Búsqueda de comunidades en grafos grandes mediante configuraciones implícitas de vectores

Victor Muntés-Mulero, Arnau Padrol-Sureda, Guillem Perarnau-Llobet *
Julian Pfeifle **

Resumen

Presentamos el algoritmo OCA para buscar comunidades solapadas en grafos grandes, como por ejemplo la Wikipedia con $1{,}6 \times 10^7$ nodos y $1{,}8 \times 10^8$ aristas. OCA se basa en la búsqueda iterativa de subconjuntos localmente óptimos para una función objetivo, representando los subconjuntos como vectores suma de una configuración virtual de vectores. Analizamos el comportamiento de dos funciones objetivo, la Laplaciana asociada a la longitud del vector suma, y la conductividad.

1. Introducción

Hasta hace relativamente poco, buscar comunidades en grafos significaba *particionar* en familias disjuntas de nodos un grafo dado $G = (V, E)$, de manera que cada familia consiga una alta puntuación con respecto de alguna medida de calidad, por ejemplo la *conductividad* [3]. Por lo general, los algoritmos para hallar tal partición dividen el grafo recursivamente en dos partes, y por tanto la búsqueda de cada nueva familia está restringida por la existencia de todas las familias halladas hasta ahora. El presente trabajo adopta un enfoque distinto, en tanto que permitimos que las comunidades se puedan solapar entre si. Eso es consistente con la observación que las comunidades pueden presentar de manera natural estructuras tanto solapadas como jerárquicas. Un claro ejemplo son las redes sociales. Hacemos énfasis en que nuestra definición de comunidad es puramente operacional y local: una comunidad es un conjunto (generalmente conexo) de nodos de G que alcanza un óptimo local de cierta función objetivo φ, en el sentido de que añadir o quitar un nodo de ella empeora el valor de φ.

La idea subyacente de OCA es representar cada nodo de G mediante un vector en un espacio vectorial, de dimensión posiblemente bastante alta, de manera que los vectores que corresponden a nodos no adyacentes son ortogonales, y todos los ángulos entre vectores "adyacentes" son iguales y agudos. Para nosotros, esta representación será siempre *implícita*, en el sentido de que nunca se llega a construir ni un sólo vector explícitamente. Una búsqueda local, apodada LOCA, encuentra una comunidad a partir de cierto subgrafo inicial, añadiendo o quitando nodos para optimizar la función objetiva. Cada ejecución de LOCA necesita tiempo $O\big(s(\log s + \log n)\Delta\big)$, donde Δ es el grado máximo y n el número de nodos de G, y s el tamaño de la comunidad hallada. El algoritmo global OCA combina los resultados de las búsquedas locales empezadas en diferentes conjuntos iniciales repartidos por todo el grafo G. De esta manera, se desvela la estructura global del grafo compuesto por comunidades solapadas y localmente óptimas.

2. Configuraciones virtuales de vectores

La inspiración de nuestra propuesta son las *representaciones ortogonales* introducidas por Lovász [5] en 1979. Se trata de una colección de n vectores unitarios $v_1, \ldots, v_n$ de manera que $\langle v_i, v_j \rangle = 0$ para todo $\{i, j\} \notin E$. Nosotros generalizamos este concepto de la siguiente manera:

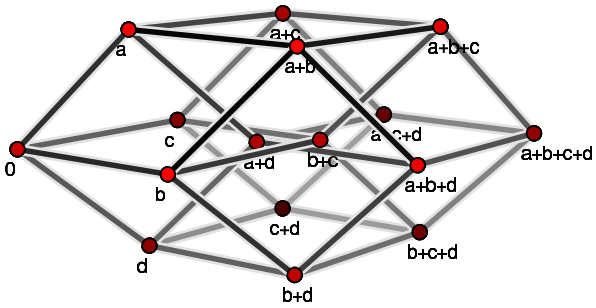
*DAMA, Universitat Politècnica de Catalunya {vmuntes, apadrol, perarnau}@ac.upc.edu
**DMA II, Universitat Politècnica de Catalunya, julian.pfeifle@upc.edu

Definición 2.1. Un conjunto $\mathcal{V} = \{v_1, \ldots, v_n\}$ de vectores unitarios en un espacio vectorial real es una *representación vectorial virtual* de G si existen $c, d \in (-1, 1)$ de manera que $\langle v_i, v_j \rangle = c$ para todo $\{i, j\} \in E$, y $\langle v_i, v_j \rangle = d$ si $\{i, j\} \notin E$.

El adjetivo *virtual* resalta el hecho que nunca tendremos que construir $\mathcal{V}$ de manera explícita. Ya que los v_i son unitarios, siempre $c, d \in [-1, 1]$; los casos extremos se corresponden con los casos degenerados en los que dos vectores se hacen paralelos (es decir, iguales) o anti-paralelos. El caso más importante para nosotros, que consideramos en exclusiva a partir de ahora, es el de $0 < c < 1$ y $d = 0$.

Definición 2.2. El *espacio de búsqueda* $\Gamma = \Gamma(G)$ asociado a $\mathcal{V}$ es el grafo con un nodo para cada subconjunto $S \subseteq V$. Una arista conecta S con S' en Γ si $S' = S \cup v$ para algún nodo $v \in V \setminus S$.

Dicho de otro modo, Γ es el grafo del cubo de dimensión $|\mathcal{V}|$. El papel de este grafo de 2^n nodos, que pensamos también como un *grafo virtual*, es formalizar el proceso de optimizar la función objetivo φ sobre todos los subconjuntos de nodos de G.



$$H_1(c) = \begin{pmatrix} 1 & c & c & 0 \\ c & 1 & c & c \\ c & c & 1 & c \\ 0 & c & c & 1 \end{pmatrix} \qquad H_2(c) = \begin{pmatrix} 1 & c & 0 & 0 \\ c & 1 & c & 0 \\ 0 & c & 1 & c \\ 0 & 0 & c & 1 \end{pmatrix}$$

El grafo $\Gamma(H_1)$

Figura 1: Dos grafos, las matrices que los representan, y el espacio de búsqueda de uno de ellos.

Ejemplo 2.3. La figura 1 muestra la representación vectorial virtual del grafo H_1 con $0 < c < 1$ y $d = 0$. Ya que b y c están conectados en H_1, pero a y d no, el ángulo $\angle(b, c)$ entre b y c es más pequeño que $\angle(a, d) = \frac{\pi}{2}$, y por tanto $b + c$ es más largo (más alejado de 0) que $a + d$.
Comparamos ahora H_1 con H_2. Puesto que $d = 0$ implica que dos vectores son ortogonales si sus nodos no están conectados, y H_1 está "más conectado" que H_2, los vectores que lo representan están más juntos que los de H_2. Por tanto, si H_1 y H_2 son subgrafos de un grafo más grande G, la suma de todos los vectores correspondientes a nodos en H_1 será más larga que la suma para H_2.

Por tanto, (el cuadrado de) la longitud Euclideana parece un buen candidato para la función objetivo para OCA: $\varphi(S) = \left\| \sum_{i \in S} v_i \right\|^2$. El ejemplo siguiente da más credibilidad a esta idea.

Ejemplo 2.4. Si $\mathcal{V}$ es una representación vectorial virtual de G y $S = \{1, 2, \ldots, k\}$ indexa un conjunto independiente de tamaño k en G, entonces $\varphi(S) = k$. Por otra parte, el vector suma de un k-clan K_k tiene longitud cuadrática $ck^2 + (1 - c)k = \Theta(k^2)$. El comportamiento diferente de subgrafos independientes y completos se vuelve más y más pronunciado según crece c.

Esperamos pues que comunidades bien conectadas entre sí tengan vectores suma más largos que comunidades más tenues. Sin embargo, y a pesar de todo, esta φ decididamente *no* es la elección adecuada, porque el vector suma más largo de todos es *siempre* $\sum_{i \in V} v_i$, que corresponde al grafo entero G (tal y como sugiere la figura 1). Con esta elección de φ, ¡OCA se comería el grafo G entero!

La lección crucial que podemos aprender del ejemplo 2.4 es el hecho de que valores más grandes de c distinguen las comunidades mejor que valores pequeños. Presentamos a continuación algunos resultados sobre valores extremos de c y cómo aproximarlos.

2.1. Cómo calcular el valor más grande posible de c

Para ver de qué manera la estructura de G restringe los posibles valores de c, codificamos los productos escalares de $\mathcal{V}$ en la matriz $G(c) = I_n + cA$, con I_n la matrix identidad $n \times n$, y A la matriz de adyacencia de G (véase la figura 1).

Teorema 2.5. *Para cualquier grafo G existe un único valor máximo para c de manera que $G(c)$ representa los productos escalares entre los miembros de una configuración vectorial virtual de G. Este valor más grande es $c = -1/\lambda_n$, con λ_n el autovalor más negativo de A.*

Demostración. Por definición, $G(c)$ representa los productos escalares entre los miembros de una configuración virtual de vectores si y sólo si la matriz se descompone como $G(c) = V^T V$, donde las columnas de la matriz V de tamaño $d \times n$ contienen las coordenadas de los vectores. Esto pasa si y sólo si $G(c)$ es *semidefinida positiva*, y por tanto la *matriz de Gram* de $\mathcal{V}$. Una caracterización de matrices semidefinidas positivas es que todos sus autovalores son no negativos. Puesto que 1 es el único autovalor de $G(0) = I_n$, existe por un lado una representación virtual de cualquier grafo G para $c = 0$, y por otro una representación virtual del grafo vacío. Suponemos pues en adelante que G no es vacío.

En este supuesto, afirmamos que la matriz de adyacencia A de G siempre tiene un autovalor negativo. Para verlo, recuérdese por una parte que los autovalores $\lambda_n \leq \cdots \leq \lambda_1$ de A son todos reales al ser A simétrica, y por otra que la traza de A es la suma de sus autovalores. Ahora, al tratarse de la matriz de adyacencia de un grafo no vacío sin bucles, la traza de A es cero pero no todos los autovalores lo son, y en consecuencia algún autovalor tiene que ser negativo; en particular, $\lambda_n < 0$.

Estamos en condiciones de demostrar la existencia de un valor máximo no nulo para c: Los autovalores de $G(c) = I_n + cA$ son $1 + c\lambda_i$, funciones lineales en c que toman valor positivo para $c = 0$. Hemos visto que al menos una de estas rectas tiene pendiente negativa, así que si c aumenta desde 0, llegará el momento en el que algún autovalor de $G(c)$ se anule por primera vez. Este valor es $c = -1/\lambda_n > 0$. □

Para aproximar el autovalor $\lambda_n < 0$, usamos el bien conocido *método de las potencias*, que nos proporciona el autovalor de mayor valor absoluto, con el siguiente resultado.

Teorema 2.6. *El autovalor más negativo λ_n de cualquier matriz simétrica A se puede aproximar a partir de cualquier cota superior $\lambda_1 \leq \kappa$ para el autovalor mayor.*

Demostración. Los autovalores de $B = A - \kappa I_n$ son $\mu_i = \lambda_i - \kappa$, con λ_i los autovalores de A. Puesto que $\mu_i \leq 0$ para todo i, el autovalor de B de máximo valor absoluto es $\mu_n < 0$, lo cual se puede aproximar con el método de la potencia. Para acabar, nótese que $|\lambda_n| = |\mu_n| - \kappa$. □

Una cota superior muy asequible para λ_1 es $\lambda_1 \leq \Delta$, con Δ el grado máximo de G [2].

3. Funciones objetivo eficientes

Para reparar el fracaso de la función objetivo "longitud de la suma al cuadrado", construimos dos nuevos grafos orientados a partir de Γ: En $\Gamma^{\uparrow}$, las aristas de Γ se orientan hacia el conjunto más grande, mientras que en $\vec{\Gamma}(\varphi)$ una arista se orienta $S \to S'$ si y sólo si $\varphi(S) < \varphi(S')$ (exceptuando casos degenerados). Por tanto, los máximos locales de φ corresponden a los sumideros de $\vec{\Gamma}(\varphi)$.

Proponemos dos funciones objetivo. Una es la *Laplaciana dirigida*, una adaptación de la noción de derivada a Γ. Intenta capturar la intuición que a pesar de que la longitud crece continuamente hasta alcanzar el grafo entero, no lo hace de manera uniforme, y las variaciones en la tasa de crecimiento contienen información sobre las comunidades. Comparamos esta función con la bien conocida *conductividad* [3].

3.1. La Laplaciana asociada al cuadrado de la longitud

Definición 3.1. El valor en v de la *Laplaciana dirigida* asociada a la función f en el grafo $\Gamma^{\uparrow}$ es

$$\mathcal{L}_{\Gamma^{\uparrow},f}(v) \;=\; f(v) - \frac{1}{\sqrt{\mathrm{indeg}(v)}} \sum_{u:u\to v} \frac{f(u)}{\sqrt{\mathrm{indeg}(u)}}.$$

Por la definición de la orientación en $\Gamma^{\uparrow}$, $\mathrm{indeg}(v) = |S|$ para cualquier nodo $v \in \Gamma^{\uparrow}$, es decir la cardinalidad del subconjunto S que corresponde a v en Γ. Ponemos $\mathcal{L}_{\Gamma^{\uparrow},f}(v) = 0$ para el nodo v que corresponde al conjunto $S = \emptyset$, para evitar la división por 0.

A partir de ahora, supondremos que $s := |S| > 1$. Entonces la Laplaciana $\mathcal{L}(S) := \mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}(S)$ es

$$\begin{aligned}\mathcal{L}(S) &= \|S\|^2 - \frac{1}{\sqrt{s(s-1)}} \sum_{v\in S} \|S \setminus v\|^2 \\ &= s - \sqrt{s(s-1)} + 2c\,\mathrm{E}_{\mathrm{in}}(S)\left(1 - \frac{s-2}{\sqrt{s(s-1)}}\right),\end{aligned}$$

donde $\mathrm{E}_{\mathrm{in}}(S)$ cuenta las aristas de G con ambos extremos en S.

Teorema 3.2. *Sea S una comunidad* grande, *$|S| \gg 0$. En este caso, un nodo $u \in V \setminus S$ mejora la Laplaciana $\mathcal{L}$ si y sólo si el número* $\mathrm{nb}_S(u)$ *de vecinos de u en S es mayor que el grado medio* $\mathrm{avdeg}(S)$ *de los nodos en S, es decir,* $\mathrm{nb}_S(u) > \mathrm{avdeg}(S)$. *En particular, nodos con un sólo vecino en S solamente son aceptados en S si S es un árbol.*

Esquema de demostración. Usando relaciones como $\langle S, u\rangle = c\,\mathrm{nb}_S(u)$, calculamos

$$\begin{aligned}\mathcal{L}(S\cup u) - \mathcal{L}(S) &= 2c\,\mathrm{nb}_S(u)\left(1 - \frac{s-1}{\sqrt{s(s+1)}}\right) \\ &\quad + 2c\,\frac{\mathrm{E}_{\mathrm{in}}(S)}{\sqrt{s}}\left(\frac{s-2}{\sqrt{s-1}} - \frac{s-1}{\sqrt{s+1}}\right) \\ &\quad + 1 - \sqrt{s}\big(\sqrt{s+1} - \sqrt{s-1}\big) \qquad (1) \\ &= c\,\mathrm{nb}_S(u)\left(\frac{3}{s} + O(s^{-2})\right) \\ &\quad + c\,\mathrm{E}_{\mathrm{in}}(S)\left(-\frac{3}{s^2} + O(s^{-3})\right) + O(s^{-2}),\end{aligned}$$

lo cual es no negativo para $s \gg 0$ si y sólo si $\mathrm{nb}_S(u) > \mathrm{E}_{\mathrm{in}}(S)/s$. □

3.2. La conductividad

Definición 3.3. La *conductividad* de un subconjunto $S \subseteq V$ en G es [3]

$$\phi(S) \;=\; \frac{\mathrm{E}_{\mathrm{out}}(S)}{\mathrm{E}_{\mathrm{out}}(S) + \min\big\{\,\mathrm{E}_{\mathrm{in}}(S),\, \mathrm{E}_{\mathrm{in}}(V\setminus S)\big\}},$$

donde $\mathrm{E}_{\mathrm{out}}$ cuenta las aristas en G con exactamente un extremo en S. Puesto que la conductividad mide la razón entre aristas salientes al total de aristas en el conjunto o su complemento (el más pequeño de los dos), intentaremos minimizarla.

Teorema 3.4. *Sea $R = \mathrm{E}_{\mathrm{out}}(S)/\,\mathrm{E}_{\mathrm{in}}(S)$, y supongamos que $\mathrm{E}_{\mathrm{in}}(S) \le \mathrm{E}_{\mathrm{in}}(V\setminus S) - \Delta$, con $\Delta = \mathrm{maxdeg}(G)$. Entonces un nodo $u \in V \setminus S$ mejora la conductividad ϕ si y sólo si $\mathrm{nb}_S(u) > \frac{\deg(u)}{R+2}$. En particular, nodos con un sólo vecino en S solamente se aceptarán en S si $\deg(u) < R + 2$.*

Demostración. Usando $\mathrm{E}_{\mathrm{out}}(S \cup u) = \mathrm{E}_{\mathrm{out}}(S) + \deg(u) - 2\,\mathrm{nb}_S(u)$, $\mathrm{E}_{\mathrm{in}}(S \cup u) = \mathrm{E}_{\mathrm{in}}(S) + \mathrm{nb}_S(u)$, y $\mathrm{E}_{\mathrm{in}}\big(V \setminus (S \cup u)\big) = \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u) + \mathrm{nb}_S(u)$, hallamos

$$\phi(S \cup u) \;=\; \frac{\mathrm{E}_{\mathrm{out}}(S) - 2\,\mathrm{nb}_S(u) + \deg(u)}{\mathrm{E}_{\mathrm{out}}(S) - \mathrm{nb}_S(u) + \deg(u) + m},$$

donde $m := \min\big\{\,\mathrm{E}_{\mathrm{in}}(S),\, \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u)\big\}$. Ya que $\mathrm{E}_{\mathrm{in}}(S) \le \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u)$, vemos que

$$\phi(S \cup u) - \phi(S) \;=\; \frac{\deg(u) - (R+2)\,\mathrm{nb}_S(u)}{(R+1)\big(T + \mathrm{nb}_{V \setminus S}(u)\big)} \tag{2}$$

con $T = \mathrm{E}_{\mathrm{in}}(S) + \mathrm{E}_{\mathrm{out}}(S)$. Eso termina la demostración. □

4. El algoritmo OCA

Presentamos dos variantes del *Overlapping Community Algorithm* OCA: la versión global OCA_g encuentra todas las comunidades en G, mientras que la versión local OCA_l sirve para explorar las comunidades de un nodo específico. En ambos casos, el motor de búsqueda local, llamado LOCA, es el mismo, y consiste en usar un método de pivot para encontrar un sumidero local en $\Gamma^{\uparrow}$ a partir de cierto conjunto inicial S_0. OCA_l selecciona conjuntos iniciales a partir del nodo a examinar y llama a LOCA. En cambio, OCA_g selecciona repetidamente conjuntos iniciales distribuidos por el grafo. Escribiremos $\mathrm{Nb}(S) = \big(\bigcup_{u \in S} \mathrm{Nb}_V(u)\big) \setminus S$ para el *conjunto* de vecinos de S, donde $\mathrm{Nb}_X(v)$ denota el conjunto de vértices en $X \subseteq V$ adyacentes a $v \in V$. Las estrategias de búsqueda empleadas son las siguientes:

Regla de pivot para LOCA: Hemos tenido las mejores experiencias con el método voraz, que en cada paso selecciona el nodo de $S \cup \mathrm{Nb}(S)$ que al suprimir o añadir a S dé el mayor incremento de φ. Para ello, LOCA mantiene una cola de prioridad $\mathcal{Q}$ con todas las *prioridades* $\Pi_+(u) = \varphi(S \cup u) - \varphi(S)$ para $u \in \mathrm{Nb}(S)$, respectivamente $\Pi_-(u) = \varphi(S \setminus u) - \varphi(S)$ para $u \in S$.

Selección de vecindario para LOCA: O bien tomamos todos los nodos a distancia $\le k$ de S_0, o bien un subconjunto aleatorio de ellos. Normalmente, usamos $k \le 2$. No hemos implementado criterios sofisticados basados en recorridos aleatorios como en [10].

Criterio de terminación para OCA: El número de nodos del grafo ya explorados aumenta mucho al principio, y después el crecimiento se ralentiza. Paramos el algoritmo cuando la tasa de crecimiento del número de nuevos nodos haya decaído debajo de cierto umbral.

Teorema 4.1. *Si G tiene n nodos y grado máximo Δ, la búsqueda local LOCA con la Laplaciana $\mathcal{L}$ se puede ejecutar en tiempo $O\big(s(\log s + \log n)\Delta\big)$, donde s es el tamaño de la comunidad encontrada. Para la conductividad ϕ, este tiempo aumenta a $O\big(s(s + \log n)\Delta\big)$.*

Demostración. Solamente consideraremos los pasos necesarios para añadir un nodo u a S; quitar nodos es muy similar. Las fórmulas (1) y (2) dicen que para ambas funciones objetivo, LOCA tiene que actualizar los valores de *todos* los $u' \in \Sigma := S \cup \mathrm{Nb}(S)$, porque todas las prioridades dependen de cantidades como E_{in} y s que cambian con las propiedades y el tamaño de S. Eso lleva a un algoritmo cuadrático en el tamaño del conjunto de la salida. Por (2), eso es lo mejor posible para la conductividad ϕ. Para la Laplaciana, podemos aumentar la eficiencia de LOCA como sigue. Sea $f_+\big(\mathrm{nb}_S(u), s, \mathrm{E}_{\mathrm{in}}(S)\big)$ el resultado de $\Pi_+(u) = \varphi(S \cup u) - \varphi(S)$ en la fórmula (1). Distinguimos los posibles casos para $u' \in \Sigma \cup \mathrm{Nb}_V(u)$:

1. Si $u' \in \mathrm{Nb}_\Sigma(u)$, reemplazamos $\Pi(u')$ por $f_+\big(\mathrm{nb}_S(u') + 1, s + 1, \mathrm{E}_{\mathrm{in}}(S) + \mathrm{nb}_S(u)\big)$. Eso pasa como mucho $\Delta = \mathrm{maxdeg}(G)$ veces.
2. Si $u' \in \Sigma \setminus \mathrm{Nb}_\Sigma(u)$, reemplazamos $\Pi(u')$ por $f_+\big(\mathrm{nb}_S(u'), s + 1, \mathrm{E}_{\mathrm{in}}(S) + \mathrm{nb}_S(u)\big)$. No sabemos cuántas veces pasa eso, porque no podemos controlar el tamaño de Σ.
3. Si $u' \in \mathrm{Nb}_V(u) \setminus \mathrm{Nb}_\Sigma(u)$, LOCA ha de calcular $\mathrm{Nb}_V(u')$.

Al desarrollar en potencias de $\frac{1}{s}$, la diferencia δ_2 entre el valor $\Pi_+(u') = f_+\big(\mathrm{nb}_S(u'), s, \mathrm{E}_{\mathrm{in}}(S)\big)$ guardado en $\mathcal{Q}$ y el valor nuevo en el caso 2 resulta ser un orden de magnitud (en potencias de $\frac{1}{s}$) más pequeño que la diferencia análogo δ_1 con el valor nuevo en el caso 1. En particular, $\frac{\delta_2}{\delta_1} < \frac{1}{100}$ para $|S| > 150$. Por tanto, después de un período inicial de duración constante C, podemos ignorar el caso 2 por completo. (Omitimos los detalles de este cálculo, y del cálculo análogo para $\varphi(S \setminus u) - \varphi(S)$. Ademas, suponemos que el número total de veces que borramos un nodo es pequeño en comparación con el número de veces que añadimos uno.) Después de este período inicial, el caso 1 efectua a lo sumo Δ actualizaciones de $\mathcal{Q}$, con un coste de $O(\log j)$ cada uno para una comunidad de tamaño j. El caso 3, finalmente, es ejecutado un máximo de Δs veces durante el algoritmo. Con las implementaciones que usamos, podemos ejecutar el cálculo de $\mathrm{Nb}_V(u')$ en tiempo $O(\log n)$. El coste total del algoritmo es, por tanto, $C' + s\,O(\log n)\Delta + \sum_{j=C+1}^{s} \Delta O(\log j) = O\big(s(\log n + \log s)\Delta\big)$. □

5. Resultados

Hemos ejecutado OCA en tres grafos artificiales y dos grafos reales, descritos a continuación.

Nombre	tipo	# nodos	# aristas
Erdös-Rényi aleatorio	generado	10^5	750353
"Word association"	real	7 211	31 798
LFR	generado	10^4–10^6	$\sim 10^5$–10^7
Árbol de margaritas	generado	10^5	$\sim 4 \cdot 10^5$
Wikipedia	real	16 986 429	176 454 501

Grafos aleatorios Erdös-Rényi. Se trata de una familia de grafos en la cual con alta probabilidad *no* existen comunidades. Generamos un grafo $G(n, p)$ según el modelo de Erdős-Rényi [1], con n nodos y una arista entre dos de ellos con probabilidad p.

"Word Association Network". Este conjunto de datos, basado en [7], se ha usado con anterioridad para la detección de comunidades en [8, 9]. Se trata de un grafo dirigido que enlaza palabras relacionadas entre sí, con pesos que reflejan el grado de relación que guardan. Lo hemos usado para evaluar la calidad de OCA_l, porque su propia estructura nos permite evaluar de manera intuitiva la calidad de las comunidades encontradas por OCA. Para nuestros experimentos, lo hemos convertido en un grafo no dirigido, conectando dos vértices si la suma de los pesos de las dos aristas que los unen excede un cierto umbral ω^*. Usando el mismo valor que [8] e ignorando nodos aislados da un grafo con 7 211 nodos y 31 798 aristas. Después de quitar todos los nodos de grado 1, queda un grafo con 5 353 nodos y 29 940 aristas.

LFR ("Communitized benchmark graphs"). Lacichinetti, Fortunato and Radicchi (en adelante, LFR) proponen en [4] un juego de pruebas para la detección de comunidades. Puesto que se trata de grafos generados por ordenador, podemos determinar de manera cuantitativa tanto el comportamiento de OCA según aumenta el tamaño del grafo, como la calidad de las comunidades halladas. Unos valores típicos que hemos considerado son: $n = 10^5$, grado medio 20, grado máximo 90, tamaño de comunidades en $(25, 150)$, exponente de la ley de potencias para la distribución de grados $\gamma = 2{,}5$, exponente para la distribución de los tamaños de las comunidades $\beta = 1{,}5$, y *mixing parameter* $\mu = 0{,}2$. Este último parametro determina la proporción de aristas que un nodo tiene en común con nodos fuera de su comunidad.

Árboles de margaritas (comunidades solapadas). No nos consta publicada ninguna familia de grafos que sirva como estándar para evaluar algoritmos de búsqueda de comunidades solapadas. Por este motivo, aquí proponemos una familia bastante sencilla con este fin. Los nodos vienen numerados por enteros consecutivos. Creamos una arista entre dos nodos u y v, con probabilidad p_1, si se cumplen las condiciones $f(u)$ y $f(v)$, donde $f(x) = (x = 0 \bmod p) \vee (x = 0 \bmod q)$, para p, q primos entre si. Además, creamos una arista entre cualquier par de nodos u y v, con probabilidad p_2, si $u = v \bmod p$ y $u, v \neq 0 \bmod q$. El grafo *margarita* resultante tiene p comunidades, una comunidad central que consiste de múltiplos de p y q, y $p-1$ comunidades solapadas con aquélla. El parámetro q determina el grado de solapamiento. Nuestro grafo de pruebas será un *árbol de margaritas*, en el que unimos varias margaritas a través de sus hojas.

Wikipedia.org. Los artículos en esta conocida enciclopedia tienen enlaces a otras páginas o sus traducciones a otros idiomas. Hemos extraído casi 17 millones de artículos y más de 180 millones de enlaces en 253 idiomas diferentes de los ficheros XML originales.

Hemos llevado a cabo los experimentos en un ordenador con un procesador con 2.33 GHz y 9 GB de RAM, corriendo bajo Linux (versión 2.6.18 del kernel). Los grafos se manejan a través de DEX, un gestor de bases de datos [6]. La implementación actual de DEX tiene dos librerías, una en C++ para manejar mapas de bits y maps, y otra en Java para tareas de alto nivel. La librería C++ contiene un buffer pool para permitir el procesamiento out-of-core de los grafos DEX.

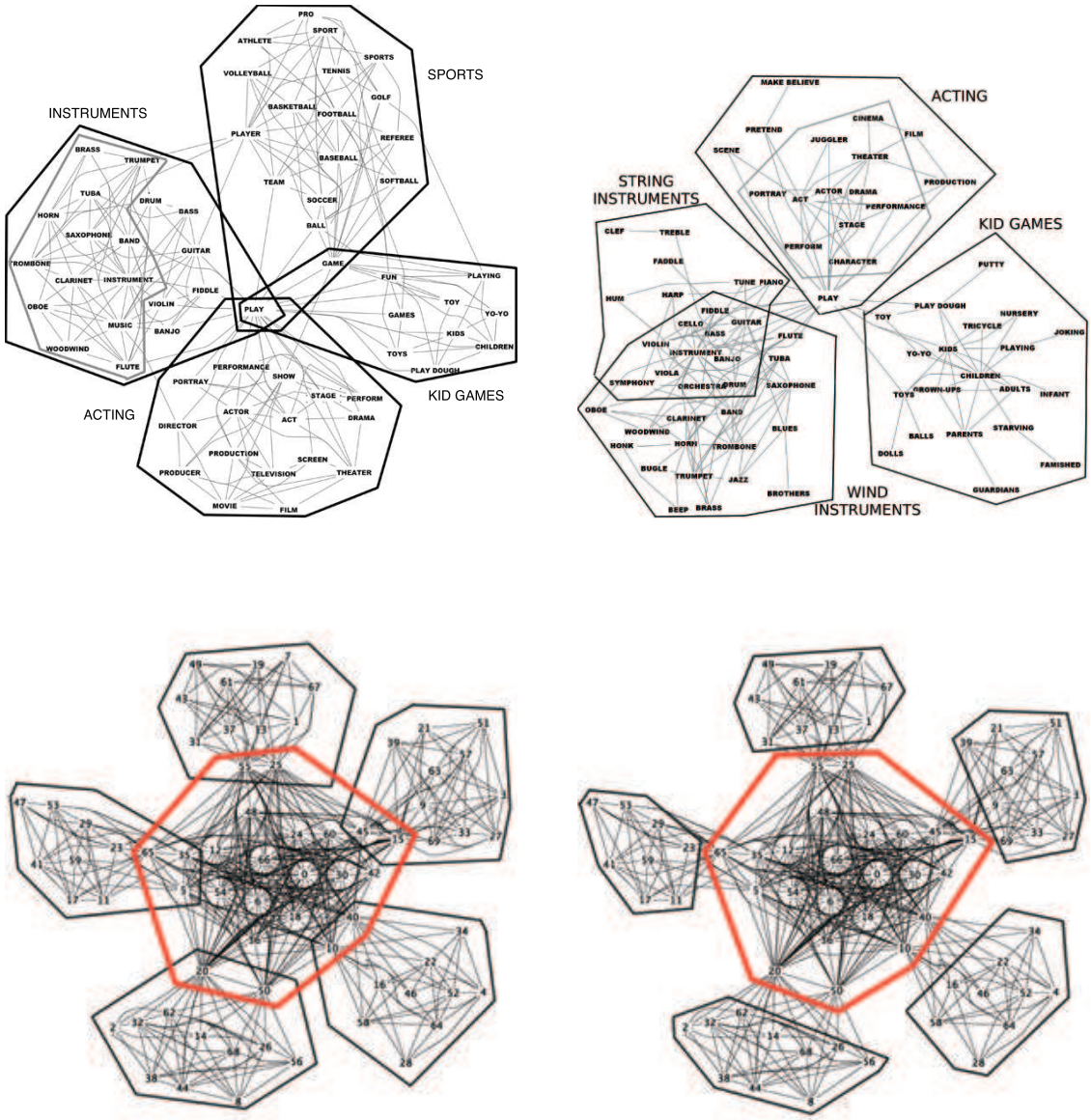


Figura 2: *Arriba:* Comunidades encontradas por LOCA en el grafo de asociaciones de palabras a partir de la palabra PLAY, por la Laplaciana (izquierda) y la conductividad (derecha). *Abajo:* Las comunidades encontradas en un grafo margarita por la Laplaciana (izquierda) y la conductividad (derecha).

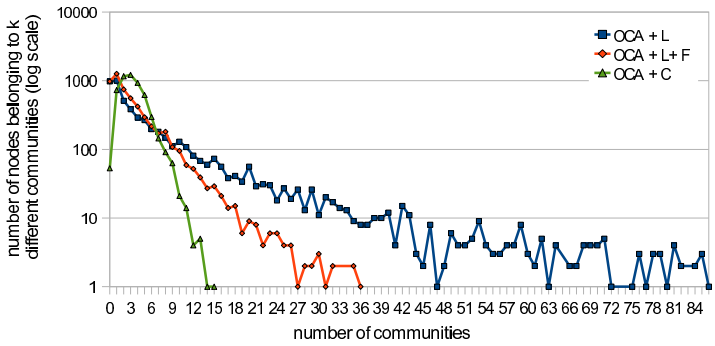


Figura 3: Número de nodos en grafos estándar LFR en función del número de comunidades a las que pertenecen. L y C se refieren a la Laplaciana y la conductividad, mientras que F es un postprocesamiento que consiste en fusionar comunidades muy similares.

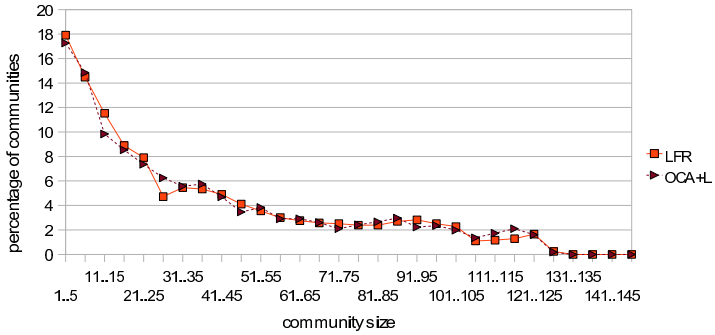


Figura 4: Validación cuántitativa de OCA con la conductividad en los grafos LFR. Mostramos hasta qué punto OCA es capaz de reconstruir la estructura de comunidades en los juegos de prueba.

Referencias

[1] P. Erdös and A. Rényi, *On random graphs, I*, Publ. Math. (Debrecen), 6 (1959), pp. 290–297.

[2] C. Godsil and G. Royle, *Algebraic graph theory.*, Graduate Texts in Mathematics. 207. New York, NY: Springer. xix, 439 p., 2001.

[3] R. Kannan, S. Vempala, and A. Vetta, *On clusterings: Good, bad and spectral*, Journal of the ACM, 51 (2004), pp. 497–515.

[4] A. Lancichinetti, S. Fortunato, and F. Radicchi, *Benchmark graphs for testing community detection algorithms*, Phys. Rev. E (Statistical, Nonlinear, and Soft Matter Physics), 78 (2008).

[5] L. Lóvasz, *On the Shannon capacity of a graph*, IEEE Trans. Inform. Theory, 25 (1979), pp. 1–7.

[6] N. Martinez-Bazán, J. Nin, S. Gómez-Villamayor, M. A. Sánchez-Martínez, V. Muntés-Mulero, and J. L. Larriba-Pey, *DEX: High-performance exploration on large graphs for information retrieval*, in CIKM '07: Proceedings of the sixteenth ACM conference on information and knowledge management, New York, NY, USA, 2007, ACM, pp. 573–582.

[7] D. L. Nelson, C. L. McEvoy, and T. A. Schreiber, *The University of South Florida word association, rhyme, and word fragment norms*, 1998.

[8] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, *Uncovering the overlapping community structure of complex networks in nature and society*, Nature, 435 (2005), p. 814.

[9] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, *Detect overlapping and hierarchical community structure in networks*, Physica A: Stat. Mech. Appl., 388 (2009), pp. 1706 – 1712.

[10] D. A. Spielman and S.-H. Teng, *A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning.* Preprint http://arxiv.org/abs/0809.3232, 2008.