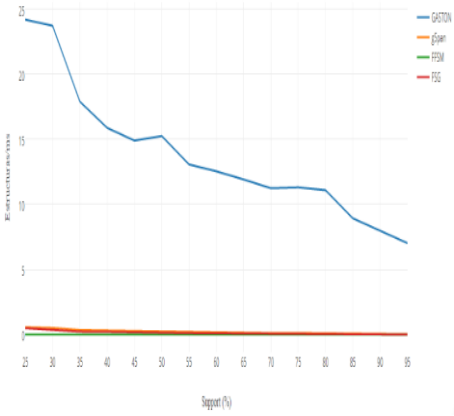

Fig. 5. Análisis de los resultados para las pruebas con grafos reales, contrastando el minimum support porcentual contra las estructuras encontradas.

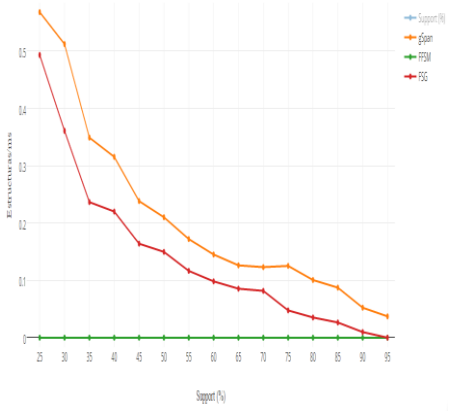

Fig. 6. Detalle de resultados para las pruebas con grafos reales de los algoritmos gSpan, FFSM y FSG.

5. Conclusiones Preliminares y Futuras Lineas de Trabajo

De lo expuesto en las secciones anteriores se pueden derivar las siguientes conclusiones con respecto a los experimentos realizados:

- La conformación de la base altera el rendimiento de los algoritmos. No es lo mismo tener estructuras con elementos únicos, como una red social, a tener estructuras con elementos repetidos, como puede ser una molécula. El tamaño de los grafos y la densidad de los mismos también afecta al comportamiento de todos los algoritmos.
- El algoritmo GASTON encuentra más resultados en menor tiempo que el resto para una base de datos compuesta por grafos chicos, de hasta aproximadamente 30 vértices y 60 aristas, pero se ve fuertemente afectado al aumentarse la densidad de la base y cuando se tienen muchos nodos repetidos.
- El algoritmo FFSM es el segundo algoritmo que más estructuras encuentra por segundo en las primeras pruebas, y el más eficiente luego de que disminuye el rendimiento del GASTON. Su rendimiento es mejor para las pruebas con grafos repetidos, ya que siempre es capaz de encontrar resultados. En los otros experimentos hay varias pruebas en la que no puede (56 y de la 66 en adelante).
- El comportamiento de los algoritmos gSpan y FSG son muy similares en las pruebas con nodos únicos, siendo el gSpan superior hacia las pruebas finales. En el caso de los experimentos con vértices repetidos, el FSG es mejor a partir de la prueba 20 hasta el final.

En resumen, se podría decir que el algoritmo GASTON es el más apropiado cuando las bases de datos no son muy densas (entre 30 vértices y 60 aristas en promedio aproximadamente) y no poseen mucha complejidad. Si las estructuras son complejas pero con un tamaño moderado (hasta 70 vértices y 90 aristas en promedio aproximadamente), el algoritmo FFSM es la mejor opción, sobre todo si los grafos cuentan con etiquetas repetidas. En el caso de grandes estructuras (más de 80 vértices y 100 aristas en promedio), las mejores opciones son el FSG y el gSpan, siendo este último el único que pudo encontrar subestructuras en todos los experimentos, por lo que se lo considera el más estable.

De las pruebas con grafos reales se pudo verificar también el hecho de que los algoritmos siempre encontraron la misma cantidad de subestructuras en cada iteración de las pruebas, por lo que no es necesario ejecutar los algoritmos reiteradas veces para tener resultados confiables una vez que se hayan definido los parámetros a utilizar.

En cuanto al trabajo futuro, la minería de grafos es un campo de constante crecimiento en la cual podrían desarrollarse las siguientes líneas de investigación:

- Ampliar los algoritmos a evaluar incluyendo otros como el SPIN [12] o CloseGraph para verificar su comportamiento.
- Ampliar las pruebas con grafos reales de manera similar a lo realizado con grafos sintéticos: usar estructuras más simples y más complejas a la base de datos utilizada en esta investigación y comparar los resultados.
- Desarrollar otras pruebas que involucren conjuntos de datos distintos en cuanto al tamaño o la conformación de las bases de datos. Podrían utilizarse grafos dirigidos o en el caso de que se usen bases de datos reales, podrían utilizarse otro tipo de estructuras que no sean moléculas.

Financiamiento

Las investigaciones que se reportan en este artículo han sido financiadas parcialmente por el Proyecto de Investigación 33A205 de la Secretaria de Ciencia y Técnica de la Universidad Nacional de Lanús (Argentina).

Referencias

[1] García-Martínez, R., Britos, P., Pesado, P., Bertone, R., Pollo-Cattaneo, F., Rodríguez, D., Pytel, P., Vanrell. J. 2011. Towards an Information Mining Engineering. En Software Engineering, Methods, Modeling and Teaching. Sello Editorial Universidad de Medellín. ISBN 978-958-8692-32-6. Páginas 83-99.

[2] Yan, X., & Han, J. (2002). gspan: Graph-based substructure pattern mining. In Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on (pp. 721-724). IEEE.

[3] Huan, J., Wang, W., & Prins, J. (2003, November). Efficient mining of frequent subgraphs in the presence of isomorphism. In Data Mining, 2003. ICDM 2003. Third IEEE International Conference on (pp. 549-552). IEEE.

[4] Holder, L. B., Cook, D. J., & Djoko, S. (1994, July). Substucture Discovery in the SUBDUE System. In KDD workshop (pp. 169-180).

[5] Ullman, J. R. (1976). An algorithm for subgraph isomorphism. Journal of the ACM (JACM), 23(1), 31-42.

[6] Kuramochi, M., & Karypis, G. (2001). Frequent subgraph discovery. In Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on (pp. 313-320). IEEE.

[7] Nijssen, S., & Kok, J. N. (2004, August). A quickstart in frequent structure mining can make a difference. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 647-652). ACM.

[8] Zaiat, J., & Romero-Zaliz, R. Mineria de datos sobre grafos: un enfoque multiobjetivo aplicado a bioremediación.

[9] Lahiri, M., & Berger-Wolf, T. Y. (2007, March). Structure prediction in temporal networks using frequent subgraphs. In Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on (pp. 35-42). IEEE.

[10] Takigawa, I., & Mamitsuka, H. (2013). Graph mining: procedure, application to drug discovery and recent advances. Drug discovery today, 18(1), 50-57.

[11] Rehman, S. U., Khan, A. U., & Fong, S. (2012, August). Graph mining: A survey of graph mining techniques. In Digital Information Management (ICDIM), 2012 Seventh International Conference on (pp. 88-92). IEEE.

[12] Huan, J., Wang, W., Prins, J., & Yang, J. (2004, August). Spin: mining maximal frequent subgraphs from graph databases. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 581-586). ACM.