Máster en Inteligencia Artificial,
Reconocimiento de Formas e Imagen Digital

Trabajo Fin de Máster

Detección de comunidades en redes complejas

Rodrigo Aldecoa

Departamento de Sistemas Informáticos y Computación

Universidad Politécnica de Valencia

Instituto de Biomedicina de Valencia

Consejo Superior de Investigaciones Científicas

Directores

Dr. Ignacio Marín
Unidad de Bioinformática
Instituto de Biomedicina de Valencia
Consejo Superior de Investigaciones Científicas

Dr. José Miguel Benedí Ruiz
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Índice general

Abstract

Muchos tipos de datos pueden ser representados mediante una red. La estructura de comunidades, una propiedad intrínseca de estas redes, puede desvelar estructuras subyacentes en dichos datos que serían difíciles de observar mediante otros análisis. Una comunidad puede ser definida como un conjunto de nodos más densamente conectados entre ellos que con el resto de la red. Su importancia radica en el principio de que se espera que nodos que se encuentran juntos en una misma comunidad compartan atributos, características comunes o relaciones funcionales. Por eso, diferentes campos científicos focalizan ahora muchos de sus esfuerzos en generar herramientas, algoritmos y modelos matemáticos que permitan una eficaz detección de estas comunidades. En este trabajo de fin de máster se resumen los avances realizados por el autor en este contexto, los cuales han sido exitosamente publicados en tres artículos en revistas internacionales. El cuerpo del trabajo se divide en tres partes, que corresponden a las tres citadas publicaciones. En primer lugar, presentamos Jerarca [1], una *suite* de algoritmos para el análisis de redes complejas. Además de convertir una red en un árbol jerárquico, es capaz de detectar la mejor partición en comunidades de la red siguiendo diferentes criterios. Segundo, proponemos un nuevo índice matemático para medir la calidad de la partición en comunidades de una red, al que denominamos Surprise [2]. Nuestros datos muestran que los resultados obtenidos por Surprise en todo tipo de redes, superan cualitativamente los obtenidos por Modularity, el criterio matemático más popular y utilizado en este ámbito. Por último, presentamos un nuevo tipo de *benchmarks,* a los que llamamos "closed", para testar algoritmos de detección de comunidades [3]. Además de poder ser utilizados para comparar el funcionamiento de diferentes algoritmos (como hacían los *benchmarks* utilizados anteriormente), los *closed benchmarks* nos aportan valiosísima información sobre la optimalidad de las soluciones proporcionadas por dichos algoritmos. En definitiva, el trabajo aquí descrito propone novedosas y contrastadas herramientas matemáticas que ayudarán a una mejor comprensión de la estructura de comunidades en redes complejas.

1. Rodrigo Aldecoa and Ignacio Marín. Jerarca: Efficient analysis of complex networks using hierarchical clustering. *PLoS ONE*, 5(7):7, 2010.

2. Rodrigo Aldecoa and Ignacio Marín. Deciphering network community structure by surprise. *PLoS ONE*, 6(9):8, 2011.

3. Rodrigo Aldecoa and Ignacio Marín. Closed benchmarks for network community structure characterization. *Physical Review E*, (in press), 2012.

1 Introducción

La actual ciudad rusa de Kalilingrado, conocida como Königsberg hasta su conquista por las tropas soviéticas en 1945, fue el escenario del nacimiento de una de las áreas de mayor auge de la matemática actual. A principios del siglo XVIII, Königsberg era uno de los núcleos científicos e intelectuales más importantes de toda Prusia. Allí convivían filósofos, pensadores y matemáticos como Immanuel Kant, Johann Georg Hamann o Christian Goldbach. La peculiar distribución de la ciudad, construida sobre el estuario del río Pregel, impide visitar sus distintos barrios sin tener que atravesar los puentes que conectan unos con otros. Esto hizo que entre los ilustrados de la época surgiese, a modo juego intelectual, el conocido posteriormente como Problema de los puentes de Königsberg. El problema consistía en saber si era posible recorrer la ciudad, pasando por sus siete puentes y cruzando sólo una vez cada uno de ellos, volviendo de nuevo al punto de partida. En 1736, Leonhard Euler consiguió responder negativamente a esta cuestión, al visualizar el problema de una forma nunca antes planteada: recurrió a una abstracción del mapa, representando cada puente mediante una línea que unía dos puntos, cada uno de los cuales representaba una región distinta de la ciudad (Figura 1.1). De este modo, se modeló el problema inicial como un grafo, el cual está compuesto por una serie de puntos, llamados nodos o vértices y un conjunto de conexiones entre ellos denominadas aristas o *links*.

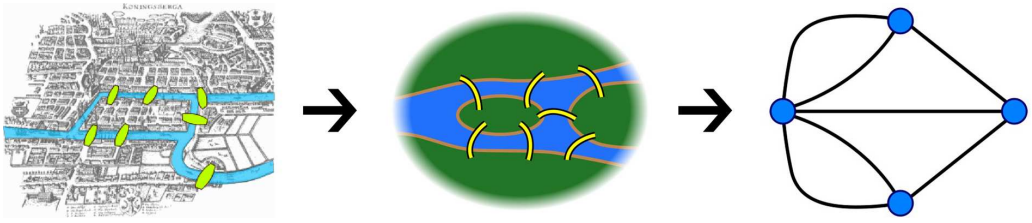

Figura 1.1: Transformación del mapa de Königsberg en un grafo

A partir de ese momento, considerado como el origen de la Teoría de Grafos, el estudio de los grafos y sus propiedades ha sido constante dentro del ámbito matemático [1]. A principios del S. XX sus aplicaciones comienzan a ser extendidas a otras áreas como la sociología, donde las relaciones interpersonales pretenden ser estudiadas mediante el análisis de redes sociales [2]. Sin embargo, no es hasta la aparición y sobre todo la amplia difusión de los ordenadores cuando el análisis de redes adquiere otra

perspectiva y pasa a ser ampliamente utilizado en la mayoría de campos científicos. La posibilidad de generar computacionalmente grandes cantidades de datos y el creciente interés por el estudio de sistemas complejos y las interrelaciones entre sus componentes, hacen que la representación mediante redes sea idónea para la resolución de muchos problemas [3]. Además de nuevas aplicaciones en sociología [4], áreas como la física [5], la informática [6] o la biología [7] se han visto beneficiadas por este nuevo enfoque. Centrándonos en esta última ciencia, la biología, hacia la cual está orientado el desarrollo de este trabajo, podemos afirmar que la modelización de problemas mediante redes es clave en multitud de ámbitos. Ahora que se intentan estudiar sistemas biológicos como un todo, que la secuenciación masiva de genomas es un hecho y que el conocimiento a nivel molecular de los organismos es cada vez mayor, el desarrollo de modelos matemáticos y herramientas para el análisis de este tipo de redes es imprescindible. Y a ello está dedicado este estudio.

1.1. Redes complejas

En 1959, Paul Erdös y Alfréd Rényi propusieron un modelo de grafos aleatorios, conocido como Erdös-Rényi, en el cual los nodos se conectan al azar [8]. Cada *link* aparece en el grafo con una probabilidad p, independientemente de la configuración del resto. Esto es lo mismo que decir que cada par de nodos tiene la misma probabilidad p de estar conectados. Cuando se empezaron a representar y estudiar redes que modelaban sistemas reales, los científicos descubrieron que sus propiedades distaban mucho de las de un grafo aleatorio [9, 10] y por ello reciben el nombre de complejas. Las redes reales poseían características particulares y distintas de las que podríamos esperar si las conexiones entre los nodos estuviesen distribuidas al azar. ¿A qué era debido aquello? La respuesta obvia era que la estructura de las redes reales estaba íntimamente ligada al funcionamiento del sistema que modelaban. Y por tanto, se podía extraer información valiosísima si se analizaban eficientemente. Gracias al interés generado sobre las redes complejas, ahora sabemos que la mayoría de ellas comparten ciertas propiedades como la heterogeneidad en el grado de sus vértices, cuya distribución produce, en la mayoría de los casos, una larga cola conocida en inglés como *heavy-tail* y característica de distribuciones de ley de potencia, Zipf o Pareto [9, 11]. Además, estas redes destacan por tener un alto coeficiente de clustering, que en cierto modo está relacionado con otras propiedades de las redes complejas que son el objeto de este estudio: su estructura de comunidades y, en muchos casos, su estructura jerárquica.

1.2. Estructura de comunidades

La estructura de comunidades, como se ha comentado anteriormente, es una propiedad de las redes complejas [13]. Una comunidad puede ser definida como un conjunto de nodos que están más densamente conectados entre ellos que con el resto de la red (Figura 1.2). La importancia de este planteamiento radica en que se espera que los nodos que están contenidos dentro de una misma comunidad compartan atributos, características comunes o relaciones funcionales (ampliamente detallado en [14]). Sin embargo, no existe una definición exacta de lo que es, o debe ser una comunidad, lo cual veremos más tarde que genera multitud de inconvenientes a la hora de dividir una red en sus distintas comunidades, lo que se conoce como partición o *clustering*.

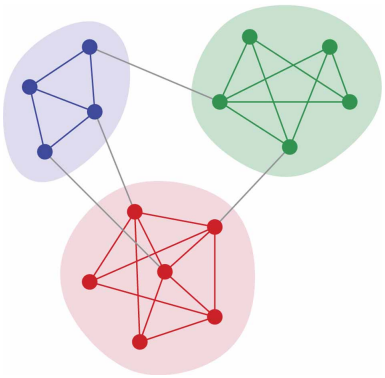

Figura 1.2: Ejemplo de red con estructura de comunidades. Los nodos de esta red están divididos en tres comunidades, donde la mayoría de las conexiones son intra-comunidad y sólo unas pocas entre comunidades. [12]

1.3. Particiones

Una partición es la división de una red en comunidades o clusters, de modo que todo nodo pertenece a algún cluster. Además, estas comunidades pueden estar jerárquicamente estructuradas, es decir, dos o más comunidades al fusionarse pueden formar una comunidad de un nivel superior. Este tipo de estructuras pueden ser representadas mediante un árbol o dendrograma. Por otro lado, en el caso de que un nodo sea asignado a más de una comunidad hablamos de particiones con solapamiento u *overlapping*. Es obvio que conforme crece el número de nodos, el número de particiones distintas que pueden obtenerse crece de una forma más que exponencial, lo cual dificulta de manera extrema la selección de la mejor partición del grafo. Cómo encontrar esta partición óptima es, probablemente, el problema abierto más importante de la investigación en estructura de comunidades. Una gran variedad de métodos y algoritmos, cada uno de ellos con su propia definición intrínseca de comunidad, han sido desarrollados para intentar extraer la partición óptima de una red. Algunos de ellos tratan de optimizar un índice global de calidad de la partición, como puede ser su Modularity [15] o Surprise [16], de los que hablaremos más tarde. Otros, sin embargo, utilizan la matriz de adyacencia para extraer información del

grafo, aplicando, por ejemplo, métodos espectrales [17]. Además, estimaciones de máxima verosimilitud [18], técnicas de campos de la física como la interacción de partículas [19] o elementos extraídos de Teoría de la Información [20] son sólo unos pocos ejemplos de métodos que han sido aplicados con relativo éxito a la búsqueda de comunidades. Pero, debido a la vaga definición de comunidad, surge otro problema: la mayoría de veces, las soluciones que devuelven los algoritmos divergen entre ellas. Es necesario, por tanto, compararlas y poder distinguir cuál es la mejor de todas. En este sentido, puede ser útil el definir una función global de calidad, que asigne a cada partición un valor numérico para, de este modo, poder compararlas cuantitativamente y elegir de entre ellas la de mayor valor como mejor. La selección de esta función de calidad no es trivial, ya que hemos de recordar que la definición de qué es o cómo debe ser una comunidad está pobremente definida. Por tanto, dependiendo de esta definición, la partición óptima puede variar entre las distintas candidatas.

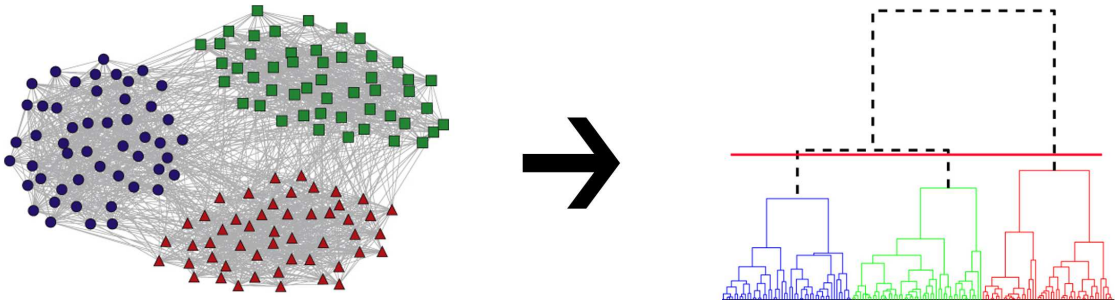

Figura 1.3: Estructura de comunidades a partir de un dendrograma. Tras representar la red como un árbol jerárquico, éste puede cortarse a un determinado nivel y de este modo crear una partición. En la figura, al cortar el dendrograma por la línea roja horizontal conseguimos una partición de tres comunidades que representa la red analizada. (Figura modificada de [21])

1.4. Modularity

La función de calidad más popular se denomina Modularity y fue propuesta por Newman y Girvan en 2004 [15]. Se basa en la idea de que una distribución en clusters no es lo que se espera por azar en una red, y por tanto, trata de cuantificar la intensidad de esta estructura de comunidades comparando la densidad de links dentro y fuera de cada comunidad con la densidad que esperaríamos si los links estuviesen distribuidos aleatoriamente en la red. A este modelo estadístico se le debe dotar de un modelo nulo que especifique qué es lo que se espera por azar. En este caso, probablemente influidos por artículos que tuvieron gran impacto en ese momento ([5, 10, 22] entre otros), los autores consideran que la distribución de grados de los nodos es una propiedad intrínseca de la red y proponen un modelo

nulo siguiendo este principio. La siguiente fórmula es la utilizada para calcular la Modularity (Q) de una partición determinada:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \ \delta(C_i, C_j) \tag{1.1}$$

donde A es la matriz de adyacencia del grafo, k_i el grado del nodo i y m el número de *links*. Según su modelo nulo, el número esperado de links entre dos nodos i y j es $\frac{k_i k_j}{2m}$. La función δ vale 1 si los nodos i y j están en la misma comunidad ($C_i = C_j$) y cero en otro caso.

Sin embargo, unos años más tarde, en 2007, Fortunato y Barthélemy demostraron matemáticamente que la optimización de esta medida sufre de un límite de resolución [23]. Esto significa que es incapaz de detectar comunidades de tamaño menor que un límite que viene determinado por el número de links del grafo y su patrón de conexiones.

1.5. Benchmarks

Dado que cualquier evaluación mediante una función global de calidad viene determinada por la definición de comunidad en la que se basa dicha función, debemos utilizar otros métodos para una efectiva comparación entre algoritmos. Para ello, como ocurre en muchos otros campos, la utilización de *benchmarks* puede ayudarnos. En 2002 Girvan y Newman proponen un tipo de *benchmark* [13] en el cual una red de 128 nodos se divide en cuatro comunidades de igual tamaño. Cada nodo a su vez está conectado, de media, con 16 nodos de su misma comunidad y no existen links entre comunidades. En ese momento podemos asumir que la estructura de comunidades real que existe es la partición en 4 comunidades iguales y que cualquier algoritmo que funcione razonablemente bien es capaz de detectar esa partición como óptima. A continuación, la estructura de comunidades inicial se empieza a degradar, aleatorizando para ello los links, que pasan de ser intracomunitarios a conectar nodos entre distintas comunidades. Como el proceso de aleatorización es progresivo, podemos observar hasta qué punto de degradación un algoritmo es capaz de seguir reconociendo la partición original. Parece lógico que el algoritmo que siga reconociendo la "partición óptima" durante un mayor tramo de degradación de la red original será el mejor. Cabe destacar que el *benchmark* descrito es poco realista, conociendo que en las redes reales la distribución de links es altamente heterogénea y los tamaños de las comunidades también pueden variar enormemente [9, 5]. Por ello, Lancichinetti *et al.* propusieron un modelo de *benchmark* mediante el cual se pueden generar redes cuyas distribuciones de los grados de los nodos y del tamaño de las

comunidades siguen leyes de potencia [24]. Aunque el proceso de degradación de la red no se realiza como en el caso anterior, sí que podemos afirmar que es equivalente. Y por tanto, la comparación entre algoritmos se realiza de forma análoga.

1.6. Medidas para comparar particiones

En los benchmarks anteriores, conforme la estructura de comunidades se hace más difusa, no es suficiente con determinar si la solución obtenida por un método es idéntica o no a la partición inicial. Dado que es más difícil para los algoritmos el reconocer esa estructura, sus soluciones pueden no ser iguales a ella y errar ligeramente (o gravemente, si el algoritmo no es bueno). Por eso, es conveniente disponer de una medida que nos indique el grado de similitud entre dos particiones, en este caso entre la devuelta por un algoritmo y la partición original. La comparación entre dos particiones no es trivial. Aunque el número de nodos clasificados es el mismo, tanto el número de comunidades como el tamaño de éstas puede ser distinto entre dichas particiones. Debido a esta complejidad se han propuesto diversas medidas desde distintas aproximaciones, las más importantes basadas en el conteo de pares y Teoría de la Información. Las medidas basadas en conteo de pares consideran como positivos los pares que están en un mismo grupo tanto en una como en otra partición. Lo mismo ocurre para los que están en distintos grupos en una partición, también deberían estarlo en la otra. Por otro lado, penalizan a los nodos que se encuentran juntos en una partición y separados en otra. Cada medida utiliza una fórmula distinta para calcular su valor, pero siempre basándose en este conteo de pares. Los índices más conocidos son *Jaccard* [25] y *Rand Index* [26]. Éste último tiene su versión corregida para eliminar posibles aciertos debido al azar, el *Adjusted Rand Index* [27]. Otras medidas tratan de evaluar cuánto pueden adivinar de una partición si ya se conoce la otra. La información necesaria para hallar la segunda partición se calcula mediante conceptos básicos de Teoría de la Información como son la entropía o la información mutua. Las medidas más utilizadas al respecto son la *Normalized Mutual Information* [28] y la *Variation of Information* [29]. La primera toma valores entre 0 y 1, siendo 1 cuando ambas particiones son idénticas y 0 cuando son independientes. Por su parte, la *Variation of Information* puede ser considerada como una distancia y toma un valor de 0 cuando las dos particiones son iguales, aumentado conforme aumenta su disimilitud.

2 *Jerarca*: Generación de una *suite* de algoritmos de detección de comunidades

2.1. Introducción

Como se ha comentado anteriormente, el *clustering* jerárquico puede ser utilizado para la detección de comunidades en una red y presenta varias ventajas sobre otros procedimientos. Por un lado, es un método completamente no supervisado y, por tanto, no es necesario especificar *a priori* el numero de comunidades presentes. Además, la generación de un árbol jerárquico permite no sólo cortarlo a un determinado nivel y extraer una partición, sino también observar cómo los nodos de la red (que son las hojas del árbol) se van combinando en grupos de niveles superiores. Sin embargo, el desarrollo de estrategias de *clustering* jerárquico es particularmente problemático en grafos no dirigidos y no ponderados. Aunque este tipo de grafos son simples en su configuración y por eso se han centrado en ellos la mayor parte de los estudios en este campo, presentan serias dificultades a la hora de definir comunidades. En este primer trabajo intentamos resolver uno de estos problemas, el conocido como *"Ties in proximity problem"* (discusión detallada en [30]). En este tipo de redes, la distancia entre dos nodos (la cual es utilizada en el proceso de *clustering* como medida de disimilitud) suele ser definida como el mínimo número de *links* que deben ser recorridos para unir uno con otro. Este cálculo, cuando se aplica a redes de cientos o miles de nodos como es el caso de la mayoría de redes reales importantes, produce un número desorbitado de empates o *ties*. Esto hace que sea imposible construir un árbol jerárquico razonable a partir de estas distancias entre nodos. Además, el problema causado por los empates en este primer nivel se extiende a cada paso del *clustering* jerárquico, ya que son posibles un enorme número de aglomeraciones (o divisiones) alternativas. Varios autores han intentado resolver este problema generando medidas de similitud entre nodos distintas a sus distancias [31, 32, 33]. Sin embargo es difícil justificar el uso de cualquiera de ellas.

Hace algunos años, en el grupo del Dr. Ignacio Marín plantearon una estrategia para resolver el problema de los empates [30]. Calculaban las distancias entre unidades mediante el algoritmo de Floyd y las denominaban *distancias primarias*. Como en

estas distancias, el número de empates es extremadamente alto, el primer paso de su método consistía en generar, mediante *clustering* jerárquico convencional (e.g., *average linkage*), un gran número de particiones alternativas de la red. De este modo podían hacer una estimación empírica del número de particiones en las que dos nodos habían sido incluidos en diferentes *clusters*. Normalizando este valor respecto al número total de particiones generadas, se conseguía una distancia para cada par de nodos, creando así una matriz de *distancias secundarias*. Por último, a partir de esta matriz se generaba un dendrograma del cual se extraían la partición final. Este método descrito fue denominado por los autores *iterative cluster analysis* e implementado en el programa UVCluster [30]. Más tarde fue exitosamente aplicado a redes reales de datos biológicos [34, 35, 36]. Sin embargo, la red de mayor tamaño analizada mediante UVCluster fue de 632 unidades [36] ya que el coste del algoritmo es $\mathcal{O}(n^3)$, siendo n el número de nodos.

En el trabajo realizado aquí, describimos una *suite* de programas denominada Jerarca, que contiene algoritmos nuevos y más eficientes para realizar *iterative hierarchical clustering*. Uno de ellos es básicamente una implementación mas rápida de UVCluster. Los otros dos programas, RCluster y SCluster proporcionan formas alternativas de generar la matriz de distancias secundarias a partir del grafo. A la hora de generar la jerarquía entre nodos a partir de la matriz, se han implementado dos algoritmos filogenéticos bien conocidos: UPGMA y Neighbor-Joining [37, 38, 39]. Finalmente, Jerarca incluye dos diferentes criterios matemáticos para extraer la mejor partición de los dendrogramas. Por un lado, el índice conocido como *modularity* (Q) [15] que ha sido ampliamente utilizado en todo tipo de trabajos sobre estructura de comunidades. Por otro lado, como alternativa, se incluye una modificación de un índice basado en una distribución hipergeométrica sugerido en el artículo que describe UVCluster [30]. Jerarca produce además una serie de ficheros útiles para editar, analizar y visualizar los resultados.

2.2. Métodos

La Figura 2.1 muestra el diagrama de flujo de Jerarca. El código ha sido escrito en *C++* y está libremente disponible en *http://jerarca.sourceforge.net*, donde también se puede encontrar la ayuda necesaria para realizar correctamente un análisis. De modo general, podemos realizar una llamada a Jerarca mediante el siguiente comando:

jerarca <Graph file> <Iterative algorithm> <Tree algorithm> <Iterations>

donde:

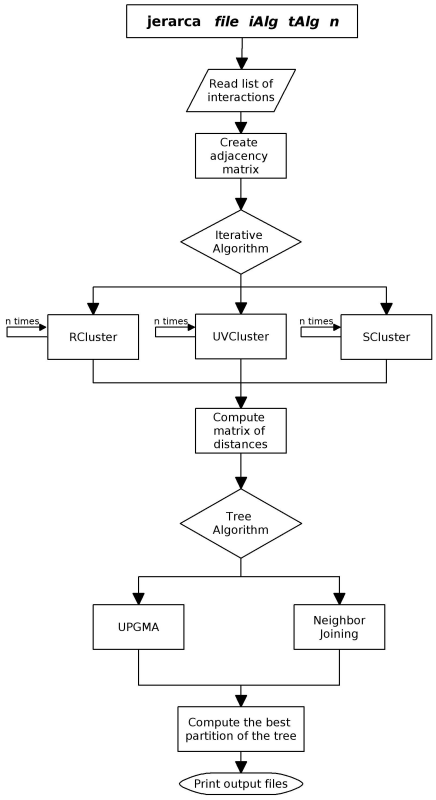

Figura 2.1: Diagrama de flujo de Jerarca. Los cuatro parámetros de entrada son *file* (lista de pares de nodos que representan los *links* del grafo), *iAlg* (algoritmo iterativo a utilizar), *tAlg* (algoritmo jerárquico a utilizar) y n (número de interacciones que se ejecutarán).

Graph file: Es el nombre del fichero que describe la red. Este fichero ha de contener una lista de links, cada uno de ellos en una línea y representado por dos nodos separados por un tabulador o espacio.

Iterative algorithm (uv, r, s, all): Se debe seleccionar el algoritmo con el que se desea generar la matriz de distancias secundarias (UVCluster, RCluster, SCluster o todos en paralelo)

Tree algorithm (u, nj, all): Selección del algoritmo que transformará la matriz de distancias secundarias en un árbol jerárquico (UPGMA, Neighbor-Joining o los dos en paralelo)

Iterations: Número de iteraciones. Número de particiones alternativas que generará el algoritmo iterativo deseado para calcular la matriz de distancias secundarias.

En la Figura 2.2 podemos ver detallado cómo sería un análisis con Jerarca. El fichero de entrada donde se describe el grafo consiste en una lista de *links* que Jerarca transforma internamente a una matriz de adyacencia. Esta es toda la información

necesaria para que el algoritmo iterativo (UVCluster, RCluster o SCluster) genere la matriz de distancias secundarias, la cual será la entrada del algoritmo de *clustering* jerárquico (UPGMA o Neighbor-Joining). Una vez construido el dendrograma, Jerarca utiliza los índices Q (modularity) y H (basado en una distribución hipergeométrica) para seleccionar la altura a la que se debe "cortar" el árbol y así extraer su mejor partición.

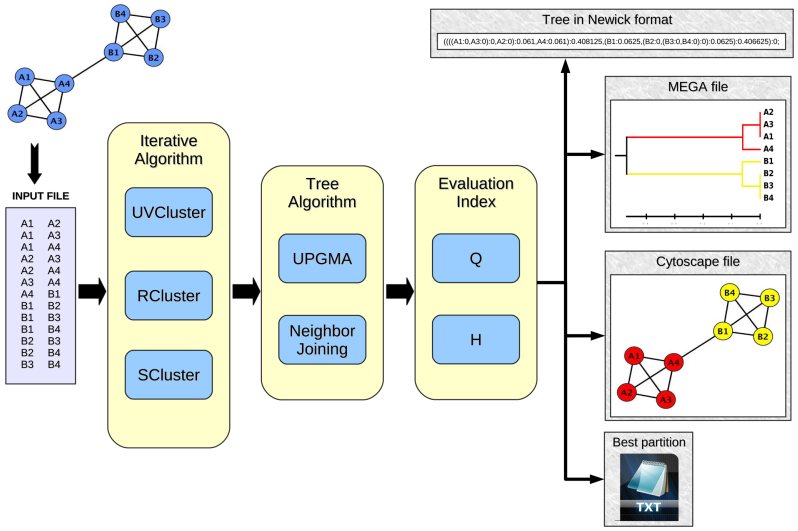

Figura 2.2: Un análisis con Jerarca. El usuario especifica el fichero de entrada donde el grafo está representado. Se analiza por el programa a través de diferentes algoritmos y produce cuatro ficheros de salida: el árbol en formato Newick, un fichero compatible con MEGA, un fichero que contiene los atributos de los nodos compatible con Cytoscape y un fichero de texto plano donde se recoge la partición óptima del árbol.

2.2.1. Iterative Algorithms

Recientemente, conseguimos desarrollar nuevas ideas para la generación de la matriz de distancias secundarias que son la principal novedad que aporta Jerarca. En primer lugar pensamos en una forma de acelerar la ejecución de UVCluster [30]. Este algoritmo contenía un parámetro llamado *Affinity Coefficient* (AC) que permitía ajustar el grado de relajación del *clustering*, de modo que cuanto más bajo era su valor nodos más alejados podían encontrarse dentro del mismo *cluster*. El máximo valor $AC = 100$ implica que únicamente nodos que estuviesen a un paso, es decir, conectados por un *link*, podían aparecer juntos (véase [30] para una explicación más detallada). Es significativo que en todos los trabajos donde se aplicó UVCluster a redes biológicas el valor fuese siempre $AC = 100$ [34, 35, 36]. Si utilizásemos este valor por defecto, el *clustering* podría ser realizado simplemente con la matriz de

adyacencia de la red y no sería necesario calcular las distancias primarias. De este modo evitamos utilizar el algoritmo de Floyd para calcular las distancias entre cada par de nodos, cuya complejidad es $\mathcal{O}(n^3)$. Gracias a este salto cualitativo y a optimizaciones menores en el código conseguimos que el código de UVCluster implementado en Jerarca tenga un coste de $\mathcal{O}(n^2)$. Además de la nueva implementación de UVCluster, dos nuevos algoritmos han sido introducidos: RCluster y SCluster. Ambos siguen la filosofía general del primero, aunque generan la matriz de distancias secundarias de manera alternativa. En la Figura 2.3 podemos ver resumidos los bucles principales de los tres algoritmos, los cuales pasamos a detallar brevemente a continuación:

UVCluster

En cada iteración selecciona un nodo al azar y lo introduce en un *cluster* junto con todos los nodos que puedan formar un *clique* con él. Suponiendo que es un grafo en el que no hay nodos desconectados, el *cluster* formado deberá tener al menos dos nodos, el seleccionado y alguno de los que conectan con él (dos nodos conectados por un *link* también es un *clique*). En caso de que haya más nodos a distancia de un paso de todos los que existen dentro del *cluster*, también son añadidos. De este modo se consigue el *clique* o subgrafo completamente conectado. Cuando no se pueden añadir más nodos al *cluster*, éstos se eliminan del grafo y se comienza nuevamente con un nodo seleccionado al azar de los restantes de la red. El algoritmo continúa hasta acabar con todos los nodos. En ese momento, disponemos de una partición del grafo, todos los nodos están asignados a algún *cluster*. Este proceso se repetirá tantas veces como iteraciones haya especificado el usuario al llamar al programa. UVCluster es un algoritmo voraz que tiende a favorecer el descubrimiento de comunidades muy compactas.

RCluster

El segundo algoritmo, RCluster (R de *random*), también genera *cliques*. Sin embargo, en vez de empezar por un nodo y ampliar el *clique* hasta su máximo tamaño como hacía UVCluster, en cada paso RCluster fusiona dos clusters aleatoriamente elegidos. El nuevo *cluster* debe ser también un *clique*. Para realizar esto, el programa en un primer momento considera cada nodo por separado como un *cluster* de una unidad. A continuación selecciona aleatoriamente dos clusters y comprueba si al fusionarse forman un *clique*, si es así elimina los dos clusters y forma uno de nivel superior. Seguirá repitiendo este paso hasta que no queden más clusters "fusionables". Sin embargo, conforme avanza el proceso de *clustering*, la probabilidad de encontrar dos *clusterings* que se puedan fusionar si se eligen aleatoriamente se reduce enormemente. En ese momento RCluster pasa a una segunda estrategia, calcula la matriz de

adyacencia de la partición actual, donde $A_{ij} = 1$ si los clusters i y j son fusionables y $A_{ij} = 0$ en caso contrario. Aleatoriamente va fusionando clusters y recalculando la matriz en cada paso hasta que no existen $1s$. Hay que destacar que al contrario que en UVCluster, en RCluster múltiples *cliques* se van formando de forma paralela. Sin embargo, el proceso de búsqueda de clusters fusionables enlentece el algoritmo dando una complejidad temporal de $\mathcal{O}(n^2 \log n)$.

SCluster

Por último SCluster (S de simple), que es el algoritmo más rápido de los tres, corriendo en tiempo $\mathcal{O}(n \log n)$. SCluster simplemente selecciona al azar un nodo y forma un *cluster* con ese nodo y todos los que están conectados a él. Estos nodos se retiran del grafo, se vuelve a elegir un nodo al azar de los restantes y se realiza la misma acción. El proceso continúa hasta que no existen más nodos. La diferencia con los dos algoritmos anteriores es que aquí los nodos de un mismo *cluster* no tienen por qué formar un *clique*, sino simplemente estar conectados con el nodo seleccionado. Aunque en principio puede parecer una técnica demasiado sencilla para la detección de comunidades, más tarde veremos que es el algoritmo más potente de los tres en la mayoría de casos.

UVCluster

```
create the list of nodes
WHILE the list of nodes is not empty:
  create a new cluster C
  select a random node N
  add N to C
  remove N from the list of nodes
  WHILE possible:
    select a node N' connected to every node in C
    add N' to C
    remove N' from the list of nodes
  END WHILE
END WHILE
```

RCluster

```
assign each node Ni to a cluster Ci
n = 0
WHILE  n < number_of_nodes:
   n = n + 1
   randomly select two clusters Ci, Cj
   IF every node in Ci is connected to every node in Cj
        merge Ci and Cj
        n = 0
   END IF
END WHILE

create matrix M (num_of_clusters x num_of_clusters)
FOREACH pair of clusters Ci, Cj:
   IF every node in Ci is connected to every node in Cj
      M(Ci, Cj) = 1
   ELSE
     M(Ci, Cj) = 0
   END IF
END FOREACH

WHILE exists any M(Ci, Cj) = 1
   choose a random pair or clusters Ci, Cj for which M(Ci, Cj) = 1
   merge Ci and Cj
   remove Ci and Cj from M
   add the new formed cluster to M
   recalculate M
END WHILE
```

SCluster

```
create the list of nodes
WHILE the list of nodes is not empty:
  create a new cluster C
  select a random node N
  add N to C
  add to C every node N' connected to N
  remove every node in C from the list of nodes
END WHILE
```

Figura 2.3: Bucle principal de los tres algoritmos de *iterative clustering* implementados en Jerarca. Cada iteración define una partición de la red asignando los nodos a clusters. El bucle es repetido tantas veces como número de iteraciones ha especificado el usuario.

2.2.2. Selección de la mejor partición

La ejecución de alguno de los algoritmos anteriores genera la matriz de distancias secundarias entre nodos. Jerarca implementa dos algoritmos filogenéticos bien conocidos para transformar esa matriz en un árbol jerárquico: UPGMA y Neighbor-Joining. Una vez creado el dendrograma, Jerarca lo recorre de la raíz hacia las hojas para extraer la mejor partición. Aunque Neighbor-Joining genera un árbol sin raíz, en este caso el árbol se ha enraizado en el punto medio, técnica conocida como *mid-point rooting* [40].

Partiendo de la raíz, cada dicotomía en el árbol genera una nueva partición, de la cual se evalúa su calidad en cuanto a estructura de comunidades se refiere. Para ello, Jerarca implementa dos criterios matemáticos. El primero de ellos es la función Modularity (Q) [15] que ha sido utilizada en multitud de estudios. Q mide la distribución de *links* intra e inter-communitarios en una determinada partición comparada con el número de *links* esperados por azar dada una distribución específica de los grados de los nodos [41]. El segundo índice (llamado H) está basado en una distribución hipergeométrica acumulada de los *links* propuesta en el artículo que describía UVCluster [30]. La definición de H es la siguiente

$$H = -log \sum_{j=p}^{min(M,n)} \frac{\binom{M}{j}\binom{F-M}{n-j}}{\binom{F}{n}} \tag{2.1}$$

donde F es el número máximo de *links* en el grafo (para un grafo de$k$ nodos, $F = \frac{k(k-1)}{2}$), n es el número de *links* reales del grafo, M es el número máximo de *links* intra-communidad posibles dada una determinada partición y p es el número de *links* intra-communidad reales que aparecen en el grafo. El índice H mide la probabilidad de obtener la actual configuración de *links* en la partición asumiendo una distribución aleatoria de conexiones *intracluster* e *intercluster*. Debido al signo negativo del logaritmo, cuanto mayor sea H, mejor será la partición de la red ("más inesperada").

2.2.3. Ficheros generados

Jerarca produce cuatro tipos de ficheros de salida (Figura 2.2). Sus nombres, generados automáticamente, incluyen una referencia a los algoritmos y criterios de evaluación utilizados (e.g., un nombre típico sería *Filename_partitionH_SCluster_Upgma.txt*). Cada uno de los cuatro ficheros contiene distinta información sobre el análisis y se pueden diferenciar por su extensión:

.meg: Contiene la matriz de distancias secundarias obtenida a partir de la mejor partición del árbol de acuerdo con el índice Q o H. Este fichero puede ser directamente importado desde el software MEGA 4 (que permite hacer todo tipo de análisis filogenéticos y de *clustering* jerárquico) [42] para posteriores análisis.

.att: Contiene la mejor partición del árbol, especifica para cada nodo el *cluster* al que pertenece. El formato está diseñado para poder ser importado en Cytoscape versión 2.x (en este momento el software más utilizado para visualización y tratamiento de redes) [43].

.txt: Guarda la mejor partición de la red en formato de texto plano. Contiene información adicional como el número de clusters, valor de Q o H que ha dado lugar a esa partición y la asignación de cada nodo a un *cluster*.

.nwk: Este fichero guarda el árbol jerárquico en formato Newick. Al ser un formato estándar y muy utilizado, este fichero puede ser utilizado casi por cualquier software de tratamiento o visualización de árboles como es el caso de MEGA.

2.3. Resultados

La velocidad de los programas ha sido probada en varios benchmarks. Aquí describimos los resultados para tres de ellos, que constan de una red artificial y dos reales:

- Benchmark A: Generamos un grafo sintético con estructura de comunidades conocida, el cual constaba de 512 nodos divididos en 16 comunidades de igual tamaño. Todos los nodos de cada comunidad estaban completamente conectados, creando de este modo 16 *cliques* desconectados. Más tarde el grafo era "degradado" progresivamente, un cierto porcentaje de *links* eran eliminados y, el mismo porcentaje de los restantes eran aleatorizados entre los nodos. De este modo se conseguía que las comunidades se fuesen haciendo poco a poco más y más difusas. Las redes generadas mediante este proceso son similares a los grafos *Caveman* propuestos por Watts [44].

- Benchmark B: Compuesto por las proteínas que forman los 408 complejos proteicos establecidos en la base de datos CYC2008 (http://wodaklab.org/cyc2008; [45]) de la levadura *Saccharomyces Cerevisiae*. Descargamos de la base de datos BIOGRID [46] las interacciones proteína-proteína que se han caracterizado entre esas proteínas hasta el momento. El grafo final contiene 1604 nodos/proteínas y 14171 *links*/interacciones.

- Benchmark C: Descargamos de BioGRID el conjunto completo de las interacciones proteína-proteína de *Saccharomyces Cerevisiae*. Con estos datos construimos una red de 5735 nodos y 51134, que representan el interactoma completo del organismo. Es decir, todas sus proteínas y las interacciones conocidas entre ellas.

El *benchmark* A fue diseñado específicamente para probar la calidad de las particiones obtenidas por los distintos algoritmos implementados en Jerarca. Para ello generamos redes con diferentes porcentajes de degradación. En este contexto, un porcentaje de degradación del 10 % significa que el 10 % de los *links* han sido eliminados de la configuración inicial y que, de los restantes, un 10 % han sido aleatorizados. El barajeo aleatorio de *links* se realiza eliminando un *link* y posteriormente añadiéndolo entre dos nodos seleccionados al azar. En el artículo original de UVCluster los autores sugerían realizar un número de iteraciones igual al de 10 veces el número de nodos de la red [30]. Por tanto, para cada red, realizamos 5000 iteraciones de Jerarca con el parámetro *all* tanto para el algoritmo iterativo como para el jerárquico. Esto significa que se realizaron 12 (= 3 alg. iterativo x 2 alg. jerárquico x 2 criterios de partición) análisis para cada una de ellas. Con valores entre 0 % y 30 % de degradación, todos los algoritmos recuperaron la estructura de comunidades original sin ningún error. Sin embargo, a partir del 40 % empezaban a surgir pequeños errores, por lo que nos centramos en este caso. Para los 6 dendrogramas construidos al utilizar los 3 algoritmos iterativos y los 2 jerárquicos, las particiones óptimas devueltas por los dos índices de calidad implementados en Jerarca (Q y H) fueron exactamente las mismas. En todos los casos excepto en uno, un nodo era clasificado erróneamente. Sólo la combinación de SCluster y UPGMA conseguía recuperar correctamente la partición original. Este ejemplo muestra como todos los algoritmos funcionan eficientemente incluso cuando la estructura es bastante críptica (un 40 % de degradación significa que sólo alrededor de un tercio de los *links* originales permanecen). Por otro lado, también se muestra la ventaja de utilizar cuando sea posible todas las combinaciones de algoritmos, dado que algunos pueden funcionar mejor que otros en ciertos casos.

Todos los tests fueron llevados a cabo en un ordenador PC compatible con un procesador Intel Core 2 Quad Q8200 a 2.33 GHz y 4GB de RAM, con Linux 2.6 como sistema operativo. Los análisis del *benchmark* A fueron muy rápidos. Para los 12 análisis de cada red descritos en el párrafo anterior (5000 iteraciones por análisis) sólo fueron necesarios entre 30 y 75 segundos. Los grafos más degradados fueron los más costosos. Para probar la velocidad de los programas en redes reales de gran tamaño, utilizamos los benchmarks B y C. Para el *benchmark* B (1604 nodo), 16000 iteraciones tardaron 3.25 horas utilizando RCluster, mientras que para UVCluster y SCluster el coste fue de 2 minutos y menos de un minuto respectivamente. Esta gran diferencia a nivel de coste temporal se debe al hecho de que esta red contiene módulos densamente conectados (grupos de nodos altamente conectados) que favorecen las estrategias voraces implementadas en UVCluster y SCluster. Un análisis

del *benchmark* C (5735 nodos) costó 40 minutos con SCluster y 3 horas con UVCluster. La ejecución de RCluster en este *benchmark* no la realizamos por la cantidad de tiempo que consumiría, aunque estimamos que tardaría alrededor de 300 horas.

Resumiendo, los nuevos algoritmos implementados en Jerarca nos permiten el análisis de grandes redes. Como los tiempo aquí detallados demuestran, simplemente con un ordenador personal se puede tratar con problemas de varios miles de nodos en un tiempo razonable, utilizando SCluster y UVCluster. Además, para redes de hasta 1000 nodos, el usuario puede incluir también RCluster en sus análisis y obtener los resultados en cuestión de minutos o unas pocas horas.

2.4. Discusión

Debido a que la cantidad de información biológica crece a pasos agigantados, uno de los principales objetivos de la bioinformática es la generación de algoritmos eficientes que sean capaces de tratar con grandes *datasets*. En análisis de redes, el cuello de botella de la estrategia de *clustering jerárquico iterativo* es precisamente que el algoritmo de *clustering* debe ser repetido un gran número de veces para generar un amplio conjunto de iteraciones de modo que sean una buena representación de la estructura subyacente del grafo. La segunda parte del análisis, la construcción del dendrograma aplicando un algoritmo filogenético, se ejecuta una única vez y por tanto su coste es despreciable respecto al programa en general. Como se ha comentado anteriormente, las aplicaciones de UVCluster estaban limitadas por la cantidad de tiempo necesaria para efectuar un solo análisis de una red de tamaño medio/grande. Por tanto, era necesaria una optimización de esta estrategia. Aplicando ciertas restricciones (fijando el valor de AC a 100), conseguimos reducir cualitativamente la complejidad temporal del algoritmo. Aunque tradicionalmente estaba limitado a análisis de redes por debajo de 1000 nodos, ahora hemos visto que es capaz de ejecutarse sobre redes de varios miles de nodos en unas pocas horas. Esto permite el estudio de interesantes *datasets*, como el interactoma completo del organismo eucariota *Saccharomyces Cerevisiae* (*benchmark* C).

Una segunda ventaja significativa de Jerarca es que además incluye dos nuevos algoritmos, RCluster y SCluster, que proporcionan estrategias alternativas de calcular las distancias secundarias entre nodos de la red. RCluster genera aleatoriamente múltiples clusters al mismo tiempo, evitando el agrupamiento voraz de UVCluster. Sin embargo, el coste de este cambio fundamental hace el algoritmo más lento que UVCluster. Con SCluster ocurre justamente lo contrario: es el más rápido y voraz de los tres. A pesar de su aparente simplicidad, su funcionamiento es el apropiado (véase los resultados para el *benchmark* A detallados anteriormente). Dado que Jerarca permite la ejecución de varios análisis de forma paralela, recomendamos utilizar los

tres algoritmos para redes de hasta 1000 nodos. Un análisis completo de estas redes requiere menos de dos horas (ver Resultados). Con redes más grandes, de hasta 10000 unidades, tanto UVCluster como SCluster se pueden utilizar, ya que sus análisis tardarían unas pocas horas. SCluster, incluso, podría utilizarse para análisis de redes de mayor tamaño. Esto podría ser muy interesante en campos como el análisis de co-expresión o redes de interacción genética, en los cuales el número de nodos (en este caso corresponderían a genes) puede ser de decenas de miles. Sin embargo, hay que tener en cuenta que estos algoritmos son fácilmente paralelizables, ya que cada iteración es independiente del resto y pueden ser distribuidas en distintos procesadores y los resultados promediados al final.

Además de UPGMA, que ya se incluía en la versión original de UVCluster, Jerarca implementa una estrategia alternativa de construcción del árbol jerárquico utilizando el algoritmo Neighbor-Joining, probablemente el algoritmo más utilizado para construir árboles a partir de una matriz de distancias. Nosotros sugerimos obtener ambos árboles (su generación es casi instantánea) en los análisis, ya que la congruencia entre los dos métodos puede ser muy útil para comprender los resultados. Una ventaja adicional de Jerarca respecto a UVCluster es la posibilidad de determinar la calidad de una partición mediante dos criterios estadísticos (Q y H). Añadimos estas opciones considerando que los usuarios pueden estar interesados no sólo en obtener una representación jerárquica de la red, si no en cómo la red puede ser dividida en clusters o comunidades (ver Introducción). La estrategia utilizada para obtener las particiones es muy sencilla, dado que el árbol se recorre de las raíz a las hojas. Por tanto, el número de particiones posibles es reducido, igual al número de nodos. Una última ventaja de Jerarca es el conjunto de ficheros de salida que genera. La posibilidad de exportar directamente los datos a programas como MEGA o Cytoscape permitirá a los usuarios tanto realizar análisis adicionales como obtener sofisticadas representaciones gráficas de los resultados.

Jerarca junto con su código fuente se encuentra libremente disponible bajo la licencia GNU GPLv3 en *http://jerarca.sourceforge.net*. La estructura modular del código permite fácilmente introducir nuevas características al programa. Nuevos algoritmos, tanto iterativos como para construir árboles, además de nuevos índices estadísticos para la evaluación de las particiones pueden ser fácilmente añadidos.

3 *Surprise:* Propuesta de una nueva medida de calidad de las particiones

3.1. Introducción

Un paso clave para entender las propiedades de una red consiste en determinar sus comunidades, grupos de nodos altamente conectados. No obstante, la mejor manera de establecer la estructura de comunidades de una red es todavía un problema abierto. Muchas estrategias han sido utilizadas (puede verse un amplio resumen en [14]), siendo la más popular la maximización de la Modularity (Q), propuesta por Newman y Girvan [15]. Esta medida Q, sin embargo, tiene el inconveniente de verse afectada por un límite de resolución: su maximización es incapaz de detectar comunidades más pequeñas que un cierto umbral que depende del tamaño de la red y de su patrón de conexiones [23]. Desde que se encontró este problema, no se ha vuelto a proponer ningún otro parámetro global para sustituir Q. Aunque se han sugerido estrategias alternativas (búsqueda de patrones estructurales locales, optimizaciones multi-nivel de Q), ninguna de ellas ha conseguido una aceptación general [14].

Hace algunos años, el grupo del Dr. Ignacio Marín sugirió la posibilidad de determinar la estructura de comunidades de una red evaluando la distribución de *links* intra e inter-comunitarios con una distribución hipergeométrica acumulada [30]. De acuerdo a esto, encontrar la estructura de comunidades óptima de una red es equivalente a maximizar el siguiente parámetro:

$$S = -log \sum_{j=p}^{min(M,n)} \frac{\binom{M}{j}\binom{F-M}{n-j}}{\binom{F}{n}} \tag{3.1}$$

donde F es el número máximo de *links* en el grafo (para un grafo de$k$ nodos, $F = \frac{k(k-1)}{2}$), n es el número de *links* reales del grafo, M es el número máximo de *links*

intra-communidad posibles dada una determinada partición y p es el número de *links* intra-communidad observados en el grafo. Denominamos a este parámetro S, de *Surprise*, ya que efectivamente mide la "sorpresa" (improbabilidad) de encontrar por azar una partición con ese enriquecimiento de *links* intracomunitarios en un grafo aleatorio.

En este trabajo, mostramos propiedades de S que hacen que sea actualmente el mejor parámetro global para estimar la estructura de comunidades de una red. Utilizando *benchmarks* estándar y otros creados por nosotros y un conjunto de los mejores algoritmos de detección de comunidades, mostramos que la maximización de S a menudo nos caracteriza óptimamente la estructura de comunidades del grafo. Cuando aplicamos esta medida a redes reales obtenemos algunas soluciones lógicas (la mayoría mejores que las obtenidas maximizando Q) pero otras veces aparecen particiones inesperadas que demuestran las limitaciones que la utilización de estrategias erróneas han arrojado sobre este campo de estudio.

3.2. Resultados

Evaluar el funcionamiento de un parámetro que mida la estructura de comunidades de una red requiere tanto un conjunto de algoritmos eficientes como un conjunto de *benchmarks*, los cuales consisten en redes sintéticas de estructura conocida. En este estudio seleccionamos seis algoritmos (ver Métodos) que fueron probados sobre dos tipos de *benchmarks*, LFR y RC. Los *benchmarks* LFR (Lancichinetti-Fortunato-Radicchi) se caracterizan por generar redes en las cuales tanto los grados de los nodos como los tamaños de las comunidades siguen leyes de potencia [24]. Por otro lado, los *benchmarks* RC (*Relaxed Caveman*) generan redes que comienzan con todos los nodos de cada comunidad formando un *clique*. Posteriormente, esta estructura se va relajando mediante la generación de *links* entre comunidades [44]. Posteriormente, los *benchmarks* LFR y RC fueron divididos en "abiertos" y "cerrados". Los *benchmarks* abiertos son aquellos que se han utilizado siempre [13, 24, 47], en los cuales se analizan un conjunto de redes con características similares y distintas densidades de *links* intra e inter-comunitarios. Conforme la proporción de *links* entre comunidades crece, las redes tienden hacia la aleatoriedad. En los *benchmarks* cerrados también se empieza con una red cuya estructura de comunidades se conoce, sin embargo, al degradarla no tiende hacia la aleatoriedad, si no que se transforma en una segunda estructura que también es bien conocida.

Para cada *benchmark* estimamos S y Q con los seis algoritmos. Los máximos valores de S y Q que se obtienen (S_{max} y Q_{max}) proporcionan las particiones que se utilizarán para compararlas con la estructura de comunidades conocida. Para medir la congruencia entre las particiones estimadas y las conocidas utilizamos la *Normalized Mutual Information* (NMI) como es común en trabajos anteriores [28, 24, 48].

3.2.1. Open benchmarks

La Figura 3.1a y Figura 3.1b muestran los resultados obtenidos en 4 *benchmarks* LFR abiertos que difieren en número de unidades y tamaños de comunidad [24] (ver Métodos). En la Figura 3.1a se puede apreciar que las soluciones calculadas mediante la maximización de la S caracterizan perfectamente la estructura de comunidades de la red ($NMI_S = 1$) incluso cuando ha sido altamente degradada con multitud de *links* inter-comunitarios, generados por el incremento del *mixing parameter* μ hasta 0.5 - 0.7 (ver Métodos para la definición de μ). Si μ sigue aumentando, la partición original ya no es la elegida por ningún algoritmo ($NMI_S < 1$). Este hecho sugiere que la estructura de comunidades original ya no existe. La maximización de S mejora cualitativamente la maximización de Q (Figura 3.1b): $NMI_S > NMI_Q$ en 2827/3600=78.5 % de los casos, $NMI_Q > NMI_S$ en sólo un 4.1 % de los casos. Cabe destacar también que $NMI_Q << NMI_S$ en grafos aleatorios o quasi-aleatorios, sugiriendo que maximizar Q sobreimpone, en estos casos, comunidades constreñidas por el límite de resolución intrínseco de Q. Además es significativo que la maximización de S presenta mejores valores de NMI media que los obtenidos por cualquier algoritmo por separado en estos mismos *benchmarks* [48]. Dependiendo de la configuración de la red y el valor del parámetro μ, la máxima S es obtenida por distintos algoritmos (Figura 3.2a).

El descubrimiento del límite de resolución de Q mostró que la posibilidad de detectar con exactitud una estructura de comunidades se ve gravemente afectada dichas comunidades son de distinto tamaño [23]. Sin embargo, por su propia construcción, los tamaños de comunidad de los *benchmarks* LFR son muy similares. Sus *Pielou's evenness indexes* (PI) [49] varían entre 0.96 y 0.98 en los 4 *benchmarks* utilizados aquí, cercanos al valor máximo posible (PI = 1 para comunidades de igual tamaño). Como considerábamos que era crítico probar S en situaciones más extremas, construimos los *benchmarks* RC, que tienen *PIs* muy inferiores. La Figura 3.1c y Figura 3.1d muestran los resultados en los *benchmarks* RC abiertos, con Degradación (D; ver Métodos) progresiva de la estructura original. Esta estructura es detectada con exactitud maximizando S, disminuyendo suavemente cuando D se incrementa (Figura 3.1c). De nuevo, los resultados obtenidos maximizando S mejoran claramente sobre los obtenidos maximizando Q (Figure 1d; $NMI_S > NMI_Q$ en 848/900=94.2 % de los casos, mientras que $NMI_Q > NMI_S$ en sólo un 3.3 % de los casos). Como ocurría en los *benchmarks* LFR, ninguno de los algoritmos consigue la mejor S en todos los casos (Figura 3.2b).

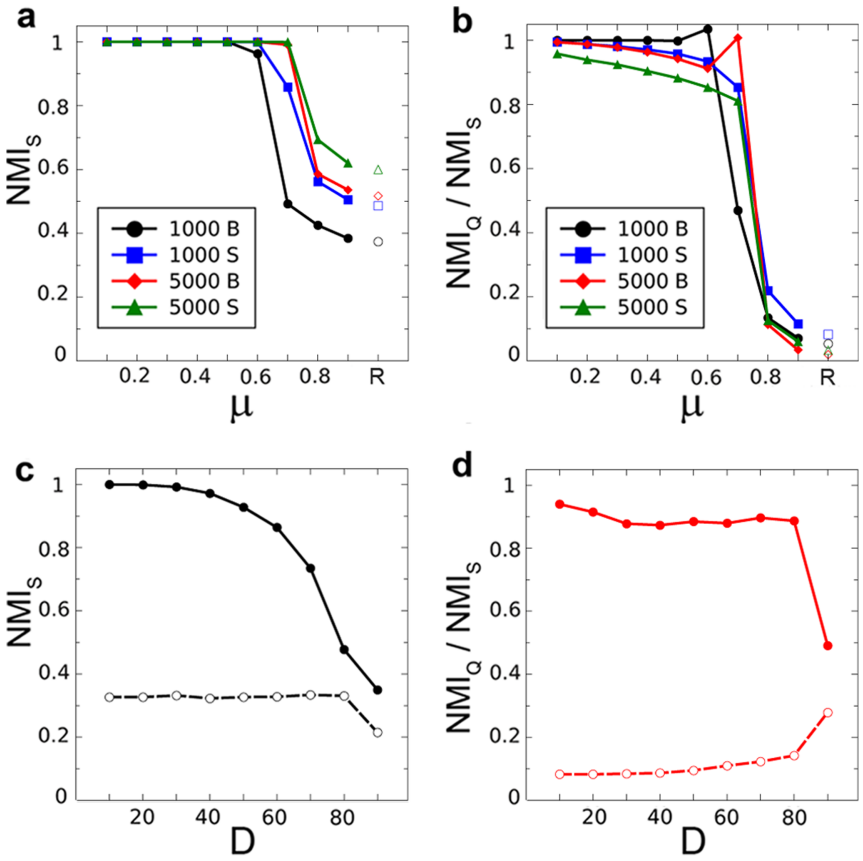

Figura 3.1: Resultados en los *benchmarks* LFR y RC abiertos. a) Resultados para los cuatro *benchmarks* LFR estándar abiertos. B y S indican que la red está compuesta por comunidades grandes o pequeñas respectivamente y 1000 o 5000 el número de nodos. μ es el *mixing parameter*, que mide la degradación de la estructura de comunidades. NMI mide la congruencia entre la estructura de comunidades deducida y la real. Cada punto esta basado en 100 redes distintas; los errores estándar de la media son demasiado pequeños para poder ser visualizados. También se muestran los valores obtenidos en 100 redes aleatorias (*random* R). b) Comparación entre la maximización de S y Q en los *benchmarks* LFR. Se muestra el ratio NMI_Q/NMI_S, que es casi siempre menor que 1. c) Resultados para los *benchmarks* RC. El parámetro *Degradation* (D) indica el porcentaje de *links* eliminados y aleatorizados. Cada punto está basado en 100 redes, los errores asociados de nuevo son demasiado pequeños para visualizarlos. Para cada valor de D se muestran los resultados de 100 redes aleatorias (círculos vacíos). Comparación entre la maximización de S y Q en los *benchmarks* RC.

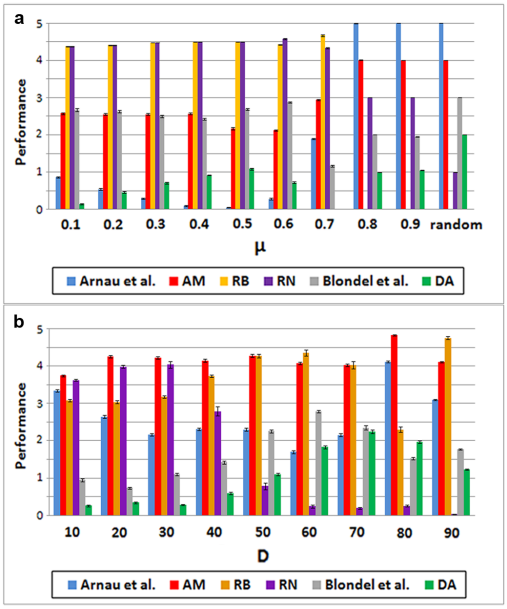

Figura 3.2: *Performance* media de los algoritmos en los *benchmarks* LFR y RC abiertos. Algoritmos utilizados: Arnau *et al.* [30], Aldecoa y Marín (AM) [47], Rosvall y Bergstrom (RB) [20], Ronhovde y Nussinov (RN) [19], Blondel *et al.* [50] y Duch y Arenas (DA) [51]. a) Ejemplo de los resultados obtenidos en los *benchmarks* LFR, 5000 nodos y comunidades grandes. Definimos *performance* de un algoritmo como $P = 6 - average\,rank$. Por tanto, el máximo valor $P = 5$ significa que un algoritmo ha sido el mejor en todas las redes, mientras que $P = 0$ que ha sido el peor en todas. Ningún algoritmo consigue los mejores resultados en todos los casos. b) Resultados obtenidos en los *benchmarks* RC. El valor de *performance* se calcula de la misma forma.

3.2.2. Closed benchmarks

Los resultados presentados anteriormente indican que utilizar S_{max} como índice para detectar estructura de comunidades tiene ventajas obvias sobre Q_{max}. Sin embargo, estas medidas no permiten evaluar cómo de óptima es la partición obtenida, dado que desconocemos qué valor potencial puede alcanzar la NMI en cada caso. Para solventar este problema generamos *benchmarks* LFR y RC cerrados, en los cuales tenemos una estimación a priori del valor máximo de los valores de NMI. Los resultados se muestran en las Figura 3.3 y Figura 3.4 (LFR y RC respectivamente). En todos los casos en los que S_{max} se utiliza, se observa un patrón casi perfectamente simétrico.

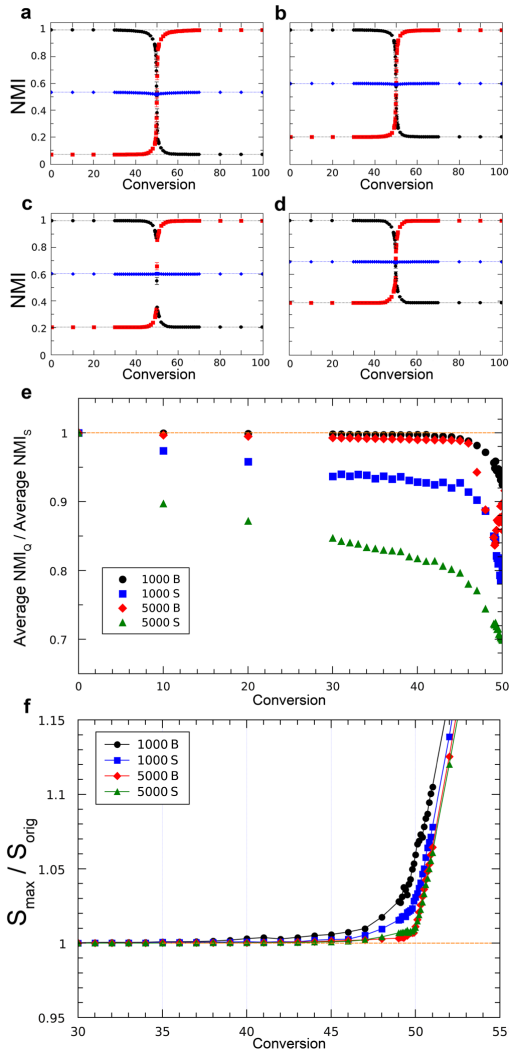

Figura 3.3: Resultados para los *benchmarks* LFR cerrados. a) *Benchmark* LFR con 1000 nodos y comunidades grandes. Para cada valor de *Conversion* (C), se muestran las NMIs entre la partición S_{max} y la estructura de comunidades inicial (círculos negros) o la final (cuadrados rojos). Los resultados simétricos hacen que la NMI media (diamantes azules) caiga con exactitud en una línea recta cuyo valor constante es $(1 + NMI_{IF})/2$. Cada punto está basado en 100 análisis independientes. b-d) *Benchmarks* LFR con, respectivamente, 1000 nodos y comunidades pequeñas (b), 5000 nodos y comunidades grandes (c) y 5000 nodos y comunidades grandes (d). Los resultados son muy similares a los mostrados en el panel a). e) Los valores medios de NMI obtenidos maximizando Q son peores que los obtenidos maximizando S, especialmente cuanto más degradada está la red y más complicado es recuperar la estructura inicial (valores de C altos). Este efecto se incrementa considerablemente al aumentar el número de nodos y reducir el tamaño de las comunidades, debido al límite de resolución de Q. Los resultados para $C > 50$ son simétricos a los expuestos aquí. f) Ratio $S_{max}/S_{orig} \geq 1$, es decir, se ha encontrado la estructura original o una diferente con un valor de S mayor.

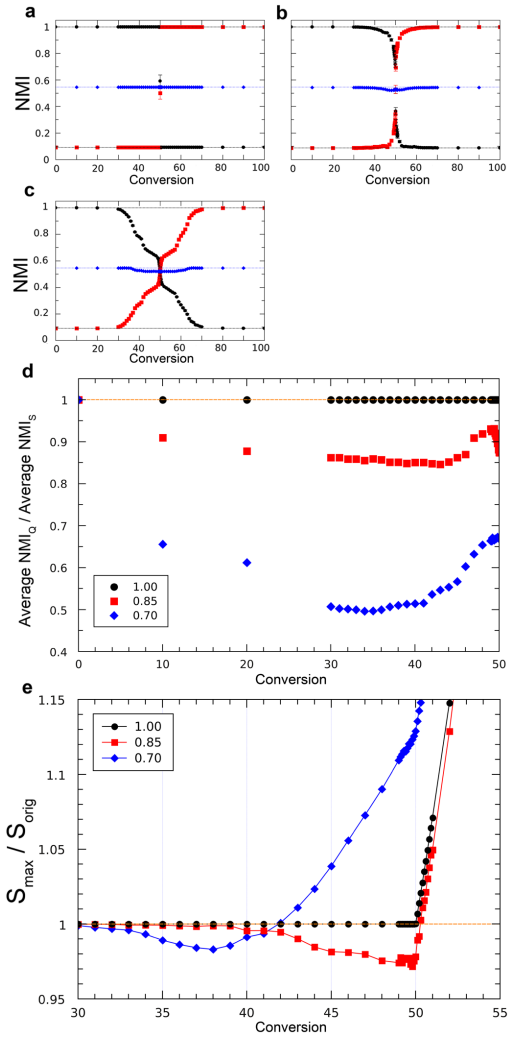

Figura 3.4: Resultados para los *benchmarks* RC cerrados. Utilizamos como ejemplos tres redes con diferentes heterogeneidades en los tamaños de comunidades. Índices Pielou (PI) iguales a 0.70, 0.85 y 1.00). a) PI=1 (comunidades de igual tamaño); b) PI=0.85; c) PI=0.70. Los resultados son análogos a los de la Figura 3.3, excepto que los patrones no son tan perfectamente simétricos en las redes más heterogéneas (paneles b y c; los diamantes azules se desvían ligeramente de la línea recta). d) Los valores medios de NMI cuando utilizamos Q son mucho peores respecto a S que en los *benchmarks* LFR, probablemente debido a la mayor heterogeneidad en el tamaño de las comunidades. e) $S_{max}/S_{orig} > 1$ en las redes con tamaños de comunidad desiguales. Los algoritmos utilizados no son capaces de conseguir los máximos valores de S, que como mínimo corresponden a la estructura original (S_{orig}). Esto puede contribuir a los ligeros fallos en la simetría de los paneles b y c. El hecho de que $S_{orig} \gg 1$ con $C < 50$ y PI=0.70 (diamantes azules) implica que los algoritmos están detectando estructuras muy diferentes a la inicial.

En el proceso de conversión de la estructura original en la final (incrementando el parámetro *Conversion*; ver Métodos), los descensos de NMI de la primera estructura son compensados por los incrementos de la segunda. La media de ambas NMIs es por tanto aproximadamente constante y tiene un valor idéntico o muy cercano a $\frac{1+NMI_{IF}}{2}$, donde NMI_{IF} se calcula comparando la estructura inicial y la final (Figura 3.3e, Figura 3.4d). Por otro lado, observamos que en los *benchmarks* LFR, S_{max} es siempre mayor o igual que S_{orig} (Figura 3.3f). En cambio, en los *benchmarks* RC no ocurre lo mismo (Figura 3.4e). Por tanto, podemos afirmar que a veces los algoritmos aquí utilizados son incapaces de detectar particiones con valores máximos de S (ya que existe al menos una partición, la original, con una S mayor). Esto podría explicar los ligeros desvíos de la simetría de NMI observados en algunos *benchmarks* RC (diamantes azules en Figura 3.4b y Figura 3.4c).

3.2.3. Redes reales

La Figura 3.5 muestra los resultados de S_{max} para tres redes reales. El primer ejemplo está basado en la base de datos CYC2008, la cual consta de 1604 que pertenecen a 324 complejos proteicos [45]. La congruencia entre las comunidades detectadas utilizando S_{max} y los complejos descritos es casi perfecta, $\mathrm{NMI_S} = 0.91$. En la Figura 3.5a se muestran las 11 comunidades de tamaño > 20, de las 313 detectadas, para observar el nivel de detalle obtenido. En cambio, la maximización de Q proporciona una clasificación tosca en sólo 24 comunidades con $\mathrm{NMI_Q} = 0.57$. Las 5 comunidades de mayor tamaño casi cubren la red entera (Figura 3.5b). Estos resultados muestran la altísima precisión de S cuando hay abundantes comunidades pequeñas, una situación en la cual Q, afectada por su límite de resolución, fracasa estrepitosamente. En la Figura 3.5c se muestra, como control positivo, los resultados de un *benchmark* clásico de estructura bien conocida, el *College football network* [13]. La congruencia con la partición esperada es de nuevo muy alta $\mathrm{NMI_S} = 0.93$. Por último, la Figura 3.5d muestra los resultados de otro bien conocido ejemplo, la red del *Zachary's Karate club* [52, 13]. Durante años se ha estudiado esta red social como una partición en dos comunidades. Sin embargo, los análisis aplicando S sorprendentemente devuelven 19 comunidades, 12 de ellas compuestas por un nodos aislados (Figura 3.5d).

3.3. Discusión

En este estudio hemos mostrado el potencial de maximizar el parámetro global Surprise (S) para caracterizar la estructura de comunidades presente en redes complejas. Los resultados indican que funciona cualitativamente mejor que el índice global más utilizado hasta el momento, la *Modularity* de Newman y Girvan [15]. Las ventajas

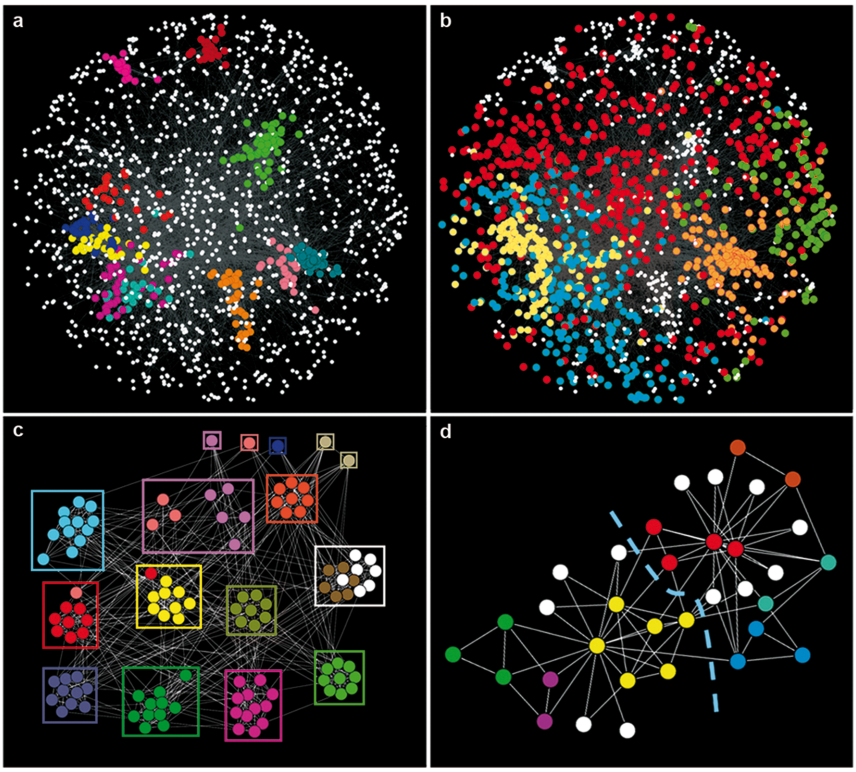

Figura 3.5: Aplicación de los análisis S_{max} a redes reales. Estructura de comunidades de la red CYC2008 (a,b), *College football network* (c) y *Zachary's karate club* (d), de acuerdo a la maximización de S (paneles a, c, d) o la de Q (panel b). En el panel c, se muestra la estructura de comunidades conocida mediante cuadrados. La línea discontinua del panel d divide la red en las dos comunidades que supuestamente existen. Esta división de la red no se sustenta en absoluto según los análisis de S_{max}: mientras que para dos comunidades $S = 13{,}61$, la partición óptima $S_{max} = 25{,}69$ divide la red en 19 comunidades. Doce de ellas son nodos aislados (representados en blanco).

de S sobre Q pueden no ser tan sorprendentes si tenemos en cuenta que sus fundamentos teóricos son distintos. La Q de Newman y Girvan se basa en una definición simple de comunidad, una región de la red en la que hay una densidad de *links* inesperadamente alta. Sin embargo, el número de nodos dentro de cada comunidad no afecta al valor de Q [13]. Por el contrario, S evalúa tanto el número de *links* como de nodos en cada comunidad (ver Ecuación 3.1). Por tanto, S asume implícitamente una definición de comunidad mucho más compleja: un número preciso de unidades para las cuales se encuentra una densidad de *links* que es estadísticamente inesperada dadas las características de la red entera. En este contexto de comparación entre ambas medidas es también muy significativo que, mientras que algunos de los algoritmos utilizados en este estudio fueron diseñados para maximizar Q, ninguno de ellos trataba de maximizar S. Por lo tanto, nuestros resultados podrían estar sub-

estimando claramente la potencia de S para detectar comunidades. Un ejemplo claro de esta subestimación se aprecia en la Figura 3.4e: los valores máximos de S en algunos casos no son detectados. Las pocas excepciones donde $\mathrm{NMI_Q} > \mathrm{NMI_S}$ podrían ser explicadas también por este motivo, algunos algoritmos maximizan Q mientras que no hay ninguno de los que hemos probado que maximice específicamente S.

Los *benchmarks* abiertos, comúnmente utilizados, son útiles para evaluar de forma general el funcionamiento de diferentes algoritmos, pero no permiten establecer cómo de óptimos son los resultados obtenidos. Por eso hemos diseñado los *benchmarks* cerrados, en los cuales una estructura de comunidades inicial es progresivamente transformada en una segunda, también conocida, estructura de comunidades. Teniendo en cuenta que ambas estructuras de comunidad son idénticas, se puede demostrar que, en cualquier punto de la conversión de una la otra, la media de las NMIs de la solución encontrada respecto a las estructuras inicial y final debería aproximarse a un valor constante ($\frac{1+NMI_{IF}}{2}$), si esta solución es óptima (ver Métodos). Esta propiedad nos permite establecer la calidad intrínseca de la partición obtenida. A menudo la maximización de S proporciona soluciones óptimas, y en las que no, es capaz de aproximarse a la estructura de comunidades de la red con una precisión muy alta. A partir de este momento se abren dos líneas de investigación obvias. Primero, la generación de algoritmos nuevos que maximicen específicamente el parámetro Surprise. Segundo, la construcción de un conjunto de *benchmarks* cerrados estándar, que pueda ser utilizado para probar cualquier algoritmo futuro.

Cuando la maximización de S se aplica a redes reales obtenemos dos tipos de resultados. Por un lado, en las redes CYC2008 y *College football network*, esperábamos encontrar una clara división en comunidades que correspondiese a los complejos proteicos (CYC2008) y a las conferencias a las cuales pertenecían los equipos (*College football network*), dado que las conexiones intra-complejos e intra-conferencias son muy abundantes (e.g. Fig 5c). Y esos son los resultados obtenidos por S_{max}. Por otro lado, la estructura del club de karate Zachary está lejos de ser obvia (e.g. Figura 3.5). Por tanto encontrar que, de acuerdo con S_{max}, la red contiene algunas comunidades pequeñas y aparte muchos nodos aislados es, al menos a priori, no tan inesperado. Una pregunta obvia sería entonces por qué la comunidad científica ha centrado todos sus esfuerzos en diseñar algoritmos y medidas para establecer obligatoriamente dos comunidades en esta red (e.g., [15, 13, 52, 53] entre otras muchas). Esto puede ser causa de un *bias* psicológico, al cual el uso de algoritmos mediocres parece haber contribuido. Este hecho muestra hasta qué punto los prejuicios humanos pueden contaminar los análisis en este tipo de problemas pobremente definidos.

3.4. Métodos

Algoritmos utilizados para maximizar S y Q

En este estudio utilizamos seis de los mejores algoritmos disponibles, seleccionados bien por su excepcional funcionamiento en *benchmarks* artificiales o bien por su éxito en análisis previos sobre redes reales [30, 47, 48, 28, 34, 36]. Fueron los siguientes: 1) UVCluster [30, 47]: realiza *iterative hierarchical clustering*, generando dendrogramas. Los mejores valores de S y Q fueron obtenidos recorriendo estos dendrogramas de la raíz a las hojas. 2) SCluster [47]: también realiza *iterative hierarchical clustering*, pero en este caso utilizando una estrategia alternativa que es más rápida y en muchos casos mas acertada que la implementada en UVCluster. 3) Algoritmo dinámico de Rosvall y Bergstrom [20]: un algoritmo que formula el problema de la detección de comunidades como un problema de compresión de la información. 4) Algoritmo multiresolución basado en *Potts Model* [19]: trata de minimizar el Hamiltoniano de un Modelo de Potts a diferentes escalas de resolución, es decir, buscando comunidades de distintos tamaños. 5) *Fast Modularity optimization* [50]: diseñado para maximizar Q. Devuelve múltiples particiones, de las cuales se seleccionaron los mejores valores de Q y S para nuestro análisis. 6) *Extremal Optimization* [51]: Un algoritmo divisivo también diseñado para maximizar Q. Todos los análisis fueron desarrollados con los parámetros por defecto de los algoritmos.

Características de los *benchmarks*

En primer lugar utilizamos los *benchmarks* LFR, que fueron desarrollados recientemente para probar algoritmos de detección de comunidades [24]. En particular, seleccionamos cuatro configuraciones estándar de estos *benchmarks* ya estudiadas por otros autores [48]. Las redes analizas tenían 1000 ó 5000 unidades y fueron construidas con dos configuraciones distintas de tamaños de comunidad (Big (B): 20-100 nodos/comunidad; Small (S): 10-50 nodos/comunidad). Para cada una de las cuatro combinaciones (1000B, 1000S, 5000B, 5000S), generamos 100 redes para cada valor del *mixing parameter* μ, que variaba entre 0.1 y 0.9 [48]. μ es el porcentaje medio de *links* de un nodo que lo conectan con nodos de otras comunidades. Lógicamente, incrementar μ debilita la estructura de comunidades de la red. Cuando $\mu = 0{,}9$ las redes son quasi-random, como se comentará luego.

Cuando descubrimos que estos *benchmarks* LFR generaban redes cuyas comunidades tenían tamaños muy similares, decidimos implementar *benchmarks* RC donde las comunidades son mucho mas variables. Todas las redes de estos *benchmarks* fueron creadas con 512 nodos divididos en 16 comunidades. Generamos cien redes con tamaños de comunidades aleatorios, determinados usando un modelo de *broken-stick*

[54]. Este modelo proporciona comunidades con tamaños altamente heterogéneos. Más tarde, degradamos la estructura de dichas redes de una forma similar al efecto que produce incrementar μ en los *benchmarks* LFR. Inicialmente, en un *benchmark* RC todos los nodos de cada comunidad son subgrafos completamente conectados (i.e., *cliques*). A continuación, esta obvia estructura de comunidades se va haciendo cada vez más difusa, primero eliminando un porcentaje de *links* y después aleatorizando el mismo porcentaje de *links* de entre los restantes. A este porcentaje común lo llamamos *Degradation* (D). Cuando D=10 % significa que un 10 % de los *links* originales han sido eliminados del grafo y un 10 % de los restantes han sido aleatorizados, es decir, eliminados y vueltos a añadir entre dos nodos al azar.

En los *benchmarks* LFR y RC descritos aquí es posible comparar redes con una estructura de comunidades obvia (generadas con valores bajos de D o μ) con otras cada vez más aleatorias cuando se incrementaban esos parámetros. A este tipo de *benchmarks* les hemos llamado abiertos ("open"). Además, hemos creado un nuevo tipo de *benchmarks* que denominamos cerrados ("closed"). En ellos, los *links* se barajean de una manera dirigida, para convertir la estructura original en una segunda estructura también conocida. De este modo, es posible monitorizar cuándo la estructura original es sustituida por la final de acuerdo con las soluciones proporcionadas por S_{max} y Q_{max}. En los *benchmarks* LFR y RC cerrados, las redes iniciales fueron las mismas descritas en los párrafos anteriores, con $\mu = 0{,}1$ (LFR) o $D = 0$ (RC) respectivamente, y las redes finales fueron obtenidas reetiquetando los nodos aleatoriamente. Por tanto, las redes inicial y final tienen la misma estructura de comunidades, aunque los nodos dentro de cada comunidad son diferentes. Definimos el parámetro *Conversion* (C) como el porcentaje de *links* exclusivos de la red inicial que son sustituidos por *links* presentes sólo en la red final (i.e., C=0: estructura inicial presente; C=100: estructura final presente).

Simetría de NMI como medida de calidad en *benchmarks* cerrados

En nuestros *benchmarks* cerrados esperamos un comportamiento simétrico de los valores de NMI respecto a las particiones inicial y final. Imaginemos que estimamos una partición de acuerdo a un cierto criterio. Consideremos ahora la siguiente desigualdad triangular:

$$NMI_{IE} + NMI_{EF} \leq 1 + NMI_{IF} \tag{3.2}$$

donde $\mathrm{NMI_{IE}}$ es la *Normalized Mutual Information* calculada entre la estructura inicial (I) y la partición estimada (E), $\mathrm{NMI_{EF}}$ es la *Normalized Mutual Information*

entre la partición final (F) y la estimada y, por último, $\mathrm{NMI_{IF}}$ es la *Normalized Mutual Information* entre las particiones inicial y final. La Ecuación 3.2 es cierta si las estructuras de I, F y E son idénticas (i.e., tanto el número como el tamaño de comunidades es igual, aunque no necesariamente los nodos dentro de cada comunidad tienen que ser los mismos). Esto se extrae del hecho que:

$$1 - NMI_{XY} = \frac{VI_{XY}}{H(X) + H(Y)} \tag{3.3}$$

donde $\mathrm{VI_{XY}}$ es la *Variation of Information* de las particiones X e Y [29] y $H(X)$ y $H(Y)$ son las entropías de X e Y respectivamente. Dado que VI es una métrica [29], satisface la desigualdad triangular

$$VI_{AB} + VI_{BC} \geq VI_{AC} \tag{3.4}$$

Si, como se indica, las estructuras de todas las particiones son idénticas, todas sus entropías tienen el mismo valor. En ese caso, la siguiente desigualdad se puede deducir de Ecuación 3.3 y Ecuación 3.4:

$$(1 - NMI_{AB}) + (1 - NMI_{BC}) \geq (1 - NMI_{AC}) \tag{3.5}$$

A partir de esta desigualdad, y sustituyendo A, B y C por I, E y F respectivamente, podemos deducir la Ecuación 3.2. Dicha fórmula, por tanto, significa que la media de $\mathrm{NMI_{IE}}$ y $\mathrm{NMI_{EF}}$ puede tomar un valor máximo de $\frac{1+NMI_{IF}}{2}$. La Ecuación 3.2 se mantendrá aproximadamente cierta si las entropías de I, E y F son muy similares (i.e., muchas comunidades idénticas). En nuestros *benchmarks* cerrados las estructuras I y F son idénticas y convertimos la primera en la segunda progresivamente. Por tanto esperamos que la partición óptima a lo largo de esta conversión sea similar a ambas. De ahí que desviaciones del valor medio esperado $\frac{1+NMI_{IF}}{2}$, sean preocupantes, ya que probablemente significan que la partición óptima no ha sido encontrada. Por otro lado, encontrar valores iguales a $\frac{1+NMI_{IF}}{2}$ es un indicador muy fuerte de que sí que ha sido encontrada.

Es importante señalar que, aunque NMI ha sido la medida más utilizada para esta finalidad en este campo [48, 24, 28], utilizar la *Variation of Information* tiene importantes ventajas en los *benchmarks* cerrados: Ecuación 3.4 puede ser utilizada en vez de Ecuación 3.2, sin necesidad de tener en cuenta las entropías de las particiones.

Redes reales

Dos de las tres redes exploradas, *College football* y *Zachary's Karate networks*, han sido ampliamente utilizadas en multitud de trabajos en el campo de detección de comunidades (e.g., refs [15, 13, 52, 53, 55]). La tercera red se obtuvo a partir de la base de datos de complejos proteicos CYC2008 [45]. Esta base de datos contiene información sobre 408 complejos proteicos de la levadura Saccharomyces Cerevisiae. Como había proteínas que aparecían en varios complejos, decidimos asignar cada una de ellas al complejo de mayor tamaño del que fuese parte y así evitar solapamientos. Este paso, mediante el cual el número de complejos se redujo a 324, era necesario para poder calcular correctamente la NMI. Una vez cada proteína estaba asignada a un cluster no-solapante, descargamos de BioGRID [46] las interacciones proteína-proteína caracterizadas hasta el momento para esas proteínas. El grafo final contenía 1604 nodos (proteínas) y 14171 conexiones (interacciones).

4 *Closed benchmarks:* Una novedosa y potente herramienta para testar algoritmos de detección de comunidades

4.1. Introducción

Durante los últimos años, multitud de métodos han sido propuestos para extraer la partición óptima en comunidades de una red. Mientras que algunas de estas estrategias tratan de maximizar una función global de calidad como su Modularity (Q) [15] o Surprise (S) [16], otras buscan la partición óptima minimizando la compresión de la información que mejor describe la red [20], minimizando el Hamiltoniano de un modelo de Potts que representa el grafo [19] o deducen el modelo de máxima-verosimilitud que mejor representa la estructura de la red [18], por nombrar sólo algunos ejemplos. Sin embargo, ninguno de estos algoritmos consigue los mejores resultados en todas las situaciones. Su funcionamiento varía enormemente dependiendo de la topología de la red analizada [48, 16].

Han sido numerosos los *benchmarks* que se han propuesto para comparar el rendimiento de algoritmos de detección de comunidades. Los primeros estaban basados en un modelo conocido como *planted l-partition* [56]. El más popular entre ellos es el *benchmark* de Girvan y Newman (GN) [13], en el cual una red de 128 nodos se divide en 4 comunidades de igual tamaño, donde cada nodo está conectado con otros 16 nodos de su propia comunidad. Este grafo inicial puede ser degradado progresivamente reemplazando esos *links* intra-comunidad por *links* entre comunidades, manteniendo constante el grado medio de los nodos. De este modo, la estructura de comunidad es cada vez más difusa y, por tanto, más complicada de recuperar para un algoritmo. Los *benchmarks* Relaxed Caveman (RC) [44, 47, 16] se basan en un concepto similar. En ellos la red inicial está compuesta por un conjunto de *cliques* de distintos tamaños y se realiza un proceso de degradación idéntico al descrito en el *benchmark* GN. Nótese que las comunidades las comunidades GN y RC son, por definición, subgrafos Erdős-Rényi [8] en los cuales, a lo largo del proceso de degradación, cada par de nodos está conectado con la misma probabilidad p. Esa característica hace que

estos benchmarks no sean muy apropiados para representar redes del mundo real, ya que es bien conocido que este tipo de redes muestran distribuciones de los grados de los nodos mucho más heterogéneas [57, 58]. Con esta idea en mente, Lancichinetti, Fortunato y Radicchi desarrollaron un nuevo tipo de *benchmarks*, llamado LFR [24], en el cual tanto los tamaños de las comunidades como la distribución del grado de los nodos se pueden ajustar a una ley de potencia. En los *benchmarks* LFR, la fracción de *links* μ que un nodo comparte con nodos de otras comunidades se puede modificar. Aumentar μ (denominado por los autores *mixing parameter*) produce un comportamiento análogo al proceso de degradación descrito para los *benchmarks* GN y RC. Es decir, la proporción de *links* inter-comunitarios crece y las comunidades originales desaparecen gradualmente. A todos estos *benchmarks* (GN, RC, LFR), les hemos llamado "*open*", dado que el resultado final de la degradación es abierto (i.e., la estructura de comunidades final de la red es indeterminada, tendiendo a un grafo aleatorio).

En este artículo, describimos en detalle un nuevo tipo de *benchmark*, llamados "*closed*", basado en la conversión de una red con estructura de comunidades conocida en otra red cuyas comunidades son también conocidas. Ya introdujimos el concepto de benchmark cerrado en un trabajo anterior [16] y mostramos cómo este tipo de benchmark puede ser exitósamente utilizado para comparar algoritmos de detección de comunidades. Aquí, explicamos su funcionamiento, presentamos algunos ejemplos y discutimos su potencial y las ventajas que estos *benchmarks* presentan sobre los *open benchmarks*. Además mostramos que la evolución dirigida de la red hacia un final cerrado permite monitorizar con exactitud el progreso de esta transformación y evaluar la calidad de una partición en cualquier momento del proceso.

4.2. Características de los *closed benchmarks*

El concepto principal detrás de los *closed benchmarks* es la transformación dirigida de una red en otra por medio del barajeo de *links*. El punto de partida es una red cuya estructura de comunidades es conocida *a priori*. Cualquier tipo de grafo y estructura de comunidades es válida como red inicial. El algoritmo entonces genera la red "final". Las redes inicial y final están estrictamente relacionadas. La estructura de comunidades de la red final es idéntica a la de la inicial, pero las etiquetas de los nodos se han aleatorizado. Para convertir la red inicial en la final es necesario la reconexión de los *links* de una manera dirigida, un proceso representado en la Figura 4.1. Los detalles del procedimiento son los siguientes:

1. Los *links* presentes tanto en la red inicial como en la final no serán reconectados con ningún otro nodo.

2. En cada paso, uno de los *links* "reconectables" es eliminado y aleatoriamente añadido entre dos nodos que estén conectados sólo en la red final. El parámetro

Conversion (C) se define como el porcentaje de *links* reconectables que han sido modificados en un punto del proceso de transformación de una red en otra.

3. La red puede ser guardada para análisis posteriores en cualquier momento de este proceso de conversión. Por tanto, se puede obtener un gran conjunto de estructuras intermedias entre las redes inicial y final en el que se pueden testar algoritmos de detección de comunidades.

4. El proceso termina cuando se llega a la estructura final.

Una característica significativa de los *closed benchmarks* es que, durante el proceso de conversión y a causa del reconexionado dirigido de los *links*, nos acercamos a la estructura final a la misma velocidad que nos alejamos de la inicial. Si llamamos D a la distancia entre ambas redes, podemos afirmar que la estructura a una distancia x del inicio estará también a una distancia $D - x$ del final del benchmark. Este hecho, junto con la idéntica topología de ambos extremos, produce un conjunto de estructuras que es simétrico respecto al 50 % del proceso de transformación. Es decir, cuando $C = 50\,\%$, la estructura de la red está, en promedio, a la misma distancia tanto de la red inicial como de la final. Dados estos patrones de la evolución de la red, podemos asumir que su estructura de comunidades sufre un comportamiento similar. Como veremos más tarde, este comportamiento es crucial para la evaluación de particiones en los *closed benchmarks*.

Cualquier *benchmark* lleva asociada una o varias medidas de rendimiento. En el caso de comparaciones entre *clusterings*, se han desarrollado numerosos métodos, basados en el conteo de pares, *cluster matching* o índices basados en Teoría de la Información (discutidos en [59, 14]). Entre los del último tipo, la Variation of Information (VI) [29] es una distancia útil para medir la disimilitud entre dos particiones, A y B ($\mathrm{VI_{AB}}$). En nuestro contexto, consideramos que esta medida tiene claras ventajas sobre otras, principalmente debido a su naturaleza métrica. Esto implica que VI es positiva definida, una distancia simétrica -que es una propiedad altamente deseable cuando se comparan *clusterings*- y, más importante para nuestros propósitos, satisface la desigualdad triangular [29]. Este hecho resulta ser muy útil para la evaluación en los *closed benchmarks*. En estos *benchmarks*, tenemos dos estructuras de comunidades conocidas, las redes inicial (I) y final (F). Además, el método genera un conjunto de estructuras intermedias estimadas (E) cuyas comunidades también se pueden determinar. Podemos deducir de la desigualdad triangular de VI la siguiente fórmula:

$$VI_{IE} + VI_{EF} \geq VI_{IF} \tag{4.1}$$

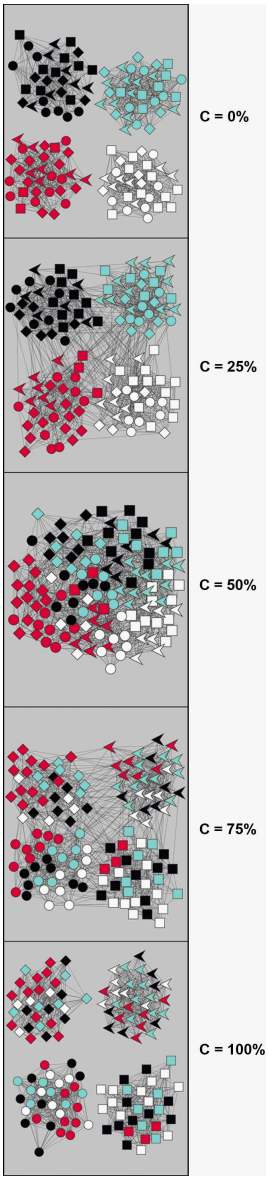

Figura 4.1: Proceso de conversión en un closed benchmark. En este caso la red de partida es el *benchmark* GN. Los *links* son reconexionados progresivamente de la red inicial ($C = 0\,\%$) a la final ($C = 100\,\%$). El color de los nodos viene definido por la comunidad inicial a la que pertenecen, mientras que su forma corresponde a la comunidad final en la que están contenidos.

Por tanto, la suma de $\mathrm{VI_{IE}}$ y $\mathrm{VI_{EF}}$ está acotada inferiormente por $\mathrm{VI_{IF}}$, que es constante, dado que las particiones de las redes inicial y final son fijas. Si el reconexionado de la red todavía no ha comenzado, la partición estimada es la misma que la inicial, $I = E$, y de ahí $VI_{IE} = 0$ y $VI_{EF} = VI_{IF}$, lo cual satisface la igualdad en la Ecuación 4.1. Cuando la conversión comienza, y dado que la red se aproxima a la estructura final a la misma velocidad que se aleja de la inicial, $\mathrm{VI_{IE}}$ debería aumentar tanto como $\mathrm{VI_{EF}}$ disminuya. Por tanto, a menos que la estructura de la red llegue a ser muy diferente de las redes inicial y final a lo largo del proceso de conversión (e.g., como se describe en el siguiente párrafo), esto debería hacer que la igualdad $VI_{IE} + VI_{EF} = VI_{IF}$ sea cierta en todo momento. Una deducción importante es que si, para una partición estimada (E) dada, la suma de $\mathrm{VI_{IE}}$ y $\mathrm{VI_{EF}}$ se aleja del valor constante $\mathrm{VI_{IF}}$, entonces E puede no ser la partición óptima [16]. De ahí que desviaciones del valor esperado $\mathrm{VI_{IF}}$ puedan indicar comportamientos subóptimos de un determinado algoritmo.

Si estructuras ajenas, muy distintas de la inicial y final se forman durante el proceso de conversión, podemos encontrar $VI_{IE} + VI_{EF} > VI_{IF}$ incluso si la partición es óptima. Se puede entender fácilmente si asumimos que la estructura intermedia se vuelve completamente aleatoria. En ese momento, se pueden dar dos situaciones, dependiendo de la densidad de *links* en el grafo. Si, en un determinado punto del reconexionado, la estructura intermedia llega a ser una sola comunidad que contiene todos los nodos -como se espera en un grafo alea-

torio con una densidad muy alta de *links*- entonces, $VI_{IE} = H(I)$ y $VI_{EF} = H(F)$, donde $H(I)$ y $H(F)$ son las entropías de las particiones inicial y final. Dado que $VI_{IF} = H(I) + H(F) - 2M(I, F)$, donde $M(I, F)$es la Información Mutua entre las particiones inicial y final, tenemos que $VI_{IE} + VI_{EF}$ debe ser mayor que VI_{IF}. Esto deriva del hecho que $M(I, F) = 0$ si y sólo si I y F son independientes, que no es el caso aquí. Por otro lado, si la densidad de links es baja y la red se aleatoriza, la estructura de comunidades puede aproximarse a una situación en la cual cada nodo está aislado en una comunidad distinta. Si esto es cierto, se puede demostrar que $VI_{IE} = log\, N - H(I)$ y $VI_{EF} = log\, N - H(F)$, donde N es el número total de nodos. En este caso, tendríamos que $VI_{IE} + VI_{EF} \gg VI_{IF}$. Por tanto, si un algoritmo está funcionando perfectamente ($VI_{IE} + VI_{EF} = VI_{IF}$) hasta un cierto punto de la conversión y cuando avanza la conversión encontramos que $VI_{IE} + VI_{EF} > VI_{IF}$, puede deberse a dos razones. Primero, un mal funcionamiento del algoritmo que es incapaz de definir correctamente las comunidades. O segundo, la emergencia de estructuras de comunidades ajenas y potencialmente aleatorias. Esta interesante situación será ilustrada en un caso particular más tarde.

4.3. Tests

4.3.1. Configuración

Como se menciona anteriormente, las características particulares de una red puede influenciar en gran medida la capacidad de un algoritmo dado para detectar su estructura de comunidades. Por este motivo, realizamos pruebas en redes artificiales que variaban en tamaño, distribución del grado de los nodos, número de comunidades y tamaño de éstas. Este último parámetro ha demostrado ser crucial en la detección de comunidades [16, 48]. Existen dos razones principales para explicar el efecto de la variación del tamaño de comunidades. Primero, las redes con una distribución de tamaños de comunidades muy asimétrica se degradan más rápidamente que aquellas con comunidades de igual tamaño, a causa de la rápida destrucción de los *clusters* pequeños. Y segundo, una distribución asimétrica afecta enormemente al funcionamiento de determinados algoritmos. Por ejemplo, cualquier algoritmo que maximice una medida popular de detección de comunidades como es la Modularity (Q) de Newman y Girvan, tendrá problemas para detectar las comunidades pequeñas, ya que Q se ve afectada un límite de resolución [23].

Una forma sencilla de medir y comparar la distribución de tamaños de comunidad es usar el Pielou's Index (PI), que cuantifica cómo de similares son los grupos en los que se divide un sistema. Este índice toma un valor de 1 si todos los grupos son del mismo tamaño y decrece cuando aumenta la varianza en los tamaños [49]. En este estudio, elegimos como puntos de partida cuatro redes artificiales con diferentes valores de PI que corresponden a cuatro open benchmarks ya publicados. Los

nombraremos de acuerdo a la siguiente convención: 1) Girvan-Newman (GN) [13]: Mencionado anteriormente. Una red de 128 nodos se divide en cuatro comunidades de igual tamaño ($PI = 1$). Los nodos se conectan exclusivamente con miembros de su comunidad con un grado medio de 16; 2) Lancichinetti-Fortunato-Radicchi, con comunidades pequeñas (LFR_S) [24, 48]: Una red de 5000 nodos. El grado medio de los nodos es 20, su grado máximo 50, el exponente de la distribución de grados es -2 y el exponente de la distribución de tamaños de comunidades es -1. Los tamaños de las comunidades varían entre 10 y 50 nodos (de ahí el nombre *"small communities"*). Entre las muchas redes que pueden ser generadas con estos parámetros, nosotros escogimos una al azar que contiene 195 comunidades de tamaños similares ($PI = 0{,}98$); 3) Relaxed Caveman con Pielou's Index = 0.75 (RC75). Ya que era necesaria una distribución de tallas de comunidades más asimétrica para analizar el comportamiento de los algoritmos en redes de todo tipo, generamos una red de 512 nodos con $PI = 0{,}75$, que corresponde a una división en 16 comunidades, donde cada una de ellas incluye de 2 a 196 nodos. En la configuración RC75, la red inicial consistía en comunidades desconectadas entre sí, cada una de ellas formando un *clique*; y 4) Relaxed Caveman $PI = 0{,}50$ (RC50): Tiene todavía una mayor variación en los tamaños de comunidades. La red inicial también constaba de 512 nodos divididos en 16 cliques, pero ahora el mayor de todos contenía 354 nodos. En la Figura 4.2 podemos ver gráficamente los patrones de las conexiones de cada red inicial y la variación de tallas de comunidad. Una vez obtenidas, se modificaban progresivamente incrementando C, resultando finalmente de cada una de ellas un conjunto de 101 redes, cubriendo todo el rango desde C=0 (estructura inicial presente) a C=100 (estructura final presente). Los correspondientes open benchmarks, con las mismas estructuras de de comunidades iniciales y degradándose aleatoriamente también fueron analizadas, siguiendo métodos estándar descritos en publicaciones anteriores (e.g., [13, 44, 24]). También describimos más tarde algunos *closed benchmarks* con estructuras iniciales aleatorias.

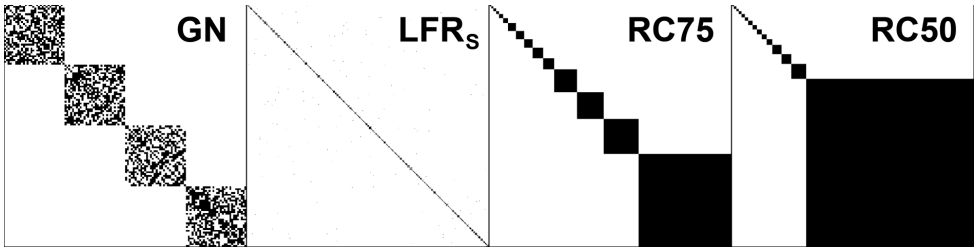

Figura 4.2: Representación gráfica de las matrices de adyacencia de las cuatro redes iniciales utilizadas en los tests. Los nodos están ordenados de acuerdo a las comunidades a las que pertenecen. Las diferencias en los tamaños relativos de las comunidades son evidentes.

4.3.2. Algoritmos

Se utilizaron dos algoritmos de detección de comunidades que han mostrado resultados excelentes en estudios recientes, Infomap [20] y SCluster [47]. Infomap entiende la búsqueda de la estructura de comunidades como un problema de compresión de la información, detectando comunidades al comprimir la topología de la red. Ha conseguido excelentes resultados en los *benchmarks* LFR [16, 48]. Por otro lado, SCluster utiliza una estrategia completamente distinta. Utilizando *clustering* jerárquico, el algoritmo calcula las distancias entre pares de nodos a partir de soluciones parciales, mediante *iterative hierarchical clustering* [47, 30]. Después construye un árbol jerárquico del que extrae la partición con Surprise [16] máxima como solución óptima. Surprise es una función de calidad que estima la bondad de una partición basándose en la comparación entre el grafo y el modelo nulo generado por una distribución aleatoria de *links* [16, 30]. SCluster ha demostrado que es capaz de extraer particiones de alta calidad cuando trabaja con redes cuyas comunidades son de tamaños muy distintos [16, 47]. Además, como una tercera manera de detectar la mejor partición de una red, seleccionamos de entre las soluciones de Infomap y SCluster aquella con mayor Surprise, dado que se ha demostrado que la maximización de Surprise no sólo mejora cualitativamente sobre la Modularity Q, sino que además supera las soluciones generadas por cualquier algoritmo en solitario [16].

4.3.3. Resultados

La Figura 4.3 ilustra los resultados de los tres métodos en nuestros cuatro *closed benchmarks*. Cada partición estimada a lo largo del proceso de conversión es comparada, utilizando la Variation of Information (VI) tanto con la estructura de comunidades inicial (círculos negros) como con la final (cuadrados rojos). $VI = 0$ significa que las particiones comparadas son exactamente la misma. Anteriormente hemos mencionado cómo la suma de la Variation of Information de un punto estimado a las particiones inicial y final ($VI_{IE} + VI_{EF}$) debería óptimamente ser constante e igual a la VI entre la partición inicial y la final (VI_{IF}). Para una mejor visualización, mostramos la mitad de esta suma ($\overline{V} = [VI_{IE} + VI_{EF}]/2$) como una línea discontinua. Si la partición es óptima, esperamos que $\overline{V} = VI_{IF}/2$.

Las gráficas muestran cuán diferente es el proceso de detección de comunidades dependiendo del algoritmo utilizado y de la topología de la red analizada. Cuando utilizamos la red GN como entrada, Infomap funciona muy bien (Figura 4.3a). La VI entre la partición final y la estimada ($\mathrm{VI_{IE}}$, círculos negros) es cero a lo largo de la primera mita del *benchmark*. Además, cuando la conversión rompe el 50 %, la VI entre la partición estimada y la final ($\mathrm{VI_{EF}}$, cuadrados grises) se comporta de la misma manera. Es decir, el algoritmo reconoce la estructura inicial hasta $C = 49\,\%$ y la final por encima de $C = 51\,\%$. Esto no ocurre al aplicar SCluster (Figura 4.3e),

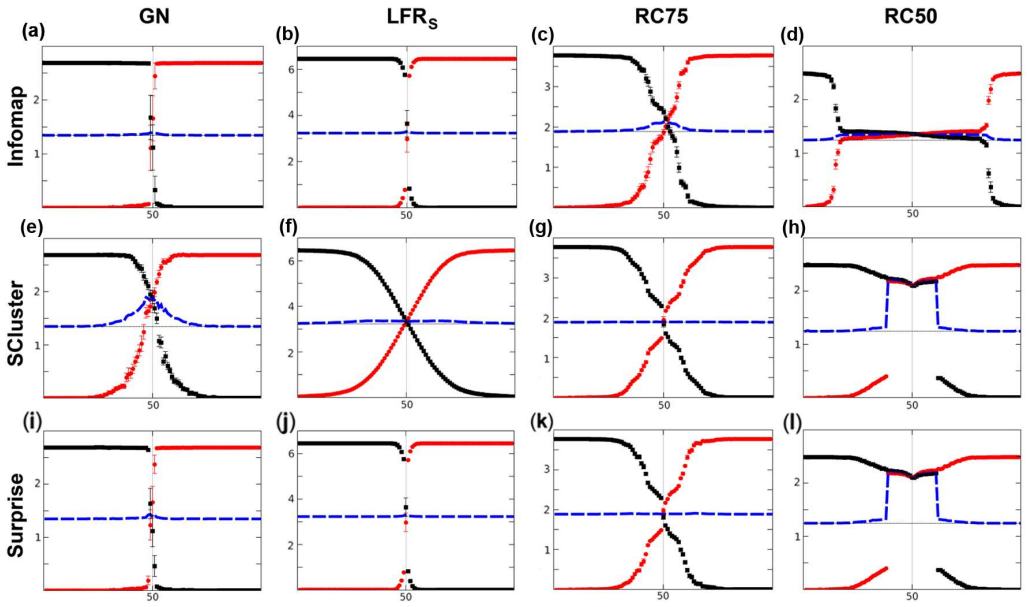

Figura 4.3: Comportamiento de la Variation of Information en los cuatro benchmarks. Los círculos negros representan el valor de VI entre la partición inicial y las estimadas ($\mathrm{VI_{IE}}$). Los cuadrados rojos la VI entre las particiones estimadas y la final ($\mathrm{VI_{EF}}$). $\overline{V}$ aparece como una línea discontinua azul, que debe seguir una recta si el funcionamiento del algoritmo es óptimo durante el proceso de conversión completo (i.e., $VI_{IE} + VI_{EF} = VI_{IF}$).

que sólo reconoce la partición inicial hasta $C = 30\,\%$ y empieza a reconocer la estructura final a partir de $C = 70\,\%$. Como cabía esperar, $\overline{V}$ muestra gráficamente esta diferencia en el funcionamiento de los algoritmos. Mientras que en la gráfica de Infomap $\overline{V}$ cae en una línea recta casi perfecta, coincidiendo con $\mathrm{VI_{IF}}/2$, las particiones estimadas por SCluster producen una desviación importante de la línea en el intervalo 30-70 %, donde ya habíamos detectado que las comunidades estaban erróneamente estimadas.

Cuando el punto de partida del *benchmark* es la red $\mathrm{LFR_S}$, Infomap produce de nuevo una gráfica simétrica, donde $\overline{V}$ concuerda casi perfectamente con $VI_{IF}/2$ (Figura 4.3b). SCluster también presenta un comportamiento simétrico, aunque se desvía ligeramente de los valores óptimos (Figura 4.3f), i.e., de nuevo funcionando peor que Infomap. En estos dos ejemplos, las tallas de las comunidades eran iguales o muy similares ($PI \approx 1$) y se espera que se degraden, en promedio, a la misma velocidad. La partición original por tanto, está presente durante la primera mitad de la conversión ($VI_{IE} \approx 0$) y entonces, la estructura de comunidades pasa a ser la final instantáneamente ($VI_{EF} \approx 0$). Por otro lado, cuando analizamos redes donde la distribución de tamaños de comunidad es altamente asimétrica (RC75, RC50), el funcionamiento de los algoritmos cambia radicalmente. En las pruebas en RC75, Infomap ya no muestra un comportamiento simétrico (Figura 4.3c), ya que $\overline{V} >$

$VI_{IF}/2$ cuando $C = 40 - 60\,\%$. Por el contrario, SCluster sí que presenta un patrón simétrico con $\overline{V} = VI_{IF}/2$ (Figura 4.3g). Podemos ver cómo la VI entre la partición inicial y la estimada ($\mathrm{VI_{IE}}$, círculos negros) es igual a cero hasta alrededor del 30 % y entonces comienza a incrementarse. Cabe destacar que, en un open benchmark (e.g., refs [13, 44, 24]) esta sería la única información de la que dispondríamos. Por tanto, podríamos pensar que a partir de $C = 30\,\%$, el algoritmo es incapaz de reconocer la partición óptima.Sin embargo, un mal funcionamiento del algoritmo no es la única explicación para estos patrones. Alternativamente, es posible que la partición inicial ya no sea la óptima porque la estructura de comunidades ha cambiado realmente. Los *closed benchmarks* ofrecen una forma sólida de saber si esta hipótesis es correcta. En los paneles c, g y k de la Figura 4.3 podemos ver que, aunque $\mathrm{VI_{IE}}$ empieza a crecer rápido, $\mathrm{VI_{EF}}$ decrece a la misma velocidad. Es decir, que la estructura de comunidades de la partición inicial se está transformando hacia la final mucho antes del punto $C = 50\,\%$, un patrón que es debido a la rápida destrucción de las comunidades pequeñas, típico de *benchmarks* con bajo PI. Este comportamiento es imposible de demostrar en ninguno de los *benchmarks* publicados anteriormente, aunque es crítico para la evaluación de los algoritmos. Ahora, podemos afirmar que el comportamiento de SCluster es óptimo, dado que $\overline{V}$ sigue una línea recta: satisface la igualdad en Ecuación 4.1 durante todo el proceso de conversión. En el último caso, RC50, el resultado de los algoritmos es algo distinto que el del resto de *benchmarks*. Infomap parece colapsar rápidamente, con $\overline{V}$ alejándose de la recta óptima en seguida, cuando $C \geq 10 - 12\,\%$ (Figura 4.3d). En el caso de SCluster (Figura 4.3h), los valores de $\overline{V}$ siguen en la recta un rato más (hasta $C = 30\,\%$ aproximadamente), pero luego el algoritmo comienza a detectar estructuras ajenas, lejos de las particiones inicial y final ($\overline{V} > VI_{IF}/2$). Estos comportamientos son debidos a la extrema asimetría de la distribución del tamaño de las comunidades, ya que existe un grupo enorme que domina la red (Figura 4.2d). Debido a esto, conforme avanza el proceso de conversión, se forma un grafo quasi-aleatorio. Infomap lo interpreta incluyendo la mayoría de los nodos de la red en una sola comunidad. Por tanto, como ya hemos discutido anteriormente, $\overline{V}$ se aproxima a $H(I)$ (que en este ejemplo toma un valor de 1.38). SCluster, por otro lado, divide la red en grupos muy pequeños, aislando la mayoría de nodos. Por eso, $\overline{V}$ alcanza valores mucho mayores que $VI_{IF}/2$, por las razones comentadas antes.

La Figura 4.3 3i-3l muestran la evolución de cada *benchmark* utilizando como partición estimada aquella con mayor Surprise entre las soluciones de Infomap y SCluster. Como era de esperar [16], esta estrategia siempre elige la mejor partición de las dos. La igualdad en Ecuación 4.1 se satisface durante todo el tiempo en las tres primeras redes. En el cuarto caso, el patrón es idéntico al producido por SCluster. Los valores de Surprise del *benchmark* RC50 sugieren que la interpretación de SCluster, de definir muchos *clusters* pequeños cuando existe una estructura intermedia *quasi-random* es preferible al sugerido por Infomap (dominado por un enorme *cluster*). Esto concuerda perfectamente con el hecho de que SCluster funciona mejor en este benchmark, como se ha indicado más arriba, principalmente en el rango

$C = 12 - 30\,\%$.

También es posible utilizar grafos aleatorios como redes de partida en un *closed benchmark*. La comparación con estos *benchmarks* basados en redes aleatorias puede contribuir a determinar si una red tiene estructura de comunidades significativa o no, un tema que ha suscitado bastante atención en los últimos tiempos [60, 61]. Para tratar este asunto, generamos cuatro tipos de grafos aleatorios, cada uno de ellos con el mismo número de nodos y *links* que los de las redes iniciales descritas anteriormente, pero distribuidos aleatoriamente. Dado que para generar un *closed benchmark* debemos asumir una estructura de comunidades *a priori*, elegimos la de mayor Surprise de entre las generadas por Infomap y SCluster. En la Figura 4.4 podemos ver los resultados de los análisis sobre los cuatro *benchmarks*. Como ocurría en el caso anterior, Infomap devuelve particiones cuyos nodos (Figura 4.4a-c) o al menos más del 90 % de ellos pertenecen a una sola comunidad (Figura 4.4d). El valor de $\overline{V}$ observado es la entropía de la partición inicial (o final) $H(I) = H(F)$, dado que si todos los nodos están en una única comunidad, $H(E) = 0$. Por otro lado, SCluster genera soluciones con un número muy alto de comunidades (Figura 4.4e-h), entendiendo que incluso un grafo aleatorio contiene un cierto grado de estructura de comunidades. En estos *benchmarks* de grafos aleatorios es interesante observar la degradación extremadamente rápida que sufren las particiones cuando sólo un 1 % de los *links* ha sido reconexionado (Figura 4.4). Si se comparan con sus redes análogas no-aleatorias, $\mathrm{VI_{IE}}$ aumenta instantáneamente, lo cual es un comportamiento esperado en redes con estructura de comunidades pobremente definida. Este tipo de comparaciones entre patrones de Variation of Information permite evaluar la robustez de una red, de una manera similar a la utilizada mediante otros métodos [60].

4.4. Discusión

El desarrollo de métodos que puedan detectar con exactitud estructura de comunidades en redes es crítico en muchos campos científicos, ya que pueden desvelar relaciones subyacentes entre los elementos de un sistema. Por tanto, es de gran importancia el comparar y evaluar esos métodos en un conjunto de *benchmarks* artificiales de modo que seleccionemos uno, o una combinación de ellos que pueda producir resultados fiables cuando se analizan redes del mundo real. En la literatura se han propuesto varios *benchmarks* estándar, la mayoría de ellos del tipo que nosotros llamamos *open*: empiezan con una red con estructura de comunidades bien definida y se va degradando mediante un reconexionado aleatorio de sus *links*. Durante este proceso, las comunidades desaparecen gradualmente hacia un *"open end"* donde la estructura de comunidades es indeterminada. Este tipo de *benchmarks* es útil para comparar el comportamiento relativo de los algoritmos pero inadecuado para evaluar su calidad intrínseca (i.e., si las soluciones obtenidas son óptimas o no).

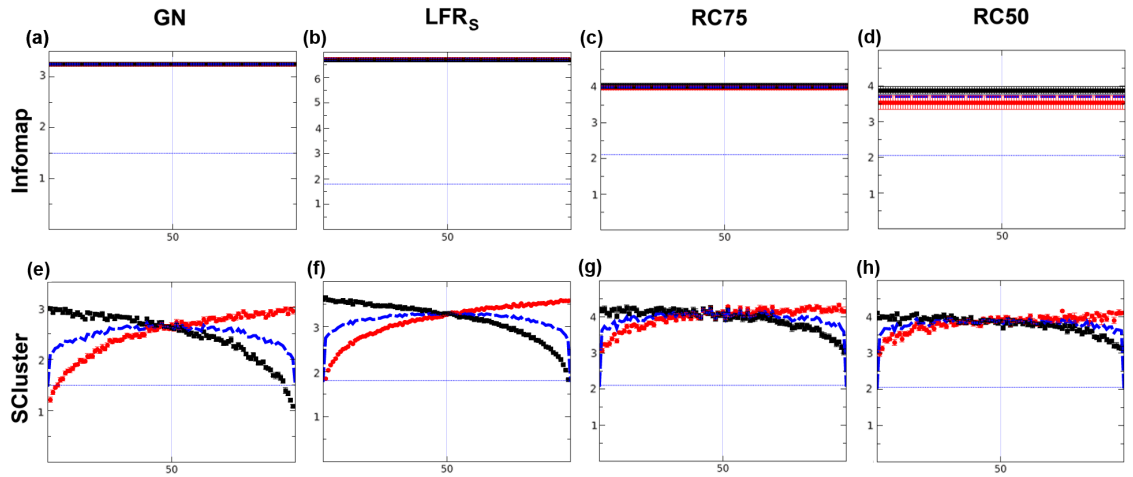

Figura 4.4: Redes aleatorias con el mismo número de nodos que los correspondientes benchmarks indicados en la parte superior. Como en Figura 4.3, la línea discontinua corresponde al valor $\overline{V}$, mientras que el rojo y negro corresponden a los valores $\mathrm{VI}_{\mathrm{EF}}$ y $\mathrm{VI}_{\mathrm{IE}}$ respectivamente. Los valores de $\mathrm{VI}_{\mathrm{EF}}$, $\mathrm{VI}_{\mathrm{IE}}$ y $\overline{V}$ coinciden ampliamente/completamente en los análisis de Infomap, apareciendo como una sola línea o líneas paralelas juntas. Nótese que tan pronto como comienza la conversión, $VI_{IE} + VI_{EF} \gg VI_{IF}$. Las diferencias entre Infomap y SCluster son debidas a la diferente manera que tienen de interpretar las estructuras aleatorias: como un sólo *cluster* (Infomap) o como muchos *clusters* individuales (SCluster).

En este artículo, hemos descrito en profundidad los closed benchmarks, que también degradan una red inicial con comunidades conocidas pero ahora evolucionando hacia una segunda red cuya estructura de comunidades también es conocida. Esta evolución se realiza mediante un reconexionado dirigido de los links de la red inicial a la final y permite controlar la progresión de la estructura entre ambas. También hemos mostrado que la Variation of Information proporciona información valiosa sobre la bondad de una partición y su posible optimalidad: la configuración de nuestros *closed benchmarks* nos permite acotar inferiormente la VI esperada utilizando la desigualdad triangular, que debe satisfacer al ser una métrica. Otra mejora relevante sobre los *open benchmarks* es el hecho de que cualquier red puede ser utilizada como entrada para el proceso de degradación, lo que permite realizar análisis sobre una gran variedad de topologías. Estas características claramente representan mejoras cualitativas sobre los *benchmarks* publicados hasta el momento.

Como hemos discutido, puede haber configuraciones con distribuciones de tamaños de comunidad altamente asimétricas, como la red RC50 (Figura 4.3), donde la igualdad de Ecuación 4.1 no se cumple durante todo el recorrido de la conversión. Sin embargo, este comportamiento en situaciones tan extremas no disminuye el poten-

cial de nuestra estrategia dado que, incluso entonces, hay varias condiciones que un buen algoritmo debe satisfacer. Primero, cuando el 50 % de los *links* se han reconexionado, $\mathrm{VI_{IE}}$ debe ser, en promedio, igual a $\mathrm{VI_{EF}}$. Segundo, la partición inicial debe ser reconocida mejor que la final durante la primera mitad del *benchmark* y, de ahí en adelante, el comportamiento debe ser exactamente el contrario. Tercero, un buen algoritmo producirá soluciones donde $VI_{IE} + VI_{EF} = VI_{IF}$ a lo largo de un mayor recorrido del proceso de conversión que un mal algoritmo. En resumen, las propiedades de los *closed benchmarks* hacen de ellos una herramienta valiosa para el desarrollo y evaluación de métodos computacionales que puedan caracterizar con exactitud la estructura de comunidades de una red.

5 Conclusión

Las primeras ideas del trabajo aquí expuesto comenzaron cuando me incorporé, en junio de 2009 al grupo del Dr. Ignacio Marín en el Instituto de Biomedicina de Valencia. A lo largo de este tiempo, esas ideas fueron madurando, convirtiéndose poco a poco en experimentos, análisis, datos, gráficas, figuras y finalmente en manuscritos y publicaciones. Este trabajo ha sido realizado íntegramente en el grupo del Dr. Marín y, en mi opinión, constituye una sólida línea de investigación que ha dado lugar a una serie de publicaciones de gran calidad científica y que ha aportado notables avances al campo de la estructura de comunidades, los cuales paso a resumir a continuación:

En primer lugar, generamos una serie de novedosos algoritmos de detección de comunidades, basados en *iterative hierarchical clustering* (estrategia ideada anteriormente por el grupo del Dr. Marín). Se encuentran implementados en una *suite* de programas llamada Jerarca, que ha sido liberada bajo dominio público. De entre estos algoritmos, cabe destacar especialmente SCluster, el cual ha demostrado ser uno de los mejores que se pueden encontrar en la literatura, sobre todo cuando tratamos con redes cuyas comunidades tienen tamaños muy distintos. Este hecho queda patente en subsiguientes artículos también descritos en este trabajo. En 2010 Jerarca fue publicado en *PLoS ONE* (factor de impacto: 4,411).

Segundo, proponemos un índice para evaluar la calidad de la estructura de comunidades de una red. Mediante una serie de extensos análisis en todo tipo de redes, conseguimos demostrar que esta medida, a la que llamamos Surprise, supera cualitativamente a la hasta ahora más conocida y utilizada medida para este fin, la Modularity de Newman y Girvan. Es por eso que esperamos que Surprise sustituya en poco tiempo a Modularity en la mayoría de estudios sobre detección de comunidades. El amplio y detallado análisis sobre Surprise y sus comparaciones con otros métodos dieron lugar a otra publicación en 2011, también en *PLoS ONE*.

Por último, nos dimos cuenta que era necesario crear un nuevo tipo de *benchmarks*, ya que la información que proporcionaban los que existían hasta ese momento era insuficiente para entender correctamente el comportamiento de los algoritmos y su habilidad definiendo comunidades. Tras semanas de discusión, conseguimos idear los *closed benchmarks*, que gracias a la naturaleza métrica de la Variation of Information nos permiten monitorizar la transformación de la estructura de comunidades de una red. Además, son capaces de obtener valiosa información sobre la optimalidad de

una partición dada. El manuscrito de los *closed benchmarks* acaba de ser aceptado (en enero de 2012) en la revista *Physical Review E* (factor de impacto: 2,352) y será publicado en breve.

Estas mejoras abren nuevas rutas en el campo de la estructura de comunidades. Por eso, ya tenemos en marcha varios proyectos continuando esta línea de trabajo, algunos de ellos muy avanzados. Esperamos finalizarlos exitosamente, que den lugar a nuevas publicaciones y que junto con lo aquí expuesto conformen una tesis doctoral en un futuro cercano.

Bibliografía

[1] B Bollobás. *Modern Graph Theory*, volume 184. Springer, 1998.

[2] Linton C Freeman. *The development of social network analysis*. Empirical Press, 2004.

[3] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[4] S Wasserman and K Faust. Social network analysis: methods and applications. *Cambridge University Press*.

[5] Reka Albert and Albert-Laszlo Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):78, 2001.

[6] Javier Borge-Holthoefer and Alex Arenas. Semantic networks: Structure and dynamics. *Entropy*, (5):1264–1302, 2010.

[7] Albert-László Barabási and Zoltán N Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[8] P Erdös and A Rényi. On random graphs. *Publ Math Debrecen*, 6(290-297):290–297, 1959.

[9] D J Watts and S H Strogatz. Collective dynamics of "small-world"networks. *Nature*, 393(6684):440–2, 1998.

[10] S H Strogatz. Exploring complex networks. *Nature*, 410(6825):268–76, 2001.

[11] M E J Newman. Power laws pareto distributions and zipf's laws. *Contemporary Physics*, 46:323–351, 2005.

[12] M E J Newman. Communities, modules and large-scale structure in networks. *Nature Physics*, 8(1):25–31, 2011.

[13] M Girvan and M E J Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, 2002.

[14] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[15] M E J Newman and M Girvan. Finding and evaluating community structure in networks. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 69(2 Pt 2):16, 2004.

[16] Rodrigo Aldecoa and Ignacio Marín. Deciphering network community structure by surprise. *PLoS ONE*, 6(9):8, 2011.

[17] Hua-Wei Shen and Xue-Qi Cheng. Spectral methods for the detection of network community structure: a comparative analysis. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(10):13.

[18] M E J Newman and E A Leicht. Mixture models and exploratory data analysis in networks. *Proc Natl Acad Sci USA*, 104(23):9564–9569, 2007.

[19] Peter Ronhovde and Zohar Nussinov. Multiresolution community detection for megascale networks by information-based replica correlations. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 80(1 Pt 2):016109, 2008.

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

[21] Mina Zarei, Keivan Aghababaei Samani, and Gholam Reza Omidi. Complex eigenvectors of network matrices give better insight into the community structure. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(10):P10018.

[22] S N Dorogovtsev and J F F Mendes. Evolution of networks. *Advances in Physics*, 51(4):1079–1187, 2002.

[23] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America*, 104(1):36–41, 2007.

[24] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 78(4 Pt 2):6, 2008.

[25] P Jaccard. Nouvelles recherches sur la distribution florale. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 44(223-270):223–270, 1908.

[26] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[27] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[28] Leon Danon, Jordi Duch, Albert Diaz-Guilera, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):10.

[29] M Meila. Comparing clusterings - an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007.

[30] Vicente Arnau, Sergio Mars, and Ignacio Marín. Iterative cluster analysis of protein interaction data. *Bioinformatics*, 21(3):364–378, 2005.

[31] A W Rives and T Galitski. Modular organization of cellular networks. *Proceedings of the National Academy of Sciences of the United States of America*, 100(3):1128–1133, 2003.

[32] Hongchao Lu, Xiaopeng Zhu, Haifeng Liu, Geir Skogerbø, Jingfen Zhang, Yong Zhang, Lun Cai, Yi Zhao, Shiwei Sun, Jingyi Xu, and et al. The interactome as a tree-an attempt to visualize the protein-protein interaction network in yeast. *Nucleic Acids Research*, 32(16):4804–4811, 2004.

[33] Andy M Yip and Steve Horvath. Gene network interconnectedness and the generalized topological overlap measure. *BMC Bioinformatics*, 8(1):22, 2007.

[34] J Ignasi Lucas, Vicente Arnau, and Ignacio Marín. Comparative genomics and protein domain graph analyses link ubiquitination and rna metabolism. *Journal of Molecular Biology*, 357(1):9–17, 2006.

[35] Antonio Marco and Ignacio Marín. A general strategy to determine the congruence between a hierarchical and a non-hierarchical classification. *BMC Bioinformatics*, 8(1471-2105):442, 2007.

[36] Antonio Marco and Ignacio Marín. Interactome and gene ontology provide congruent yet subtly different views of a eukaryotic cell. *BMC systems biology*, 3(1):69, 2009.

[37] R R Sokal and C D Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28(22):1409–1438, 1958.

[38] N Saitou and M Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[39] Masatoshi Nei and Sudhir Kumar. *Molecular Evolution and Phylogenetics*, volume 25. Oxford University Press, 2000.

[40] James S Farris. Estimating phylogenetic trees from distance matrices. *The American Naturalist*, 106(951), 1972.

[41] Aaron Clauset, M E J Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):1–6, 2004.

[42] Koichiro Tamura, Joel Dudley, Masatoshi Nei, and Sudhir Kumar. Mega4: Molecular evolutionary genetics analysis (mega) software version 4.0. *Molecular Biology and Evolution*, 24(8):1596–1599, 2007.

[43] Melissa S Cline, Michael Smoot, Ethan Cerami, Allan Kuchinsky, Nerius Landys, Chris Workman, Rowan Christmas, Iliana Avila-Campilo, Michael Creech, Benjamin Gross, and et al. Integration of biological networks and gene expression data using cytoscape. *Nature Protocols*, 2(10):2366–2382, 2007.

[44] Duncan J Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*, volume 107. Princeton University Press, 1999.

[45] Shuye Pu, Jessica Wong, Brian Turner, Emerson Cho, and Shoshana J Wodak. Up-to-date catalogues of yeast protein complexes. *Nucleic Acids Research*, 37(3):825–831, 2009.

[46] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H Lackner, Jürg Bähler, Valerie Wood, and et al. The biogrid interaction database: 2008 update. *Nucleic Acids Research*, 36:D637–D640, 2008.

[47] Rodrigo Aldecoa and Ignacio Marín. Jerarca: Efficient analysis of complex networks using hierarchical clustering. *PLoS ONE*, 5(7):7, 2010.

[48] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 80(5 Pt 2):056117, 2009.

[49] E C Pielou. The measurement of diversity in different types of biological collections. *Journal of Theoretical Biology*, 13(1):131–144, 1966.

[50] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

[51] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 72(2 Pt 2):027104, 2005.

[52] W W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[53] Linton C Freeman. Finding groups with a simple genetic algorithm. *The Journal of Mathematical Sociology*, 17(4):227–241, 1993.

[54] Robert H Macarthur. On the relative abundance of bird species. *Proceedings of the National Academy of Sciences of the United States of America*, 43(3):293–295, 1957.

[55] M E J Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582, 2006.

[56] Anne Condon and Richard M Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[57] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):11, 1999.

[58] S Boccaletti, V Latora, Y Moreno, M Chavez, and D Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

[59] Nguyen Xuan Vinh, Jilen Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[60] Brian Karrer, Elizaveta Levina, and M E J Newman. Robustness of community structure in networks. *Physical Review E*, 77(4):10, 2007.

[61] Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):18, 2011.