PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

COMUNIDADES EN REDES SOCIALES

JOSÉ IGNACIO GARRIDO MUÑOZ

Profesor Guía: **Wenceslao Palma Muñoz**
Profesor Co-referente: **Ignacio Araya Zamorano**

Carrera: **Ingeniería Civil Informática**

DICIEMBRE, 2014

Esta obra va dedicada primeramente a Dios por darme la vida y sabiduría necesaria en este largo proceso de formación, a mis padres Héctor y Edy por entregarme ambos su inmenso amor y apoyo, y las herramientas esenciales para construir mi camino del éxito, a mi compañera de vida Elizabeth por estar conmigo durante toda esta etapa, haciéndome ver lo simple que puede ser la vida y a mi querida casa de estudios por dotarme de conocimientos indispensables.

Resumen

El uso de los grafos para modelar sistemas complejos es creciente en multitud de ámbitos. Son extremadamente útiles para representar interacciones, relaciones sociales e intercambio de información en Internet. Analizando la estructura de estas redes, comprendiendo como interaccionan sus distintos elementos, se puede entender mejor el comportamiento del sistema en su conjunto. A menudo, los nodos que conforman estos grafos tienden a formar grupos altamente conectados entre sí y con otros grupos. Esta propiedad es conocida como estructura de comunidades solapadas y este trabajo se ha centrado en el problema de la detección de comunidades solapadas en redes dinámicas y su caracterización.

Para entender la problemática desde su base, en este trabajo se estudia la teoría de grafos desde sus definiciones básicas hasta las características generales, luego se hace un breve análisis de las redes sociales desde tres perspectivas: a nivel de elementos, de grupos y de redes. A continuación, se detallan los tipos de problemas en la búsqueda de comunidades, tales como comunidades con información global, local, solapadas y dinámicas, se mencionan y describen algunos de los algoritmos más relevantes para la detección de comunidades disjuntas y solapadas. Posteriormente, se trata el tema central del presente trabajo, las redes dinámicas, enfocándose en el análisis e implementación del algoritmo dinámico AFOCS (*Adaptive Finding Overlapping Community Structure*) [34], un algoritmo adaptativo para la detección, actualización y rastreo de la evolución de comunidades solapadas en redes dinámicas no dirigidas y sin pesos, apuntando a la actualización de forma rápida y eficiente como consecuencia de la inserción y/o eliminación de nodos y/o aristas dentro de una red compleja. De forma complementaria, se formula una extensión para analizar grafos ponderados no dirigidos, con el propósito de tomar en cuenta la importancia del peso en las relaciones entre nodos, y poder establecer un criterio adicional para formar una comunidad. Luego para comprobar el impacto de la incorporación de pesos en AFOCS, se compara el rendimiento del método frente a otros algoritmos dinámicos.

Palabras Clave: comunidad, algoritmo, grafo, solapamiento, dinámica, redes.

Abstract

The use of graphs to model complex systems belonging to different domains such as social networks and biology is increasing. They are extremely useful to represent interactions, social relations and information exchange between nodes. Analizing the structure of these complex systems as graphs helps to understand the behaviour of the systems as a whole. This works focuses on the problem of detecting overlapping communities in dynamics graphs.

An extension to the dynamic algorithm AFOCS is proposed to analize dynamics weighted and undirected graphs. A detailed experimental evaluation shows that the proposed approach achieves competitive results w.r.t state of the art approaches.

Keywords: community, algorithm, graph, overlapping, dynamics, networks.

Índice

Lista de Figuras

Lista de Tablas

1. Introducción

Las personas y organizaciones forman comunidades en redes o grupos sociales. Este tipo de agrupamiento llamado comunidad o cluster se basa en relaciones de alta afinidad entre un grupo específico de elementos o nodos, los cuales están densamente conectados entre sí y muy poco conectados con nodos de otras comunidades [19]. La condicionante anterior, referente a la poca conectividad entre nodos de diferentes comunidades es el motivo de estudio del presente trabajo, enfocándose en el análisis de comunidades solapadas o superpuestas en redes dinámicas [34], donde uno o más nodos pueden pertenecer a dos o más comunidades, estableciéndose relaciones entre elementos de diferentes grupos. Además, la mayoría de las redes sociales evolucionan con frecuencia en el tiempo debido a la alta dinámica de los miembros participantes de la red. En términos prácticos, si bien cualquier cambio pequeño no parece ser tan significativo en la estructura de la red, los cambios que presentan mayor frecuencia y duración conllevan a una transformación impredecible de sus comunidades, en particular cuando dichas comunidades se solapan. Dicho escenario conduce a la necesidad de reidentificar las nuevas comunidades, lo cual es una tarea difícil especialmente en redes dinámicas debido a que la topología de estas redes cambia muy rápidamente. Una solución trivial al problema anterior, sería ejecutar reiteradamente métodos estáticos [36, 28, 22] para la detección de comunidades solapadas para encontrar nuevas comunidades ante cualquier cambio en la red. Al efectuar este procedimiento, se generan ciertos inconvenientes como altos tiempos de respuesta y recursos computacionales al analizar redes muy grandes, y la ejecución innecesaria del algoritmo en partes locales de la red que no son afectadas con los cambios de topología. Entonces, una manera de solucionar el problema sería actualizar de forma adaptativa la estructura de comunidad actual, basándose únicamente en su historia evolutiva y en los cambios únicos de la red, para no caer en una redetección innecesaria.

En virtud de lo anterior, se estudia el algoritmo dinámico AFOCS (*Adaptive Finding Overlapping Community Structure*) [34], método adaptativo para la detección, actualización y rastreo de la evolución de comunidades solapadas en grafos dinámicos no dirigidos y sin pesos. El algoritmo consta de dos etapas, primero mediante FOCS (*Finding Overlapping Community Structure*) se detectan todas las posibles comunidades solapadas en la red, y luego se aplica AFOCS para actualizar estas comunidades como consecuencia de la inserción y/o eliminación de nodos y/o aristas. Con el fin de manejar eficazmente los cambios en el grafo, AFOCS se descompone en eventos más simples, de forma que cada evento se maneja de forma rápida y con esto se obvia la necesidad de reidentificar en todo momento la estructura de comunidad en toda la red. También se propone una extensión para la detección de comunidades solapadas en AFOCS, esto es, agregando peso a las aristas pero conservando su simetría, de esta forma, se plantea una abstracción de la red a un grafo ponderado no dirigido. Los resultados de esta extensión se contrastan en pruebas experimentales, donde los resultados se comparan con otros algoritmos dinámicos.

2. Objetivos

A continuación se presentan los objetivos generales y específicos del presente trabajo.

2.1. Objetivos Generales

Estudiar e implementar AFOCS (*Adaptive Finding Overlapping Community Structure*), algoritmo de detección de comunidades solapadas en redes dinámicas [34].

2.2. Objetivos Específicos

- Analizar documentos de investigaciones previas para generar conocimiento base en el desarrollo de la problemática.

- Formular una extensión en AFOCS para la detección de comunidades solapadas en redes ponderadas no dirigidas.

- Implementar y realizar pruebas con la nueva extensión de AFOCS, involucrando detección de solapamiento y operaciones adaptativas en redes sociales dinámicas, con el fin de estudiar su conformación y evolución de comunidades, contrastando con el rendimiento de otros algoritmos dinámicos.

3. Elementos de la Teoría de Grafos

Para comprender que los grafos son realmente una red, es necesario estudiar algunos conceptos básicos y formales y notaciones de teoría de grafos, junto con algunas propiedades fundamentales que caracterizan una red. A continuación se detallan en forma matemática y textual definiciones básicas que permiten comprender de mejor manera el tema.

3.1. Definiciones básicas

La teoría de grafos según el contexto de estudio, forma parte de las ciencias de la computación, que estudia las propiedades de los grafos, estructuras que se componen de tres partes, el conjunto de vértices, nodos o puntos, el conjunto de aristas, enlaces o líneas que pueden ser dirigidas o no dirigidas y los pesos que pueden contener o no las aristas.

3.1.1. Grafos y grados de vértices

Un grafo es un conjunto de vértices que se pueden conectar entre sí por medio de aristas, en esencia, cada arista del grafo une exactamente dos vértices. Un grafo es definido de la siguiente forma.

Definición 3.1. *: Un grafo G consiste de una colección de V **vértices** y de una colección de E **aristas**, denotándose como $G = (V, E)$. Cada arista $e \in E$, se dice que une dos vértices, que son denominados como **puntos extremos**. Si e es el enlace $u, v \in V$, entonces se escribe como $e = \langle u, v \rangle$. Los vértices u y v en este caso se dice que son adyacentes. La arista e se dice que es incidente con los vértices u y v respectivamente.*

A medida que se avanza en la lectura se escribirá $V(G)$ y $E(G)$ para denotar el conjunto de vértices y aristas asociados al grafo G, respectivamente. Es importante tener en cuenta que una arista se puede representar como una tupla no ordenada de dos vértices, es por dicha razón, que (u, v) representa que los vértices u y v son adyacentes.

Un grafo que no tiene enlaces o varias aristas es llamado **grafo simple**. Del mismo modo, no hay nada que prohíba que un grafo no tenga vértices en lo absoluto, lo que implica en este caso que no tendrá aristas, lo cual se denomina como **grafo vacío**. Caso contrario, sucede cuando un grafo sencillo tiene n vértices, con cada vértice adyacente a cada otro vértice, es lo que se conoce como **grafo completo**. Un grafo completo con n vértices es comúnmente denotado como k_n.

Un grafo puede ser dirigido o no dirigido, bipartito o ponderado, en donde las aristas pueden contener pesos o no, la Figura 1 representa algunos ejemplos de estos diferentes tipos de grafos.

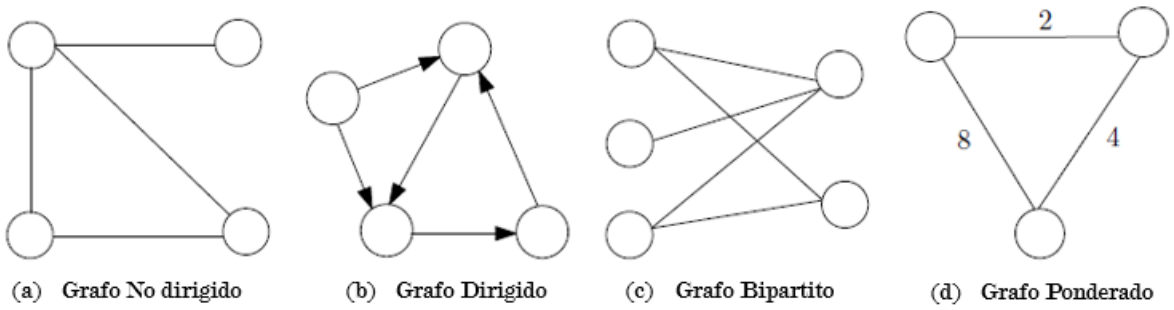


Figura 1: Ejemplo de diferentes tipos de grafos. En el caso del *grafo dirigido* (b), las flechas indican la direccionalidad de cada arista. En el *grafo ponderado* (d) los valores asociados con cada arista representan pesos (un grafo ponderado puede ser dirigido o no dirigido).

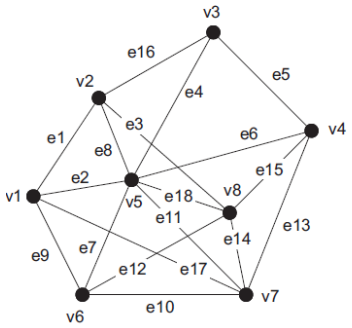
Definición 3.2. *: Un grafo $G_B = (V_h, V_a, E_b)$ es llamado **bipartito** si los nodos del conjunto V pueden ser particionados en dos conjuntos disjuntos V_h y V_a, donde $V = V_h \cup V_a$, de tal manera que cada arista $e \in E_b$ conecta a un nodo del conjunto V_h con un nodo del conjunto V_a. Es decir, $e = (i, j) \in E \Rightarrow i \in V_h$ y $j \in V_a$. En otras palabras, no existen aristas entre nodos de la misma partición.*

En muchas situaciones prácticas, es conveniente referirse a los vecinos de un vértice, en términos grafo-teóricos, los vecinos de un vértice u son formados por los vértices que son adyacentes a v, en otras palabras, aquellos vértices a los que v se ha unido mediante una arista. La denotación matemática de los vecinos de un vértice es la que sigue.

Definición 3.3. *: Para cualquier grafo G y vértice $v \in V(G)$, el conjunto de vecinos $N(v)$ de v es el conjunto de vértices (que no sea v) adyacentes a v, es decir:*

$$N(v) \stackrel{def}{=} \{w \in V(G) | v \neq w, \exists e \in E(G) : e = \langle u, v \rangle\} \tag{1}$$

Una propiedad importante de un vértice es el número de aristas incidentes en él, este número de enlaces se llama **grado de un vértice**.



$V(G) = \{v_1, \ldots, v_8\}$
$E(G) = \{e_1, \ldots, e_{18}\}$

$e_1 = \langle v_1, v_2 \rangle$	$e_{10} = \langle v_6, v_7 \rangle$
$e_2 = \langle v_1, v_5 \rangle$	$e_{11} = \langle v_5, v_7 \rangle$
$e_3 = \langle v_2, v_8 \rangle$	$e_{12} = \langle v_6, v_8 \rangle$
$e_4 = \langle v_3, v_5 \rangle$	$e_{13} = \langle v_4, v_7 \rangle$
$e_5 = \langle v_3, v_4 \rangle$	$e_{14} = \langle v_7, v_8 \rangle$
$e_6 = \langle v_4, v_5 \rangle$	$e_{15} = \langle v_4, v_8 \rangle$
$e_7 = \langle v_5, v_6 \rangle$	$e_{16} = \langle v_2, v_3 \rangle$
$e_8 = \langle v_2, v_5 \rangle$	$e_{17} = \langle v_1, v_7 \rangle$
$e_9 = \langle v_1, v_6 \rangle$	$e_{18} = \langle v_5, v_8 \rangle$

Figura 2: Un ejemplo de un grafo con ocho vértices y 18 aristas.

Definición 3.4. *: El número de aristas incidentes con un vértice v se denomina grado de v, denotado como $\delta(v)$. Si existen bucles éstos se cuentan dos veces.*

Un ejemplo claro es el de la Figura 2, en el caso del vértice v_1 existen 4 aristas incidentes, tenemos que $\delta(v_1) = 4$. La Tabla 1 muestra el resumen considerando todos los nodos del grafo.

Vértices	Grado	Aristas incidentes	Vecinos
v_1	4	$\langle v_1, v_2\rangle, \langle v_1, v_5\rangle, \langle v_1, v_6\rangle, \langle v_1, v_7\rangle$	v_2,v_5,v_6,v_7
v_2	4	$\langle v_1, v_2\rangle, \langle v_2, v_3\rangle, \langle v_2, v_5\rangle, \langle v_2, v_8\rangle$	v_1,v_3,v_5,v_8
v_3	3	$\langle v_2, v_3\rangle, \langle v_3, v_4\rangle, \langle v_3, v_5\rangle$	v_2,v_4,v_5
v_4	4	$\langle v_3, v_4\rangle, \langle v_4, v_5\rangle, \langle v_4, v_7\rangle, \langle v_4, v_8\rangle$	v_3,v_5,v_7,v_8
v_5	7	$\langle v_1, v_5\rangle, \langle v_2, v_5\rangle, \langle v_3, v_5\rangle, \langle v_4, v_5\rangle, \langle v_5, v_6\rangle, \langle v_5, v_7\rangle, \langle v_5, v_8\rangle$	v_1,v_2,v_3,v_4,v_6,v_7,v_8
v_6	4	$\langle v_1, v_6\rangle, \langle v_5, v_6\rangle, \langle v_6, v_7\rangle, \langle v_6, v_8\rangle$	v_1,v_5,v_7,v_8
v_7	5	$\langle v_1, v_7\rangle, \langle v_4, v_7\rangle, \langle v_5, v_7\rangle, \langle v_6, v_7\rangle, \langle v_7, v_8\rangle$	v_1,v_4,v_5,v_6,v_8
v_8	5	$\langle v_2, v_8\rangle, \langle v_4, v_8\rangle, \langle v_5, v_8\rangle, \langle v_6, v_8\rangle, \langle v_7, v_8\rangle$	v_2,v_4,v_5,v_6,v_7

Tabla 1: Grados, aristas incidentes y vecinos de los vértices de la Figura 2.

Teorema 3.1. *: Para todos los grafos G, la suma de los grados de los vértices es dos veces el número de aristas, es decir,*

$$\sum_{v \in V(G)} \delta(v) = 2 \cdot |E(G)| \tag{2}$$

En el contexto de las redes sociales, el grado de un vértice es un concepto potente, ya que se pueden utilizar los grados de un vértice para expresar la importancia de una persona dentro de un grupo social. Es más, el ordenamiento de los vértices de un grafo por el grado de vértices, permite obtener una visión de cómo una red de este tipo está organizada.

3.1.2. Secuencia de grado

Al listar los grados de los vértices de un grafo se obtiene la **secuencia de grado**. Por lo general los grados de los vértices se muestran en orden descendente, en cuyo caso se hace referencia a una **secuencia ordenada de grado**. Un ejemplo claro, es considerar los ocho vértices del grafo G de la Figura 2, en donde se tienen los siguientes grados de vértices:

Vértices	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
Grado	4	4	3	4	7	4	5	5

Tabla 2: Grados de los vértices de la Figura 2.

Al ordenar estos grados en orden descendente, se tiene la siguiente secuencia de grados:

$$[7, 5, 5, 4, 4, 4, 4, 3]$$

Si cada vértice tiene el mismo grado, entonces se tiene un **grafo regular**. En el grafo k-regular cada vértice tiene grado k, en un caso especial, grafos 3-regular se denominan **grafos cúbicos**.

Para determinar si una secuencia ordenada de grados corresponde a la representación gráfica de un grafo, Havel [24] y Hakimi [23] postulan el siguiente teorema de construcción que justifica el *Algoritmo de Havel-Hakimi* para construir un grafo a partir de una secuencia de grado.

Teorema 3.2. *(Havel-Hakimi): Considerar una lista $s = [d_1, d_2, ..., d_n]$ de n números en orden descendente. La lista es gráfica si y sólo si $s^* = [d_1^*, d_2^*, ..., d_{(n-1)}^*]$ de $n-1$ números*

también es gráfica, donde:

$$d_i^* = \begin{cases} d_{i+1} - 1 & ;\ desde\ i{=}1,2,\ldots,d_1 \\ d_{i+1} & ;\ en\ otro\ caso \end{cases} \tag{3}$$

En simples palabras, las condiciones para que se cumpla el Teorema 3.2 son las siguientes:

- La suma de los grados de la secuencia debe ser par.
- $max(d_i) < n$
- $d_1, d_2, ..., d_n \geq 0$.
- Eliminar el primer elemento de la secuencia de grados.
- Restar una unidad a los elementos restantes de la secuencia de grados.
- Si se alcanza una sucesión de ceros, entonces la secuencia de grados inicial es un grafo.

3.1.3. Subgrafos y grafos lineales

Otro concepto importante de grafos es el de **subgrafos**. Se dice que un grafo H es un subgrafo de G si H consiste en un subconjunto de aristas y vértices de G. Formalmente se tiene lo siguiente:

Definición 3.5. *: Un grafo H es un subgrafo de G si $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$ tal que para todo $e \in E(H)$ con $e = \langle u, v \rangle$, tenemos que $u, v \in V(H)$. Cuando H es un subgrafo de G, se denotará $H \subseteq G$.*

Por ejemplo, la Figura 3 muestra el llamado *grafo cúbico*, con 8 vértices y tres de sus subgrafos.

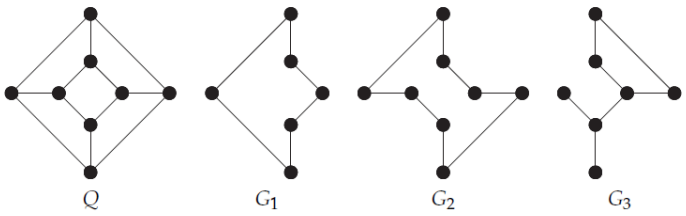


Figura 3: Grafo Cúbico. El grafo cúbico Q con 8 vértices y tres subgrafos G_1, G_2 y G_3.

Es importante considerar subgrafos formados por un subconjunto específico de vértices, éstos son llamados **grafos inducidos**, ya que son construidos mediante la adopción de un subconjunto V^* de vértices y agregando cada arista del grafo original que conecta dos vértices de V^*. Formalmente, se tiene:

Definición 3.6. *: Considerar un grafo G y un subconjunto $V^* \subseteq V(G)$. El subgrafo inducido por V^* tiene conjunto de vértices V^* y conjunto de aristas E^* definido por:*

$$E^* \stackrel{def}{=} \{e \in E(G) | e = \langle u, v \rangle\ ; u, v \in V^*\} \tag{4}$$

Del mismo modo, si $E^* \subseteq E(G)$, el subgrafo inducido por E^* tiene conjunto de aristas E^* y un conjunto de vértices V^* definido por:

$$V^* \stackrel{def}{=} \{u, v \in V(G) | \exists e \in E^* : e = \langle u, v \rangle\} \tag{5}$$

El subgrafo inducido por V^* o E^* es escrito como $G[V^*]$ o $G[E^*]$, respectivamente.

Algo relacionado con la noción de subgrafo es el **grafo lineal**.

Definición 3.7. *: Considere un grafo simple $G = (V, E)$. El grafo lineal de G, denotado como $L(G)$ es construido desde G mediante la representación de cada arista $e = \langle u, v \rangle$ desde E mediante un vértice v_e en $L(G)$, y que unen dos vértices v_e y v_{e^*} si y solo si las aristas e y e^* son incidentes con el mismo vértice en G.*

A modo de ejemplo, el grafo representado en la Figura 4 (a) contiene cuatro vértices y seis aristas. Su correspondiente grafo lineal, mostrado en la Figura 4 (b) consiste en seis vértices.

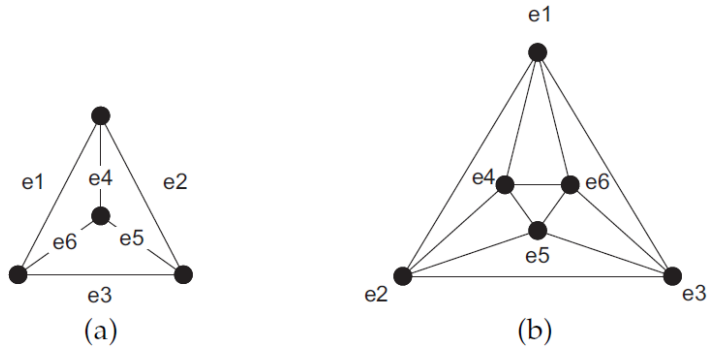


Figura 4: Grafo Lineal.(a) Un grafo G y (b) su grafo lineal $L(G)$.

3.2. Representación gráfica de grafos

Hasta el momento los grafos se han descrito en términos de vértices y aristas. Ahora se verá su representación mediante gráficos. Esto es particularmente importante cuando se tienen que representar grafos muy grandes mediante un proceso automatizado por computadora.

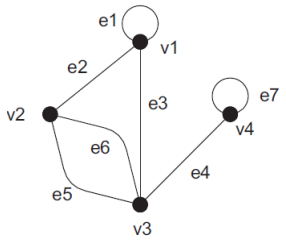
3.2.1. Estructura de datos

Existen diferentes formas de representar grafos. Talvez la manera más atractiva es mediante la **matriz de adyacencia**. Suponer que se tiene un grafo G con n vértices y m aristas. La matriz de adyacencia correspondiente no es más que una matriz A con n filas y n columnas con entradas $A[i, j]$ que denota el número de aristas que unen los vértices v_i y v_j. A modo de ilustración, la Figura 5 muestra un grafo simple acompañado de su matriz de adyacencia.

La matriz de adyacencia posee las siguientes propiedades:

- Una matriz de adyacencia es simétrica, esto es para todo $i, j, A[i, j] = A[j, i]$. Esta propiedad refleja el hecho de que una arista es representada como un par desordenado de vértices $e = \langle v_i, v_j \rangle = \langle v_j, v_i \rangle$.

- Un grafo G es simple si y sólo si para todo $i, j, A[i,j] \leq 1$ y $A[i,i] = 0$. Dicho más simple, no puede haber a lo más una arista que una los vértices v_i y v_j, y en particular, no puede haber ninguna arista que una un vértice a sí mismo.
- La suma de los valores en la fila i es igual al grado del vértice v_i, es decir, $\delta(v_i) = \sum_{j=1}^{n} A[i,j]$.

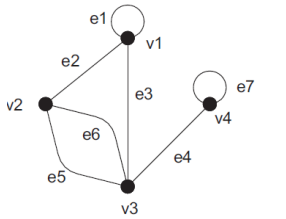


	v_1	v_2	v_3	v_4
v_1	2	1	1	0
v_2	1	0	2	0
v_3	1	2	0	1
v_4	0	0	1	2

Figura 5: Un grafo con su correspondiente matriz de adyacencia.

Como alternativa, se puede utilizar la **matriz de incidencia** de un grafo para su representación gráfica. Una matriz de incidencia M del grafo G consiste en n filas y m columnas de tal manera que $M[i,j]$ cuenta el número de veces que la arista e_j es incidente con el vértice v_i. Notar que $M[i,j]$ toma valores 0, 1 o 2: una arista no es incidente con el vértice v_i, el vértice v_i es exactamente uno de los puntos extremos, y cuando existe un bucle que une al vértice v_i así mismo. La Figura 6 muestra la matriz de incidencia para el grafo de la Figura 5. Las propiedades que posee una matriz de incidencia son las siguientes:

- Un grafo G no tiene bucles si y sólo si para todo $i, j, M[i,j] \leq 1$.
- La suma de todos los valores de la fila i es igual al grado del vértice v_i. En términos matemáticos, esto es expresado como $\forall i : \delta(v_i) = \sum_{j=1}^{m} M[i,j]$.
- Debido a que cada arista tiene exactamente dos no necesariamente puntos extremos distintos, se sabe que para todo $j, \sum_{i=1}^{n} M[i,j] = 2$.



	e_1	e_2	e_3	e_4	e_5	e_6	e_7
v_1	2	1	1	0	0	0	0
v_2	0	1	0	0	1	1	0
v_3	0	0	1	1	1	1	0
v_4	0	0	0	1	0	0	2

Figura 6: Un grafo con su correspondiente matriz de incidencia.

Uno de los problemas con el uso de la matriz de adyacencia o con la matriz de incidencia, es que sin más optimizaciones el número total de elementos para representar un grafo es n o m, respectivamente. Esto no es muy eficiente cuando se trabaja con grafos muy grandes, especialmente cuando el número de aristas es relativamente pequeño.

Una forma más eficiente de representación, y usado en la práctica, es elaborar una **lista de aristas**. En este caso, simplemente se enumeran todas las aristas de un grafo G especificando cada arista con los vértices que son incidentes. Se debe tener en cuenta que esta representación crece linealmente con el número de aristas. Por ejemplo, la representación de la Figura 6 es:

$$(\langle v_1, v_1\rangle, \langle v_1, v_2\rangle, \langle v_1, v_3\rangle, \langle v_2, v_3\rangle, \langle v_2, v_3\rangle, \langle v_3, v_4\rangle, \langle v_4, v_4\rangle)$$

3.2.2. Isomorfismo de Grafos

Las representaciones de los grafos son independientes de cómo se dibuja un grafo. No importa si se representa un grafo mediante la matriz de adyacencia, matriz de incidencia o lista de aristas, lo importante es atribuir correctamente las etiquetas a los vértices y a las aristas. De esta forma se pondrán tener iguales representaciones de grafos pero con diferentes dibujos (ver Figura 7). Esta noción de similitud se conoce con el nombre de **isomorfismo** de grafos.

Definición 3.8. *: Considere dos grafos $G = (V, E)$ y $G^* = (V^*, E^*)$. G y G^* son isomorfos si existe una correlación de uno es a uno $\phi : V \to V^*$ tal que para cada arista $e \in E$ con $e = \langle u, v\rangle$, existe una única arista $e^* \in E^*$ con $e^* = \langle \phi(u), \phi(v)\rangle$.*

Dicho de otra manera, dos grafos G y G^* son isomorfos si de manera única se pueden mapear los vértices y aristas de G a los de G^*, de tal manera que si dos vértices eran unidos en G por un número determinado de aristas, sus homólogos en G^* estarán unidos por el mismo número de aristas.

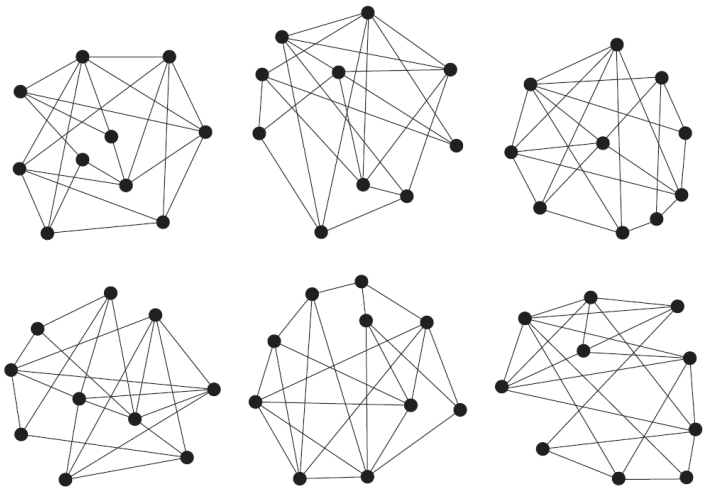


Figura 7: Seis diferentes dibujos de grafos con la misma representación, es decir, grafos isomorfos.

En muchos casos, es relativamente sencillo comprobar si dos grafos son isomorfos, ya que es necesario que se cumplan ciertos requisitos. Por ejemplo, los dos grafos deben tener el mismo número de vértices y aristas, pero el requisito más importante es que, es necesario que sus respectivas secuencias ordenadas de grados sean iguales. Desafortunadamente estas condiciones no son necesarias para que dos grafos sean isomorfos. Esencialmente, esto significa que una vez que todas las condiciones necesarias se hayan cumplido, se tendrá que recurrir a un método de ensayo y error.

3.3. Conectividad

Cada vértice v podría ser alcanzado por cualquier otro vértice w, en el sentido de que se podría indicar una cadena de vértices adyacentes desde v a w. Para entender mejor el concepto de conectividad entre nodos de un grafo, se describirán algunos conceptos básicos.

Definición 3.9. *: Un subgrafo H de G es llamado* ***componente*** *de G si H está conectado y no figura conectado a un subgrafo de G con más vértices o aristas. El número de componentes de G es denotado por $\omega(G)$.*

La noción de conectividad es importante, especialmente cuando se considera la robustez de redes. Robustez en este contexto, significa que tan bien la red permanece conectada cuando se quitan vértices o aristas. Hay muchas redes en las cuales la robustez de uno u otro modo desempeña un papel fundamental. Ahora se formaliza esta idea al considerar lo que se conoce como **corte de vértice y arista**.

Definición 3.10. *: Para un grafo G sea $V^* \subset V(G)$ y $E^* \subset E(G)$. V^* es llamado un* ***vértice de corte*** *si $\omega(G-V^*) > \omega(G)$. Si V^* se compone de un solo vértice v, entonces v es llamado un* ***corte de vértice****. Del mismo modo, si $\omega(G-E^*) > \omega(G)$ entonces E^* es llamado una* ***arista de corte****. Si E^* se compone de una sola arista e, entonces e es llamado un* ***corte de arista****.*

La notación usada $G-V^*$ indica el subgrafo inducido $G[V(G)^*]$. Por convención se denota $\kappa(G)$ para indicar el tamaño mínimo de un corte de vértice para el grafo G, y del mismo modo, $\lambda(G)$ denota el tamaño mínimo de un corte de arista para el grafo G. Pues resulta que, $\kappa(G) \leq \lambda(G)$, sino también que $\kappa(G)$ es menor o igual que el grado del vértice mínimo. Esta propiedad se formula en el siguiente teorema:

Teorema 3.3. *: $\kappa(G) \leq \lambda(G) \leq min\{\delta(v)|v \in V(G)\}$*

Para probar que $\kappa(G) \leq \lambda(G)$, considerar un grafo G con $\lambda(G) = k$. Entonces, si $\kappa(G) \geq k$ para algún k se dice que el grafo G es **k-conectado**. Del mismo modo, el grafo G es **k-aristas-conectadas** si $\lambda(G) \geq \kappa(G)$. Por último, un grafo para el cual $\kappa(G) = \lambda(G) = min\{\delta(v)|v \in V(G)\}$ se dice que esta **óptimamente conectado**. Si el grafo es simple entonces se debe cumplir el Teorema 3.3 (ver Figura 8).

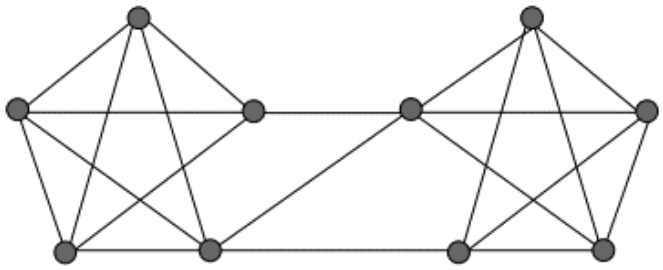


Figura 8: Grafo G en donde $\kappa(G) = 2, \lambda(G) = 3, \delta(G) = 4$.

Definición 3.11. *: Considerar un grafo G y una colección P de $(u,v)-$caminos en G, con $u, v \in V(G)$. P es* ***vértice independiente*** *si para todo $(u,v)-$caminos $P_1, P_2 \in P$ se tiene que $V(P_1) \cap V(P_2) = \{u,v\}$. La colección es* ***arista independiente*** *si para todos sus $(u,v)-$caminos P_1 y P_2, se tiene que $E(P_1) \cap E(P_2) = \phi$.*

En otras palabras, dos $(u, v)-$caminos P_1 y P_2 son vértice independiente si sólo comparten los vértices u y v, en cambio, son arista independiente si no tienen ninguna arista en común. Usando el camino de independencia, se llega a uno de los teoremas más fundamentales en la teoría de grafos, formulado por el matemático *Karl Menger*.

Teorema 3.4. *(Menger): Sea G un grafo conexo y u y v dos vértices no adyacentes en G. El mínimo número de vértices en un vértice de corte que separa a u y v es igual al máximo número de pares de caminos de vértices independientes entre u y v. Análogamente, el mínimo número de aristas en una arista de corte que separa a u y v, es igual al máximo número de pares de caminos de aristas independientes entre u y v.*

Para determinar el número mínimo de aristas de un grafo k-conectado, se debe estudiar el denominado **grafo de Harary**.

Definición 3.12. *: Un **grafo Harary** $H_{k,n}$ es un grafo simple k-conectado con n vértices y con un mínimo número de aristas.*

Para saber el número de aristas que tiene un grafo Harary, se demostrará que $H_{k,n}$ tiene exactamente $[k \cdot n/2]$ aristas, es decir, el número natural más pequeño de aristas superiores o iguales a $k \cdot n/2$. Para este fin, se deben etiquetar los vértices en $H_{k,n}$ como $0, 1, ..., n-1$ y organizarlos gráficamente como un círculo. Siguiendo la idea de Bondy y Murty [45], se considera los siguientes tres casos de combinaciones de k y n.

- **k es par**: Se construye $H_{k,n}$, uniendo cada vértice i hasta sus $k/2$ vecinos de la mano izquierda más cercanos y sus $k/2$ vecinos de la mano derecha más cercanos.
- **k es impar, n es par**: En este caso, se construye $H_{k-1,n}$ y se agregan $n/2$ aristas uniendo vértice i a su vecino de la mano izquierda a una distancia $n/2$ (con $0 \leq i \leq \frac{n}{2}$). En otras palabras, se añaden aristas $\left\langle 0, \frac{n}{2} \right\rangle, \left\langle 1, 1+\frac{n}{2} \right\rangle, \left\langle 2, 2+\frac{n}{2} \right\rangle, ..., \left\langle \frac{n-2}{2}, n-1 \right\rangle$.
- **k es impar, n es impar**: En este caso, al igual que el anterior, primero se construye $H_{k-1,n}$ y se agregan $(n+1)/2$ aristas $\left\langle 0, \frac{n-1}{2} \right\rangle, \left\langle 1, 1+\frac{n-1}{2} \right\rangle, ..., \left\langle \frac{n-1}{2}, n-1 \right\rangle$.

Para aclarar la construcción de estos grafos, la Figura 9 muestra los grafos $H_{4,8}, H_{5,8}$ y la construcción de $H_{5,9}$ desde $H_{4,9}$.

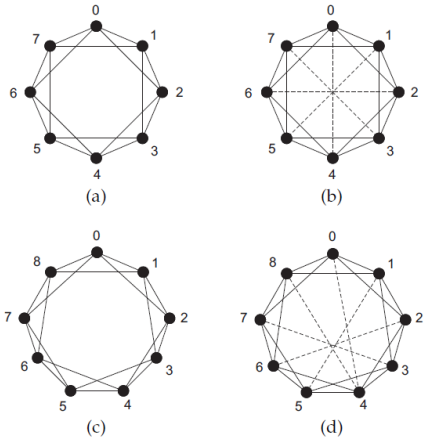


Figura 9: Varios grafos Harary: (a) $H_{4,8}$, (b) $H_{5,8}$, (c) $H_{4,9}$, y (d) $H_{5,9}$. Los bordes son los que se suman para obtener $H_{5,8}$ desde $H_{4,8}$, y $H_{5,9}$ desde $H_{4,9}$, respectivamente.

Ahora que se sabe el procedimiento para construir grafos Harary, es necesario tener en cuenta que el número mínimo de aristas para cualquier grafo simple G k-conectado es:

$$|E(H_{k,n})| = \frac{nk}{2} \tag{6}$$

4. Redes Sociales

Las redes sociales suelen representar las relaciones entre "entidades" sociales; ya sean relaciones dadas por las comunicaciones entre los miembros de un grupo u otro tipo de relaciones. Muchos análisis de redes sociales se centran en el análisis estructural de las redes, todo esto con el fin de ayudar a explicar el comportamiento social. Estos métodos son tradicionalmente usados en las ciencias sociales y del comportamiento, pero hoy en día se aplican también en el campo de la informática con el fin de analizar y estudiar estructuralmente las relaciones entre los elementos que componen una red compleja.

4.1. Análisis de Redes Sociales

El objetivo principal del análisis de redes sociales (SNA) es detectar e interpretar los patrones de las relaciones sociales entre los individuos. Sin embargo, el campo también se adentró en muchas áreas de investigación como la medicina, los sistemas de transporte, los sistemas de información, sistemas de computación y la ciencia organizacional. A pesar de que el dominio de la aplicación determina la forma apropiada de análisis, los métodos que son frecuentes en el análisis de redes se pueden distinguir por el nivel de análisis. Brandes y Erlebach [10] proponen tres niveles de análisis:

- *Análisis a nivel de elementos.* Una de las cuestiones fundamentales en este tipo de análisis es detectar la relevancia de un elemento del grafo, es decir, de un vértice o una arista: ¿Qué tan importante es este nodo en el grafo?. Habitualmente, las medidas para evaluar la importancia de un nodo se basan en su centralidad en la red. La centralidad de los elementos de las redes sociales se pueden determinar utilizando diferentes estadísticas locales, tales como el grado (entrada/salida), el coeficiente de agrupación para evaluar la importancia de un nodo, la intermediación (betweenness) para evaluar la importancia de una arista del grafo, entre muchas otras. Ejemplos de aplicación de este tipo de análisis pueden encontrarse en [9, 26].

- *Análisis a nivel de grupos.* Un objetivo común en las redes sociales es separar en grupos a los individuos que tiene conexiones fuertes. Por ejemplo, en las redes de interacción, las personas que forman un grupo en particular interactúan más estrechamente entre sí que con individuos de otros grupos de la red. Un grupo se caracteriza (entre otras cosas) por las fuertes relaciones (conexiones) entre sus miembros. El subgrafo inducido por este grupo tiene una mayor conectividad entre cada par de miembros dentro del grupo en comparación con nodos fuera del grupo. El punto de partida de todos estos conceptos se relaciona con la idea de subgrafos cohesivos [46, 43].

- *Análisis a nivel de redes.* Este enfoque pretende estudiar las propiedades del grafo en su conjunto. Las propiedades de red son señaladas para evaluar la similitud entre redes, además, los estadísticos obtenidos en la red muchas veces reflejan rasgos característicos con otras redes, por ejemplo, en un cierto dominio de aplicación. Por otra parte, observar cambios en las propiedades de una red podría proporcionar indicaciones importantes para la interpretación del análisis temporal (dinámico) de la red.

5. Detección de comunidades

Dentro del análisis de redes sociales (SNA) se encuentra la búsqueda de comunidades en grafos. En estos grafos, las entidades de una red social son representadas mediante nodos o vértices y las relaciones entre estos nodos son representadas por las aristas o arcos, así, la idea de la búsqueda de comunidades se basa en encontrar estructuras altamente cohesivas en la red. Según Chakrabarti [12] una comunidad es generalmente considerada como *"un conjunto de nodos donde cada uno de éstos está más cerca de los otros nodos que componen la comunidad que de los que no la componen"*. De igual manera Chen [13] se refiere a las comunidades en las redes sociales como *"un conjunto de las entidades de una red social de tal manera que éstas compartan rasgos comunes o una proximidad, determinada por alguna similitud entre las entidades o una métrica relacional"*. Con respecto a la afinidad entre nodos se asumirá que toda la información de la relación se contempla en la existencia y el valor de la arista. Por lo tanto, los algoritmos que se verán a continuación buscarán encontrar comunidades dentro del grafo, utilizando para ello, exclusivamente la información aportada por la topología de éste. Es importante señalar que sólo se mencionarán algunos de los algoritmos más relevantes para la detección de comunidades disjuntas y solapadas. Para una revisión más completa de la gran gama de métodos y algoritmos existentes se sugiere revisar el estudio realizado por Santo Fortunato en relación a la búsqueda de comunidades en grafos [19].

5.1. Comunidades

Según Fortunato [19], el primer inconveniente para realizar clustering (búsqueda de comunidades) en un grafo es encontrar una definición cuantitativa de comunidad. Sin embargo, Fortunato sostiene que no existe una definición universalmente aceptada, de hecho, la definición generalmente depende del sistema que se esta analizando y/o de las aplicaciones que se requieran realizar. A continuación se detallan los tipos de problemas en la búsqueda de comunidades:

- ***Búsqueda de comunidades con información Global***: Una buena partición en comunidades puede ser definida según criterios o métricas globales, es decir, tomando en cuenta las interacciones de todos los nodos del grafo. La idea central es que la inserción o no inserción de un nodo a una comunidad afectará el resultado de las pertenencias de todos los vecinos. Para abordar este tipo de problema es necesario contar con toda la información del grafo.

  Las comunidades también pueden ser descritas con respecto al grafo en conjunto. Existen definiciones basadas en la idea de que un grafo tiene la estructura de comunidad si es diferente a un grafo arbitrario. Un **grafo arbitrario** es aquel en donde cada par de vértices tienen la misma probabilidad P de ser adyacentes. Por lo tanto, se define un **modelo nulo**, es decir, un grafo similar al original en algunos aspectos estructurales, con el fin de realizar una comparación, para verificar si el grafo en estudio posee la estructura de comunidad. El modelo nulo más conocido es el propuesto por Newman y Girvan, y consiste en una versión aleatoria del grafo original, en donde las aristas son reconectadas al azar [31]. La semejanza entre cada par de nodos con respecto a alguna propiedad, no importa sólo si están conectados mediante una arista, sino, que pueden ser estructuralmente equivalente si ellos comparten los mismos vecinos, aún si el par de nodos no son adyacentes entre sí.

- ***Búsqueda de comunidades con información Local***: En redes sociales muy grandes, no es posible acceder a la información de toda la red. En estos casos se pretende descubrir comunidades con información local, es decir, conociendo sólo un grupo de nodos, los vecinos y algunas relaciones entre ellos. Chen [13] se refiere a las comunidades locales como conjuntos de nodos con conexiones densas entre los miembros, encontradas y evaluadas en base a información local y define el problema como sigue: Supongamos que existe una porción de nodos D de la cual se conocen todos los vecinos. Dado que no se posee el conocimiento de toda la red, existe un grupo S del cual sólo se conocen los nodos que son adyacentes a D pero no el resto de relaciones posibles. Así la única manera de conocer todos los vecinos de un nodo s_i perteneciente a S es visitándolo y adquiriendo la información de sus enlaces. Al realizar este proceso se integrará este nodo al conjunto D. Este procedimiento se realiza iterativamente de manera de ir agregando nodos al conjunto D y decidir cuáles debiesen ser parte de la comunidad y cuáles no. Para ello se definirá un conjunto $C \subseteq D$ el cual debe cumplir con una cierta métrica y representa la comunidad y un conjunto B de nodos que pertenecen a D y no cumplen con esta métrica. Un ejemplo de esta métrica es que para cada nodo $c_i \in C$ todos sus nodos adyacentes pertenecen a D, de manera de que la comunidad será compuesta sólo por miembros de los que se tenga toda la información y de sus vecinos, como muestra la Figura 10. Algunos métodos para la búsqueda de comunidades con información local son los propuestos por Clauset [14] en los cuales define diferentes métricas para descubrir y evaluar las comunidades.

- ***Búsqueda de comunidades superpuestas***: En general, los métodos y algoritmos tradicionales de búsqueda de comunidades asumen que un nodo puede pertenecer a una y sólo una comunidad. Sin embargo, en las redes sociales un usuario puede tener más de un grupo de amigos, por lo tanto, puede pertenecer a más de una comunidad. En la sección 5.4 se examinará un algoritmo actual para abordar este tipo de problemas.

- ***Comunidades Dinámicas***: En general las redes sociales tienden a cambiar en el tiempo, es decir, los miembros de la(s) comunidad(es) pueden ir cambiando y/o eliminando sus relaciones con los otros miembros de la comunidad(es), modificando la estructura de la red y por lo tanto, variando las estructuras de las comunidades. El problema de la búsqueda de comunidades dinámicas ha sido estudiado por diversos autores [1, 4, 35]. La idea básica es estudiar los cambios en las relaciones entre los nodos, suponiendo inicialmente un grafo $G(N, A)$ donde n es el número total de nodos participantes en todo el período de estudio. Se tiene un conjunto de nodos de $X = \{x_1, x_2, ..., x_n\}$ y una secuencia $H = \langle P_1, P_2, ..., P_t \rangle$ de observaciones donde cada P_i representa las particiones de la red del conjunto X. El estudio de las comunidades dinámicas permite responder preguntas como cuáles son las comunidades que perduran en el tiempo y cuáles no, estudiar factores internos o externos que llevan a la desaparición de una comunidad, quiénes son los nodos que conforman el *core* (centro) de una comunidad, y preguntas globales respecto a la evolución del propósito de toda la comunidad [41], entre otras tantas interrogantes. En la sección 6 se estudia con más detalle este tipo de redes, ya que es el tema central del presente trabajo.

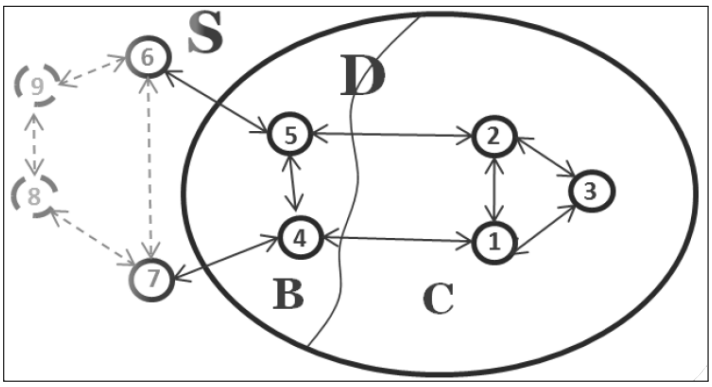


Figura 10: Problema de la búsqueda local de comunidades.

5.2. Índice de calidad: Modularidad (Q)

El término Modularidad (del inglés *Modularity*) fue presentado por Newman y Girvan [31] con la finalidad de determinar la calidad de las particiones en un grafo. Esta métrica se basa en la idea de que una distribución de comunidades o clusters no es lo que se espera por azar en una red, y por tanto, trata de medir la diferencia entre la cantidad de aristas existentes en las comunidades y el número de aristas esperadas en una red aleatoria equivalente. A este modelo estadístico se le debe acompañar de un modelo nulo que especifique que es lo que se espera por azar. Suponer el *Modelo Nulo* ϕ en donde se tiene un grafo no dirigido $G = (V, E)$ con m aristas y en la ecuación 7 se define la probabilidad entre los vértices i y j.

$$\forall (i,j) \in V : p_{i,j} = \frac{k_i k_j}{2m} \tag{7}$$

En la ecuación 7 k_i y k_j representan el grado del vértice i y j respectivamente. El valor final de la probabilidad depende de la existencia de la arista que conecta el par de vértices, por lo cual, en función de la naturaleza de la arista, el valor de la probabilidad $p_{i,j}$ entre los vértices i y j puede tomar dos formas:

$$\text{Si existe la arista } \{i,j\} \in E : 1 - p_{i,j}$$
$$\text{Si no existe la arista } \{i,j\} \notin E : -p_{i,j}$$

Entonces, el grado esperado de un nodo j es precisamente k_j y, por tanto, el número esperado de aristas es m.

En función del modelo nulo ϕ y de la ecuación 7, en la ecuación 8 se define el *Fitness de un cluster* (subconjunto de nodos) $S \subseteq V$ para todos los pares de nodos que perteneces a S.

$$Q(S) = \sum_{i,j \in S} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \tag{8}$$

Entonces la modularidad es definida como la suma de fitness individuales (ecuación 8). En consecuencia, la fórmula para calcular la modularidad en un grafo no dirigido y sin pesos se describe en la ecuación 9.

$$Q = \frac{1}{2m} \sum_{ij} \left((A_{ij} - \frac{k_i k_j}{2m}) \delta (c_i, c_j) \right) \tag{9}$$

Donde A_{ij} es el elemento de la matriz de adyacencia correspondiente a los vértices (i, j), k_i es el grado del vértice i, k_j es el grado del vértice j, m es el número de aristas totales y la función $\delta(c_i, c_j)$ es la delta de Kronecker que vale 1 si los nodos i y j están en la misma comunidad ($c_i = c_j$) y 0 si pertenecen a comunidades distintas.

Dependiendo de la aplicación y del tipo de grafo que se esta analizando, la métrica (ecuación 9) se normaliza dividiendo por $2m$ para obtener valores entre $[0, 1]$, o bien, se divide por $4m$ para obtener valores entre $[-1, 1]$.

La modularidad también se puede escribir como en la ecuación 10.

$$Q = \sum_{i}^{c} \left(e_{ii} - a_i^2\right) \tag{10}$$

Con $e_{ii} = \sum_{ij} \frac{A_{ij}}{2m} \delta\left(c_i, c_j\right)$ y $a_i = \frac{k_i}{2m}$, donde e_{ii} representa el número de aristas dentro de la comunidad y a_i el número de aristas que parte fuera de c.

Generalizando para otros tipos de grafos, la modularidad se puede definir para grafos dirigidos (ecuación 11) y para grafos no dirigidos con pesos en la aristas (ecuación 12).

$$Q = \frac{1}{m} \sum_{ij} \left((A_{ij} - \frac{k_{i_{salida}} k_{j_{entrada}}}{m}) \delta\left(C_i, C_j\right) \right) \tag{11}$$

$$Q = \frac{1}{2m} \sum_{ij} \left((A_{ij} - \frac{k_{i_{ponderado}} k_{j_{ponderado}}}{2m}) \delta\left(C_i, C_j\right) \right) \tag{12}$$

En la ecuacion 12 los pesos en las aristas deben ser no negativos, $k_{i_{ponderado}}$ y $k_{j_{ponderado}}$ son la suma de los pesos de las aristas incidentes en i y j respectivamente, A_{ij} es el peso de la arista $\{i, j\}$ y m es la suma de los pesos de todas las aristas.

Los valores que se aproximan a $Q = 1$, siendo éste el máximo valor, indican una estructura de comunidad fuerte. La modularidad no debería ser usada para comparar la calidad de la estructura de comunidades de grafos que son muy diferentes en tamaño, ya que el valor máximo de Q crece en paralelo al aumento del tamaño del grafo. Las particiones con valores Q negativos o muy cercanos a cero, puede insinuar la existencia de subgrafos con muy pocas aristas internas y muchas aristas entre comunidades (estructura multipartito). Por lo general una modularidad de 0.3 da indicios de una buena estructura de comunidad.

Para una detección basada en modularidad suponer que se tiene un grafo con estructura de comunidades pero no se sabe cuantas comunidades existen ni cuales nodos pertenecen a cada comunidad, entonces calculando la modularidad para todas las particiones (nodo a nodo) posibles se puede encontrar la estructura real de comunidades en el grafo, que es aquella partición que tiene la mayor modularidad. A continuación se describe el pseudocódigo para la detección de comunidades basado en la idea de Modularidad.

Algoritmo 1 Pseudocódigo para la detección de comunidades basado en Modularidad (Q)

Input: Conjunto de nodos de un grafo $G = (V, E)$, $V(G) = \{x_1, x_2, ..., x_n\}$
Output: Q_{max}, división de nodos

```
for k cluster desde x_1 hasta x_n do
    Tomar todas las posibles divisiones de los nodos en esos clusters.
    Calcular Q en cada caso.
end for
return Q_max, división de nodos
```

Según señala Fortunato [19] *la modularidad representa uno de los primeros intentos por lograr entender los principios del problema de clustering, integrando en su función de calidad todos los elementos esenciales, desde la definición de comunidad, pasando por la elección de un modelo nulo de comparación, hasta la expresión de solidez o fortaleza de las comunidades y particiones encontradas.* Desde ese momento muchos investigadores han tratado de extender esta definición a otros métodos para abordar la problemática del descubrimiento de comunidades en una red.

5.3. Algoritmos para la detección de comunidades disjuntas

Pues la modularidad es una función que refleja que tan buena es una partición de un grafo, mientras mayor sea el valor de Q mejor es la partición, entonces una buena estrategia es buscar maximizar esa métrica. Por tanto, dada la gran cantidad de particiones en un grafo, el problema de maximizar la modularidad es del tipo NP-Completo [8]. Es así, como se han elaborado algoritmos voraces (greedy) que permiten lograr valores aproximados al óptimo de modularidad. A continuación se presentan algunos de los algoritmos que permiten descubrir comunidades sin solapamiento basados en la optimización de la modularidad y en mapeos de paseos aleatorios (maps of random walks) en un grafo, y por último se hace énfasis en la superposición de comunidades, por lo que se describe en la sección 5.4.1 un método de percolacion clique propuesto por Palla et al. [36].

5.3.1. Algoritmo de Clauset et al.

Una de las técnicas que permiten obtener valores cercanos al óptimo de modularidad, son los algoritmos voraces (greedy). Newman [32] con su método aglomerativo jerárquico fue uno de los pioneros en el área, pero debido a los problemas de eficiencia y complejidad en el método, Clauset et al. [15] realizó cambios en la manera en que era calculada la modularidad entre los grupos (clusters), eliminando todas las operaciones innecesarias y mejorando la elección de la métrica en cada iteración.

Clauset et al. [15] propone un algoritmo jerárquico aglomerativo optimizado para la detección de comunidades en redes de dimensiones considerables. El algoritmo se basa en el cálculo de la modularidad de la ecuación 9, donde valores de Q superiores a 0.3 indican la presencia de una buena estructura de comunidad en la red o grafo.

El algoritmo voraz de Clauset et al. determina a partir de un grafo constituido ya sea por tantos nodos como comunidades, los diversos incrementos de modularidad en cada posible unión de nodos de una determinada comunidad, seleccionado un ΔQ máximo (ecuación 13) para fusionar esas comunidades.

$$\Delta Q_{i,j} = \begin{cases} \frac{1}{2m} - \frac{k_i k_j}{(2m)^2} & \text{si } i \text{ y } j \text{ estan conectados} \\ 0 & \text{otros casos} \end{cases} \tag{13}$$

$$a_i = \frac{k_i}{2m} \tag{14}$$

Entonces el algoritmo se puede definir de la siguiente manera:

1. Calcular los valores iniciales (aumento de modularidad) de $\Delta Q_{i,j}$ y a_i de acuerdo a las ecuaciones 13 y 14 respectivamente en cada posible unión de nodos, y llenar una matriz H con el mayor valor de cada fila de la matriz ΔQ.
2. Seleccionar los valores $\Delta Q_{i,j}$ más altos de H y unir las comunidades correspondientes, actualizar la matriz ΔQ, la matriz H y a_i e incrementar el valor de Q por $\Delta Q_{i,j}$.
3. Repetir el Paso 2 hasta que quede una sola comunidad.

5.3.2. Algoritmo de Blondel et al.

El algoritmo voraz (greedy) desarrollado por Blondel et al. [5] se conoce como Método de Louvain, es un método heurístico de agregación multi-nivel que se basa principalmente en la optimización de la modularidad. Es extremadamente rápido para analizar grafos con una gran cantidad de nodos y aristas, con mejores resultados en términos de modularidad que el método propuesto por Clauset et al. [15] y permite ser aplicado sobre grafos con peso.

La heurísica se divide en dos etapas las cuales son aplicadas iterativamente. En la primera etapa se asigna a cada nodo $i \in N$ a una comunidad distinta (cada nodo es una comunidad). Así, inicialmente se tiene $|N|$ comunidades distintas. Luego se considera, para cada nodo i su vecindario de nodos compuesto por $j \in V(i) \subseteq N$. Posteriormente se evalúa la ganancia de modularidad de sacar al nodo i de su comunidad y colocarlo en la comunidad de j. El nodo i se une a la comunidad que entregue la máxima ganancia de modularidad con valor positivo, de manera que si este no lo es, el nodo i continúa en la comunidad actual. Se observa que los nodos aislados nunca serán unidos a alguna comunidad, luego no afectan el resultado del proceso. Este proceso es aplicado iterativa y secuencialmente hasta que no se puedan hacer más mejoras locales en el grafo. Blondel señala que un mismo nodo puede ser analizado varias veces en este proceso, además menciona, que aunque el proceso varía según el orden de los nodos seleccionados, las pruebas realizadas señalan que esta variación no es muy significativa en términos de modularidad, pero sí en los tiempos de ejecución del algoritmo.

En el caso de existir un nodo aislado en el grafo, la ganancia de modularidad a una comunidad C puede ser calculada mediante la ecuación 15.

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \tag{15}$$

Donde $\sum_{in}$ es la suma de las aristas en C, $\sum_{tot}$ es la suma de las aristas que inciden en los nodos de C, k_i es el grado del nodo i, $k_{i,in}$ la suma de las aristas que parten en i y terminan en algún nodo de C y $m = \frac{1}{2}\sum_{i,j} A_{ij}$. Para el caso en que el nodo i pertenezca a otra comunidad, se debe evaluar el costo de remover i de la comunidad, a través de una expresión similar, más la ganancia de modularidad al utilizar la ecuación 15.

La segunda etapa del algoritmo consta en construir un nuevo grafo en el cual los nuevos nodos serán las comunidades encontradas en la etapa anterior. Luego se construyen las nuevas aristas como la suma de las aristas entre las comunidades fusionadas. Adicionalmente, se guardan las aristas dentro de una misma comunidad como una arista circular desde la comunidad i a la comunidad i. Una vez que esta etapa se completa se repite la etapa uno sobre este nuevo grafo. El autor, llama al proceso donde se utilizan estas dos etapas como una "pasada". Las pasadas son realizadas iterativamente hasta que no exista ninguna ganancia de modularidad en la etapa uno. En la Figura 11 se ilustra el proceso.

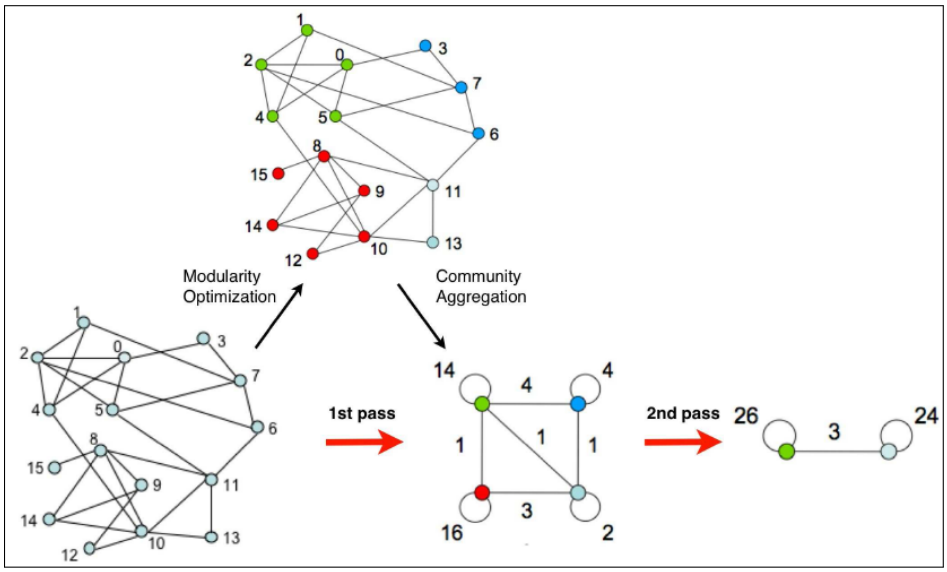


Figura 11: Algoritmo de Blondel et al. o de Louvain [5].

El algoritmo de Blondel et al. ofrece una estructura jerárquica de toda la red, en cada nivel hay un valor máximo de modularidad para cierto número de grupos. Pues, el límite de resolución se evita ya que la primera etapa del algoritmo sólo consiste en el desplazamiento individual de nodos y es poco probable que se superpongan comunidades, la superposición puede ocurrir en las fases posteriores.

5.3.3. Algoritmo de Duch & Arenas

Duch y Arenas [17] proponen un nuevo algoritmo divisivo para la optimización de la modularidad Q usando una heurística de búsqueda basada en la Optimización Extrema (del inglés Extremal Optimization, EO) algoritmo propuesto por Boettcher y Percus [7, 6]. Básicamente lo que hace el algoritmo de EO es optimizar una variable global mediante la mejora extrema de variables locales. En este caso, la variable a optimizar es Q definida en la ecuación 10. Por lo tanto, la definición de las variables locales usando el problema de optimización extrema debe estar relacionada con la contribución individual de nodos i a la sumatoria de la ecuación 10 dada una cierta partición en comunidades,

$$q_i = k_{r(i)} - k_i a_{r(i)} \tag{16}$$

donde $k_r(i)$ es el número de aristas que un nodo i perteneciente a una comunidad r tiene con nodos de la misma comunidad, y k_i es el grado del nodo i. Notar que $Q = \frac{1}{2m}\sum_i q_i$ donde

i se refiere a todos los nodos del grafo dada una cierta partición en comunidades y m es la cantidad de aristas totales en el grafo. Al re-escalar la variable local q_i por el grado del nodo i se obtiene una contribución adecuada del nodo i a la modularidad, en relación a su propio grado y normalización en el intervalo $[-1, 1]$.

$$\lambda_i = \frac{q_i}{k_i} = \frac{k_{r(i)}}{k_i} - a_{r(i)} \tag{17}$$

Se considera a λ_i como la variable local involucrada en el proceso de optimización extrema que caracteriza a un nodo i, denominándose como el fitness del nodo i.

El algoritmo de Duch para encontrar el óptimo valor de modularidad se define de la siguiente manera:

1. Inicialmente, se divide en grafo en dos particiones aleatorias, generalmente con la misma cantidad de nodos. Esta división crea una división de comunidades iniciales, donde se entiende por comunidades como componentes conectados en cada partición.

2. En cada intervalo de tiempo, el sistema auto-organiza moviendo los nodos con el menor fitness (ecuación 17) de una partición a otra. En principio, cada movimiento implica el recálculo del fitness de muchos nodos.

3. El proceso de repite hasta que se alcanza un estado óptimo con el máximo valor de modularidad Q. Luego, se eliminan las aristas entre las particiones y se procede recursivamente con cada componente conectado resultante. El proceso termina cuando la modularidad Q no puede optimizarse más.

Un ejemplo claro es lo que se muestra en la Figura 12-izquierda, donde se toma el ejemplo clásico del grafo del club de karate Zachary, dicho grafo se divide en dos particiones aleatorias, donde el número inicial de comunidades en este caso es cinco (Figura 12-derecha). Luego comienza el proceso de auto-organización: los nodos con "peor fitness" son seleccionados y se mueven desde una partición a otra partición, este movimiento provoca un recálculo del fitness de todos los demás nodos. Se calcula el nuevo valor para la modularidad Q, y se repite el proceso hasta que no existan cambios por mejorar.(ver Figura 13).

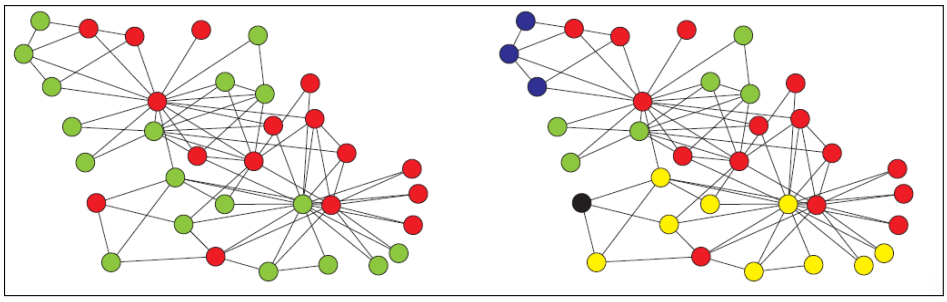


Figura 12: Izquierda: Inicialización aleatoria del grafo del club de karate Zachary en dos particiones, rojo y verde. Derecha: Cinco comunidades diferentes identificadas como componentes conectados en cada partición. Cada color define a una comunidad distinta.

La aplicación del algoritmo al grafo del club de karate Zachary proporciona el valor óptimo de modularidad después de tres iteraciones recursivas. El grafo de descompone en cuatro comunidades con un valor para la modularidad de 0.419.

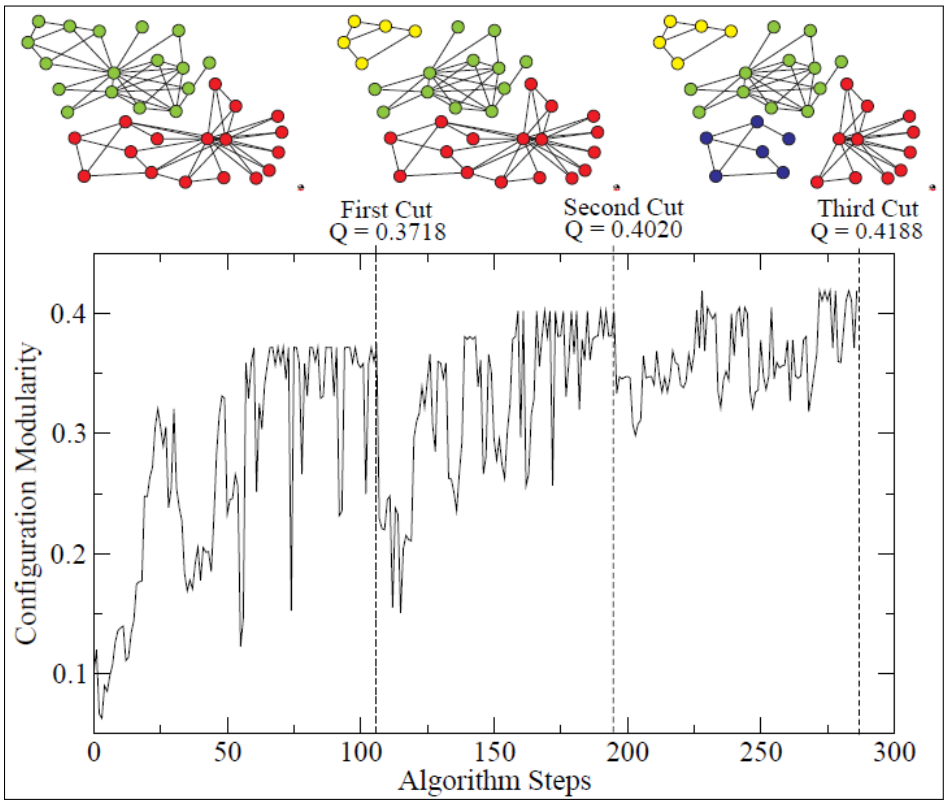


Figura 13: Arriba: Grafos luego de eliminar aristas en cada corte recursivo. Abajo: Evolución del valor de Q en cada etapa del proceso adaptativo. Las barras de separación indican las divisiones recursivas del grafo realizada la máxima Q.

5.3.4. Algoritmo Infomap

La forma tradicional de identificar comunidades en grafos dirigidos y con pesos ha sido simplemente ignorar la direcciones y los pesos de las aristas, con lo que se descarta información valiosa sobre la estructura del grafo. Pues, mapeando el flujo de todo el grafo por las interacciones locales entre los nodos, se logra conservar la información sobre la direccionalidad de las relaciones y los pesos de las aristas.

Rossvall y Bergstrom [42] proponen un algoritmo basado en compresión, llamado Infomap, el cual se basa en la idea de búsqueda de una descripción comprimida de un paseo aleatorio en un grafo dado.

El procedimiento del algoritmo Infomap es el siguiente:

1. A cada nodo se le da un nombre codificado (código Huffman [25]).
2. Cada cluster recibe un nombre codificado (código Huffman [25]).

3. Los nombres de los nodos pueden volver a ser utilizados (reciclado), siempre y cuando no se repitan en el mismo cluster.

4. El procedimiento de reciclado permite ahorrar el espacio requerido por la asignación de un nombre diferente a cada nodo.

5. Cuando un nodo pasa de un cluster a otro deberá indicar el nombre del nuevo cluster.

6. Si el grafo tiene una fuerte estructura de comunidad, entonces el reciclaje de los nombres de los nodos es conveniente.

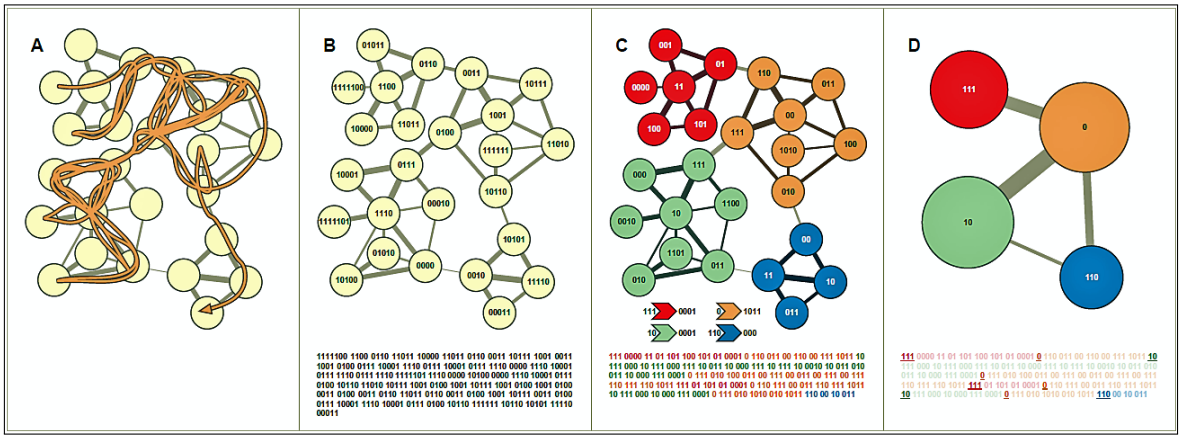


Figura 14: Detección de comunidades comprimiendo la descripción del flujo de información en grafos. En (A) se describe la trayectoria de un paseo aleatorio por el grafo, la línea naranja denota la trayectoria. En (B) mediante código de Huffman [25] se le asigna un nombre único a cada nodo del grafo. Los 314 bits que se muestran bajo el grafo describen la trayectoria de lo mostrado en (A), se comienza con 1111100 para el primer nodo en el paseo de la esquina superior izquierda, 1100 para el segundo nodo, etc., y terminando con 00011 para el último nodo del paseo aleatorio en la esquina inferior derecha. En (C) se muestra una descripción de 2 niveles del paseo aleatorio, en el que los principales clusters reciben nombres únicos, pues los nombres de los nodos dentro de los clusters se reutilizan. Los códigos de entrada y salida de cada cluster se muestran a la izquierda y derecha de las flechas bajo el grafo, respectivamente. Usando este código, se puede describir el paseo en 243 bits mostrados bajo el grafo en (C). Los tres primeros bits 111 indican que el paseo comienza en el cluster de color rojo, el código 0000 especifica el primer nodo del paseo. En (D) se muestran los nombres de los clusters y no las ubicaciones dentro de los mismos, proporcionando un óptimo granulado del grafo completo.

5.4. Algoritmos para la detección de comunidades solapadas

Los algoritmos anteriormente explicados dividen el grafo en comunidades donde cada nodo pertenece a sólo una comunidad. En las redes sociales, las personas representadas por nodos, pueden pertenecer a varias comunidades. Es así como nacen los *métodos de superposición* que tienen por objetivo identificar las comunidades dentro de un grafo, las cuales comparten uno o más nodos, tal como muestra la Figura 15.

Un método para enfrentar este problema consiste en obtener una división de la red y luego localmente buscar nodos que puedan pertenecer a más de una comunidad, obteniendo

así comunidades solapadas. Diversos autores [2, 3, 48] han propuesto métodos para detectar comunidades superpuestas, pero el método más popular y exitoso es el expuesto por Palla et al. [36] denominado algoritmo CPM (Clique Percolation Method, *Método de Percolación Clique*), que es implementado en el software libre Cfinder disponible en `http://cfinder.org`.

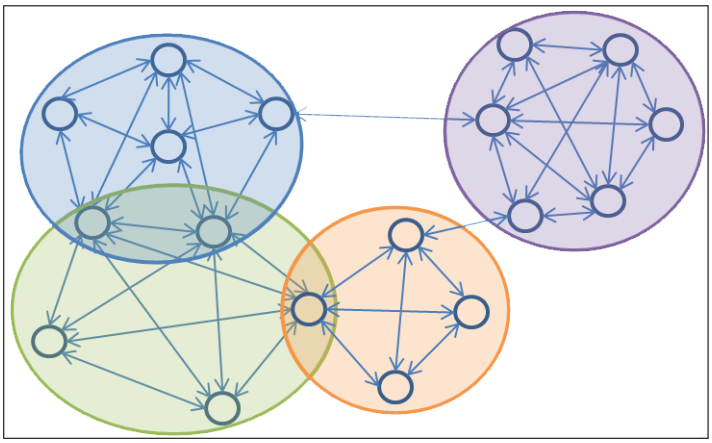


Figura 15: Comunidades solapadas.

5.4.1. Algoritmo CPM

Una de las técnicas más populares y exitosas para identificar comunidades solapadas es la expuesta por Palla et al. [36] llamada Método de Percolación Clique (CPM). Un clique, es un subgrafo mínimo en donde todos sus nodos están conectados entre sí (grafo completo).

Este algoritmo se basa en que es muy probable que las aristas internas de una comunidad formen cliques debido a que las comunidades son densas en aristas internas. Por otro lado, es probable que entre comunidades no se formen cliques debido a que no debiesen existir muchas aristas entre ellas. Palla et al. utilizan el término *k-clique* para referirse a un subgrafo de k elementos que forma un clique (ver Figura 16), luego define el término de adyacencia entre dos k-clique señalando que son adyacentes si entre ellos comparten $k-1$ nodos. Posteriormente, se define una cadena de k-clique como la unión de dichos elementos adyacentes. A continuación, señala que dos k-clique estarán conectados si ambos son parte de una misma cadena de k-clique. Así finalmente, Palla define a una comunidad de k-clique como el subgrafo con el mayor número de éstos conectados. Luego se deben buscar, a través de una heurística, los diferentes k-clique de la red para ir armando las comunidades según las definiciones anteriores. Si bien incialmente el método CPM sólo servía para redes no dirigidas y sin peso, Palla et al. extendieron su uso para ser utilizado sobre redes dirigidas [37], y Farkas para la utilización de pesos en las aristas [18]. La complejidad del algoritmo depende de muchos factores, pero en el peor caso es una complejidad exponencial. Los autores señalan de la capacidad del algoritmo de tratar con grafos del orden de 10^5 nodos. El método CPM y sus mejoras están disponibles en el software CFinder disponible en `http://cfinder.org`.

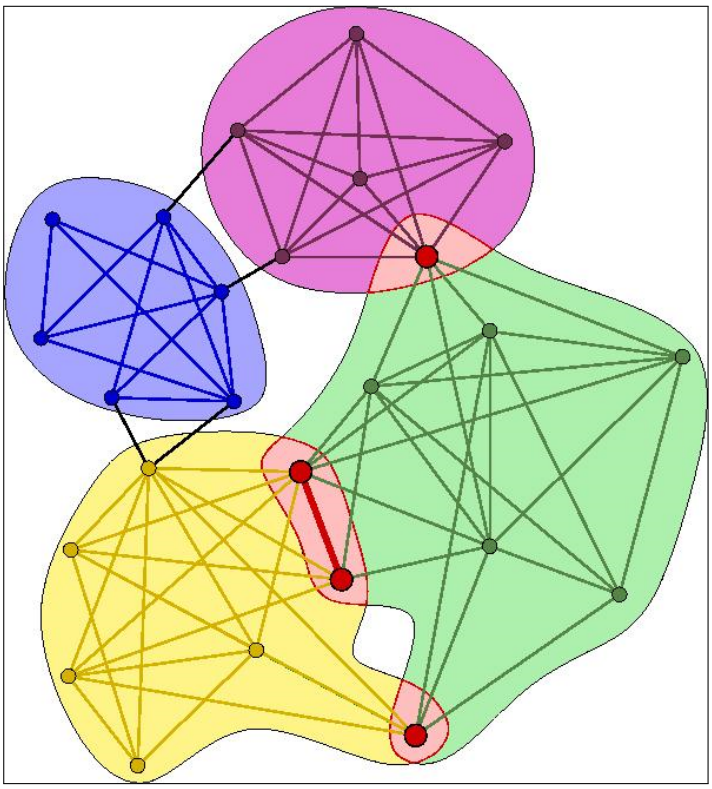


Figura 16: Comunidades k-*clique*, con $k = 4$.

6. Redes dinámicas

6.1. Notación básica

Sea $G = (V, E)$ un grafo no dirigido y sin pesos que representa una red, donde V es el conjunto de N nodos y E es el conjunto de M aristas. Denotemos por $\Omega = \{C_1, C_2, C_k\}$ la estructura de comunidades del grafo. En contraste con la estructura de comunidades disjuntas, no se restringe la intersección vacía de dos comunidades, es decir, se permite que $C_i \cap C_j = \emptyset$.

Para un nodo $u \in V$, se tiene que d_u, $N(u)$, $Com(u)$, denotan el grado, nodos vecinos y el conjunto de comunidadesa las que pertenece el nodo u, respectivamente. Para cualquier $C \subseteq V$, se define C^{in} y C^{out} como el conjunto de aristas internas y el conjunto de aristas externas de C (fuera de la comunidad), respectivamente.

6.2. Modelo de red dinámica

Sea $G_0 = (V_0, E_0)$ la red original de entrada y $G_t = (V_t, E_t)$ una imagen de la red original registrada en el tiempo t. Denotemos ΔV_t y ΔE_t respectivamente como los conjuntos de nodos y aristas que se añaden o eliminan en la red en el momento t. Notar que $\Delta G_t = (\Delta V_t, \Delta E_t)$ describe los cambios en términos generales de toda la red. La imagen de la red en el tiempo $t+1$ se expresa como una combinación de la red anterior junto con el cambio o imagen actual $G_{t+1} = G_t \cup \Delta G_t$. Por último, una red dinámica φ se define como una secuencia de imágenes de la red evolutiva: $\varphi = (G_0, G_1, G_2, ...)$.

6.3. Función de densidad

Con el fin de cuantificar la bondad de una comunidad identificada, es decir, medir la proporción de aristas internas a la comunidad C respecto al total de sus posibles aristas internas, se utiliza la conocida función de densidad ψ [20] definida como:

$$\psi(C) = \frac{\left|C^{in}\right|}{\binom{|C|}{2}} \text{ donde } C \subseteq V \tag{18}$$

Donde $\left|C^{in}\right|$ es la cantidad de aristas internas de la comunidad C y $|C|$ es la cantidad de nodos existentes. Mientras más se aproxima C a un clique de su tamaño, mayor es su valor de densidad $\psi(C)$. Para que C se considere una comunidad local se propone la función $\tau(C)$, que es el umbral necesario de densidad interna que debe poseer la comunidad C.

$$\tau(C) = \frac{\sigma(C)}{\binom{|C|}{2}} \text{ donde } \sigma(C) = \binom{|C|}{2}^{1-\frac{1}{\binom{|C|}{2}}} \tag{19}$$

Aquí $\sigma(C)$ es el umbral del número de aristas internas que basta para que C se considere una comunidad local. Particularmente, un subgrafo inducido por C es una comunidad local si y sólo si $\psi(C) \geq \tau(C)$ o equivalentemente $\left|C^{in}\right| \geq \sigma(C)$. Las funciones $\tau(C)$ y $\sigma(C)$ son procedimientos locales en la comunidad candidata C por lo que no se requiere de ningún parámetro de entrada por parte del usuario, además por la Proposición 1 dichas funciones $\tau(C)$ y $\sigma(C)$ van en aumento.

Proposición 1. *La función $f(n) = n^{1-\frac{1}{n}}$ es estrictamente creciente para $n \geq 4$ y $lim_{n\to\infty} f(n) = n$.*

6.4. Definición del problema

La idea es encontrar un conjunto de nodos que maximicen la función de densidad interna global $\psi(\Omega) = \sum_{C \in \Omega} \psi(C)$, ya que cuanto mayor es la densidad interna de una comunidad, más clara sería su estructura.

Dada una red dinámica $\varphi = (G_0, G_1, G_2, ...)$ donde G_0 es el grafo de entrada y $G_1, G_2, ...$ son imágenes del grafo original obtenidas a través de una colección de topologías de la red evolutiva $\Delta G_1, \Delta G_2, ...$ en el tiempo. El problema es detectar y actualizar de manera eficiente la estructura de comunidades solapadas C_t en cualquier tiempo t utilizando solamente la información de la imagen anterior C_{t-1}, así como hacer un seguimiento de la evolución de las comunidades.

En la siguiente sección se presenta el desarrollo del método compuesto por dos fases: (1) mediante FOCS identificar la estructura básica de comunidades solapadas dada la imagen original de una red estática y (2) a través de AFOCS efectuar la actualización, y seguimiento de la evolución de las comunidades solapadas en una red dinámica.

6.5. FOCS (Búsqueda de estructura de comunidades solapadas)

En primera instancia, se descubre rápidamente la estructura básica de comunidades solapadas. En general, FOCS trabaja hacia la clasificación de los nodos del grafo en diferentes grupos o comunidades, con el fin de localizar todas las partes posibles densamente conectadas en la red, acto seguido, combina las comunidades que se encuentran muy solapadas entre sí, es decir, aquellas que comparten una subestructura significativa. En FOCS, β (el umbral de solapamiento) define la cantidad de subestructura que dos comunidades pueden compartir. FOCS permite $|C_i \cap C_j| \geq 2$ para cualquier subconjunto C_i, C_j de V, por lo tanto, permite que las comunidades se superpongan, no sólo en un vértice, sino también en una parte de toda la comunidad.

6.5.1. Búsqueda de comunidades locales

Las comunidades locales conectan partes de la red cuyas densidades internas son mayores o iguales que un cierto parámetro. En FOCS, este parámetro se determina automáticamente basado en la función $\tau(\cdot)$ y en el tamaño de cada parte correspondiente. Particularmente, una comunidad local se define sobre la base de una conexión (u, v) cuando el número de conexiones internas dentro del subgrafo inducido por $C \equiv \{u, v\} \cup (N(u) \cap N(v))$ excede $\sigma(C)$, o en otras palabras, cuando $\psi(C) \geq \tau(C)$ (figura 17(a)). Sin embargo, hay un problema que eventualmente pudiera surgir durante este procedimiento: la existencia de subcomunidades en una comunidad más grande. En la práctica, se detectan comunidades grandes unificadas por comunidades más pequeñas (subcomunidades) si la comunidad más grande es en sí densamente conectada. Con el fin de filtrar este caso no deseado, se impone $\psi(\cup_{i=1}^{s} C_i) < \tau(\cup_{i=1}^{s} C_i) \forall s = 1... |C|$. FOCS detecta aquellas comunidades locales que están compuestas por 4 o más nodos, identificando comunidades pequeñas al final del proceso de búsqueda. Esto tiene sentido en términos de redes sociales, ya que generalmente una comunidad esta compuesta por más de 3 participantes. El Algoritmo 2 describe el procedimiento de búsqueda, en donde su tiempo de complejidad es $O(dM)$ con $d = max_{v \in V} d_v$.

Algoritmo 2 Búsqueda de comunidades locales

Input: $G = (V, E)$
Output: Una Colección de comunidades Ω_r

```
for (u, v) ∈ E do
  if Com(u) ∩ Com(v) = ∅ then
    Dejar que C = {u, v} ∪ N(u) ∩ N(v);
    if |C^in| ≥ σ(C) and |C| ≥ 4 then
      Definir C como una comunidad local;
      /*Se incluye C dentro de la estructura de comunidades*/
      Ω_r = Ω_r ∪ {C};
    end if
  end if
end for
```

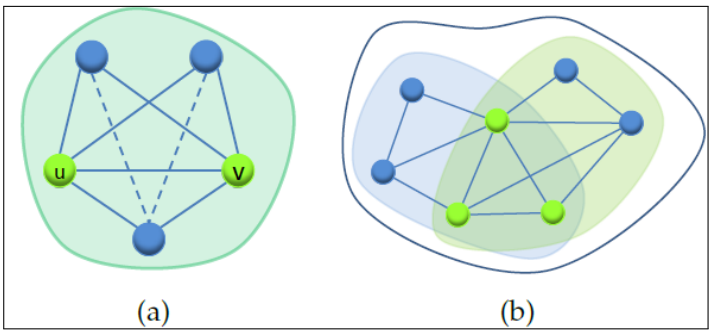


Figura 17: (a) Una comunidad local C definida por una arista (u,v). Donde $\psi(C) = 0{,}9 \geq \tau(C) = 0{,}725$ (b) Combinación de dos comunidades locales que comparten una subestructura significativa (OS score $= 1{,}027 \geq \beta = 0{,}8$).

6.5.2. Combinación de comunidades solapadas

Una vez realizado el proceso de búsqueda, la estructura de comunidad del grafo se representa como una colección de (posiblemente solapadas) partes densas de la red junto con los valores atípicos encontrados. Es posible que algunas de esas partes densas puedan compartir subestructuras importantes, por lo que deben unirse si es que están muy solapadas entre sí. Para lo anterior, se propone una función de puntuación de solapamiento (*Overlapping Score*) para dos comunidades, definida como:

$$OS(C_i, C_j) = \frac{|I_{ij}|}{min\{|C_i|, |C_j|\}} + \frac{\left|I_{ij}^{in}\right|}{min\left\{\left|C_i^{in}\right|, \left|C_j^{in}\right|\right\}} \tag{20}$$

Donde $I_{ij} = C_i \cap C_j$. La función $OS(C_i, C_j)$ valora la importancia de los nodos y aristas en común que se comparten entre C_i y C_j. En comparación con la métrica de distancia que se sugiere en [30], la función $OS(\cdot, \cdot)$ no sólo tiene en cuenta la fracción de nodos en común, sino que también considera la cantidad de aristas en común entre dos comunidades, lo que es esencial para su combinación. Por otra parte, la función $OS(\cdot, \cdot)$ es simétrica y escala bien con el tamaño de cualquier comunidad, y cuanto mayor sea su valor, las comunidades en consideración más deberían combinarse. En este proceso, las comunidades C_i y C_j se fusionan

si $OS(C_i, C_j) \geq \beta$ (ver figura 17(b)).

Algoritmo 3 Combinación de comunidades solapadas

Input: Estructura de comunidades Ω_r
Output: Una refinada estructura de comunidades Ω_f

```
1: Ω_f ← Ω_r
2: for C_i, C_j ∈ Ω_r and !done do
3:    if OS(C_i, C_j) ≥ β then
4:       C ← Combinar C_i y C_j;
5:       /*Actualizar la estructura actual*/
6:       Ω_f = (Ω_f \ {C_i ∪ C_j}) ∪ C;
7:       done ← false;
8:    end if
9: end for
```

El tiempo de complejidad del Algoritmo 3 es $O(N_0^2)$ donde N_0 es el número de comunidades locales.

6.5.3. Revisión de nodos no asignados

Cuando se ejecutan los dos algoritmos anteriores, siguen existiendo nodos y/o aristas sobrantes debido a su poca atracción con el resto del grafo. Debido a su tamaño limitado el Algoritmo 2 se salta las comunidades pequeñas de un tamaño menor a 4 nodos, y por lo tanto, puede dejar de lado nodos sin asignar. Estos nodos no se tocarán en el Algoritmo 3 ya que no pertenecen a ninguna comunidad local, y en consecuencia, se mantendrán sin asignación. Por lo tanto, se debe volver a estos nodos, ya sea para agruparlos en comunidades o clasificarlos como valores atípicos en función de su conectividad. Alternativamente, este proceso puede ser pensado como una comunidad tratando de incorporar nodos adyacentes sin asignar que son similares a la comunidad de acogida. Para ello, se necesita una función fitness con la finalidad de cuantificar la *similitud* entre un nodo u y su comunidad vecina C. Se utiliza la siguiente función fitness usada en [28, 21, 30]:

$$F_C = \frac{\left|C^{in}\right|}{2\left|C^{in}\right| + \left|C^{out}\right|} \text{ donde } C \subseteq V. \tag{21}$$

Teniendo en cuenta la ecuación 21, una comunidad C incorporará cualquier nodo adyacente sin asignar de máxima similitud de manera *greedy*, siempre y cuando que el nodo recién unido no se reduzca por la comunidad del valor fitness actual. Si no existe tal nodo, C es definida como una comunidad final del grafo. En tanto, los nodos que no fueron incorporados a ninguna comunidad mediante este último procedimiento, se identifican como valores atípicos. El Algoritmo 4 detalla este procedimiento en detalle.

Algoritmo 4 Revisión de nodos no asignados

Input: Una refinada estructura de comunidades $\Omega_f = \left\{C'_1, C'_2, ..., C'_t\right\}$
Output: Una básica estructura de comunidades $\Omega = \{C_1, C_2, ..., C_k\}$

```
1: Ω = Ω_f
2: for u ∈ V and Com(u) = ∅ do
3:    NC(u) = {C_j ∈ Ω | u es adyacente a C_j};
4:    for C_j ∈ NC(u) do
5:       if F_{C_j ∪ {u}} ≥ F_{C_j} then
6:          C_j ← C_j ∪ {u};
7:          Com(u) ← Com(u) ∪ {j};
8:       end if
9:    end for
10:   if Com(u) = ⊘ then
11:      Clasificar a u como valor atípico;
12:   end if
13: end for
```

6.6. AFOCS (Búsqueda adaptativa de estructura de comunidades solapadas)

En particular, AFOCS se usa para actualizar adaptativamente y rastrear las comunidades del grafo, que fueron previamente inicializadas por FOCS. Se debe tener en cuenta que FOCS sólo se ejecuta una vez en G_0, después AFOCS se hará cargo y manejará todos los cambios introducidos en el grafo. Suponer que $G = (V, E)$ y $\Omega = \{C_1, C_2, ..., C_n\}$ es el grafo actual y su correspondiente estructura de comunidades solapadas, respectivamente. Para cada comunidad C de G, por definición el número de aristas que unen C con otras comunidades es menor que el número de aristas que existen dentro de la propia comunidad C.

Intuitivamente, la adición de aristas internas o la eliminación de aristas externas entre comunidades de G, hará que éstas se fortalezcan y, en consecuencia, la estructura de G será más clara. Del mismo modo, la eliminación de aristas internas y la adición de aristas externas, disminuirá la densidad interna de una comunidad, y como resultado, se verá afectada la estructura de G. El proceso de actualización es muy complicado y difícil, puesto que cualquier cambio insignificante en la topología del grafo, posiblemente, podría conducir a una transformación impredecible de la estructura de comunidad. Con el fin de reflejar estos cambios en una red compleja, el modelo de grafo subyacente se actualiza con frecuencia por cualquier inserción o extracción de un nodo o un conjunto de nodos, o una arista o un conjunto de aristas. En base a lo anterior, los cambios en el grafo se pueden ver como una colección de eventos simples, cuyos detalles son los siguientes:

- $newNode(V + u)$: Un nuevo nodo u y su(s) arista(s) adyacente(s) se insertan.
- $removeNode(V - u)$: Un nodo u y su(s) arista(s) adyacente(s) se eliminan del grafo.
- $newEdge(E + e)$: Una nueva arista e que conecta a dos nodos existentes se inserta.
- $removeEdge(E - e)$: Una arista e es eliminada del grafo.

6.6.1. Inserción de nuevo nodo

Cuando un nuevo nodo u y sus aristas asociadas son introducidas en el grafo, las posibilidades son (1) u puede venir sin arista adyacente o (2) con muchas aristas que conectan una o más posiblemente comunidades solapadas. Si u no tiene arista adyacente, simplemente u se une en el conjunto de valores atípicos y se preserva la estructura de comunidad actual. El caso interesante sucede, y lo hace normalmente, cuando u viene con varias aristas que conectan una o más comunidades existentes. Dado que las comunidades de la red pueden solaparse entre sí, es necesario determinar a cuales primero u debe unirse con el fin de maximizar la densidad interna ganada. Por la Proposición 2, se da una condición necesaria para un nuevo nodo con el fin de unirse a una comunidad existente, es decir, el algoritmo unirá al nodo u en C si el número de aristas que tiene u para C es suficiente:

$$d_{ui} > \frac{2\left|C^{in}\right|}{|C|-1} \tag{22}$$

Sin embargo, el no cumplimiento de esta condición no implica que u no deba pertenecer a C, ya que potencialmente puede reunir alguna subestructura de C para formar una nueva comunidad (Figura 18), razón por la cual también se maneja dicha posibilidad. El Algoritmo 5 presenta el procedimiento.

Algoritmo 5 Inserción de un nuevo nodo

Input: Estructura de comunidad actual Ω_{t-1}
Output: Una estructura de comunidad actualizada Ω_t

$C_1, C_2, ..., C_k \leftarrow$ Comunidades adyacentes de u;
for $i = 1$ **to** k **do**
  if $d_{ui} > \frac{2|C_i^{in}|}{|C_i|-1}$ **then**
    $C_i \leftarrow C_i \cup \{u\}$;
  else
    $C \leftarrow N(u) \cap C_i$;
    if $\psi(C) \geq \tau(C)$ and $|C| \geq 4$ **then**
      $C_i \leftarrow C_i \cup \{u\}$;
    end if
  end if
end for
/*Comprobar nuevas comunidades desde valores atípicos*/
for $v \in C_0$ and $Com(v) \cap Com(u) = \emptyset$ **do**
  $C \equiv N(u) \cap N(v)$;
  if $\psi(C) \geq \tau(C)$ and $|C| \geq 4$ **then**
    Definir C como una nueva comunidad;
  end if
end for
Combinar comunidades solapadas sobre $C_1, C_2, ..., C_k$ y C_0;
Actualizar Ω_t;

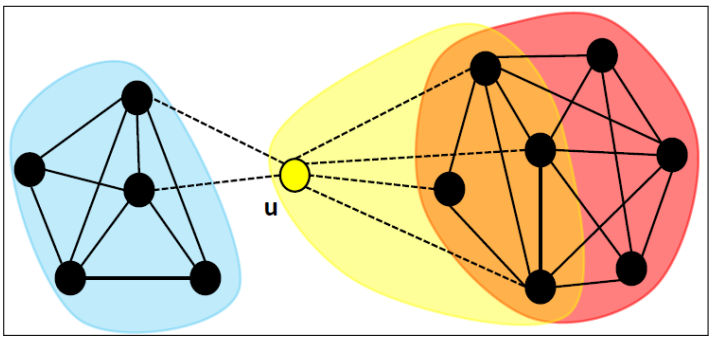


Figura 18: Cuando un nuevo nodo u es insertado, u podría reunir algunos nodos de una comunidad existente (rojo) para formar una nueva comunidad (amarillo).

Proposición 2. *Suponer que u es un nodo recién insertado con aristas d_{ui} para cada comunidad adyacente C_i, u se unirá en C_i si $d_{ui} > \frac{2|C^{in}|}{|C|-1}$.*

6.6.2. Inserción de nueva arista

Cuando se inserta en el grafo una nueva arista $e = (u, v)$ que conecta dos nodos existentes u y v, se pueden dar cuatro escenarios diferentes: (1) e es la única arista dentro de una sola comunidad C, (2) e está dentro de la intersección de dos (o más) comunidades, (3) e se une a dos comunidades que se encuentran separadas y (4) e está cruzando comunidades solapadas. Si e está totalmente dentro de una comunidad C, su presencia va a fortalecer la densidad interna de C, ya que la adición de e no debe dividir a la comunidad actual C en subestructuras más pequeñas. La misma reacción se aplica en el segundo subcaso cuando e esta dentro de la intersección de dos comunidades, ya que sus densidades interiores se incrementan. Así, en estos dos primeros casos, la estructura actual del grafo queda intacta. El manejo de los dos últimos subcasos es complicado, ya que cualquiera de ellos pueden, o no, tener efecto sobre la estructura actual del grafo o de forma impredecible formar una nueva comunidad, y además se pueden solapar o combinar con otros (Figura 19). Se sabe que cuando una nueva arista (u, v) se introduce entre dos comunidades disjuntas C_u y C_v, ni u ni v deben trasladarse a C_u o C_v. Sin embargo, todavía existe una posibilidad de que la inserción de esta nueva arista, junto con algunas subestructuras de C_u o C_v, sea suficiente para formar una nueva comunidad que puede solaparse no sólo con C_u y C_v, sino también con otras comunidades. Los otros subcasos se manejan de manera similar. El Algoritmo 6 describe este procedimiento.

Algoritmo 6 Inserción de una nueva arista (u, v)

Input: Estructura de comunidad actual Ω_{t-1}
Output: Una estructura de comunidad actualizada Ω_t

if $((u, v) \in$ a una sola comunidad OR $(u, v) \in C_u \cap C_v)$ **then**
  $C_t \leftarrow C_{t-1}$;
else
  if $Com(u) \cap Com(v) = \emptyset$ **then**
    $C \leftarrow N(u) \cap N(v)$;
    if $\psi(C) \geq \tau(C)$ **then**
      Definir C como una nueva comunidad;
      Comprobar para combinar en $Com(u), Com(v)$ y C;
    else
      for $D \in Com(u)$(or $D^{'} \in Com(v)$) **do**
        if $\psi(D \cup \{v\}) \geq \tau(D)$(or $\psi(D^{'} \cup \{u\}) \geq \tau(D^{'})$) **then**
          $D \leftarrow D \cup \{v\}$ (or $D^{'} \leftarrow D^{'} \cup \{u\}$);
        end if
      end for
      Combinar comunidades solapadas sobre $D^{'}s$(or $D^{'}$);
    end if
    Actualizar Ω_t;
  end if
end if

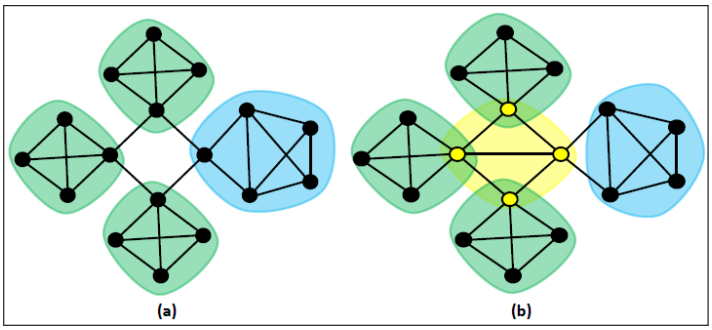


Figura 19: (a) El grafo con 4 comunidades disjuntas (no solapadas) (b) Cuando se añade la arista central, el centro de nodos forma una nueva comunidad (amarillo).

6.6.3. Eliminación de nodo

Cuando un nodo u se elimina del grafo, como consecuencia, todas sus aristas adyacentes también se eliminarán. Si u es un nodo atípico, simplemente se elimina u y sus correspondientes aristas de la estructura actual y las comunidades del grafo se mantendrán sin cambios. En situaciones en que u no es un nodo atípico, la comunidad resultante es compleja de tratar. Para dar una idea de este efecto, considerar los dos ejemplos ilustrados en la Figura 20. En el primer ejemplo, cuando C es casi un clique completo, la eliminación de cualquier nodo no va a romper esa comunidad. Sin embargo, si se elimina un nodo se tienden a unir los otros nodos dentro de una comunidad, la subestructura sobrante se divide en una más pequeña, junto con un nodo que posteriormente se fusionó con una de sus comunidades cercanas. Por

lo tanto, la identificación de los restos de la estructura de C es tarea vital una vez que el nodo u se elimina de C. Para controlar dicho escenario, primero se examina la densidad interna de C excluyendo el nodo eliminado u. Si el número de aristas internas aún es suficiente, por ejemplo $\psi(C \setminus \{u\}) \geq \tau(C \setminus \{u\})$, se pueden mantener con seguridad las comunidades actuales del grafo. De lo contrario, se aplica el Algoritmo 2 en el subgrafo inducido por $C \setminus \{u\}$ para identificar rápidamente las subestructuras sobrantes en C, y luego dejar que estas subestructuras (comunidades) busquen un conjunto de nodos sin asignar de manera tal que les ayuden a aumentar sus densidades internas. Por último, a nivel local, se comprueba para la combinación de comunidades, en su caso, mediante el uso de un algoritmo similar al Algoritmo 3.

Algoritmo 7 Eliminación de un nodo

Input: Estructura de comunidad actual Ω_{t-1}
Output: Una estructura de comunidad actualizada Ω_t

```
for C ∈ Com(u) and ψ(C \ {u}) < τ(C \ {u}) do
    LC ← Comunidades locales por el Algoritmo 2 en C \ {u};
    for C_i ∈ LC and |C_i| ≥ 4 do
        S_i ← nodos tal que ψ(C_i ∪ S_i) ≥ τ(C_i ∪ S_i);
        C_i ← C_i ∪ S_i;
    end for
    Combinar comunidades solapadas sobre LC;
end for
Actualizar Ω_t;
```

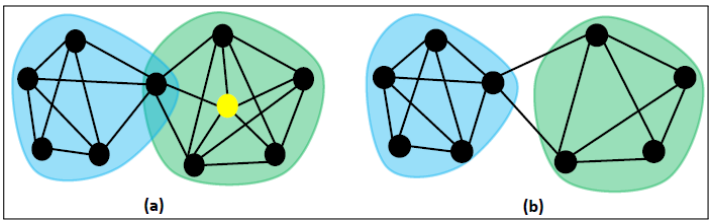


Figura 20: (a) Dos comunidades solapadas (b) Cuando se retira el nodo central, la nueva estructura se compone de dos comunidades disjuntas (no solapadas).

6.6.4. Eliminación de arista

Cuando una arista $e = (u, v)$ es eliminada, ocurre algo similar al insertar una nueva arista (1) e está entre dos comunidades disjuntas, (2) e está dentro de una única comunidad, (3) e está dentro de la intersección de dos (o más) comunidades y, finalmente, (4) e está cruzando comunidades solapadas. En el primer subcaso, cuando e cruza dos comunidades disjuntas, su eliminación hará que la estructura del grafo sea más clara (ya que ahora habrán menos aristas entre comunidades), y por lo tanto, las comunidades actuales se mantendrán sin cambios. Cuando e está dentro de una única comunidad C, el manejo de su eliminación es complicado, ya que esto puede conducir a una transformación impredecible de dicha comunidad: C podría ser inalterada o bien descompuesta en partes más pequeñas si contiene subestructuras que sean menos atractivas unas con otras, como se muestra en la Figura 21. Por lo tanto, el problema de identificar la estructura de la comunidad restante se convierte en la parte central

no sólo para este caso, sino también para los demás. Para resolver lo anterior, primero se verifica la densidad interna de la comunidad restante y, una vez más se utiliza el Algoritmo 2 y a nivel local se identifican las subestructuras sobrantes. A continuación, se comprueba la combinación de comunidades, ya que estas estructuras posiblemente pueden solaparse con las comunidades existentes en el grafo. El procedimiento detallado se describe en el Algoritmo 8.

Algoritmo 8 Eliminación de una arista

Input: Estructura de comunidad actual Ω_{t-1}
Output: Una estructura de comunidad actualizada Ω_t

if (u, v) es una arista atípica **then**
  $\Omega_t = (\Omega_{t-1} \setminus \{u, v\}) \cup \{u\} \cup \{v\}$;
else if $d_u = 1$ (or $d_v = 1$) **then**
  $\Omega_t = (\Omega_{t-1} \setminus C(u)) \cup \{u\} \cup C(v)$;
else if $C \equiv C(u) \cap C(v) = \emptyset$ **then**
  $\Omega_t = \Omega_{t-1}$;
else if $\psi(C \setminus (u, v)) < \tau(C \setminus (u, v))$ **then**
  $LC \leftarrow$ Comunidades locales por el Algoritmo 2 en $C \setminus (u, v)$;
  Definir cada $L \in LC$ como una comunidad local de Ω_{t-1};
  Combinar comunidades solapadas en $L's$;
end if
Actualizar Ω_t;

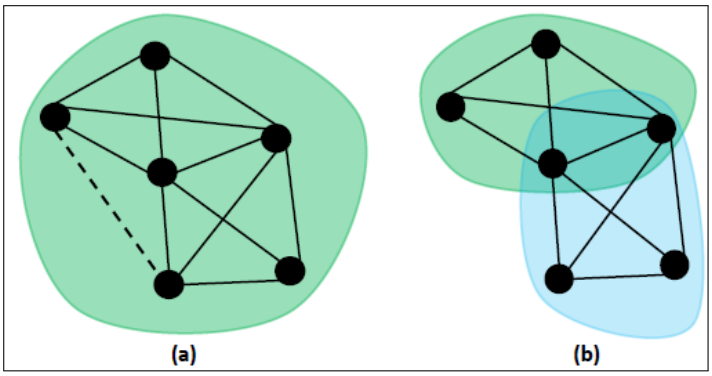


Figura 21: (a) La comunidad de origen (b) Cuando se quita la arista de puntos, la comunidad se divide en dos comunidades solapadas.

6.7. Extensión para grafos ponderados no dirigidos

Teniendo en cuenta los pesos en las aristas del grafo, se hace una abstracción de la red a un grafo ponderado no dirigido.

6.7.1. Definiciones relacionadas

Definición 1. *(grafo ponderado no dirigido) Sea* $G = (V, E, W)$ *un grafo ponderado no dirigido en donde* V *es el conjunto de todos los nodos,* $E = \{\langle u, v\rangle \,|u, v \in V\}$ *representa el conjunto de todas las aristas,* $W = \{w_{uv} \in \Re | u, v \in V;\}$ *es el peso del conjunto de aristas,* w_{uv} *es el peso de la arista* (u, v).

Definición 2. *(interactividad) Para cualquier comunidad $C_i \in \Omega$, si $u \in V$, tenemos que w_u^{in} denota la suma de los pesos de las aristas desde el nodo u hacia los otros nodos de la comunidad C_i. Si w_u^{in} no es igual a cero, entonces existe interactividad entre el nodo u y la comunidad C_i.*

Definición 3. *(comunidad atípica) Para cualquier comunidad $C_i \in \Omega$, si $u \in V$ y $w_u^{in} = 0$, entonces u es llamado un valor atípico en relación con la comunidad C_i.*

Definición 4. *(comunidad frontera) Para cualquier $C_i, C_j \in \Omega$, si $u \in C_i$ y u es interactivo tanto con C_i como con C_j, entonces u es llamado nodo frontera de la comunidad C_i, y el conjunto $F_i = \{u | u \in C_i, u$ es un nodo frontera de $C_i\}$ es llamado frontera de la comunidad C_i.*

Definición 5. *(nodo puente) Si $C = C_i \cap C_j$ y $u \in C$ y u es un vecino muy cercano de al menos un nodo dentro de C_i y C_j, entonces u es llamado nodo puente entre C_i y C_j.*

6.7.2. Modelo de red dinámica

Sea $G_0 = (V_0, E_0, W_0)$ la red original de entrada y $G_t = (V_t, E_t, W_t)$ una imagen de la red original registrada en el tiempo t. Denotemos ΔV_t, ΔE_t y ΔW_t respectivamente como los conjuntos de nodos, aristas y pesos de aristas que se añaden o eliminan en la red en el momento t. Notar que $\Delta G_t = (\Delta V_t, \Delta E_t, \Delta W_t)$ describe los cambios en términos generales de toda la red. La imagen de la red en el tiempo $t+1$ se expresa como una combinación de la red anterior junto con el cambio o imagen actual $G_{t+1} = G_t \cup \Delta G_t$. Por último, una red dinámica φ se define como una secuencia de imágenes de la red evolutiva: $\varphi = (G_0, G_1, G_2, ...)$.

6.7.3. Función de pesos

La idea es maximizar la función $W(\Omega) = \sum_{C \in \Omega} W(C)$ [16] de forma tal que cuanto más se acerque a 1, la función de densidad ψ [20] podrá alcanzar su óptimo valor. Tomando en cuenta esta extensión con pesos en las aristas, la detección de comunidades solapadas mediante FOCS se verá limitada por el factor numérico de cada enlace, entre más alto el valor, o más cercano a la media con respecto al valor máximo permitido, y dependiendo de la densidad interna de aristas en la comunidad, la estructura de comunidad será más fuerte.

$$W(C) = \frac{1}{2w} \sum_{u,v \in C} (w_{u,v} - \frac{w_u^{in} * w_v^{in}}{2w}) \in [0, ..., 1] \tag{23}$$

Donde $w_{u,v} \in [1, ..., 10]$ (enteros) es el peso de la arista (u, v), considerando que un menor peso denota baja afinidad entre nodos, en cambio, una arista con un peso muy cercano a diez expresa una fuerte relación entre dos nodos, w_u^{in} es la suma de los pesos de las aristas incidentes en u, w es la suma de los pesos de todas las aristas de la comunidad C. En base a la ecuación anterior, la función de densidad $\psi(C)$ será dependiente del peso de cada uno de los enlaces de la comunidad C (ecuación 24), esto con el fin de garantizar la bondad de dicha comunidad no sólo tomando en cuenta la cantidad de aristas internas existentes, sino también sus pesos.

$$\psi_w(C) = \psi(C) * W(C) \tag{24}$$

De esta forma, la aplicación de la ecuación 23 será reflejada en FOCS a través de la búsqueda de comunidades locales en una red ponderada no dirigida.

Algoritmo 9 Búsqueda de comunidades locales ponderadas

Input: $G = (V, E, W)$
Output: Una Colección de comunidades Ω_r

```
for (u, v) ∈ E do
  if Com(u) ∩ Com(v) = ∅ then
    Dejar que C = {u, v} ∪ N(u) ∩ N(v);
    if ψ_w(C) ≥ τ(C) and |C| ≥ 4 then
      Definir C como una comunidad local;
      /*Se incluye C dentro de la estructura de comunidades*/
      Ω_r = Ω_r ∪ {C};
    end if
  end if
end for
```

6.7.4. Procesando cambios en los pesos de las aristas

Se tiene que $\Delta G = (\Delta G_1, \Delta G_2, \Delta G_3, ..., \Delta G_k)$ representa la evolución del grafo durante el tiempo Δt, donde ΔG_i representa un cambio importante. Las modificaciones de los pesos de las aristas en el período de tiempo Δt se pueden expresar mediante los siguientes eventos:

- $IncreaseTheEdgeWeight(W + \Delta W)$: Se incrementa el peso de una arista.
- $ReduceTheEdgeWeight(W - \Delta W)$: Se disminuye el peso de una arista.

Los cambios en los pesos de las aristas afectarán directamente el valor de densidad interna de las comunidades, y por ende se producirán cambios en sus estructuras. Si se aumenta o disminuye el peso de una arista (u, v) que pertenece a la comunidad C_i, la afinidad y densidad interna de dicha comunidad se fortalecerá o debilitará, debido al aumento o disminución del peso en el grado de los nodos u y v. Si la arista (u, v) es un nexo de conexión entre las comunidades disjuntas C_i y C_j, entonces su respectiva estructura de comunidad será más o menos evidente al aumentar o reducir el peso de la arista (u, v). Para otros casos, como el aumento o disminución del peso en una arista que afecta la relación entre dos comunidades solapadas, se generarán diversos cambios, en donde será necesario aplicar FOCS en el área local afectada.

6.8. Resultados experimentales

En esta sección, se presentan resultados reales de AFOCS implementando la extensión para grafos ponderados no dirigidos, se comparan los resultados frente a otros algoritmos dinámicos como iLCD [11], OSLOM [29] y LabelRankT [47]. Las pruebas se llevaron a cabo en una computadora con procesador intel core i5, con 4 gigabyte de RAM, en entorno windows.

Métricas: Las métricas a evaluar son:

1. *Información Mutua Normalizada (NMI) [28]:* Enfocada en el solapamiento de comunidades, mide la cantidad de información que se comparte entre un par de comunidades C_1, C_2, sus valores están en el rango [0,1], donde $NMI(C_1, C_2) \cong 1$ si sus estructuras son idénticas y $NMI(C_1, C_2) \cong 0$ si el par de comunidades son totalmente independientes.

2. *Tiempo de ejecución (seg.):* Mide la cantidad de tiempo en segundos que un algoritmo se demora en la detección de comunidades solapadas, como también en el procesamiento de cada imagen del grafo original.

3. *Modularidad (Q) [31]:* Mide la calidad de las comunidades encontradas en términos de cohesión interna, es decir, valora la cantidad de aristas dentro una comunidad por sobre aquellas que se encuentran entre comunidades. Valores Q negativos o muy cercanos a cero, puede insinuar la existencia de subgrafos con muy pocas aristas internas y muchas aristas entre comunidades. Por lo general una modularidad de 0.3 da indicios de una buena estructura de comunidad.

6.8.1. Estadísticas Grafo sintético

Conjunto de datos: Se ha usado un grafo ponderado no dirigido generado por el benchmark LFR [27], estructurado para evaluar la detección de comunidades solapadas. Este tipo de redes contienen solapamiento de comunidades de diferentes tamaños que capturan las características internas de las redes del mundo real.

Configuración: Los parámetros introducidos en el benchmark LFR para generar el grafo sintético son: número de nodos $N = 1000$, grado promedio $k = 15$, grado máximo $k_{max} = 50$, mínimo exponente de la secuencia de grado $t_1 = 2$, mínimo exponente de distribución de tamaño de comunidad $t_2 = 1$, tamaño mínimo comunidad $c_{min} = 20$, tamaño máximo comunidad $c_{max} = 50$, cantidad de comunidades a las que pertenece un nodo $o_m = 2$, parámetro de mezclado μ=0.1. Para la detección de comunidades solapadas en AFOCS+W se ha ingresado como parámetro de entrada el umbral de solapamiento β=0.5.

En la figura 22 se realiza la comparación entre el NMI y μ (parámetro de mezclado) que indica la relación entre la conectividad de un nodo con su comunidad y el resto del grafo, acá se muestra que a medida que aumenta μ el algoritmo con peor rendimiento es iLCD, puesto que sus comunidades detectadas se hacen menos definidas y es mayor su dificultad para detectar comunidades solapadas, mientras que AFOCS es competitivo frente a OSLOM, siendo este último 4.5 % más efectivo, ya que a medida que las comunidades se hacen menos definidas sigue siendo capaz de encontrar correctamente dichas comunidades.

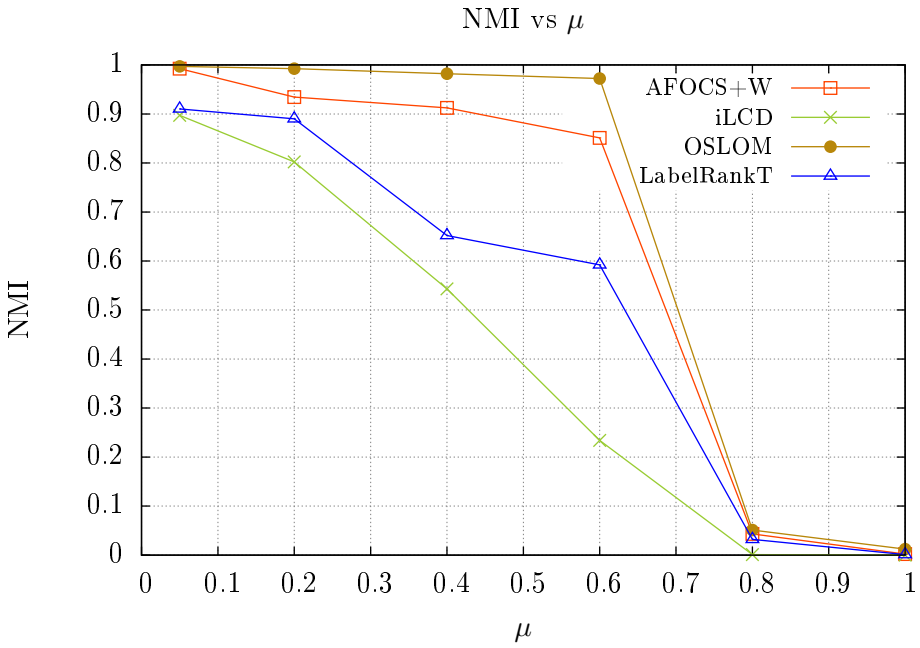


Figura 22: Resultados de NMI vs μ de AFOCS frente a otros algoritmos dinámicos, en un grafo ponderado no dirigido. (ver datos en tabla A3)

Se procesaron seis imágenes del grafo original, las primeras cinco imágenes representan 768 inserciones de aristas cada una, mientras que la última imagen contiene dos eliminaciones de nodos y una eliminación de arista. Como se muestra en la figura 23, LabelRankT comienza con el más alto NMI, pero a medida que evoluciona el grafo mediante la inserción de aristas, cada vez van existiendo menos comunidades afines entre sí, por tanto, la cantidad de aristas entre comunidades disminuye progresivamente, caso similar ocurre en OSLOM e iLCD, mientras que AFOCS presenta resultados alentadores, a medida que se avanza en el tiempo la detección de comunidades solapadas es más evidente aún con la eliminación de nodos y arista, el NMI aumenta paralelamente con el paso de imágenes, siendo 50.2 % más efectivo que LabelRankT.

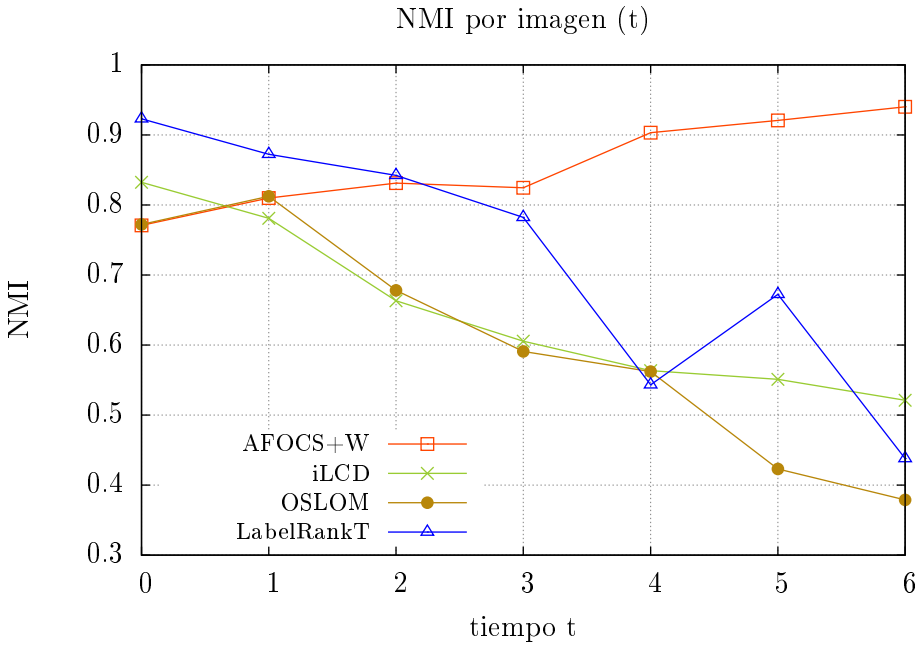


Figura 23: Resultados de NMI según cada imagen del grafo original. Se hace la comparación de la calidad de detección de comunidades solapadas en una red dinámica con pesos según AFOCS, OSLOM, iLCD y LabelRankT. (ver datos en tabla A4)

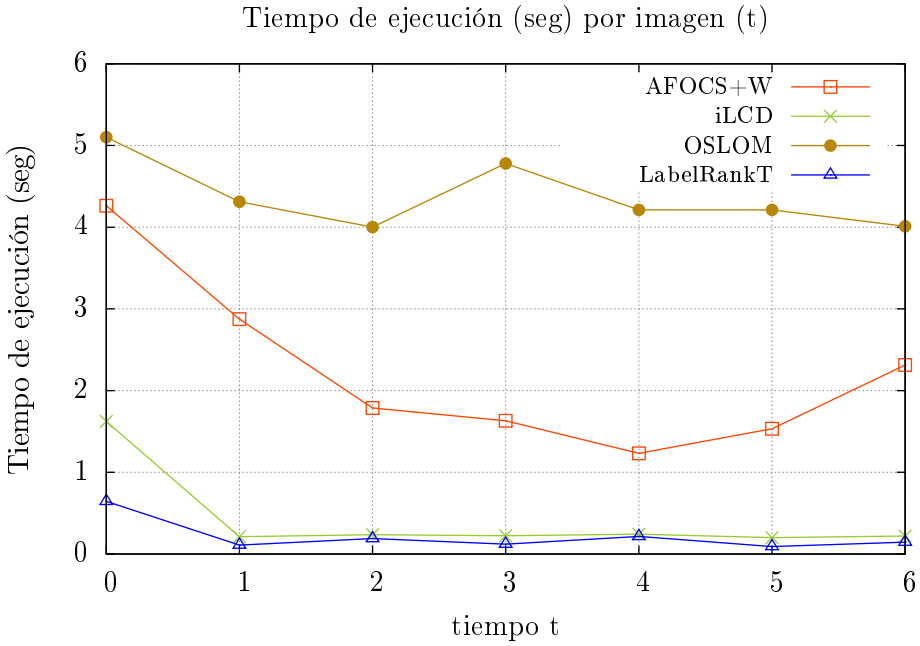


Figura 24: Tiempo de ejecución de AFOCS y otros algoritmos dinámicos según la evolución del grafo original. (ver datos en tabla A5)

El tiempo de procesamiento ilustrado en la figura 24 presenta diferencias de acuerdo a la complejidad y calidad de detección de cada algoritmo, puesto que iLCD a medida que procesa cada imagen detecta menos solapamiento entre comunidades, por ende, su tiempo de ejecución es linealmente mínimo, caso similar ocurre con LabelRankT, con la salvedad que este último es capaz de detectar correctamente más comunidades solapadas que iLCD. Los altos niveles de NMI de AFOCS penalizan en cierta medida sus tiempos de procesamiento, siendo 10.3x más lento que LabelRankT. Entre más alto sea el NMI y menor sea el tiempo de procesamiento, mejor será el algoritmo, AFOCS presenta una alta calidad de detección de solapamiento en una red dinámica, su tiempo de ejecución a pesar de no sera tan bajo, tiende a disminuir a medida que se insertan aristas, por lo que en términos generales presenta resultados prometedores.

6.8.2. Estadísticas Grafo real

Conjunto de datos: Se ha usado un grafo ponderado no dirigido que modela un extracto de la red social Facebook, representando un circulo social anónimo entre diversas personas según intereses comunes.

Configuración: El grafo esta compuesto por 63732 nodos y 408553 aristas. La evolución de la red social esta representada por 25 imágenes dado un tiempo t, cada imagen representa la creación de entre 16000 a 17000 nuevas relaciones dentro de grupos sociales, produciéndose la inserción o migración de entre 500 a 7000 personas en nuevas comunidades sociales según intereses comunes, la cantidad de nodos y aristas insertados/as en cada imagen se grafican en la figura 27 y figura 26, respectivamente. Para la detección de comunidades solapadas en AFOCS+W se ha ingresado como parámetro de entrada el umbral de solapamiento β=0.7.

En la figura 25 se realiza la comparación entre el NMI y μ, se muestra que a medida que aumenta μ las comunidades se hacen menos definidas, por lo que se repite el escenario para iLCD como el algoritmo con peor rendimiento, puesto que en función del aumento del parámetro de mezclado presenta mayores dificultades para detectar correctamente comunidades solapadas, mientras que AFOCS sigue siendo competitivo transformándose en el segundo algoritmo con mejor rendimiento de NMI, solo con un 4.9 % menos efectivo que OSLOM.

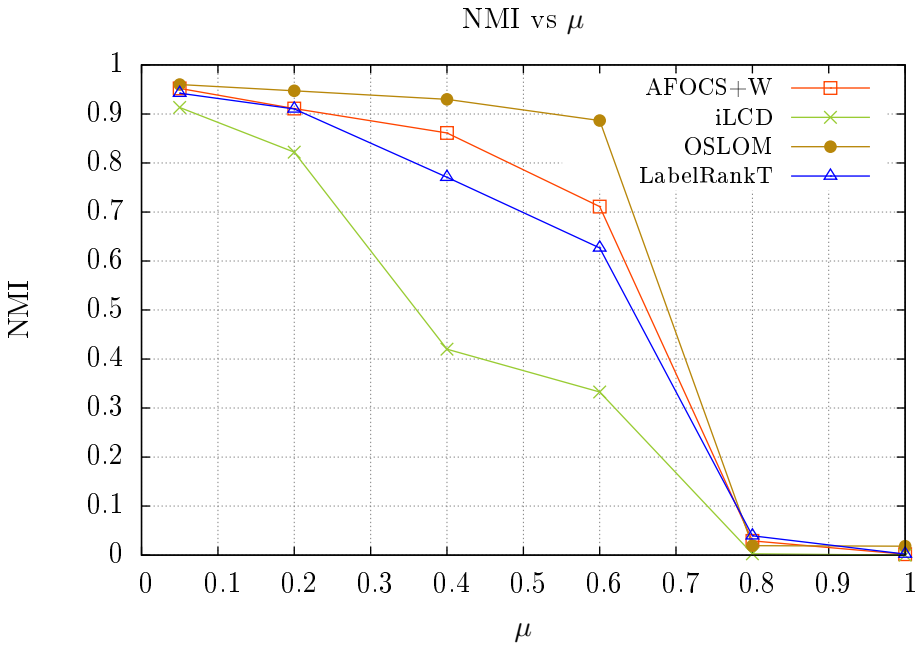


Figura 25: Resultados de NMI vs μ de AFOCS frente a otros algoritmos dinámicos, en un grafo que modela un extracto de la red social Facebook. (ver datos en tabla A6)

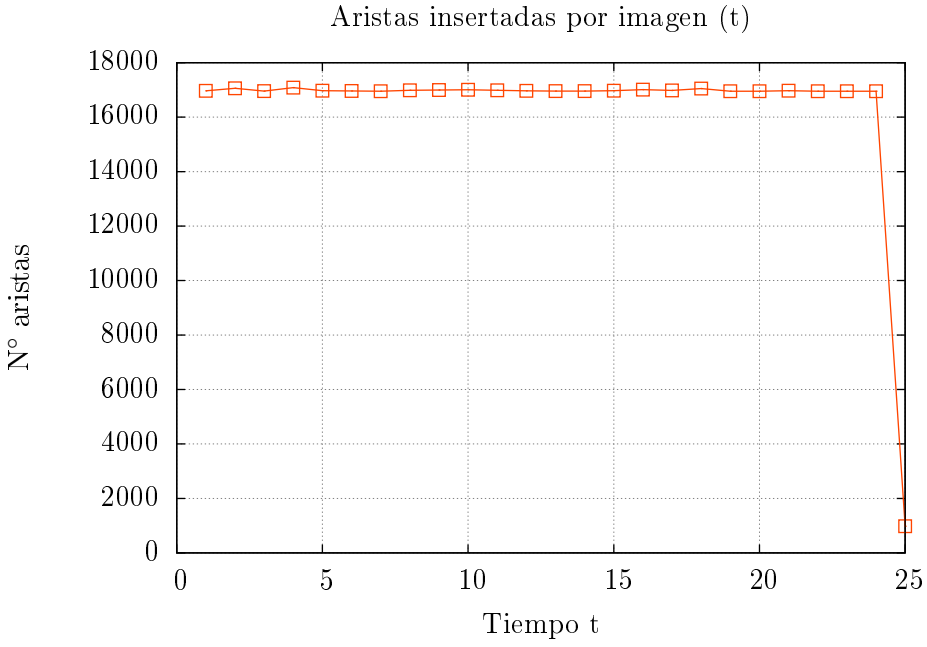


Figura 26: La estructura de cada imagen conforme a la inserción de nuevas aristas en la red Facebook dado un tiempo t. (ver datos en tabla A7)

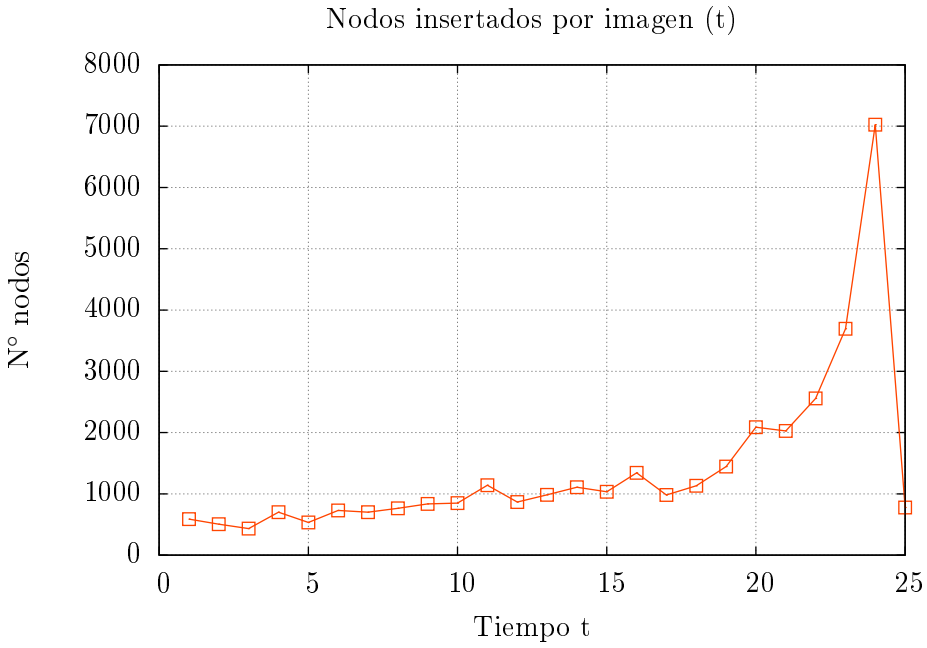


Figura 27: La estructura de cada imagen conforme a la inserción de nuevos nodos en la red Facebook dado un tiempo t. (ver datos en tabla A7)

La red dinámica presenta 25 cambios a través del tiempo, compuesta cada una por inserción de nodos y aristas como se ilustra en la figura 27 y figura 26, respectivamente. Al observar la figura 28, nuevamente LabelRankT comienza con el más alto NMI, manteniendo su alza hasta $t = 7$, en $t = 8$ comienza a detectar comunidades poco afines entre sí, disminuyendo progresivamente el nivel de solapamiento, caso similar ocurre en OSLOM e iLCD, en cambio, AFOCS si bien presenta el peor comienzo en NMI da luces de resultados alentadores, ya que a medida que se avanza en el tiempo, la detección de comunidades solapadas es más evidente, debido a la gran cantidad de información que comparten miembros con miembros de otras comunidades, el NMI aumenta paralelamente con el paso de imágenes, sólo hasta $t = 20$ donde se tiende a estacionalizar el comportamiento de la red, traduciéndose en una efectividad de 6.9 % por sobre su perseguidor LabelRankT.

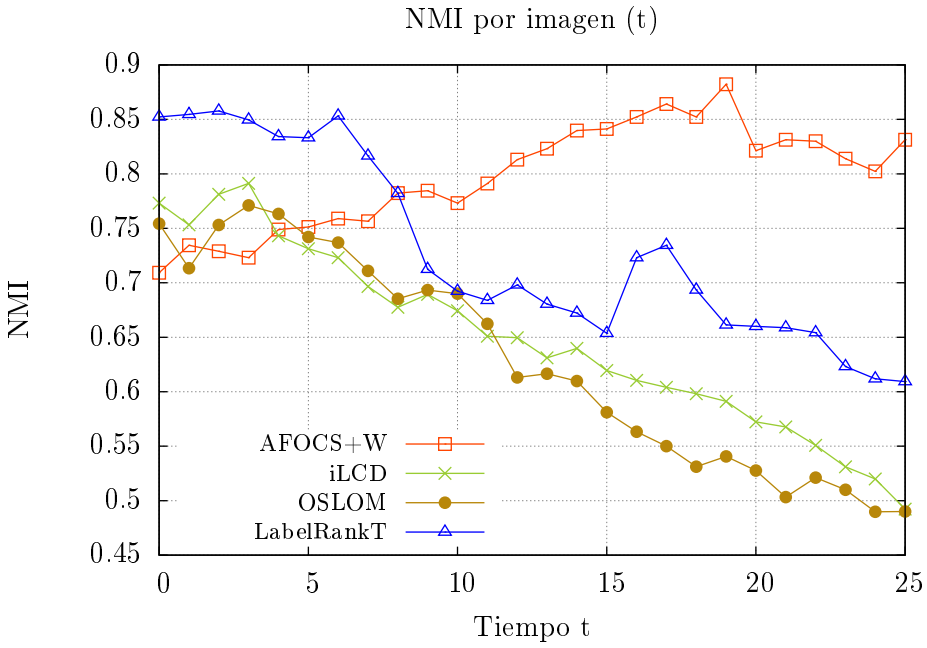


Figura 28: Resultados de NMI según cada imagen del grafo original. Se hace la comparación de la calidad de detección de comunidades solapadas en un extracto de Facebook contrastando los resultados de AFOCS, OSLOM, iLCD y LabelRankT. (ver datos en tabla A8)

A medida que las comunidades son más definidas en cuanto a los niveles de información que se comparten entre ellas, el tiempo de procesamiento para su detección se verá afectado. Como se ilustra en la figura 29, OSLOM mantiene sus altos tiempos de respuesta para procesar cada imagen de la red original, siendo 16x más lento que el resto de los algoritmos. La alta puntuación de NMI que registra AFOCS se respalda con sus competitivos tiempos de procesamiento, si bien, al igual que iLCD y OSLOM en un comienzo la detección inicial de comunidades solapadas demanda el doble de tiempo que el resto de la serie de tiempo, a medida que se introducen cambios en la red, AFOCS es capaz de generar cambios localizados en el grafo, minimizando al máximo los tiempos de respuesta, produciéndose una diferencia de tiempo de 0.61 segundos que es casi despreciable para el tratamiento de redes grandes.

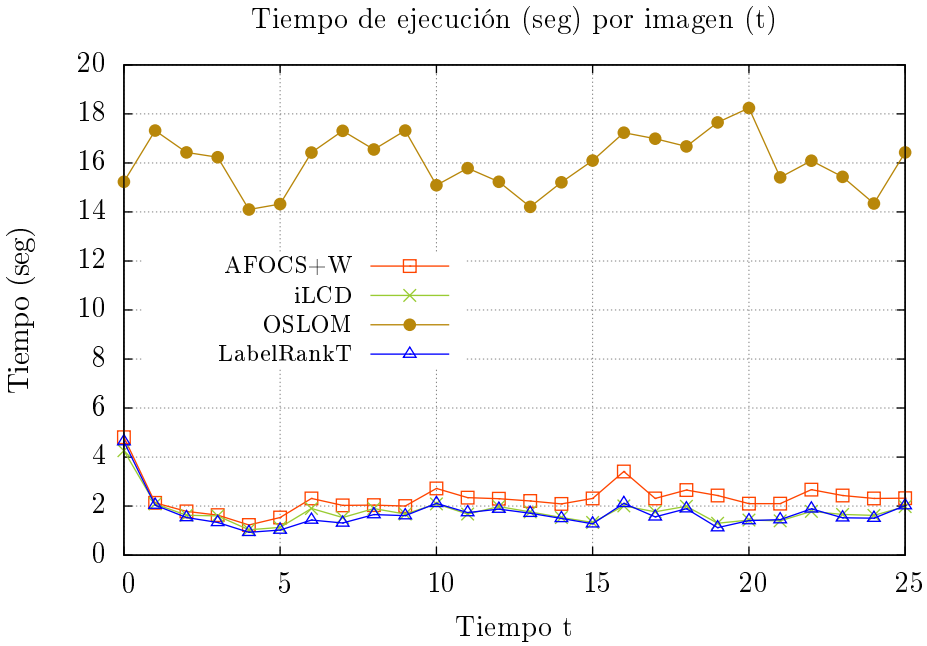


Figura 29: Tiempo de ejecución de AFOCS y otros algoritmos dinámicos según la evolución de la red Facebook en cada tiempo t. (ver datos en tabla A9)

Teóricamente una fuerte cohesión interna es importante para cada comunidad, ya que da indicios de una amplia comunicación entre la mayoría de sus miembros, pero una débil cohesión interna da señales de masivas conexiones entre grupos, apuntando a un alto nivel solapamiento entre comunidades. Como se muestra en la figura 30, la peor modularidad la presenta el algoritmo iLCD, revelando la existencia de pocas aristas dentro de cada comunidad detectada, lo que denota la conformación de muchas comunidades pequeñas que se solapan con otras, pero aún así, las comunidades se encuentran poco definidas en cuanto a su disminuida estructura interna. Valores de modularidad sobre 0.3 da señales de una buena estructura de comunidad, LabelRankT y OSLOM presentan la mejor modularidad que se mantiene conforme evoluciona la red dinámica, esto se explica contrastando con los resultados de la figura 28, donde el NMI que da señales del solapamiento de comunidades decrece a medida que se introducen cambios en el grafo, revelando la existencia de pocas aristas entre comunidades en desmedro de una fuerte cohesión interna. En el caso de AFOCS, muestra los valores de modularidad más bajos después de iLCD, presentando un 38.4 % de peor cohesión interna que LabelRankT, sus valores se encuentran en ciertos márgenes de aceptación a pesar del alto solapamiento que es capaz de detectar.

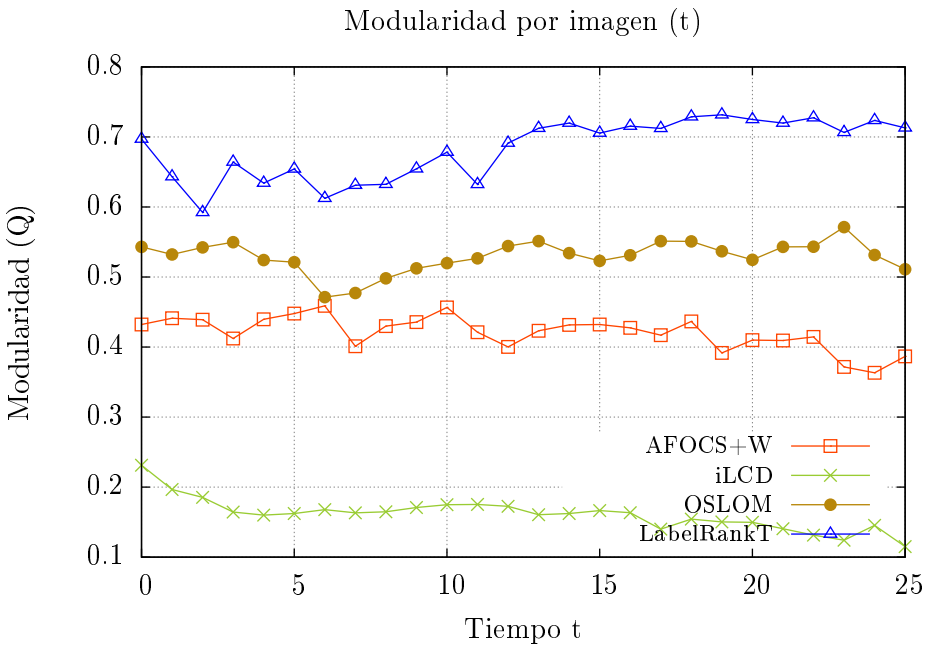


Figura 30: Resultados de Modularidad (Q) según la evolución de la red Facebook. Se hace la comparación de la calidad de las comunidades encontradas dentro de la red Facebook contrastando los resultados de AFOCS+W, OSLOM, iLCD y LabelRankT. (ver datos en tabla A10)

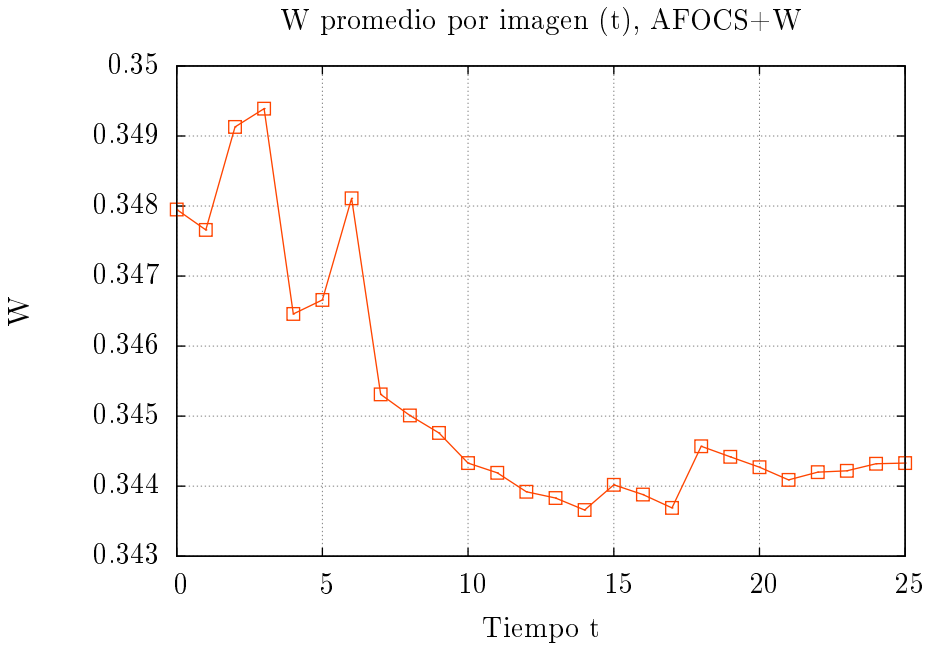


Figura 31: Resultados de W al evaluar las comunidades en cada imagen de la red Facebook. (ver datos en tabla A11)

En la figura 31 se muestran los valores promedios de la función de pesos (W) al evaluar todas las comunidades detectadas luego del procesamiento de cada imagen realizado por el algoritmo AFOCS+W, se vislumbra un comportamiento descendente de W a medida que se generan cambios en el grafo original, quedando en manifiesto que en primera instancia muchas comunidades candidatas pueden ser afectadas indirectamente para que sean consideradas o no como comunidades locales o finales. Sin embargo, en la figura 33 se observa que la densidad promedio de las comunidades detectadas luego de procesar cada imagen tiene tendencia creciente, lo que da cuenta del aumento en el número de comunidades detectadas como se muestra en la figura 32, donde a medida que se procesan más cambios en el grafo el número de comunidades y la densidad de cada comunidad aumenta notoriamente, sobre todo en $t = 24$ donde el número de comunidades aumenta a casi el doble que las detectadas en $t = 23$, esto se produce por la alta cantidad de nodos insertados en $t = 24$. A pesar de que los valores promedios de la función de pesos no son tan altos, aveces no impide en este caso que se formen más comunidades producto de los altos valores que entrega la función de densidad.

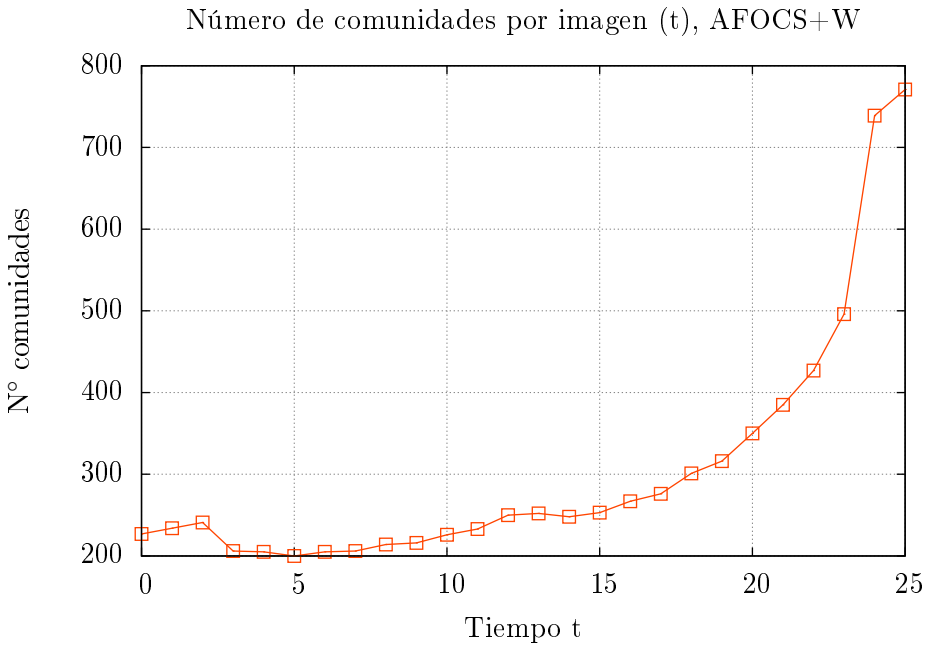


Figura 32: Estructura de comunidades detectadas por AFOCS+W luego de procesar cada imagen de la red Facebook. (ver datos en tabla A12)

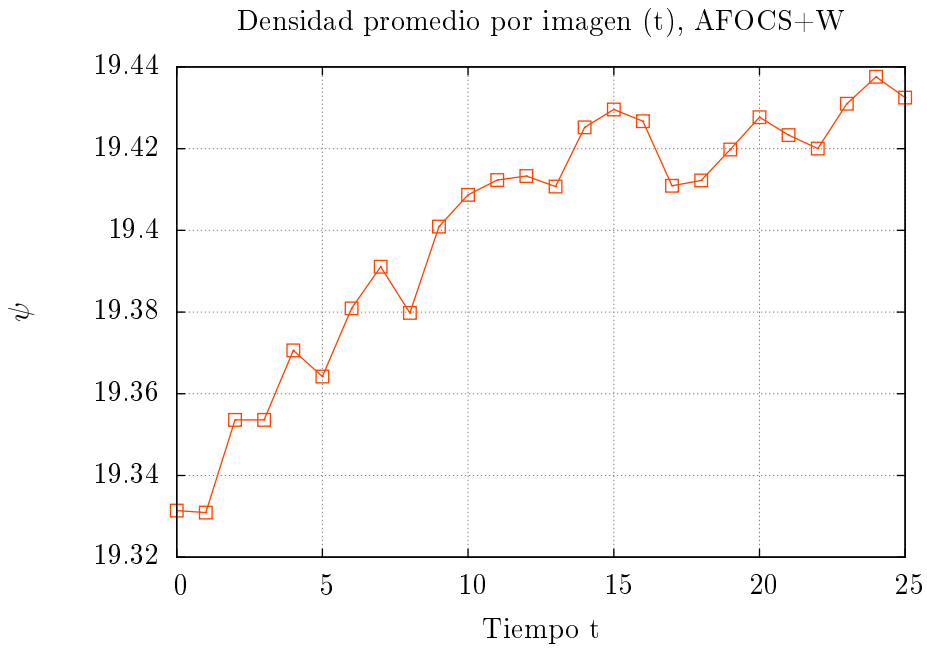


Figura 33: Función de densidad promedio de las comunidades detectadas por AFOCS+W luego de procesar cada imagen de la red Facebook. (ver datos en tabla A13)

7. Herramientas para análisis de grafos

A continuación se presentan diversas herramientas que permiten efectuar un análisis y evaluación de grafos.

IGraph

IGraph es una librería de código abierto, distribuida bajo licencia GPL, para el estudio y análisis de redes/grafos. Los principales objetivos de esta librería son proveer un conjunto de tipos de datos y funciones para una fácil implementación de algoritmos de grafos, un manejo rápido de grandes grafos con millones de nodos y aristas, y permitir un rápido prototipado por medio de un lenguaje de alto nivel como R. IGraph permite manipular tanto grafos dirigidos como no dirigidos, no admite la implementación de hipergrafos. Por otro lado, cuenta con implementaciones de problemas típicos de teoría de grafos como árboles de expansión mínima y flujo de red, también implementa algoritmos para algunos métodos de análisis estructural dentro de un grafo. IGraph puede ser instalado como un librería del lenguaje C, como un paquete de R, como un módulo de extensión de Python o como una extensión de Ruby. IGraph permite la detección de comunidades, implementando los siguientes algoritmos: método divisivo de intermediación propuesto por Newman y Girvan [31], método voraz (greedy) propuesto por Clauset et al. [15] explicado en la sección 5.3.1, algoritmo basado en los vectores propios de la matriz de modularidad (*Leading Eigenvector*) propuesto por Newman [33], algoritmo *Spin Glass* propuesto por Reichardt [40], algoritmo *Walk Trap* propuesto por Pascal Pons [38] y el algoritmo Infomap propuesto por Rossvall y Bergstrom [42] que es explicado en la sección 5.3.4. IGraph se puede descargar desde el sitio web `http://igraph.org`.

Gephi

Gephi es una herramienta para la exploración, navegación y análisis de grafos. Permite interactuar con las distintas representaciones, manipular las estructuras, formas y colores que revelan propiedades ocultas. Utiliza un motor de renderizado 3D para mostrar grandes grafos en tiempo real y para acelerar la exploración. Gephi se destaca por ser una herramienta libre de código abierto y que corre tanto en Windows, Linux como Mac. Está desarrollado en JAVA y se distribuye bajo licencia GNU GPL 3. Resulta ideal para desplegar gráficos representados mediante grafos, complejos gráficos de visualización de datos utilizados en análisis de redes sociales. Soporta la representación de grafos dirigidos, no dirigidos y mixtos, y por el momento no permite implementar hipergrafos. Uno de los aspectos más importantes cubiertos por Gephi es la interacción en tiempo real, permite modificar propiedades de los nodos y aristas al mismo tiempo que se modifica la representación o layout del grafo. Así mismo, permite realizar agrupaciones, filtrado, manipulación, navegación y proveer un fácil acceso a los datos. Gephi dispone del código fuente para su utilización y de una API denominada Gephi Toolkit para desarrollar aplicaciones propias basadas en dicha herramienta. Gephi permite la detección de comunidades en grafos, implementando el Algoritmo de Blondel et al. explicado en la sección 5.3.2. Gephi se puede descargar desde el sitio web `https://gephi.org`.

Cytoscape

Cytoscape es una plataforma de software gratis y de código abierto de bioinformática integral, diseñada para la visualización de rutas biológicas y redes de interacción molecular. Aunque Cytoscape fue diseñado originalmente para la investigación biológica, ahora es una

plataforma general para el complejo análisis y visualización de grafos.

Cytoscape permite la adición de plugins para el análisis de redes complejas, en materia de detección de comunidades, se cuenta con el plugin GLay, el cual ofrece una variada colección de algoritmos de análisis y detección de comunidades y funciones de diseño, dinámicamente con enlaces desde iGraph. Los algoritmos que se implementan en GLay son el método voraz (greedy) propuesto por Clauset et al. [15] explicado en la sección 5.3.1, método voraz (greedy) propuesto por Wakita [44], algoritmo *Walk Trap* propuesto por Pascal Pons [38], algoritmo *Label Propagation* propuesto por Raghavan et al. [39], método divisivo de intermediación propuesto por Newman y Girvan [31], algoritmo basado en los vectores propios de la matriz de modularidad (*Leading Eigenvector*) propuesto por Newman [33], algoritmo *Spin Glass* propuesto por Reichardt [40] y el método voraz (greedy) propuesto por Clauset et al. [15] en la versión de iGraph (que es más rápido que la versión de GLay). Cytoscape se puede descargar desde el sitio web `http://www.cytoscape.org`, y los plugins se pueden descargar desde la App Store de Cytoscape accediendo a `http://apps.cytoscape.org`.

Radatools

Radatools es una conjunto de programas de libre distribución para el análisis de redes complejas. Se incluyen algoritmos para la detección de comunidades, ellos son el algoritmo basado en los vectores propios de la matriz de modularidad (*Leading Eigenvector*) propuesto por Newman [33] y el algoritmo de Duch [17] basado en la optimización extrema de la modularidad que es explicado en la sección 5.3.3. Radatools se puede descargar desde el sitio web `http://deim.urv.cat/~sgomez/radatools.php`.

Cfinder

Cfinder es un software gratuito para la búsqueda y visualización de comunidades solapadas, implementa el algoritmo Percolación Clique (CPM) propuesto por Palla et al. [36]. Cfinder mediante el algoritmo CPM ofrece rapidez y eficiencia para agrupar datos representados por grandes grafos, como redes sociales y redes genéticas. Cfinder se puede descargar desde el sitio web `http://cfinder.org`.

8. Conclusión

El presente trabajo ha servido para indagar en detalle sobre la minería de grafos enfocada desde las redes sociales, las cuales son modeladas desde una perspectiva novedosa. La inmersión en el estado del arte y el estudio de los elementos matemáticos de la teoría de grafos ha ayudado para tener el conocimiento base acerca de las propiedades que poseen este tipo de estructuras, y como estas propiedades son importantes para concebir el concepto de comunidad dentro de una misma red.

La aplicación de distintos tipos de algoritmos de acuerdo a un modelo matemático determinado, ha dado a conocer las definiciones intrínsecas del concepto de comunidad. Comprender la lógica e implementación de los métodos tradicionales de detección de comunidades disjuntas y solapadas, ha permitido ampliar el conocimiento para estudiar algoritmos más avanzados. La utilización de métricas que ayudan a medir la calidad de detección de comunidades de los algoritmos, se transforman en herramientas indispensables para apoyar el mejoramiento de métodos existentes o la creación de algoritmos nuevos.

Es así, como el propósito final del presente trabajo ha sido estudiar la estructura de comunidades solapadas en redes dinámicas, aquellos grafos que cambian con el tiempo y generan progresivamente nuevas estructuras de comunidad. En primera instancia, se estudió e implementó AFOCS, algoritmo dinámico capaz de detectar comunidades solapadas en redes estáticas y dinámicas mediante dos fases, la primera comienza con FOCS en que se detectan las primeras comunidades solapadas del grafo original, luego aplicando una medida de puntuación se realiza un refinamiento de detección de comunidades solapadas, donde aquellas que se encuentran muy unidas se fusionan en una sola comunidad, teniendo como consecuencia aumentar la densidad interna total de la nueva estructura de comunidad, un punto importante, es que FOCS tiene siempre en consideración la existencia de nodos atípicos, con el objetivo de analizar su integración una vez completada la etapa de refinamiento, para conseguir que la densidad interna de cada comunidad aumente y su estructura sea aún más definida. En la segunda fase, se ejecuta AFOCS, donde analiza la conformación de comunidades solapadas en función de los cambios introducidos en el grafo, es decir, se basa en la historia evolutiva de la red para actualizar de manera adaptativa la estructura de comunidad, esto se efectúa tomando en cuenta el resultado de los cambios introducidos en la última imagen del grafo al realizar una inserción y/o eliminación de un nodo y/o arista, atacando solamente la(s) zona(s) afectada(s) por el cambio.

Luego de estudiar el algoritmo dinámico AFOCS sobre grafos no dirigidos y sin pesos, se propuso el desafío de integrar pesos en las aristas (conservando su simetría), a modo que la detección y conformación de comunidades también dependa del peso en la relación entre cada par de nodos y no sólo de la cantidad de aristas internas que posee cada comunidad. Al realizar las pruebas implementando la extensión de AFOCS para grafos ponderados no dirigidos, se llegaron a resultados competitivos, puesto que el algoritmo sigue siendo capaz de encontrar una buena definición de comunidades solapadas junto con mostrar un aceptable nivel de cohesión interna de cada comunidad, con lo cual cumple con el principio de la métrica más popular utilizada para medir la calidad de las particiones, la modularidad.

Anexo A Tablas de resultados experimentales

μ	**AFOCS+W**	**iLCD**	**OSLOM**	**LabelRankT**
0.05	0.9923	0.8972	0.9971	0.9102
0.2	0.9343	0.8021	0.9923	0.8901
0.4	0.9123	0.5432	0.9821	0.6521
0.6	0.8512	0.2341	0.9723	0.5922
0.8	0.0432	0.0012	0.0512	0.0322
1.0	0.0022	0.0001	0.0123	0.0012

Tabla A3: Valores de NMI según el parámetro de mezclado μ en un grafo sintético ponderado no dirigido.

t	**AFOCS+W**	**iLCD**	**OSLOM**	**LabelRankT**
0	0.7708	0.8323	0.7721	0.9231
1	0.8099	0.781	0.8123	0.8723
2	0.8312	0.6632	0.678	0.8423
3	0.8245	0.6055	0.5909	0.7822
4	0.9032	0.5634	0.5621	0.5433
5	0.9207	0.5511	0.4231	0.6723
6	0.9402	0.5212	0.3789	0.4379

Tabla A4: Valores de NMI en cada tiempo t en función de los cambios introducidos en un grafo sintético ponderado no dirigido.

t	**AFOCS+W**	**iLCD**	**OSLOM**	**LabelRankT**
0	4.2632	1.6233	5.1021	0.6452
1	2.8753	0.2123	4.3123	0.109
2	1.7872	0.2367	4.0021	0.1898
3	1.6313	0.2231	4.7812	0.1212
4	1.2320	0.2432	4.2122	0.2147
5	1.5334	0.2001	4.2130	0.0921
6	2.3134	0.2199	4.0122	0.1455

Tabla A5: Tiempo (seg.) de ejecución en el procesamiento de los cambios del grafo sintético ponderado no dirigido en cada imagen de tiempo t.

μ	**AFOCS+W**	**iLCD**	**OSLOM**	**LabelRankT**
0.05	0.9521	0.9131	0.96	0.9422
0.2	0.9109	0.8221	0.9472	0.9102
0.4	0.8611	0.4201	0.9299	0.7709
0.6	0.711	0.3328	0.8865	0.6266
0.8	0.0291	0.0027	0.01922	0.0395
1	0.0019	0.0001	0.01832	0.0021

Tabla A6: Valores de NMI según el parámetro de mezclado μ en un grafo real (facebook) ponderado no dirigido.

t	N° nodos	N° aristas
1	587	16968
2	506	17062
3	434	16957
4	700	17086
5	533	16973
6	730	16964
7	700	16953
8	765	16985
9	835	16997
10	848	17005
11	1141	16986
12	866	16965
13	985	16958
14	1107	16956
15	1034	16969
16	1344	17009
17	982	16979
18	1132	17052
19	1445	16952
20	2088	16952
21	2026	16972
22	2558	16953
23	3695	16953
24	7024	16951
25	777	980

Tabla A7: Cantidad de nodos y aristas insertadas en cada imagen de tiempo t del grafo real (facebook) ponderado no dirigido.

t	**AFOCS+W**	**iLCD**	**OSLOM**	**LabelRankT**
0	0.7092	0.77312	0.7543	0.8523
1	0.7345	0.7532	0.7134	0.8546
2	0.729	0.7812	0.7531	0.8578
3	0.7231	0.79132	0.7711	0.8496
4	0.7489	0.7431	0.7632	0.8342
5	0.7512	0.7313	0.7421	0.8331
6	0.7589	0.7231	0.7369	0.8532
7	0.7565	0.6966	0.7109	0.8165
8	0.7823	0.6775	0.6854	0.7821
9	0.7845	0.6894	0.6933	0.71239
10	0.7732	0.6743	0.69	0.6923
11	0.7912	0.6509	0.6623	0.6839
12	0.8131	0.6496	0.61312	0.6981
13	0.8231	0.6311	0.6165	0.6805
14	0.8398	0.6398	0.6098	0.6722
15	0.8412	0.6195	0.5812	0.6536
16	0.8523	0.6104	0.5633	0.7231
17	0.8641	0.6041	0.5501	0.7345
18	0.8523	0.59821	0.5312	0.6934
19	0.8823	0.5912	0.5406	0.6614
20	0.8213	0.5723	0.5276	0.66
21	0.8313	0.5677	0.5033	0.6588
22	0.8299	0.5508	0.5211	0.6543
23	0.8139	0.5311	0.50996	0.6231
24	0.8023	0.5199	0.4898	0.6119
25	0.83131	0.4923	0.49	0.6093

Tabla A8: Valores de NMI en cada tiempo t en función de los cambios introducidos en un grafo real (facebook) ponderado no dirigido.

t	**AFOCS+W**	**iLCD**	**OSLOM**	**LabelRankT**
0	4.8122	4.2632	15.2311	4.6322
1	2.1316	2.0912	17.3213	2.0423
2	1.7872	1.6276	16.4324	1.5344
3	1.6313	1.6020	16.2333	1.3356
4	1.2320	1.0330	14.0982	0.9321
5	1.5334	1.1314	14.3212	1.0232
6	2.3134	1.9022	16.4244	1.4322
7	2.03210	1.5323	17.3122	1.3123
8	2.04120	1.8998	16.5511	1.6543
9	2.00120	1.6744	17.3213	1.6112
10	2.7213	2.0901	15.0933	2.1112
11	2.3431	1.6762	15.7889	1.7345
12	2.301	1.9822	15.2323	1.8775
13	2.2111	1.7622	14.2098	1.7099
14	2.09123	1.5343	15.2092	1.499
15	2.3132	1.3341	16.0993	1.2767
16	3.41123	2.0091	17.2333	2.1092
17	2.3123	1.7662	16.9882	1.5632
18	2.6543	1.9899	16.6766	1.8922
19	2.4323	1.3001	17.6533	1.12
20	2.099	1.4332	18.2432	1.4092
21	2.0999	1.3989	15.4121	1.4453
22	2.6712	1.7863	16.0901	1.8909
23	2.4322	1.6612	15.4331	1.5234
24	2.3111	1.6231	14.3431	1.5092
25	2.3213	1.9832	16.4322	2.0167

Tabla A9: Tiempo (seg.) de ejecución en el procesamiento de los cambios del grafo real (facebook) ponderado no dirigido en cada imagen de tiempo t.

t	**AFOCS+W**	**iLCD**	**OSLOM**	**LabelRankT**
0	0.4322	0.2312	0.5431	0.697
1	0.4412	0.1966	0.5321	0.6432
2	0.439	0.1853	0.5422	0.5921
3	0.4121	0.1643	0.5498	0.6642
4	0.4396	0.16009	0.5241	0.6341
5	0.4478	0.16231	0.5212	0.6543
6	0.4589	0.1678	0.4712	0.6123
7	0.4011	0.1633	0.4772	0.6311
8	0.4299	0.1648	0.49812	0.6323
9	0.4356	0.1709	0.5123	0.6544
10	0.4566	0.1748	0.5199	0.6782
11	0.4211	0.1753	0.5266	0.6322
12	0.4001	0.17266	0.5441	0.6912
13	0.4233	0.1605	0.5512	0.7123
14	0.4317	0.1623	0.5341	0.7199
15	0.4321	0.1665	0.5231	0.7056
16	0.4274	0.1634	0.5311	0.7154
17	0.4169	0.14	0.5512	0.7122
18	0.4365	0.1543	0.5508	0.7288
19	0.3916	0.1502	0.5367	0.7316
20	0.41	0.1497	0.5245	0.725
21	0.4094	0.1405	0.5431	0.7198
22	0.4145	0.1314	0.5432	0.7274
23	0.3717	0.1241	0.57123	0.7065
24	0.3632	0.1454	0.5314	0.7237
25	0.3867	0.11508	0.511	0.7129

Tabla A10: Valores de Modularidad (Q) en función de los cambios introducidos en el grafo real (facebook) ponderado no dirigido durante cada tiempo t.

t	W
0	0.34795
1	0.34766
2	0.34913
3	0.34939
4	0.34646
5	0.34666
6	0.34811
7	0.34531
8	0.34501
9	0.34476
10	0.34433
11	0.34419
12	0.34392
13	0.34383
14	0.34366
15	0.34402
16	0.34388
17	0.34369
18	0.34457
19	0.34442
20	0.34427
21	0.34409
22	0.34420
23	0.34422
24	0.34432
25	0.34433

Tabla A11: Valores promedios de la función de pesos (W) de las comunidades detectadas durante el procesamiento de cada imagen.

t	N° Comunidades
0	227
1	234
2	241
3	206
4	205
5	200
6	205
7	206
8	214
9	216
10	226
11	233
12	250
13	252
14	248
15	253
16	267
17	276
18	301
19	316
20	350
21	385
22	427
23	496
24	739
25	771

Tabla A12: Cantidad de comunidades detectadas por AFOCS+W luego de procesar cada imagen del grafo original de la red Facebook.

t	ψ
0	19.3314
1	19.3309
2	19.3536
3	19.3536
4	19.3706
5	19.3642
6	19.3809
7	19.3911
8	19.3798
9	19.4009
10	19.4087
11	19.4123
12	19.4133
13	19.4107
14	19.4252
15	19.4296
16	19.4267
17	19.4109
18	19.4122
19	19.4198
20	19.4277
21	19.4233
22	19.42
23	19.431
24	19.4376
25	19.4325

Tabla A13: Densidad promedio de las comunidades luego de procesar cada imagen de grafo original de la red Facebook.

Referencias

[1] Lars Backstrom, Daniel P. Huttenlocher, Jon M. Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Knowledge Discovery and Data Mining*, pages 44–54, 2006.

[2] Jeffrey Baumes, Mark K. Goldberg, Mukkai S. Krishnamoorthy, Malik Magdon-ismail, and Nathan Preston. Finding communities by clustering a graph into overlapping subgraphs. In *International Association for Development of the Information Society*, pages 97–104, 2005.

[3] Jeffrey Baumes, Mark K. Goldberg, and Malik Magdon-ismail. *Efficient Identification of Overlapping Communities*. 2005.

[4] Tanya Y. Berger-wolf and Jared Saia. A framework for analysis of dynamic social networks. In *Knowledge Discovery and Data Mining*, pages 523–528, 2006.

[5] Vincenet D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics-theory and Experiment*, 10, 2008.

[6] Stefan Boettcher and Allon G. Percus. Extremal optimization for graph partitioning. *Physical Review E*, 64, 2001.

[7] Stefan Boettcher and Allon G. Percus. Optimization with Extremal Dynamics. *Physical Review Letters*, 86:5211–5214, 2001.

[8] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity – np-completeness and beyond, 2006.

[9] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20:172–188, 2008.

[10] Ulrik Brandes and Thomas Erlebach. Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]. In *Dagstuhl Seminars*, 2005.

[11] Remy Cazabet, Frédéric Amblard, and Chihab Hanachi. Detection of Overlapping Communities in Dynamical Social Networks. In *IEEE International Conference on Social Computing*, pages 309–314, 2010.

[12] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38, 2006.

[13] Jiyang Chen. Community Mining: Discovering Communities in Social Networks. University of Alberta, 2010.

[14] Aaron Clauset. Finding local community structure in networks. *Physical Review E*, 72, 2005.

[15] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70, 2004.

[16] Bhaskar DasGupta and Devendra Desai. On the Complexity of Newman's Community Finding Approach for Biological and Social Networks. *Computing Research Repository*, abs/1102.0, 2011.

[17] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72, 2005.

[18] Illes J. Farkas, Dániel Ábel, Gergely Palla, and Tamás Vicsek. Weighted network modules. *New Journal of Physics*, 9:180–180, 2007.

[19] Santo Fortunato. Community detection in graphs. *Physics Reports-review Section of Physics Letters*, 486:75–174, 2010.

[20] Santo Fortunato and Claudio Castellano. Community Structure in Graphs. 2007.

[21] Mark K. Goldberg, Stephen Kelley, Malik Magdon-Ismail, Konstantin Mertsalov, and Al Wallace. Finding Overlapping Communities in Social Networks. In *IEEE International Conference on Social Computing*, pages 104–113, 2010.

[22] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12, 2010.

[23] S. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962.

[24] Vaclav Havel. Note the existence of finite graphs. *Applications of Mathematics*, 080(4):477–480, 1955.

[25] D. A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of The Institute of Radio Engineers*, 40:1098–1101, 1952.

[26] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of The ACM*, 46:604–632, 1999.

[27] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80, 2009.

[28] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11, 2009.

[29] Andrea Lancichinetti, Filippo Radicchi, Jose' Javier Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *Computing Research Repository*, abs/1012.2, 2010.

[30] Conrad Lee, Fergal Reid, Aaron McDaid, and Neil Hurley. Detecting highly overlapping community structure by greedy clique expansion. 2010.

[31] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 2004.

[32] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69, 2004.

[33] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74, 2006.

[34] Nam P. Nguyen, Thang N. Dinh, Sindhura Tokala, and My T. Thai. Overlapping communities in dynamic networks: Their detection and mobile applications. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 85–96, New York, NY, USA, 2011. ACM.

[35] Gergely Palla, Albert-Laszlo Barabasi, and Tamas Vicsek. Quantifying social group evolution. *Nature*, 446:664–667, 2007.

[36] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[37] Gergely Palla, Illés J. Farkas, Péter Pollner, Imre Derényi, and Tamás Vicsek. Directed network modules. *New Journal of Physics*, 9:186–186, 2007.

[38] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks (long version). 2005.

[39] Usha Nandini Raghavan, Röka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76, 2007.

[40] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74, 2006.

[41] Sebastián A. Ríos, Felipe Aguilera, and Luis A. Guerrero. *Virtual Communities of Practice's Purpose Evolution Analysis Using a Concept-Based Mining Approach.* 2009.

[42] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[43] J. Scott. Social Network Analysis: A Handbook 2nd Ed. 2000.

[44] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks: [extended abstract]. In *World Wide Web Conference Series*, pages 1275–1276, 2007.

[45] W. Wallis. Graph theory with applications (j. a. bondy and u. s. r. murty). *SIAM Review*, 21(3):429–429, 1979.

[46] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications.* 1994.

[47] Jierui Xie, Mingming Chen, and Boleslaw K. Szymanski. Labelrankt: Incremental community detection in dynamic networks via label propagation. *CoRR*, abs/1305.2006, 2013.

[48] Shihua Zhang, Rui-Sheng Wang, and Xiang-Sun Zhang. Identification of overlapping community structure in complex networks using fuzzy c -means clustering. *Physica A-statistical Mechanics and Its Applications*, 374:483–490, 2007.